from visual import \*

# Constants and time step

N = 6 # number of bobs

R = 0.3 # Radius of bob (separation between bobs=1)

Ks = 50. # K of springs(masses=1)

g = 9.8 # Relative strength of gravity

gamma= 0.03 # Some friction

Rgdt = 1.0 # Rigidity factor (Rgdt=0 is a regular spring)

Dt = 0.002\*sqrt(1/g) # Full time step

Dt2 = Dt/2.0 # Midpoint time step

DiaSq = (2\*R)\*\*2 # Diameter of bob squared

RandomStart = False

if RandomStart: # Get the random number generator

from random import \*

seed()

Cbob0 = color.red

Cbob = color.yellow

Cstring = color.cyan

# Properties of the display window

window = display(title="Multiple Pendulum", width=800, height=600)

window.fullscreen = 1 # Change to 0 to get a floating window

window.range = (2\*N,2\*N,2\*N)

window.cursor.visible = 0 # Hide the mouse

window.userspin = 0 # No rotation with mouse

window.forward = (0,0,-1) # +z-axis toward you

window.lights = [vector(0,0,1)] # Vector pointing to the light source

window.ambient=0

try:

window.material=None

except:

pass

# Create the initial positions and velocitites (0,0) of the bobs

bob\_x=[0.]

bob\_y=[0.85\*N]

x\_dot=[0.]\*(N+1)

y\_dot=[0.]\*(N+1)

for k in range(1,N+1):

if RandomStart:

alpha = uniform(0,2\*pi) # 2\*pi\*random()

else:

alpha = pi/5

bob\_x.append(bob\_x[k-1]+cos(alpha))

bob\_y.append(bob\_y[k-1]+sin(alpha))

# Create the bobs

bob=[sphere(pos=(bob\_x[0],bob\_y[0]), radius=R\*0.5, color=Cbob0)]

for k in range(1,N+1):

bob.append(sphere(pos=(bob\_x[k],bob\_y[k]), radius=R, color=Cbob))

# Create the string out of N links

link = [0]\*N

for k in range(N):

link[k] = cylinder(pos=bob[k].pos,axis = bob[k+1].pos-bob[k].pos,

radius=R/3, color=Cstring)

# Create some auxiliary variables

x\_dot\_m = [0.]\*(N+1)

y\_dot\_m = [0.]\*(N+1)

dij = [0.]\*(N+1) # array with distances to previous bob

dij\_m = [0.]\*(N+1)

for k in range(1,N+1):

dij[k] = sqrt((bob\_x[k]-bob\_x[k-1])\*\*2+(bob\_y[k]-bob\_y[k-1])\*\*2)

fctr = (lambda x: (x-1.+Rgdt\*(x-1.)\*\*3)/x)

# Click the mouse to start

window.mouse.getclick() # Empty the mouse queue

# The main loop

while True :

rate(1000)

# Compute the midpoint variables

bob\_x\_m = map((lambda x,dx:x+Dt2\*dx),bob\_x,x\_dot)

bob\_y\_m = map((lambda y,dy:y+Dt2\*dy),bob\_y,y\_dot)

for k in range(1,N+1):

dij\_m[k] = sqrt((bob\_x\_m[k]-bob\_x\_m[k-1])\*\*2 +

(bob\_y\_m[k]-bob\_y\_m[k-1])\*\*2)

for k in range(1,N+1):

factor = fctr(dij[k])

x\_dot\_m[k] = x\_dot[k] - Dt2\*(Ks\*(bob\_x[k]-bob\_x[k-1])\*fctr(dij[k])

+ gamma\*x\_dot[k])

y\_dot\_m[k] = y\_dot[k] - Dt2\*(Ks\*(bob\_y[k]-bob\_y[k-1])\*factor + g

+ gamma\*y\_dot[k])

for k in range(1,N):

factor = fctr(dij[k+1])

x\_dot\_m[k] -= Dt2\*Ks\*(bob\_x[k]-bob\_x[k+1])\*factor

y\_dot\_m[k] -= Dt2\*Ks\*(bob\_y[k]-bob\_y[k+1])\*factor

# Compute the full step variables

bob\_x = map((lambda x,dx:x+Dt\*dx),bob\_x,x\_dot\_m)

bob\_y = map((lambda y,dy:y+Dt\*dy),bob\_y,y\_dot\_m)

for k in range(1,N+1):

dij[k] = sqrt((bob\_x[k]-bob\_x[k-1])\*\*2+(bob\_y[k]-bob\_y[k-1])\*\*2)

for k in range(1,N+1):

factor = fctr(dij\_m[k])

x\_dot[k] -= Dt\*(Ks\*(bob\_x\_m[k]-bob\_x\_m[k-1])\*factor

+ gamma\*x\_dot\_m[k])

y\_dot[k] -= Dt\*(Ks\*(bob\_y\_m[k]-bob\_y\_m[k-1])\*factor + g

+ gamma\*y\_dot\_m[k])

for k in range(1,N):

factor = fctr(dij\_m[k+1])

x\_dot[k] -= Dt\*Ks\*(bob\_x\_m[k]-bob\_x\_m[k+1])\*factor

y\_dot[k] -= Dt\*Ks\*(bob\_y\_m[k]-bob\_y\_m[k+1])\*factor

# Check to see if the spheres are colliding

for i in range(1,N):

for j in range(i+1,N+1):

dist2 = (bob\_x[i]-bob\_x[j])\*\*2+(bob\_y[i]-bob\_y[j])\*\*2

if dist2 < DiaSq: # Spheres are colliding

Ddist = sqrt(dist2)-2\*R

tau = norm(vector(bob\_x[j]-bob\_x[i],bob\_y[j]-bob\_y[i]))

DR = Ddist/2\*tau

bob\_x[i] += DR[0]#DR.x

bob\_y[i] += DR[1]#DR.y

bob\_x[j] -= DR[0]#DR.x

bob\_y[j] -= DR[1]#DR.y

Vji = vector(x\_dot[j]-x\_dot[i],y\_dot[j]-y\_dot[i])

DV = dot(Vji,tau)\*tau

x\_dot[i] += DV[0]#DV.x

y\_dot[i] += DV[1]#DV.y

x\_dot[j] -= DV[0]#DV.x

y\_dot[j] -= DV[1]#DV.y

# Update the loations of the bobs and the links in the string

for k in range(1,N+1):

bob[k].pos = (bob\_x[k],bob\_y[k])

link[k-1].pos = bob[k-1].pos

link[k-1].axis = bob[k].pos-bob[k-1].pos