<u>Project: Double Pendulum in Vpython</u>

Made by: Muhammad Umer

Roll no. 20K-0225

Assigned by: Sir Waqar Ahmed

Problem Statement:

Simulate a Double pendulum and demonstrate chaotic motion for the farthest end

of pendulum.

Solution:

Simulation of a double pendulum, for large motions, is a chaotic system, but for small motions it is a simple linear system.

You can change parameters in the simulation such as mass, gravity, and length of rods. You can drag the pendulum with your mouse to change the starting position.

When the angles are small in the Double Pendulum, the system behaves like the linear Double Spring. Motion is determined by simple sine and cosine functions. For large angles, the pendulum is non-linear. We regard the pendulum rods as being massless and rigid. We regard the pendulum masses as being point masses.

In simulation, the viewer will be able to observe change in motion of the both pendulums by time.

Procedure:

<u>Objects in simulation:</u>

1. Sphere of suitable radius, named ball, as our object of motion.

2. Ropes to hold the balls.

3. Stand to hold the ropes.

4. Axes and origin will be represented by curve and a sphere

Variables:

1. Gravitational field strength ‘g’
2. Mass of ball 1 ‘Mass1’
3. Mass of ball 2 ‘Mass2’
4. Diameter of balls ‘d’
5. Gap between balls ‘gap’
6. Length of ball 1 ‘Len1’
7. Length of ball 2 ‘Len2’
8. Radius of ball 1 ‘R1’
9. Radius of ball 2 ‘R2’

Initial values:

We will set the initial value of simulation height and width to 1000

Background color in vector form as vec(0,0,0) which is black

Code Block:

The following code was used to carry out the simulation results shown in results below.

```
scene.height = scene.width = 1000

scene.background = vec(0,0,0)


g = 9.8

Mass1 = 2

Mass2 = 1

d = 0.05

gap = 2*d

Len1 = 0.5

Len1display = Len1+d

Len2 = 1

Len2display = Len2+d/2

I1 = Mass1*Len1**2/12

I2 = Mass2*Len2**2/12

R1 = Len1/2

R2 = Len2/2-d/4
```

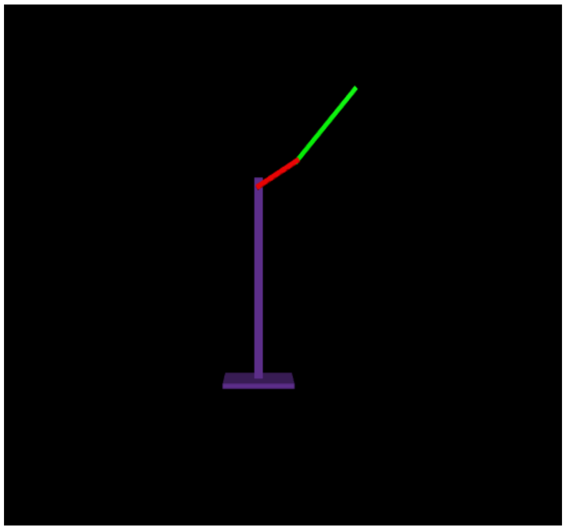
```
hpedestal = 1.3*(Len1+Len2)

wpedestal = 0.1

tbase = 0.05

wbase = 8*gap

offset = 2*gap

topofbar = vector(0,0,0)

scene.center = topofbar-vector(0,Len1,0)

pedestal = box(pos=topofbar-vector(0,hpedestal/2,0), height=1.1*hpedestal, length=wpedestal,
width=wpedestal, color=vector(0.4,0.2,0.6))

base = box(pos=topofbar-vector(0,hpedestal+tbase/2,0), height=tbase, length=wbase, width=wbase,
color=pedestal.color)

axle = cylinder(pos=topofbar-vector(0,0,offset-gap/2+d/5), axis=vector(0,0,2*(offset-gap/2+d/5)),
radius=d/4, color=color.yellow)


class pendulum:

    def __init__(self, pos=topofbar, theta1=1, theta1dot=0, theta2=0, theta2dot=0):

        self.pos = pos

        self.initial_theta1 = theta1

        self.initial_theta1dot = theta1dot

        self.initial_theta2 = theta2

        self.initial_theta2dot = theta2dot

        b1 = box(pos=pos+vector(R1,0,-(gap+d)/2), size=vector(Len1display,d,d), color=color.red)

        b2 = box(pos=pos+vector(R1,0,(gap+d)/2), size=vector(Len1display,d,d), color=color.red)

        c = cylinder(pos=pos+vector(Len1,0,-(gap+d)/2), axis=vector(0,0,gap+d), radius=axle.radius,
color=color.green)

        self.frame1 = compound([b1,b2,c])

        self.frame1.rotate(angle=self.initial_theta1-pi/2, axis=vector(0,0,1), origin=vector(0,0,0))

        self.frame1.initial_pos = vector(self.frame1.pos)

        self.frame1.initial_axis = vector(self.frame1.axis)


        b3 = box(pos=vector(R2,0,0), size=vector(Len2display,d,d), color=color.green)
        self.frame2 = compound([b3])
        self.reset()
        self.__visible = True

    def update(self, dtheta1=0, dtheta2=0):
        self.theta1 += dtheta1
        self.theta2 += dtheta2
        self.frame1.rotate(angle=dtheta1, axis=vector(0,0,1), origin=self.pos)
        axle2pos = self.frame1.pos+R1*norm(self.frame1.axis)
        self.frame2.rotate(angle=dtheta2, axis=vector(0,0,1), origin=axle2pos)
        self.frame2.pos = axle2pos+R2*norm(self.frame2.axis)

    def reset(self):
        self.theta1 = self.initial_theta1
        self.theta1dot = self.initial_theta1dot
        self.theta2 = self.initial_theta2
        self.theta2dot = self.initial_theta2dot
        self.frame1.pos = self.frame1.initial_pos
        self.frame1.axis = self.frame1.initial_axis
        axle2pos = self.frame1.pos+R1*norm(self.frame1.axis)
        self.frame2.pos = axle2pos+vector(R2,0,0)
        self.frame2.axis = vector(Len2display,0,0)
        self.frame2.rotate(angle=self.theta2-pi/2, axis=vector(0,0,1), origin=axle2pos)

    def get_visible(self):
        return self.__visible

    def set_visible(self, vis):
        self.__visible = self.frame1.visible = self.frame2.visible = vis
```

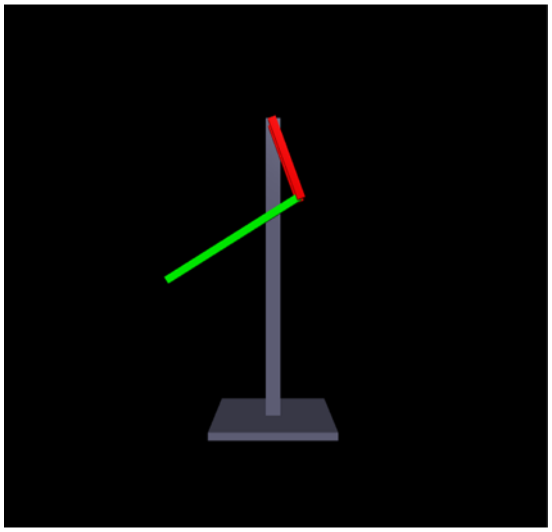
```
pend1 = pendulum(topofbar+vector(0,0,offset), 2.1, 0, 2.4, 0)

pend2 = pendulum(topofbar+vector(0,0,-offset), 2.1, 0, 2.4, 0)

pend2.initial_theta1 += 0.001

pend2.reset()

pend2.set_visible(False)


pendula = [pend1, pend2]

scene.autoscale = False


run = False


def reset():

    global t, gd, energy_check

    t = 0

    ResetRunbutton()

    for pend in pendula:

        pend.reset()

    if gd:

        gd.delete()

        gd = None

        energy_check = False

        EnergyCheck(None)


def ResetRunbutton():

    global run

    run = False

    Runbutton.text = "Run"


def Runb(r):

    global run
    run = not run
    if run:
        r.text = "Pause"
    else:
        r.text = "Run"

Runbutton = button(text='Run', bind=Runb)

scene.append_to_caption('   ')

button(text='Reset', bind=reset)

scene.append_to_caption('\n\nChoose a situation:   ')

options = ['One double pendulum', 'Two double pendula, starting with angles that differ by only 0.001
radian']

def choosependula(p):
    if p.selected == options[0]:
        if not pendula[1].get_visible(): return
        pendula[1].set_visible(False)
    else:
        if pendula[1].get_visible(): return
        pendula[1].set_visible(True)
    reset()

setmenu = menu(choices=options, selected='One double pendulum', bind=choosependula)

scene.append_to_caption('\n \n')
```

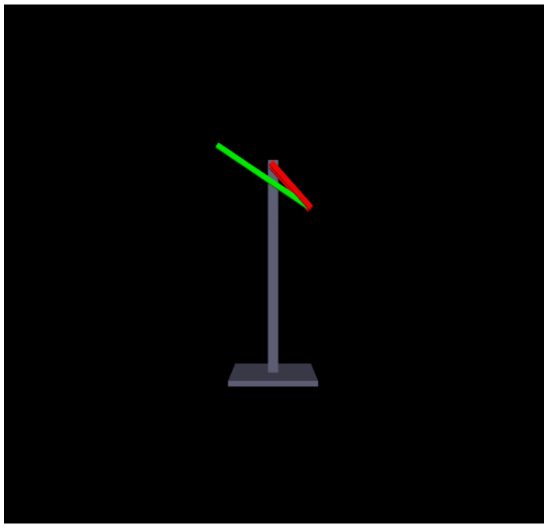
```
energy_check = False
gd = None
gK = None
gU = None
gE = None

def EnergyCheck(ec):
    global energy_check, graphing, gd, gK, gU, gE
    energy_check = not energy_check
    if energy_check:
        if not gd:
            gd = graph(width=scene.width, height=300, title='K is green, U is blue, E = K+U is red')
            t = 0
        gK = gcurve(color=color.green)
        gU = gcurve(color=color.blue)
        gE = gcurve(color=color.red)

checkbox(bind=EnergyCheck)
scene.append_to_caption(' Graph energy (for front pendulum)\n\n')



dt = 0.0002
t = 0
n = 0

C11 = (0.25*Mass1+Mass2)*Len1**2+I1
C22 = 0.25*Mass2*Len2**2+I2

while True:
    rate(1/dt)
    if not run: continue
    for pend in pendula:

        C12 = C21 = 0.5*Mass2*Len1*Len2*cos(pend.theta1-pend.theta2)
        Cdet = C11*C22-C12*C21
        a = .5*Mass2*Len1*Len2*sin(pend.theta1-pend.theta2)
        A = -(.5*Mass1+Mass2)*g*Len1*sin(pend.theta1)-a*pend.theta2dot**2
        B = -.5*Mass2*g*Len2*sin(pend.theta2)+a*pend.theta1dot**2
        pend.atheta1 = (C22*A-C12*B)/Cdet
        pend.atheta2 = (-C21*A+C11*B)/Cdet

        pend.theta1dot += pend.atheta1*dt
        pend.theta2dot += pend.atheta2*dt

        dtheta1 = pend.theta1dot*dt
        dtheta2 = pend.theta2dot*dt
        pend.update(dtheta1, dtheta2)

    t = t+dt
    n += 1

    if energy_check and n >= 100:
        n = 0
        K =
.5*((.25*Mass1+Mass2)*Len1**2+I1)*pend1.theta1dot**2+.5*(.25*Mass2*Len2**2+I2)*pend1.theta2d
ot**2+\
            .5*Mass2*Len1*Len2*cos(pend1.theta1-pend1.theta2)*pend1.theta1dot*pend1.theta2dot
        U = -(.5*Mass1+Mass2)*g*Len1*cos(pend1.theta1)-.5*Mass2*g*Len2*cos(pend1.theta2)
        gK.plot(pos=(t,K))
        gU.plot(pos=(t,U))
        gE.plot(pos=(t,K+U))
```

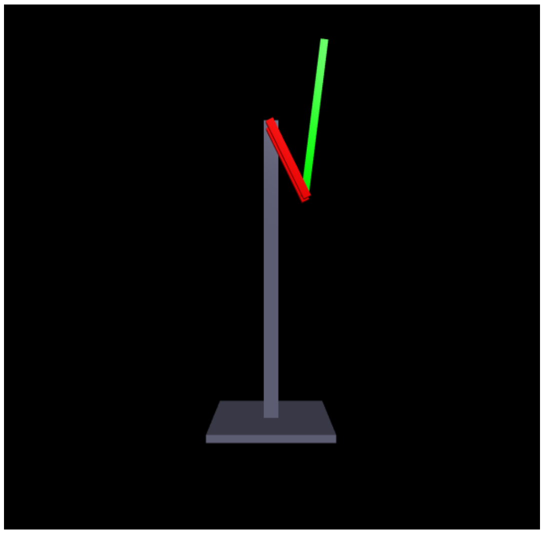
Result:

Figures below show the successful simulation of the said problem. Changing the masses of balls completely changes the motion of this double pendulum. Changing the length of rope also varies motion. The first figure shows the reset position.



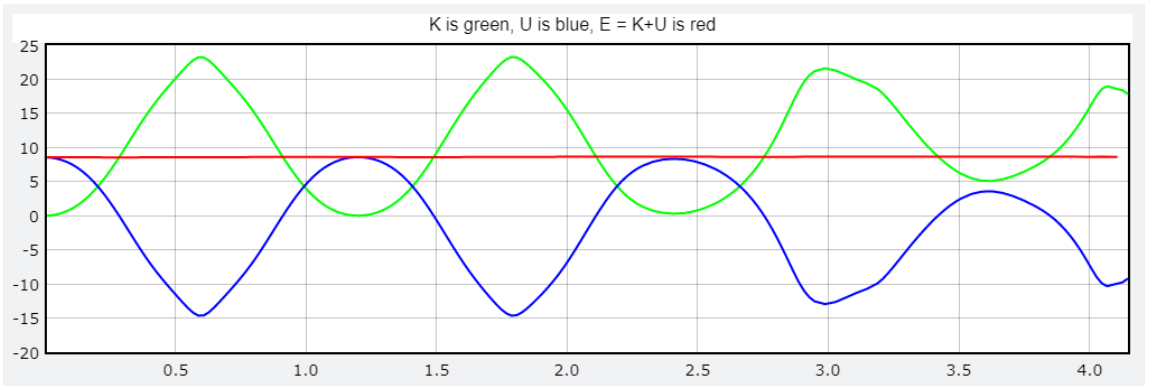




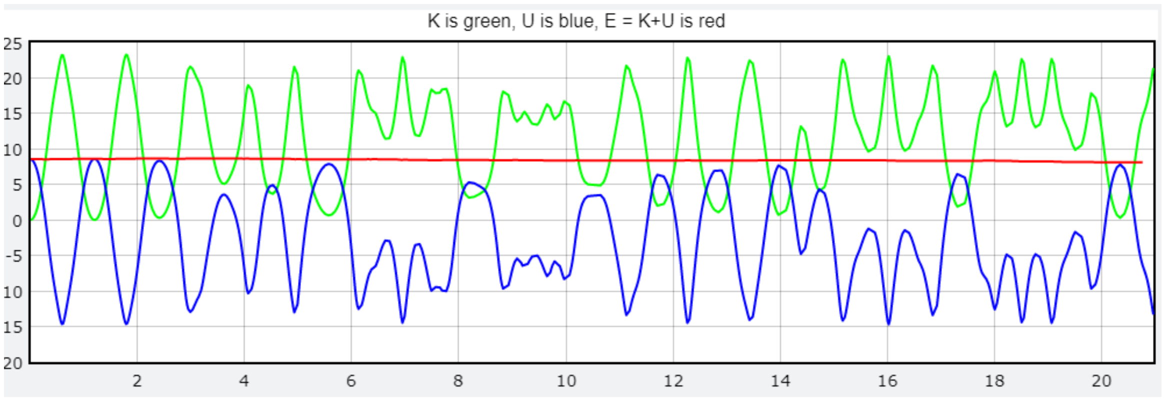


Graphs:

Graph of energy at the starting:



Graph of energy after some time:



Conclusion:

We were easily able to demonstrate the motions of double pendulum with the help of Vpython with respect to time. This demonstration shows the chaotic motion of both balls with each other. We were able to conclude that the double pendulum was highly sensitive to initial starting conditions, as shown by the increasing difference between its angles and coordinate positions between a control and variations in initial velocities and angles.