# events-example0.py

# Barebones timer, mouse, and keyboard events

from tkinter import \*

def subtract(a,b):

return [a[0]-b[0],a[1]-b[1]]

def add(a,b):

return [a[0]+b[0],a[1]+b[1]]

def magnitude(val):

return (val[0]\*\*2+val[1]\*\*2)\*\*0.5

def normalized(val):

mag = magnitude(val)

return [val[0]/mag,val[1]/mag]

class physobject(object):

def \_\_init\_\_(self,pos,vel,mass,world):

self.Mass = mass

self.Pos=pos

self.Vel=vel

self.Radius=(mass/3.14)\*\*.5

self.LastPos=pos

self.LastVel = vel

self.curgrav =[0,0]

self.World = world

def nextvelocity(self):

return add(self.LastVel,[self.curgrav[0]\*self.World.timeinterval,self.curgrav[1]\*self.World.timeinterval])

def startvelocity(self):

return add(self.LastVel,[self.curgrav[0]\*self.World.timeinterval/2,self.curgrav[1]\*self.World.timeinterval/2])

def nextpos(self):

vels = self.nextvelocity()

return add(self.LastPos,[vels[0]\*self.World.timeinterval,vels[1]\*self.World.timeinterval])

def startpos(self):

vels = self.startvelocity()

return add(self.LastPos,[vels[0]\*self.World.timeinterval,vels[1]\*self.World.timeinterval])

def pretick(self):

self.curgrav=self.curgravforce()

#(self.curgrav)

self.Vel=self.nextvelocity()

self.Pos=self.nextpos()

def tick(self):

self.LastVel=self.Vel

self.LastPos=self.Pos

def firstpretick(self):

self.curgrav=self.curgravforce()

self.Vel=self.startvelocity()

self.Pos=self.startpos()

def curgravforce(self):

force = [0,0]

for objec in self.World.Objects:

if(self.Pos==objec.Pos):

continue

delta = normalized(subtract(self.LastPos,objec.LastPos))

distance = magnitude(subtract(self.LastPos,objec.LastPos))

theforce = objec.Mass/(distance\*distance)

force = add(force,[delta[0]\*theforce,delta[1]\*theforce])

return [-force[0],-force[1]]

class world(object):

def \_\_init\_\_(self):

self.Objects=[]

self.timeinterval =0.01

def tick(self):

for thing in self.Objects:

thing.pretick()

for thing in self.Objects:

thing.tick()

def starttick(self):

for thing in self.Objects:

thing.firstpretick()

for thing in self.Objects:

thing.tick()

def defineobjects(world2):

world2.Objects.append(physobject([-150,0],[0,-14.1421356/2],30000,world2))

world2.Objects.append(physobject([150,0],[0,14.1421356/2],30000,world2))

world2.Objects.append(physobject([-600,0],[0,10],100,world2))

world2.Objects.append(physobject([1200,0],[0,7.07106781],200,world2))

world2.starttick()

####################################

# customize these functions

####################################

def init(data):

# load data.xyz as appropriate

data.world1=world()

data.scalefactor = 0.3

defineobjects(data.world1)

pass

def mousePressed(event, data):

# use event.x and event.y

pass

def keyPressed(event, data):

# use event.char and event.keysym

pass

def timerFired(data):

for i in range(0,100):

data.world1.tick()

pass

def redrawAll(canvas, data):

# draw in canvas

for object in data.world1.Objects:

canvas.create\_oval((object.Pos[0]\*data.scalefactor+object.Radius\*data.scalefactor+400,object.Pos[1]\*data.scalefactor+object.Radius\*data.scalefactor+400),(object.Pos[0]\*data.scalefactor-object.Radius\*data.scalefactor+400,object.Pos[1]\*data.scalefactor-object.Radius\*data.scalefactor+400),fill="yellow")

pass

####################################

# use the run function as-is

####################################

def run(width=300, height=300):

def redrawAllWrapper(canvas, data):

canvas.delete(ALL)

redrawAll(canvas, data)

canvas.update()

def mousePressedWrapper(event, canvas, data):

mousePressed(event, data)

redrawAllWrapper(canvas, data)

def keyPressedWrapper(event, canvas, data):

keyPressed(event, data)

redrawAllWrapper(canvas, data)

def timerFiredWrapper(canvas, data):

timerFired(data)

redrawAllWrapper(canvas, data)

# pause, then call timerFired again

canvas.after(data.timerDelay, timerFiredWrapper, canvas, data)

# Set up data and call init

class Struct(object): pass

data = Struct()

data.width = width

data.height = height

data.timerDelay = 1# milliseconds

init(data)

# create the root and the canvas

root = Tk()

canvas = Canvas(root, width=data.width, height=data.height)

canvas.pack()

# set up events

root.bind("<Button-1>", lambda event:

mousePressedWrapper(event, canvas, data))

root.bind("<Key>", lambda event:

keyPressedWrapper(event, canvas, data))

timerFiredWrapper(canvas, data)

# and launch the app

root.mainloop() # blocks until window is closed("bye!")

run(800, 800)