第五次计算物理作业

吴远清 2018300001031

第一题

首先, 我们将抛球运动分为一下三种, 由简单到复杂的模型:

- 无阻力无旋转模型
- 有阻力无旋转模型
- 有阻力有旋转模型

在此作业中, 我们导入`vpython`库来实现3维绘图, 导入`math`库来进行数学计算, 并设置常数如下:

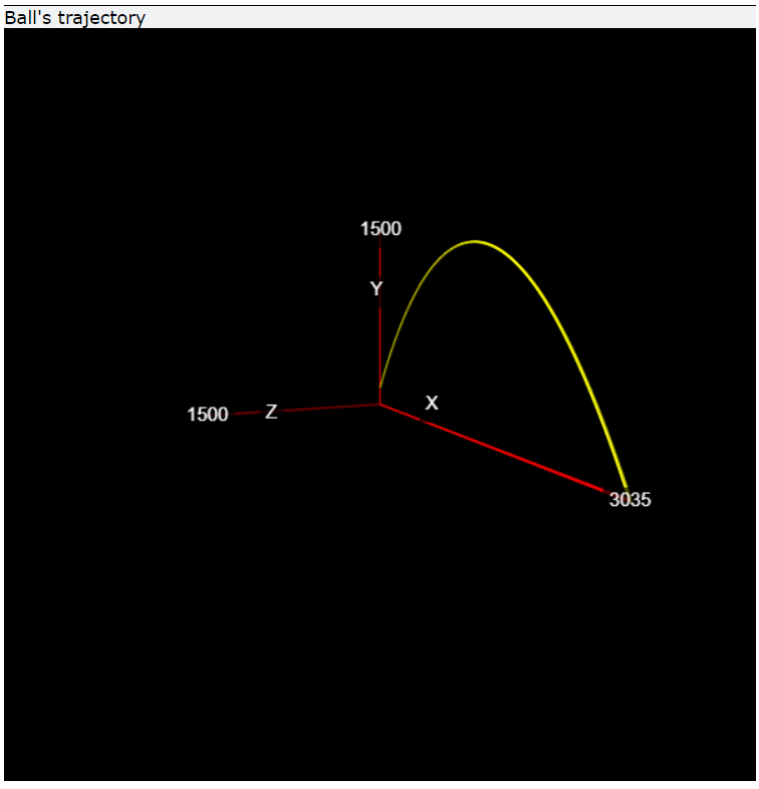
```
from vpython import *
import math

g = 9.8
zoomFactor = 100
airDensity = 1.293
```

无阻力无旋转情况

首先我们实现理想情况下的简单抛球(不考虑旋转和空气阻力),计算球体从抛出到落到地面的过程.并在此类中实现两种结果展示方式:

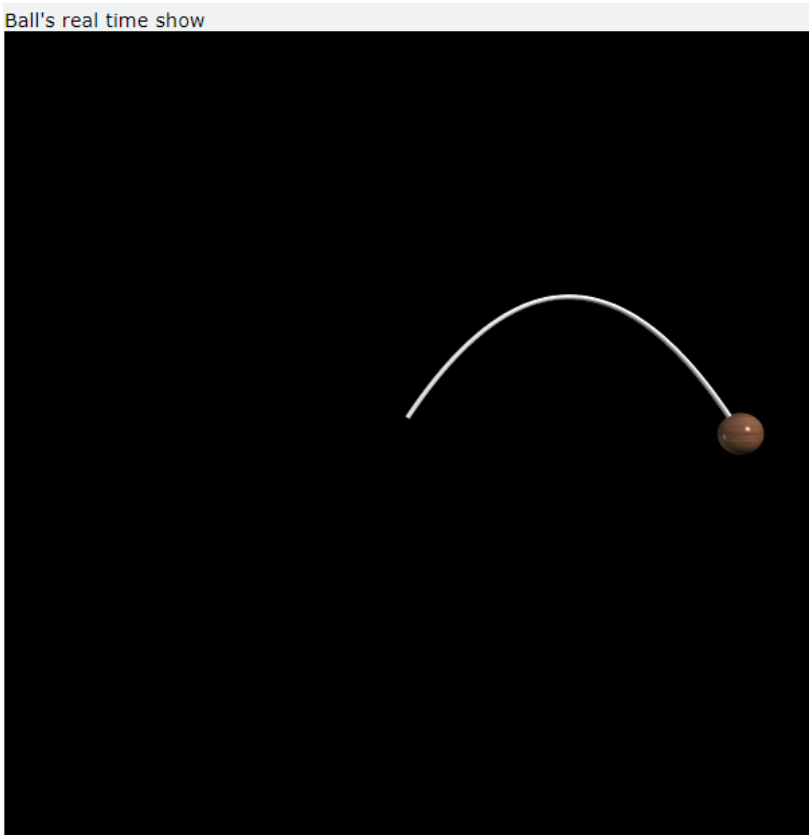
- 轨迹显示
- 实时球体显示

对于轨迹显示, 其计算结果的展示参照之前3维随机行走的代码.



但本代码中的轨迹显示可以随着最终落点的位置,而调整坐标轴的长度,并标出坐标轴的长度.

实时球体显示可以实现球体位置的显示,通过`sphere`函数来绘制球体,材质选为`textures.wood`,并且`make_trail`参数设为`True`,以显示轨迹.



代码如下:

```
class simpleBall():
    def __init__(self, x0 = 0, y0 = 1.5, z0 = 0, vx0 = 10, vy0 = 15, vz0 = 0, dt
= 0.001):
        assert y0 > 0, "Error: Positive y0 was expected, not {}.".format(y0)
        self.dt = dt
        self.t = [0]
        self.x = [x0]
        self.y = [y0]
        self.z = [z0]
        self.vx = [vx0]
        self.vy = [vy0]
        self.vz = [vz0]
    def calc(self):
        while True:
            self.t.append(self.t[-1] + self.dt)
            self.vx.append(self.vx[-1])
            self.vy.append(self.vy[-1] - g * self.dt)
            self.vz.append(self.vz[-1])
            self.x.append(self.x[-1] + self.dt * self.vx[-1])
            self.y.append(self.y[-1] + self.dt * self.vy[-1])
            self.z.append(self.z[-1] + self.dt * self.vz[-1])
            if self.y[-1] <= 0:
                break
    def trajectoryShow(self):
        xmax = max([max(self.x)*zoomFactor,1500])
        ymax = max([max(self.y)*zoomFactor,1500])
        zmax = max([max(self.z)*zoomFactor,1500])
```

```
        scene = canvas(x=0, y=0, width=600, height=600, title="Ball's
trajectory")
        xax   = curve(x=list(range(0,1500)), color=color.red, pos=
[vector(0,0,0),vector(xmax,0,0)], radius=10.)
        yax   = curve(x=list(range(0,1500)), color=color.red, pos=
[vector(0,0,0),vector(0,ymax,0)], radius=10.)
        zax   = curve(x=list(range(0,1500)), color=color.red, pos=
[vector(0,0,0),vector(0,0,zmax)], radius=10.)
        xname = label( text = "X", pos = vector(1000, 150,0), box=0)
        yname = label( text = "Y", pos = vector(-100,1000,0), box=0)
        zname = label( text = "Z", pos = vector(100, 0,1000), box=0)
        xmaxName = label(text = '{}'.format(int(xmax)),pos = vector(xmax,0,0),
box = 0)
        ymaxName = label(text = '{}'.format(int(ymax)),pos = vector(0,ymax,0),
box = 0)
        zmaxName = label(text = '{}'.format(int(zmax)),pos = vector(0,0,zmax),
box = 0)
        c = curve(x=list(range(0, len(self.t))),radius=10.0,color=color.yellow)
        for i in range(len(self.t)):
            c.append(vector(self.x[i]*zoomFactor,
self.y[i]*zoomFactor,self.z[i]*zoomFactor))
            rate(int(1/self.dt))
    def realtimeShow(self):
        scene = canvas(x=0, y=0, width=600, height=600, title="Ball's real time
show")
        ball = sphere(radius = 2, texture=textures.wood, make_trail = True)
        for i in range(len(self.t)):
            ball.pos = vector(self.x[i],self.y[i],self.z[i])
            rate(int(1/self.dt))
```

有阻力无旋转情况

我参照Nicholas书中(2.26)式:

$$\frac{B_2}{m} = 0.0039 + \frac{0.0058}{1 + exp[(v - v_d)/\Delta]}$$

来计算阻力系数, 阻力如下式:

$$F_{drag,i} = -B_2 \left|\vec{v} - \vec{v}_{wind}\right| (v_i - v_{wind,i})$$

此模型下, 我首先继承自`simpleBall`类, 并在`__init__`方法中使用`super().__init__()`来继承父类的初始化过程, 并添加新的参数.

之后, 我重写了`calc`方法,使得计算中会考虑空气阻力的影响.

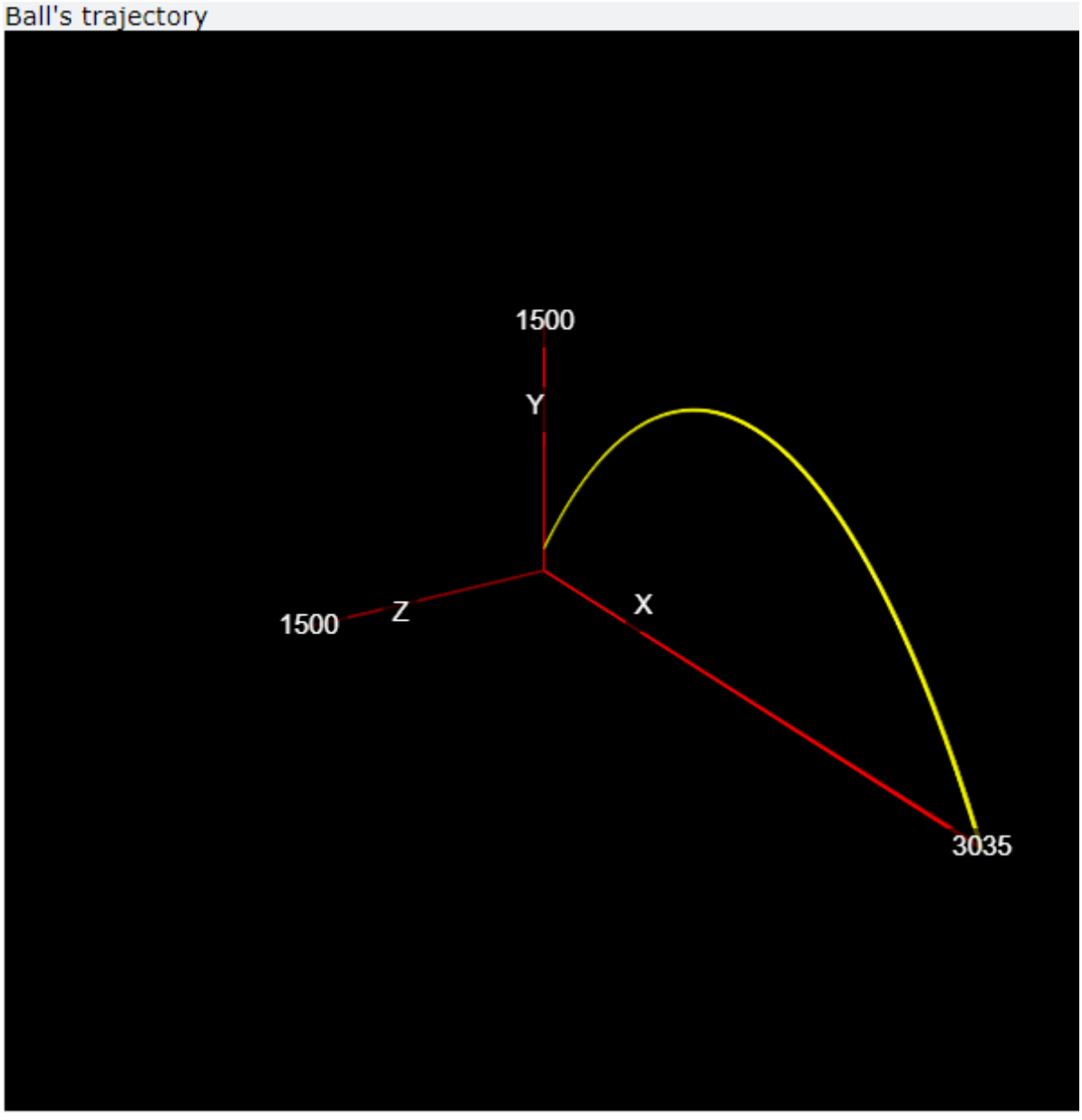
对于`trajectoryShow`和`realtimeShow`方法,直接继承自父类, 而不需要重写.

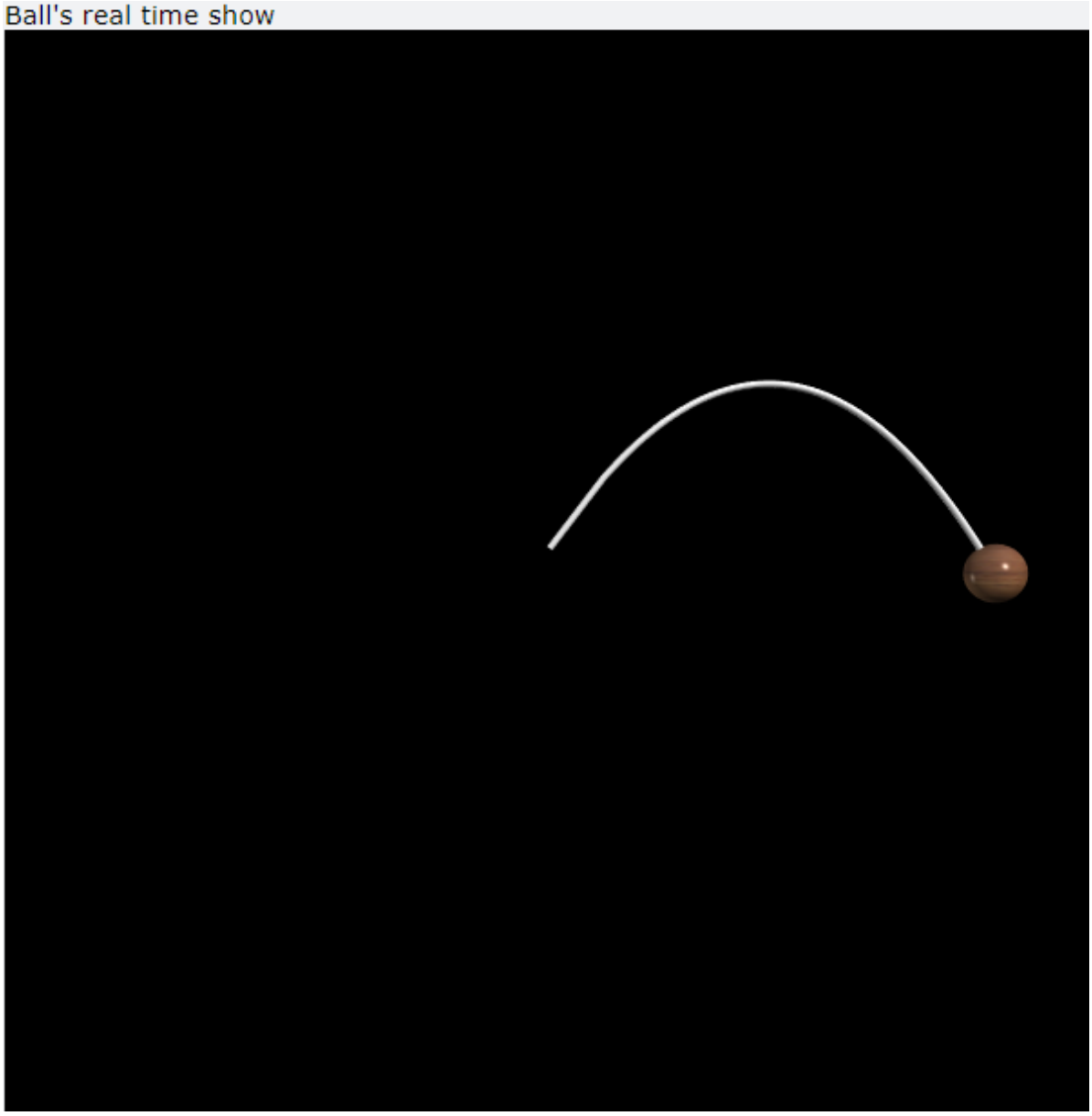
代码如下:

```
class resistanceBall(simpleBall):
    def __init__(self, vwindx = -1, vwindy = 0, vwindz = 0, mass = 0.149,vd =
35,delta = 5):
        super().__init__()
        self.vwx = vwindx
        self.vwy = vwindy
        self.vwz = vwindz
        self.m = mass
```

```
        self.vd = vd
        self.delta = delta
    def calc(self):
        while True:
            v = math.sqrt(self.x[-1]**2 + self.y[-1]**2 + self.z[-1]**2)
            dv = math.sqrt((self.x[-1] - self.vwx)**2 + (self.y[-1] -
self.vwy)**2 + (self.z[-1] - self.vwz)**2)
            self.t.append(self.t[-1] + self.dt)
            self.vx.append(self.vx[-1] - self.m * (0.0039 + 0.0058/(1 +
math.exp(v - self.vd)/self.delta)) * dv * (self.vx[-1] - self.vwx)*self.dt)
            self.vy.append(self.vy[-1] - g * self.dt - self.m * (0.0039 +
0.0058/(1 + math.exp(v - self.vd)/self.delta)) * dv * (self.vy[-1] -
self.vwy)*self.dt)
            self.vz.append(self.vz[-1] - self.m * (0.0039 + 0.0058/(1 +
math.exp(v - self.vd)/self.delta)) * dv * (self.vz[-1] - self.vwz)*self.dt)
            self.x.append(self.x[-1] + self.dt * self.vx[-1])
            self.y.append(self.y[-1] + self.dt * self.vy[-1])
            self.z.append(self.z[-1] + self.dt * self.vz[-1])
            if self.y[-1] <= 0:
                break
```

结果如下:





可以发现, 相较于未引入空气阻力时, 此模型下, 球飞行的距离和最大高度均减少.

有阻力有旋转情况

最后,我来处理题目所要求的情况.

对于旋转的棒球, 我们仅考虑旋转轴沿竖直方向, 且ω不发生改变的情况,我们参照书上的公式:

$$\overrightarrow{F_M} = S_0\vec{\omega} \times \vec{v}$$

值得考虑的是, 马格努斯力F_M的方向. 其应该垂直于$\vec{\omega}$和$\vec{v}$,考虑到$\vec{\omega}$在我们现在考虑的情况下恒沿y轴正方向, 那么$\overrightarrow{F_M}$一定位于$x-z$平面内. 垂直于$\vec{v}$在$x-z$平面内的投影. 有:

$$\theta = \arctan(v_z/v_x)$$

可以求得分量:

$$F_{M_x} = |F_M| \sin\theta$$

$$F_{M_z} = -|F_M| \cos\theta$$

其中,$|F_M|$为:

$$|F_M| = S_0 \, |\omega_0| \cdot |v| \sin(\pi - \alpha) = S_0 \, |\omega_0| \cdot |v| \sin\alpha = S_0 \, |\omega_0| \cdot |v| \frac{\sqrt{v_x^2 + v_y^2}}{|v|}$$

类似的,此情况下继承自 `resistanceBall`,并在 `__init__` 方法中使用 `super().__init__()` 来继承父类的初始化过程, 并添加新的参数.

我们重写 `calc` 方法,来考虑马格努斯力和Lateral Force的影响.

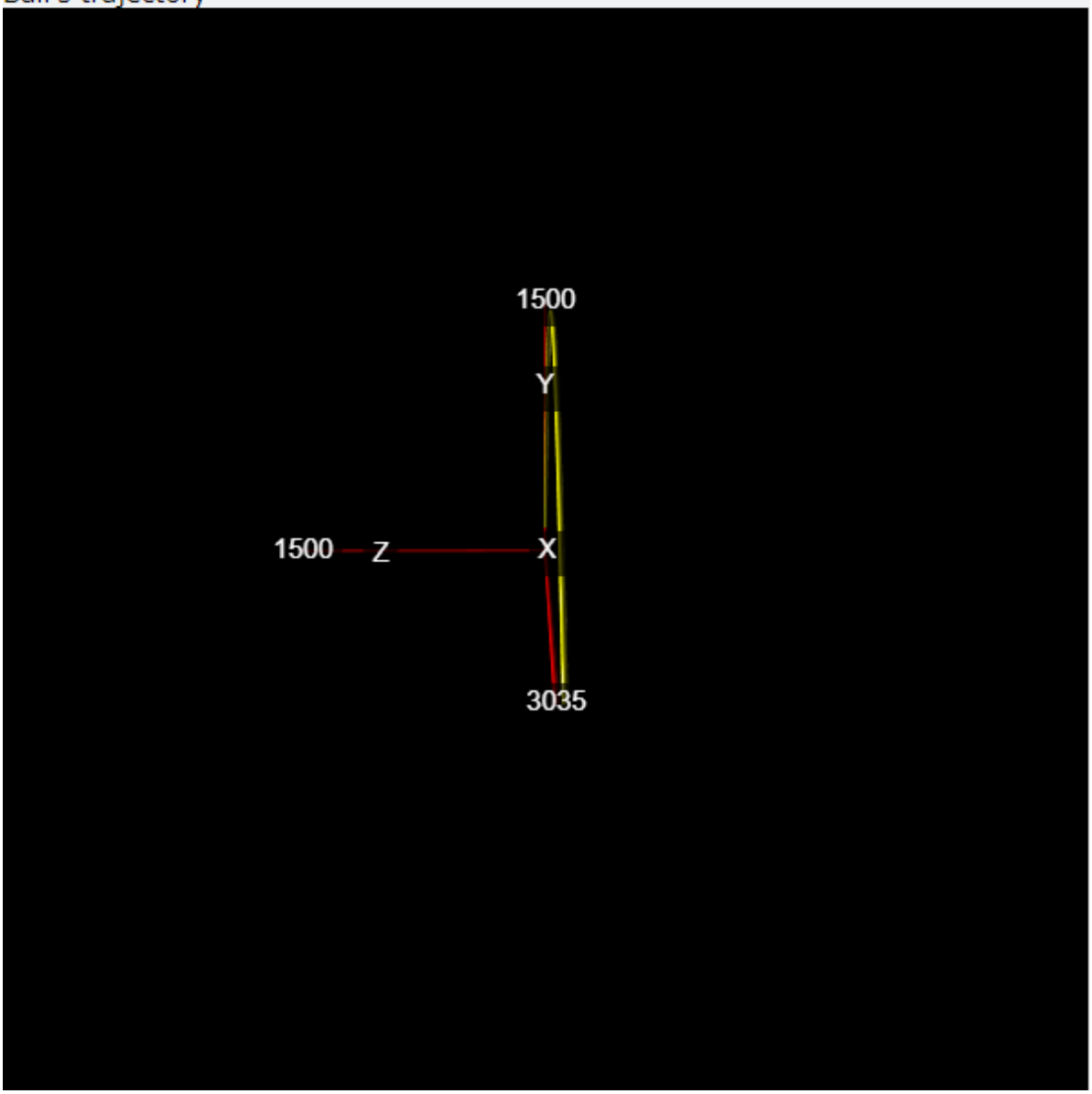
相似的, 我们不需要对 `trajectoryShow` 进行改动. 但是,我们需要重写 `realtimeSHow` 方法来实现对旋转的展示. 我们通过 `sphere` 的 `rotate` 方法来实现这一目的.

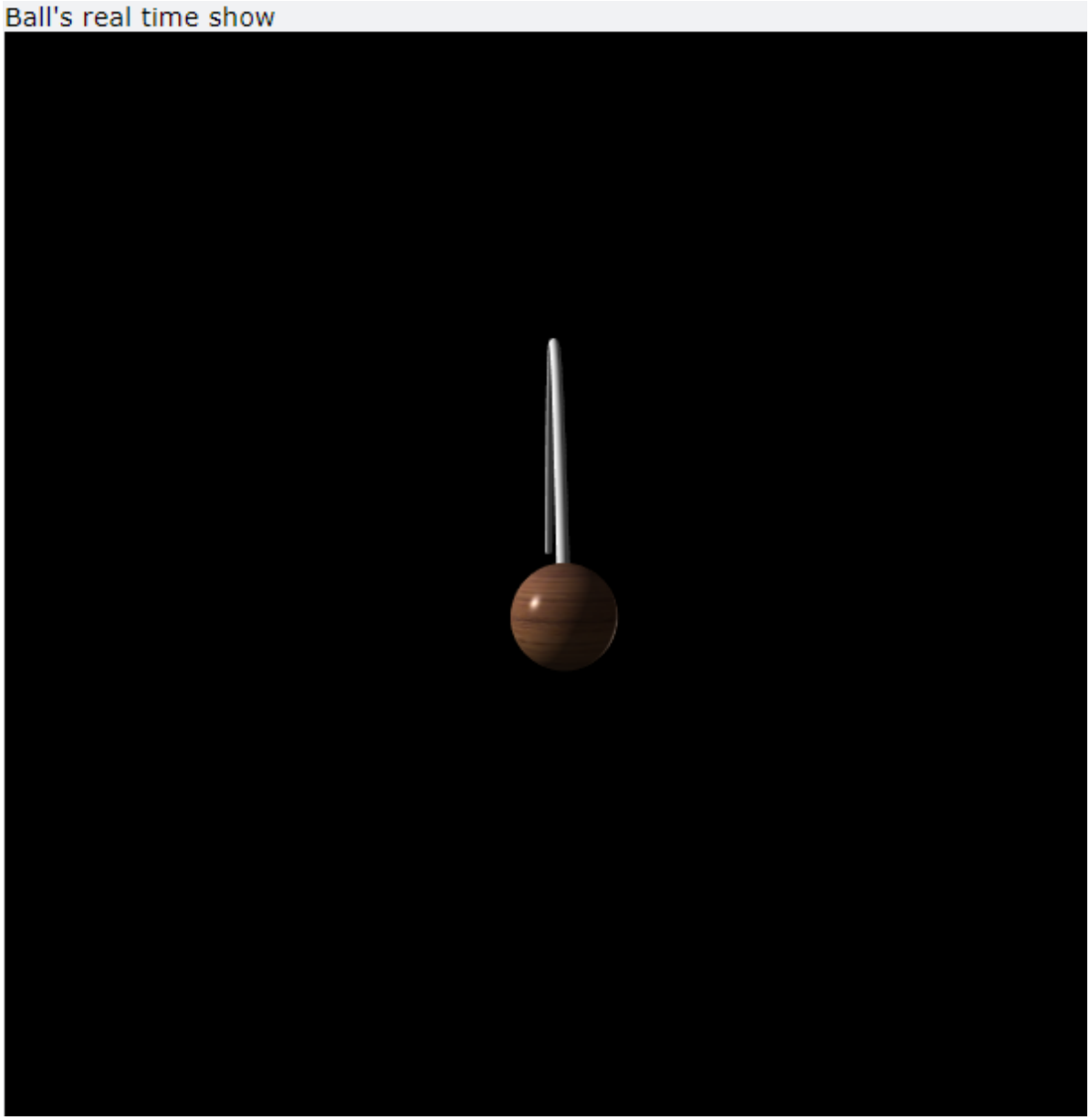
代码如下:

```
class spinResistanceBall(resistanceBall):
    def __init__(self, omega = 10, s0 = 0.00041):
        super().__init__()
        self.omega = omega
        self.angle = [0]
        self.s0 = self.m * s0
    def calc(self):
        while True:
            v = math.sqrt(self.vx[-1]**2+self.vy[-1]**2+self.vz[-1]**2)
            fm = self.s0 * self.omega * v * math.sqrt(self.vx[-1]**2 +
self.vy[-1]**2)/v
            theta = math.atan(self.vz[-1]/self.vx[-1])
            fmx = fm*math.sin(theta)
            fmz = - fm * math.cos(theta)
            alateral = g *
(sin(4*self.angle[-1])-0.25*sin(8*self.angle[-1])+0.08*sin(12*self.angle[-1])-0.
025*sin(16*self.angle[-1]))
            v = math.sqrt(self.x[-1]**2 + self.y[-1]**2 + self.z[-1]**2)
            dv = math.sqrt((self.x[-1] - self.vwx)**2 + (self.y[-1] -
self.vwy)**2 + (self.z[-1] - self.vwz)**2)
            self.t.append(self.t[-1] + self.dt)
            self.vx.append(self.vx[-1] - self.m * (0.0039 + 0.0058/(1 +
math.exp(v - self.vd)/self.delta)) * dv * (self.vx[-1] - self.vwx)*self.dt +
fmx/self.m * self.dt)
            self.vy.append(self.vy[-1] - g * self.dt - self.m * (0.0039 +
0.0058/(1 + math.exp(v - self.vd)/self.delta)) * dv * (self.vy[-1] -
self.vwy)*self.dt)
            self.vz.append(self.vz[-1] - self.m * (0.0039 + 0.0058/(1 +
math.exp(v - self.vd)/self.delta)) * dv * (self.vz[-1] - self.vwz)*self.dt +
fmz/self.m * self.dt)
            self.x.append(self.x[-1] + self.dt * self.vx[-1])
            self.y.append(self.y[-1] + self.dt * self.vy[-1])
            self.z.append(self.z[-1] + self.dt * self.vz[-1])
            self.angle.append(self.angle[-1] + self.omega * self.dt)
            if self.y[-1] <= 0:
                break
    def realtimeShow(self):
        scene = canvas(x=0, y=0, width=600, height=600, title="Ball's real time
show")
        ball = sphere(radius = 2, texture=textures.wood, make_trail = True)
        for i in range(len(self.t)):
            ball.pos = vector(self.x[i],self.y[i],self.z[i])
            if i >= 1:
```

```
                ball.rotate(angle = self.angle[i] - self.angle[i-1], axis =
vec(0,1,0), origin = ball.pos)
            rate(int(1/self.dt))
```

Ball's trajectory





可以明显看出, 相较于前两种情况, 考虑旋转时, 球的轨迹会发生z方向上的偏转.

使用图片无法展示对旋转的描绘, 可以点击如下链接, 在线查看运行结果.(由于运行环境不同, 对代码进行了重构)

https://www.glowscript.org/#/user/yqwu/folder/MyPrograms/program/ComputationalPhysics5thHomework