```
#Gravity Simulator
import tkinter
from tkinter import BOTH, HORIZONTAL, CURRENT, END
from matplotlib import pyplot

#Define window
root = tkinter.Tk()
root.title('Gravity Simulator')
root.iconbitmap('earth.ico')
root.geometry('500x650')
root.resizable(0,0)

#Define fonts and colors
#NONE use system defualts

#Define global variables
time = 0
data = {}
for i in range(1,5):
    data['data_%d' % i] = []

#Define functions
def move(event):
    """Drag the balls vertically on the canvas to set the position."""
    #If the current object clicked has the "BALL" tag, we should allow it to be moved.
    if "BALL" in main_canvas.gettags(CURRENT):
        #Record the x position of the ball and keep it the same.
        x1 = main_canvas.coords(CURRENT)[0]
        x2 = main_canvas.coords(CURRENT)[2]

        #Change the coords of the CURRENT object based on the event.y position of the
        mouse.  Recall the ball has size 10
        main_canvas.coords(CURRENT, x1, event.y, x2, event.y+10)

        #Attempt to not move the ball off the canvas.  CURRENT[3] is y2 coord
        #Above the top of the screen
        if main_canvas.coords(CURRENT)[3] < 15:
            main_canvas.coords(CURRENT, x1, 5, x2, 15)
        #Below the bottom of the screen
        elif main_canvas.coords(CURRENT)[3] > 415:
            main_canvas.coords(CURRENT, x1, 405, x2, 415)

    #Update the height label for each ball
    update_height()


def update_height():
    """Update the height labels for each ball."""
    for i in range(1,5):
        heights['height_%d' % i].config(text="Height: " + str(round(415 -
        main_canvas.coords(balls['ball_%d' % i])[3], 2)))


def step(t):
    """Advance the ball one 'step' based on time_slider value of t"""
    global time

    #loop through all 4 balls
    for i in range(1,5):
        #DO THE PHYSICS!  Negate a and v because canvas y values increase as you move
        down.
        a = -1*float(accelerations['a_%d' % i].get())
        v = -1*float(velocities['v_%d' % i].get())
        d = v*t + .5*a*t**2

        #Get the x coords for the current ball.  These remain constant
        x1 = main_canvas.coords(balls['ball_%d' % i])[0]
```

```
65          x2 = main_canvas.coords(balls['ball_%d' % i])[2]
66
67          #Move the given ball and create a dash line to mark the new position
68          if main_canvas.coords(balls['ball_%d' % i])[3] + d <= 415:
69              main_canvas.move(balls['ball_%d' % i], 0, d)
70              y2 = main_canvas.coords(balls['ball_%d' % i])[3]
71              #Draw dash line at bottom of ball
72              main_canvas.create_line(x1, y2, x2, y2, tag="DASH")
73          #The ball has hit the ground
74          else:
75              main_canvas.coords(balls['ball_%d' % i], x1, 405, x2, 415)
76
77          #Do MORE PHYSICS
78          vf = v + a*t
79          #update velocity values for each ball
80          velocities['v_%d' % i].delete(0, END)
81          velocities['v_%d' % i].insert(0, str(round(-1*vf, 2)))
82
83          #Add data for the step to the data dict
84          data['data_%d' % i].append((time, 415 - main_canvas.coords(balls['ball_%d' %
            i])[3]))
85
86      #Update heights for the given time interval
87      update_height()
88
89      #Update time
90      time += t
91
92
93  def run():
94      """RUn the entire sim until all balls are at the ground or above the screen."""
95      #Balls may start on the ground or at the top of the screen so call step() at least
        once
96      step(t_slider.get())
97
98      #Run step() until ALL balls have hit the ground or left the screen based of the y2
        coord [3]
99      while 15 < main_canvas.coords(balls['ball_1'])[3] < 415 or 15 <
        main_canvas.coords(balls['ball_2'])[3] < 415 or 15 <
        main_canvas.coords(balls['ball_3'])[3] < 415 or 15 <
        main_canvas.coords(balls['ball_4'])[3] < 415:
100         step(t_slider.get())
101
102
103 def graph():
104     """Graph distance v time for 4 balls."""
105     #Colors of the balls corresponds to colors of the graph
106     colors = ['red', 'green', 'blue', 'yellow']
107
108     for i in range(1,5):
109         #Initialize x,y values
110         x = []
111         y = []
112         #Add corresponding data to x,y values
113         for data_list in data['data_%d' % i]:
114             x.append(data_list[0])
115             y.append(data_list[1])
116         #Plot data in corresponding color
117         pyplot.plot(x, y, color=colors[i-1])
118
119     #Graph formatting
120     pyplot.title('Distance Vs. Time')
121     pyplot.xlabel('Time')
122     pyplot.ylabel('Distance')
123     pyplot.show()
124
125
```

```
126 def reset():
127     """Erase all "DASH" tags from canvas, set balls back to ground, and resent entry
        fields."""
128     global time
129 
130     time = 0
131     main_canvas.delete("DASH")
132 
133     #Clear each ball...
134     for i in range(1,5):
135         #Clear and set the velocity and accelerations
136         velocities['v_%d' % i].delete(0, END)
137         velocities['v_%d' % i].insert(0, '0')
138         accelerations['a_%d' % i].delete(0, END)
139         accelerations['a_%d' % i].insert(0, '0')
140 
141         #Reset ball to starting position
142         main_canvas.coords(balls['ball_%d' % i], 45+(i-1)*100, 405, 55+(i-1)*100, 415)
143 
144         #Clear data
145         data['data_%d' % i].clear()
146 
147     update_height()
148     t_slider.set(1)
149 
150 
151 #Define layout
152 #Create frames
153 canvas_frame = tkinter.Frame(root)
154 input_frame = tkinter.Frame(root)
155 canvas_frame.pack(pady=10)
156 input_frame.pack(fill=BOTH, expand=True)
157 
158 #Canvas frame layout
159 main_canvas = tkinter.Canvas(canvas_frame, width=400, height=415, bg='white')
160 main_canvas.grid(row=0, column=0, padx=5, pady=5)
161 
162 line_0 = main_canvas.create_line(2,0,2,415)
163 line_1 = main_canvas.create_line(100,0,100,415)
164 line_2 = main_canvas.create_line(200,0,200,415)
165 line_3 = main_canvas.create_line(300,0,300,415)
166 line_4 = main_canvas.create_line(400,0,400,415)
167 
168 balls = {}
169 balls['ball_1'] = main_canvas.create_oval(45, 405, 55, 415, fill='red', tag="BALL")
170 balls['ball_2'] = main_canvas.create_oval(145, 405, 155, 415, fill='green', tag="BALL")
171 balls['ball_3'] = main_canvas.create_oval(245, 405, 255, 415, fill='blue', tag="BALL")
172 balls['ball_4'] = main_canvas.create_oval(345, 405, 355, 415, fill='yellow', tag="BALL")
173 
174 #Input frame layout
175 #Row labels
176 tkinter.Label(input_frame, text='d').grid(row=0, column=0)
177 tkinter.Label(input_frame, text='vi').grid(row=1, column=0)
178 tkinter.Label(input_frame, text='a').grid(row=2, column=0, ipadx=22)
179 tkinter.Label(input_frame, text='t').grid(row=3, column=0)
180 
181 #Heights/Distance labels
182 heights = {}
183 for i in range(1,5):
184     heights['height_%d' % i] = tkinter.Label(input_frame, text="Height: " + str(415 -
        main_canvas.coords(balls['ball_%d' % i])[3]))
185     heights['height_%d' % i].grid(row=0, column=i)
186 
187 #Velocity entry boxes
188 velocities = {}
189 for i in range(1,5):
190     velocities['v_%d' % i] = tkinter.Entry(input_frame, width=15)
```

```
191         velocities['v_%d' % i].grid(row=1, column=i, padx=1)
192         velocities['v_%d' % i].insert(0, '0')
193 
194     #Acceleration entry boxes
195     accelerations = {}
196     for i in range(1,5):
197         accelerations['a_%d' % i] = tkinter.Entry(input_frame, width=15)
198         accelerations['a_%d' % i].grid(row=2, column=i, padx=1)
199         accelerations['a_%d' % i].insert(0, '0')
200 
201     #Time slider
202     t_slider = tkinter.Scale(input_frame, from_=0, to=1, tickinterval=.1, resolution=.01,
    orient=HORIZONTAL)
203     t_slider.grid(row=3, column=1, columnspan=4, sticky='WE')
204     t_slider.set(1)
205 
206     #Buttons
207     step_button = tkinter.Button(input_frame, text="Step",
    command=lambda:step(t_slider.get()))
208     run_button = tkinter.Button(input_frame, text="Run", command=run)
209     graph_button = tkinter.Button(input_frame, text="Graph", command=graph)
210     reset_button = tkinter.Button(input_frame, text="Reset", command=reset)
211     quit_button = tkinter.Button(input_frame, text="Quit", command=root.destroy)
212 
213     step_button.grid(row=4, column=1, pady=(10,0), sticky="WE")
214     run_button.grid(row=4, column=2, pady=(10,0), sticky="WE")
215     graph_button.grid(row=4, column=3, pady=(10,0), sticky="WE")
216     reset_button.grid(row=4, column=4, pady=(10,0), sticky="WE")
217     quit_button.grid(row=5, column=1, columnspan=4, sticky="WE")
218 
219     #Make each ball 'dragable' in the vertical direction
220     root.bind('<B1-Motion>', move)
221 
222     #Run root window's main loop
223     root.mainloop()
```