```python
from visual import*
from visual.graph import*

#BUILD A TCOORDINATE SYSTEM
wall1=box(pos=(0,0,0),size=(0.1,0.1,15),color=color.red)
wall2=box(pos=(0,0,0),size=(15,0.1,0.1),color=color.blue)
wall3=box(pos=(0,0,0),size=(0.1,15,0.1),color=color.green)

#INITIAL CONDITIONS
block=box(pos=(0,0,0),size=(2,2,2),color=color.white)
m_block=1                       #mass, size, color, position of the spring oscillator
g=9.8                           #gravitational acceleration
F_grav = vector (0, -g*m_block, 0)    #gravity of mass
spring=( helix(pos=(0,10,0),axis=(0,-10,0),radius = 0.4, thickness = 0.4))
L0=10                          #initial position, direction, size, length of the spring
ks=3                           #the spring constant
deltat=0.01                     #a brief time interval
p_block=vector(0,0,0)             #momentum of spring

#TRAJECTORY VISUALIZATION
block.trail=curve(color=block.color)
t=0
p=gcurve(color=color.yellow)
q=gcurve(color=color.red)   #distinguish two curves by different color and different name p&q

#MOTION LOOP
while 1:                   #plotting forever
    rate (200)             #set an appropriate frequency for the movement
    t=t+deltat             #the loop begins at this instant
    #Change of variable
    L =  block.pos - spring.pos  #current length of spring
    Lhat = L/ mag (L)           #direction vector of L
    s = mag (L) - L0            #axial deformation of spring
    F_spring =  -ks *s*Lhat     #restoring force of the spring
    F_net = F_grav + F_spring     #vector sum of two forces
    p_block = p_block + F_net * deltat                #the change of momentum
    block.pos = block.pos +(p_block/m_block) * deltat   #the change of position
    spring.axis=block.pos+(0,-10,0)                 #direction of restoring force and spring
    block.trail.append(pos=block.pos)                #define position of the trail

#Calling the 'plot function'
#changing the vector position into a scalar and make it positive
    if block.pos.y>0:
        pos=mag(block.pos)
    else:
        pos=-mag(block.pos)
    p.plot(pos=(t,pos))                   #plot the position versus time curve

    if p_block.y>0:
        pb=mag(p_block)
    else:
        pb=-mag(p_block)
    q.plot(pos=(t,pb))                    #plot the momentum (velocity) versus time curve
```