MECE 4510: Evolutionary Computation and Design

Project – Phase C

Jerry Zhang | UNI: jz2966 | Grace Hours Remaining: 66 – 26 = 40

Zhengyang Du | UNI: zd2219 | Grace Hours Remaining: 61 – 26 = 35

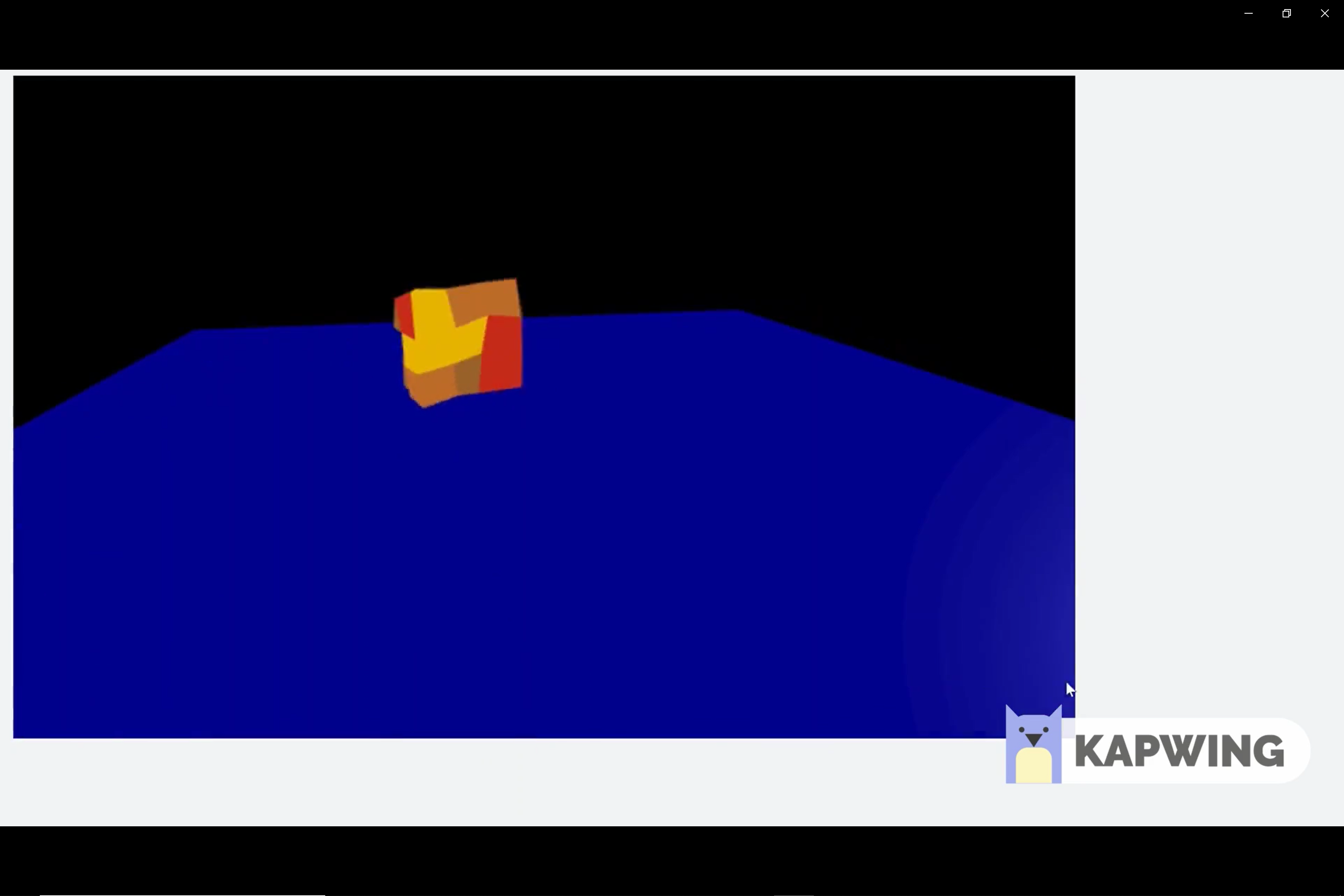
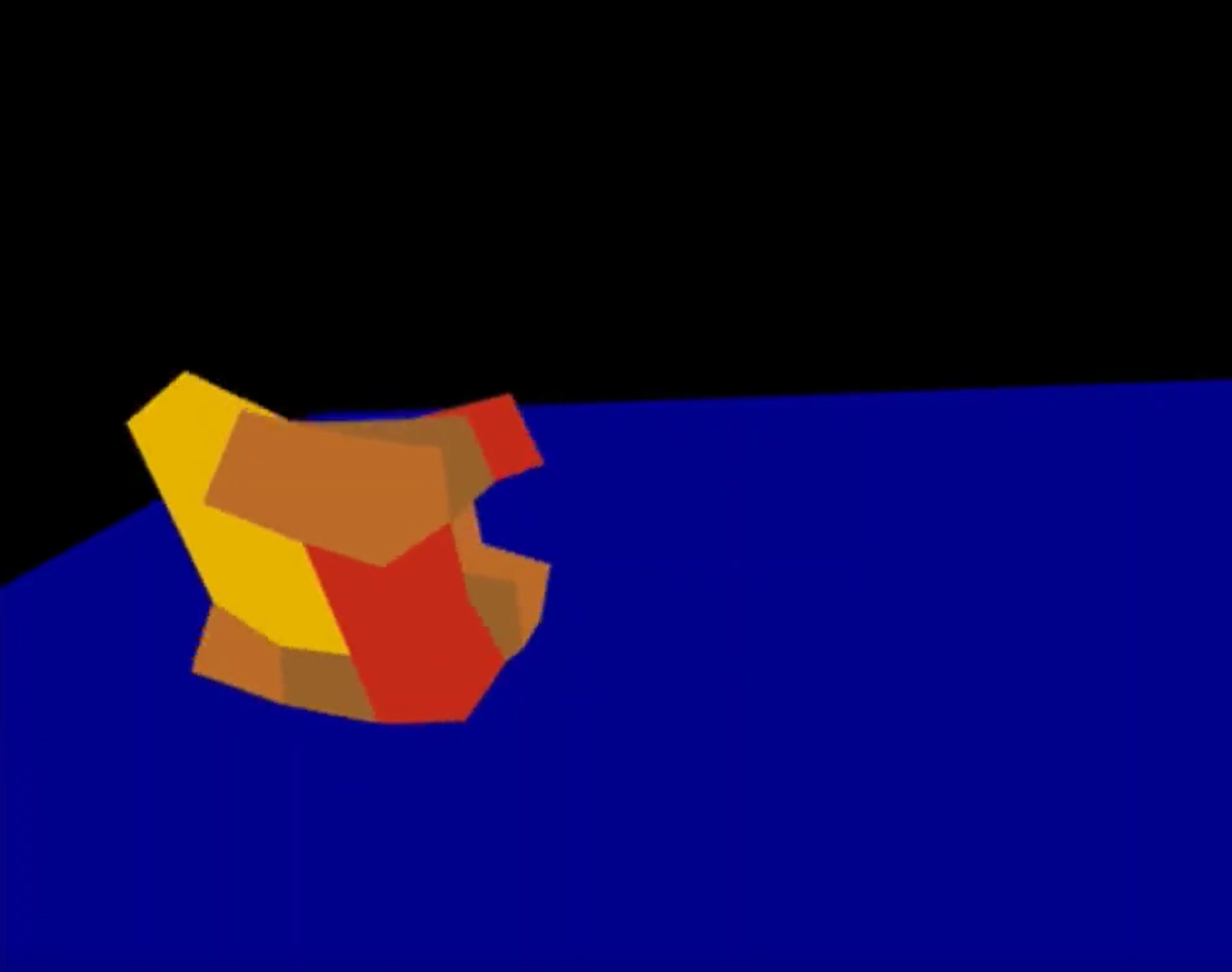
Instructor: Professor Hod Lipson

Date Submitted: 12/20/2018

**RESULTS**

Fastest Robot: <http://www.youtube.com/watch?v=WJGpLCXpisg&t=0m3s>

Speed: 5.5 m/s

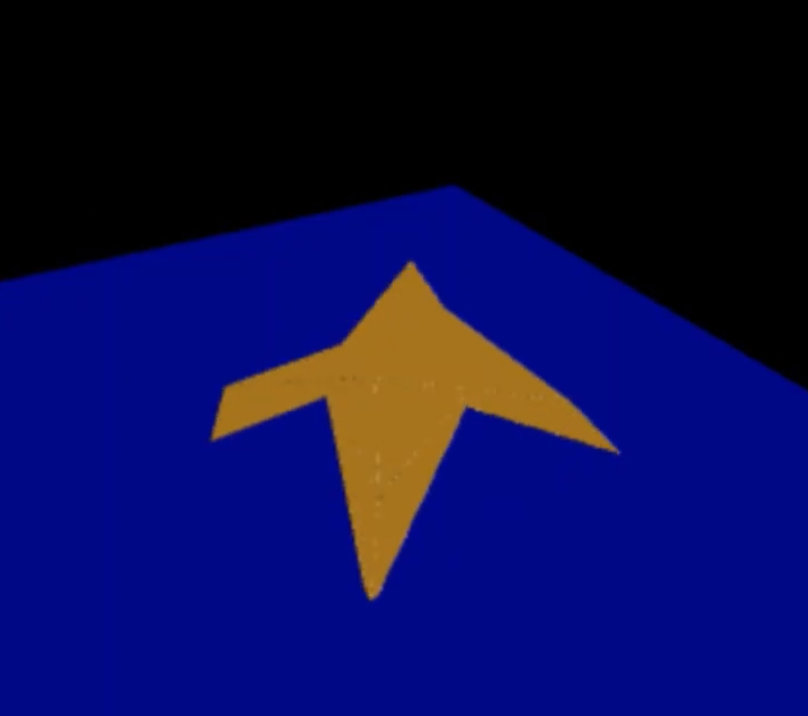
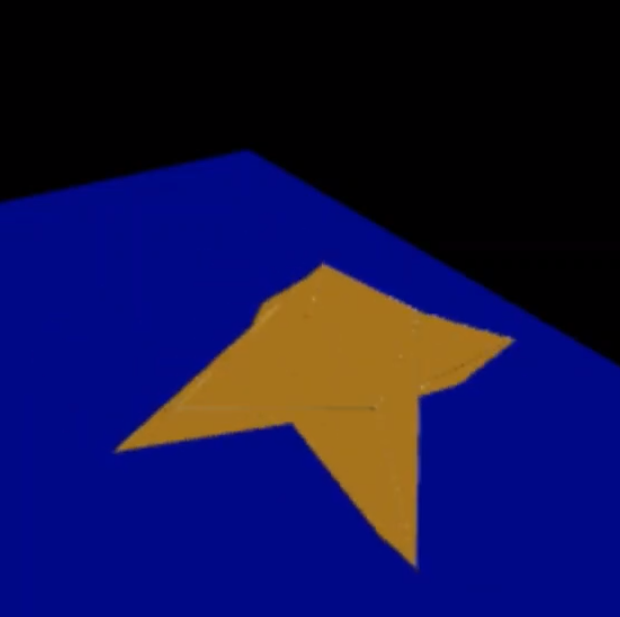
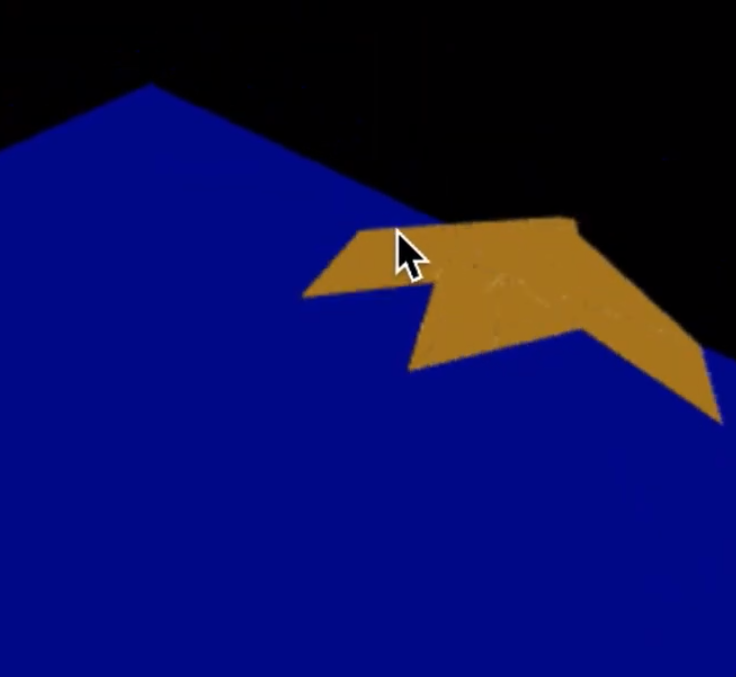
Fastest Bouncing Robot: <http://www.youtube.com/watch?v=WJGpLCXpisg&t=0m54s>

Speed: 1.7 m/s

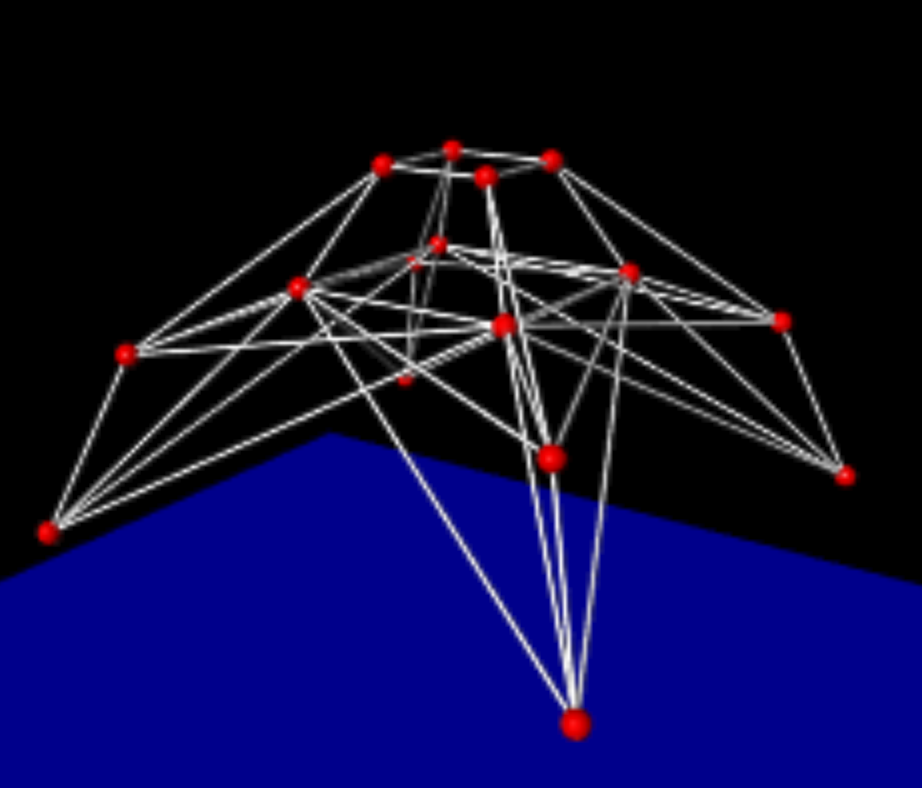
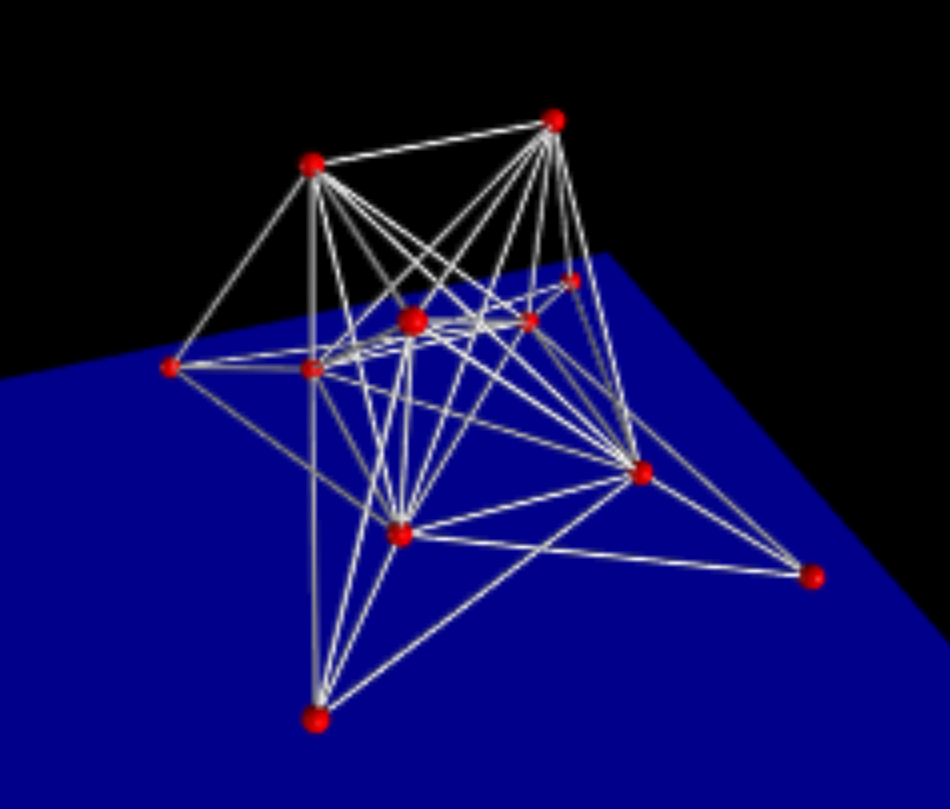
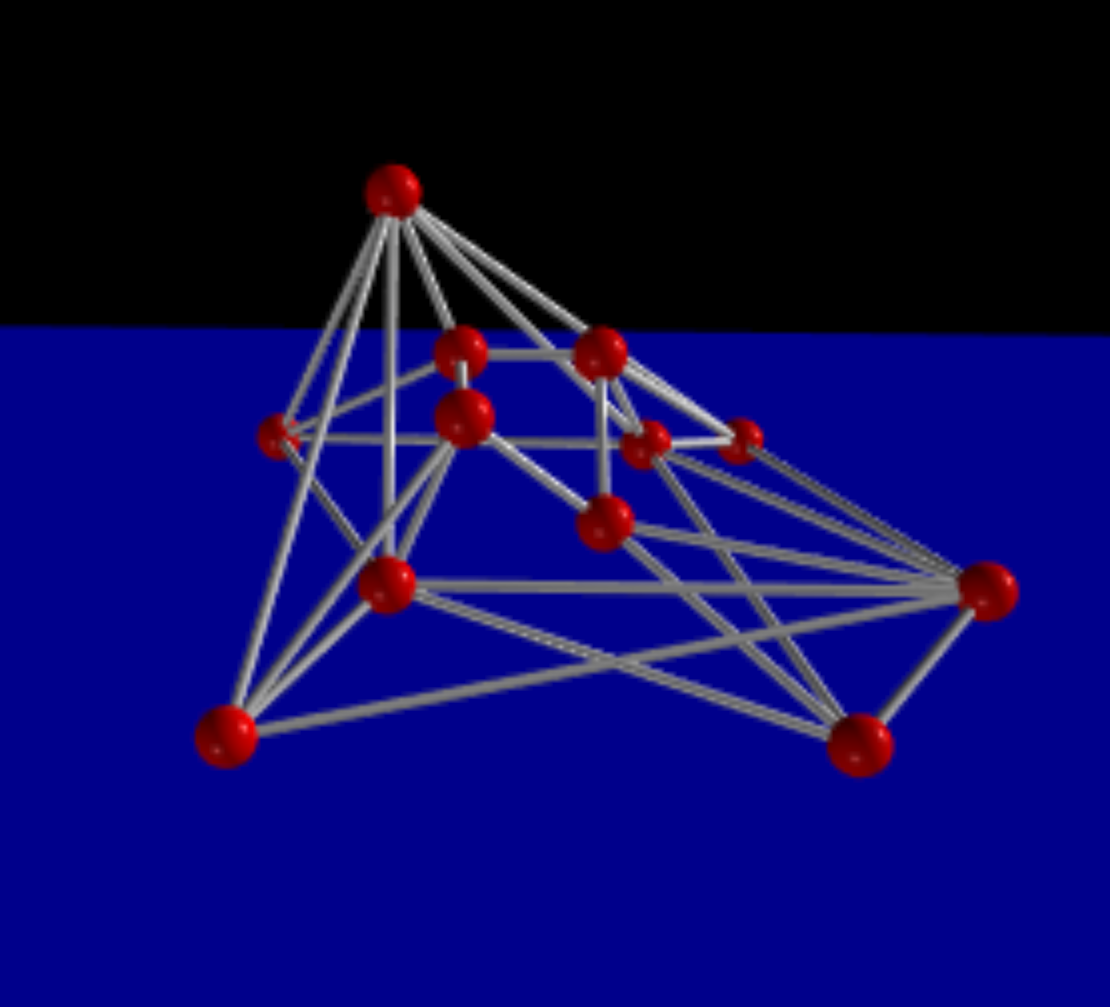
  

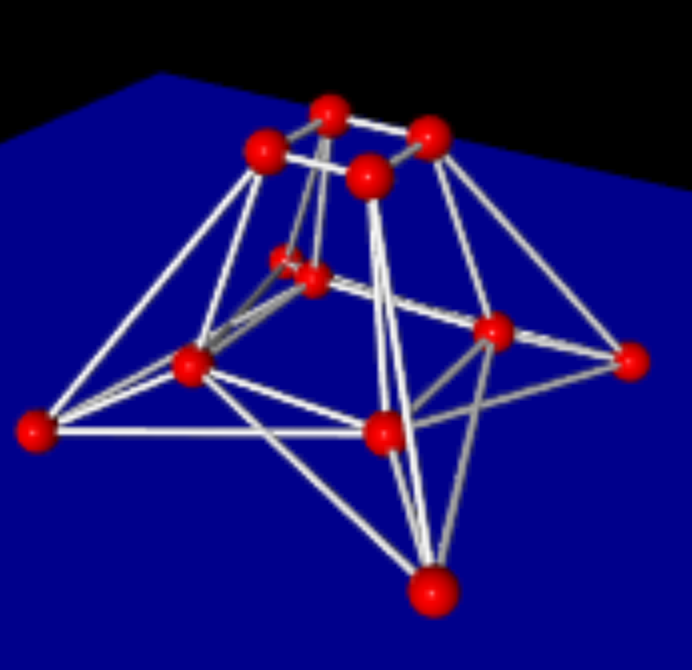
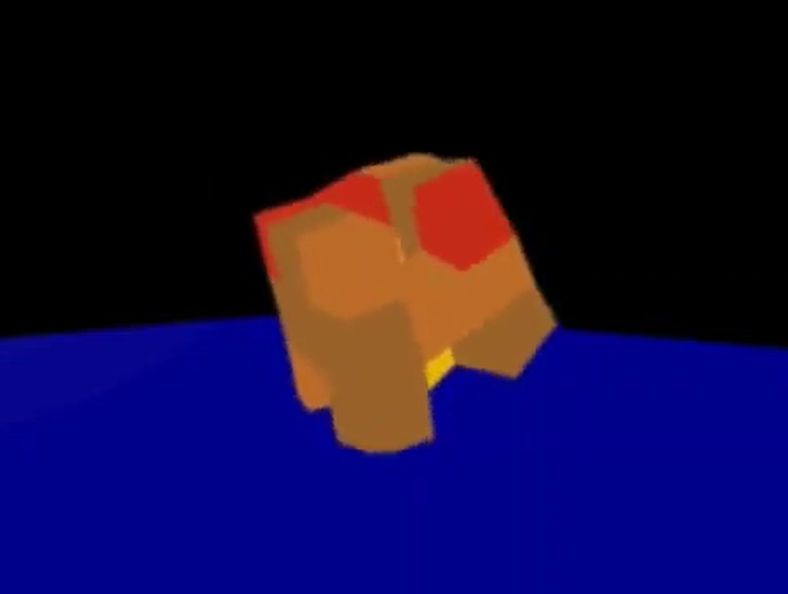
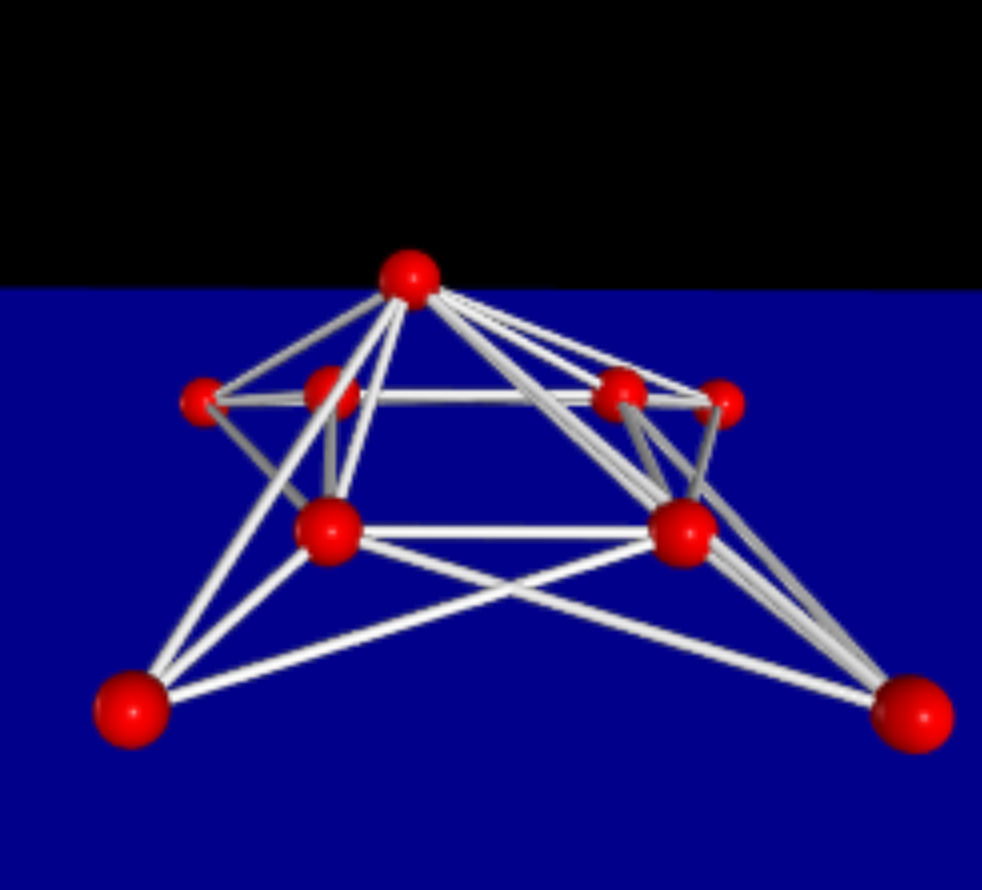
Innovative Robot: <http://www.youtube.com/watch?v=WJGpLCXpisg&t=1m44s>

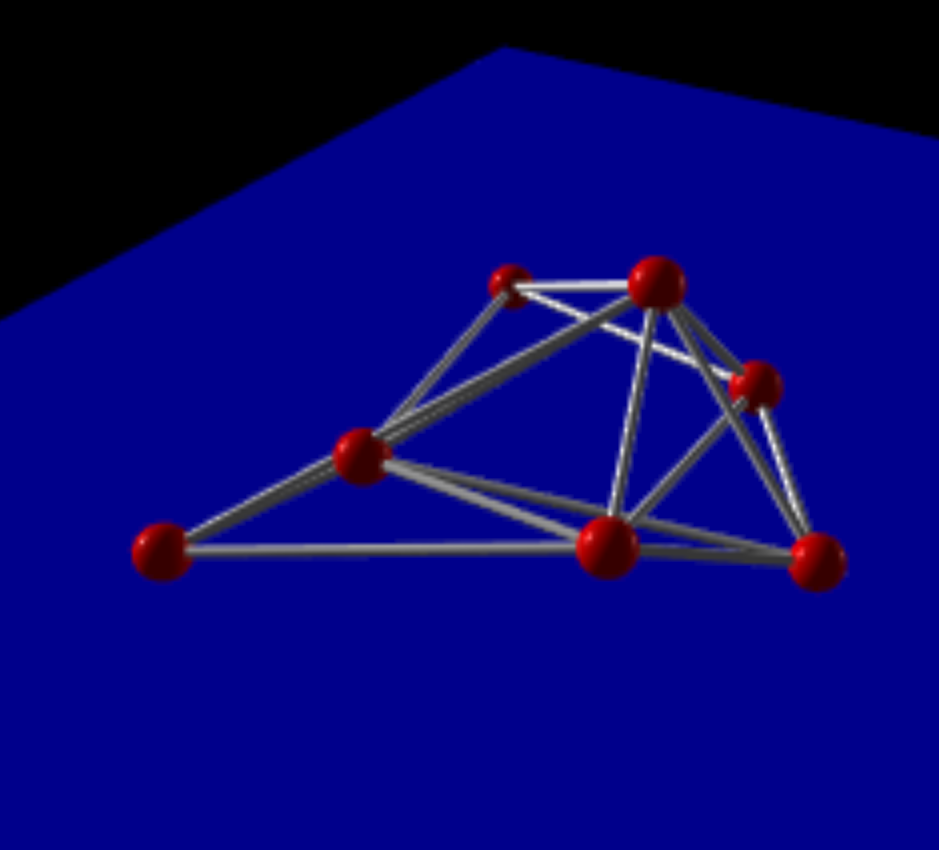
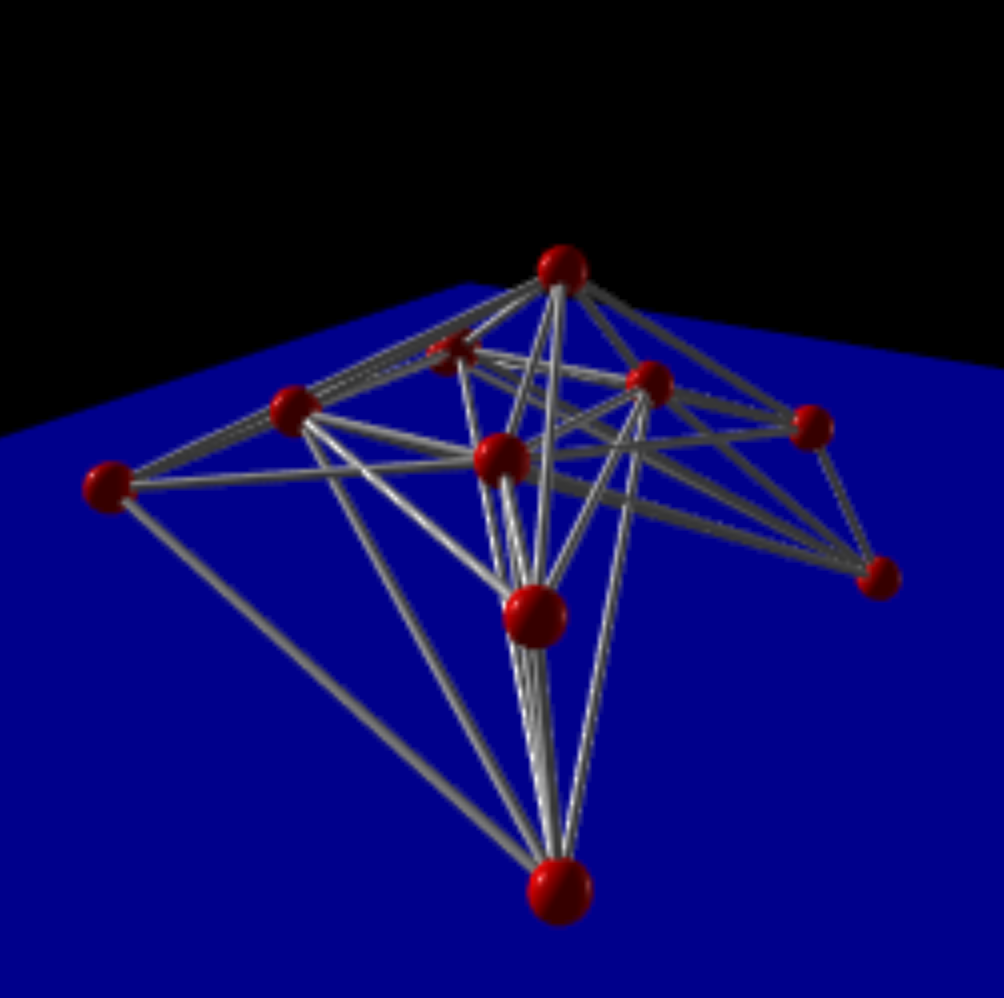
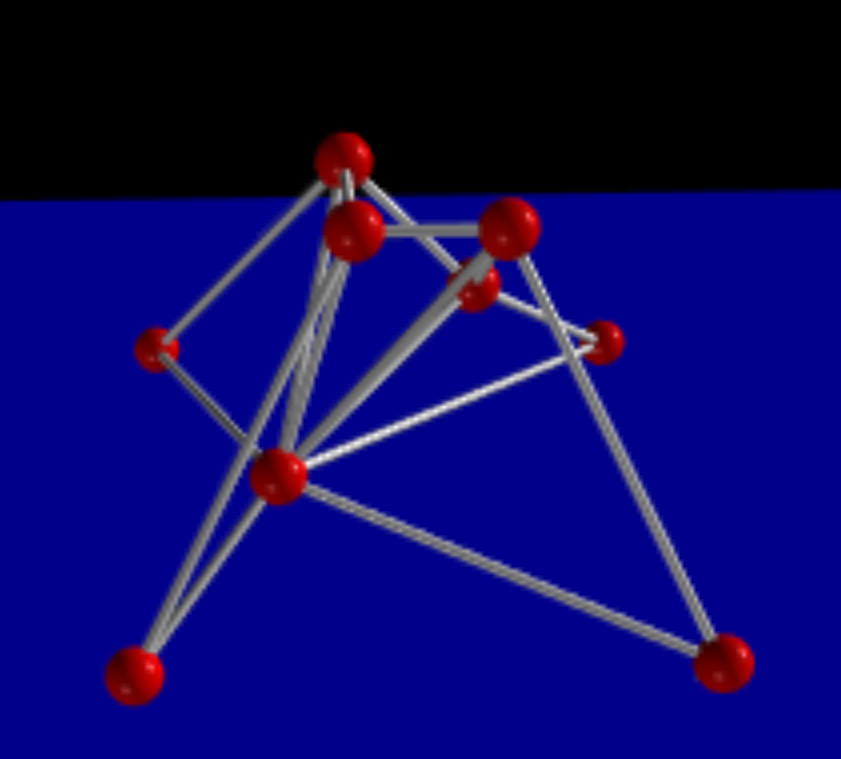
Speed: 2.2 m/s

**ROBOT ZOO**

**METHODS/PARAMETERS**

**Simulation Parameters:**

In order to simulate the aforementioned robots, the vpython package was utilized. Although aesthetically pleasing, this package is inefficient, thus it took exponentially more time to evolve our robots.

|  |  |  |  |
| --- | --- | --- | --- |
| Total Evolution Time | dT | Gravity | Spring Constant Floor |
| 2 sec | .001 sec |  | 10000 N/m |

**Robot Parameters:**

We evolved our robots with two unique approaches: one innovative approach based in inductive bias via the addition of node layers, and a second approach based on direct encoding of cube structures.

***Cube based Approach***: For this phase, we have initially created a 3 by 3 cubic robot. The structure was formed by 27 unit cubes that, which represent different materials. There were cubes that were specified to "exhale” via a sine function, "inhale" via a cosine function. Additionally, we specified empty cube areas to represent the air, and solid cubes with stiff spring constant values. The features were stored and represented by with different values in a combined array. Therefore, the robot could be evolved using these representations

In total it took over 72 hours to evolve our best cube-based robot.

|  |  |  |  |
| --- | --- | --- | --- |
| # Masses | # Springs | Total Mass | Spring Constant |
| 72 | 378 | 3.6 kg | 10000 |

**Performance: 5.5 m/s | 1.06 diameters/cycle**

***(Innovative) 4-node layer-based approach*:** This methodology was based upon our innovative robot assignment in the last assignment, which failed work properly due to array storage and manipulation issues. We managed to successfully debug our previous issues by removing trivial joints within our joint subspace and produce working robots which multi-layer node networks. Our innovative robot from prior, a dual-jointed, four-legged spider-bot, performed much more efficiently once our code was rewritten. Below are the parameters for the best evaluated robot using this methodology. In total it took over 24 hours to evolve our best spider robot. Below are the parameters:

|  |  |  |  |
| --- | --- | --- | --- |
| # Masses | # Springs | Total Mass | Spring Constant |
| 16 | 64 | 0.8 kg | 1000 N/m |

**Performance: 2.2 m/s | 0.94 diameters/cycle**

**Evolution Parameters**

As discussed before, different numbers were used as the representations for different features of the unit cubes. Once the combination of selected numbers were formed, the 3 by 3 cube then would be constructed and begin simulate. Therefore, the evolution was done by first random selecting the combination of the features and generating 10 parents. For each of the parent, its elected features would be sent to simulation for 2 seconds and calculate the distance it moved. By repeating this, we used truncation selection for our EA.

For the 4 node layer based approach, we created a function to randomly place the nodes, and decide whether there were 1 or 2 extra node layers. Then evolved accordingly. This methodology did not yield any effective robots, but instead allowed us to create a zoo.

Each length between each mass, were evolved to actuate based on an initially random and then evolved sinusoidal value. In order to evolve the cubic structure and innovative designs, a locomotion pattern was evolved between each mass. Rest length of each spring is defined as:

Where is the original rest length, and is a random value between and is a random value between [.

Based on python’s limitations, the spring evaluations per second is as follows:

**Number of Spring Evaluations per second**: 16632 springs/second.

**DISCUSSION**

**Cube based approach:**

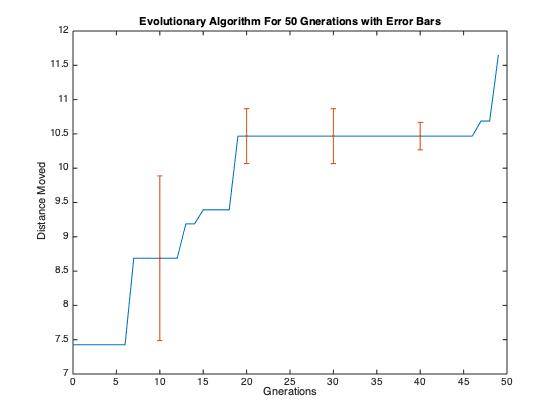
The evolving method worked well for our evolution and we were able to find the best structure for 50 generations that moved the furthest compared to the other generations, and thus we were able to create our fastest robot. For this phase, we could not figure out how to implement an efficient indirect encoding method which would give us better results. The efficiency of our code was not ideal as well that for 50 generations it already almost 70 hours. If more generations could be evaluated for the robot, there would be better results and moving features

**4-node layer-based approach:**

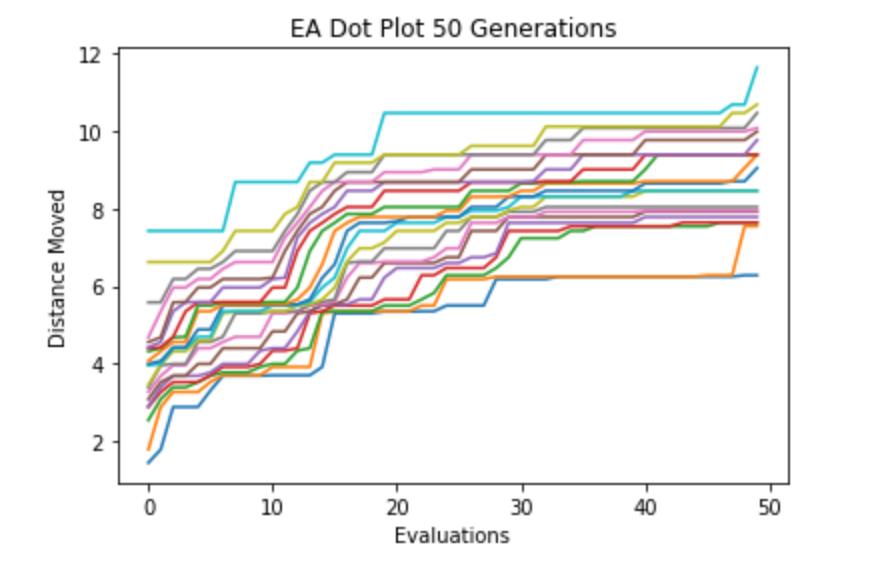
What worked well was that after fixing the joint space representation, and evolving the robot, we were able to finally get the spider robot to move effectively with its legs. One issue that we were not able to fix was the fixing the joints at the head of the spider. Just like the legs, these joints were also evolving and oscillating in congruence with the joints of the spider. One thing we also should have done differently was set up a function to optimize the spring constant associated with the longer joints. By doing this, we could give k a different magnitude based on length on the joint, and evolve accordingly.

**PLOTS**

Learning Curve with error bars:



Population Dot Plot



**APPENDIX**

Innovative Spider

##INNOVATIVE SPIDER  
  
import vpython as vp  
import itertools  
import numpy as np  
from math import \*  
import random  
import matplotlib.pyplot as plt  
  
scene = vp.canvas() # title = 'Box', width = 600, height = 400, center = vp.vector(0,0,0))  
  
floor = vp.box(pos=vp.vector(0, -1, 0), length=10, height=0.01, width=10, color=vp.color.blue)  
  
  
def getCOM(v):  
 COM = (v[0].pos + v[1].pos + v[2].pos + v[3].pos + v[4].pos + v[5].pos + v[6].pos + v[7].pos + v[8].pos + v[9].pos +  
 v[10].pos + v[11].pos + v[12].pos + v[13].pos + v[14].pos + v[15].pos) / 16  
 return COM  
  
  
# def mutation(r):  
# ran = random.randint(0, 9)  
# if ran > mutation\_rate:  
# x = random.randint(0, 25)  
# y = random.randint(0, 25)  
# r[x], r[y] = r[y], r[x]  
# return r  
# else:  
# return r  
  
  
# def Crossover(m, n):  
# M = []  
# N = []  
# for i in range(len(m)):  
# M.append(m[i])  
# N.append(n[i])  
# rand = random.randint(0, 20)  
# index = list(range(rand, rand + 6))  
# exchangem = []  
# exchangen = []  
# for i in range(len(index)):  
# exchangem.append(M[index[i]])  
# exchangen.append(N[index[i]])  
# for j in range(len(index)):  
# M[index[j]] = exchangen[j]  
# N[index[j]] = exchangem[j]  
# return (M, N)  
  
  
# dis1 = 0  
# good\_dis = []  
# best\_dis = []  
# mutation\_rate = 7  
  
v = 0  
dt = 0.001  
mass = 0.1  
g = 9.81  
k\_sp = 1000  
g\_vector = vp.vector(0, 9.81, 0)  
  
pa1 = [[-0.22910063559006733, -1.112282029832087], [-0.12552320965915562, 0.8277683557176245], [0.2806916493564143, 2.386047307095305], [-0.21257918117869412, 2.324089088655218], [0.1632157985219953, -0.6766117463033265], [0.28499906428617533, 0.7243908274508661], [-0.07006138468916454, 0.9963608542623428], [0.06734803978455728, 2.048931637992525], [-0.1470299289529832, -1.6685597747311225], [0.18271350388380087, 2.1066348014886316], [0.2876204706825857, 1.0295887674707584], [-0.05055673047115211, 1.5232367109334177], [-0.2870701020596687, 0.3254579724113027], [-0.159660431024846, 1.4627628439412437], [-0.11235498494325119, 2.0832697797374085], [-0.07132087662377565, -2.4763000824775707], [-0.28444122261989124, 2.2463882519550236], [-0.28121588033325046, 1.9107349952515849], [-0.05497958924059365, 1.68687304434511], [-0.2668878337675762, -1.4986777959127293], [-0.2879053506463822, -0.48816713513165055], [0.11772840836317394, -0.9919137449324746], [-0.2729726619184368, -1.6939715889215674], [0.1652097686083624, 0.3832285206864108], [-0.10020277135881264, -1.7461851106482147], [0.007474331196806128, 2.111587319268918]]  
  
# pa1 = [[-0.3067418516983423, -3.095653678317867], [0.07763804077278458, -1.3048665145280034], [0.5828379703308758, 1.5180361969257001], [0.47367547915651065, -2.7220395517138916], [-0.22552321293470995, 1.7826154112424444], [0.20726549326398525, 1.6544780357502988], [-0.17211864383632286, -0.7955848342845049], [0.32175582636123035, 2.827997446698695], [-0.5310339146852617, 1.1155611541531965], [-0.019840991346967374, -1.2085026594523332], [-0.10037718265406387, 1.5852872591898048], [-0.48199240941990557, 2.7515348918389098], [-0.18271629003468554, 2.278384473374639], [-0.38380837202157914, -0.4404540605379226], [0.4072091963284049, 0.7509254892570847], [-0.4291960267428591, 0.8381508741633086], [0.3653607290192744, -0.5047537648114644], [0.08147647506137867, -0.33223682494856055], [0.01865274695185737, -1.7923455820858414], [-0.1539665658152985, -1.749167772144356], [0.023110271332242194, -2.17511106639519], [-0.4011482393351309, 1.0474830483297755], [-0.4318826969885843, -1.2972886196013718], [-0.18127175023685194, -0.9499633222170929], [-0.5579365937566048, -0.15998408134772601], [0.15218002048154933, -0.1929867684553952]]  
  
# for i in range(len(pa1)):  
# pa1[i][0] - 0.4  
  
# for i in range(10):  
# pa = []  
# for j in range(26):  
# p = []  
# b = np.random.uniform(-0.4, 0.4)  
# c = np.random.uniform(-np.pi, np.pi)  
# p.append(b)  
# p.append(c)  
# pa.append(p)  
  
# pa1.append(pa)  
  
# dots = []  
  
# for a in range(2):  
# print(a, "gen")  
# for i in np.arange(0, 10, 2):  
# child1, child2 = Crossover(pa1[i], pa1[i + 1])  
  
# pa1.append(child1)  
# pa1.append(child2)  
  
# for i in range(10, 20):  
# pa1[i] = mutation(pa1[i])  
  
# total\_dis = []  
# take = []  
# for h in range(20):  
# print(h)  
ballname = ['b1', 'b2', 'b3', 'b4', 'b5', 'b6', 'b7', 'b8']  
ballvectors = [vp.vector(0, 0, 0), vp.vector(0, 0, 1), vp.vector(1, 0, 0), vp.vector(1, 0, 1),  
 vp.vector(0.25, 0.5, 0.25), vp.vector(0.25, 0.5, 0.75), vp.vector(0.75, 0.5, 0.25),  
 vp.vector(0.75, 0.5, 0.75)]  
  
for i in range(len(ballname)):  
 ballname[i] = vp.sphere(pos=ballvectors[i], radius=0.01, color=vp.color.red, f\_k=vp.vector(0, 0, 0))  
  
springvecs = [[ballvectors[0], ballvectors[4]], [ballvectors[0], ballvectors[1]],  
 [ballvectors[0], ballvectors[2]],  
 [ballvectors[1], ballvectors[5]], [ballvectors[1], ballvectors[3]],  
 [ballvectors[2], ballvectors[6]],  
 [ballvectors[3], ballvectors[7]], [ballvectors[3], ballvectors[2]],  
 [ballvectors[4], ballvectors[5]],  
 [ballvectors[4], ballvectors[6]], [ballvectors[7], ballvectors[5]],  
 [ballvectors[7], ballvectors[6]]]  
  
spring = ['s1', 's2', 's3', 's4', 's5', 's6', 's7', 's8', 's9', 's10', 's11',  
 's12'] # , 's13', 's14', 's15', 's16', 's17', 's18', 's19', 's20','s21', 's22', 's23', 's24', 's25', 's26', 's27', 's28']  
  
for i in range(len(springvecs)):  
 position = springvecs[i][1] - springvecs[i][0]  
 spring[i] = vp.cylinder(pos=springvecs[i][0], axis=position, length=vp.mag(position), radius=.002,  
 color=vp.color.white)  
  
  
  
triangles = []  
for z in itertools.combinations(ballvectors, 3):  
 triangles.append(z)  
  
T = list(range(56))  
for i in range(len(triangles)):  
 T[i] = vp.triangle(v0= vp.vertex(pos=triangles[i][0]), v1=vp.vertex(pos=triangles[i][1] ), v2=vp.vertex(pos=triangles[i][2]), texture = "https://i.imgur.com/eQueRtf.jpg")# "https://i.imgur.com/MjzWrv2.png")  
  
  
  
  
ballname2 = ['b9', 'b10', 'b11', 'b12']  
ballvectors2 = [vp.vector(-0.5, -0.25, -0.5), vp.vector(-0.5, -0.25, 1.5), vp.vector(1.5, -0.25, 1.5),vp.vector(1.5, -0.25, -0.5)]  
for i in range(len(ballname2)):  
 ballname2[i] = vp.sphere(pos=ballvectors2[i], radius=0.01, color=vp.color.red, f\_k= vp.vector(0,0,0))  
springvecs2 = []  
  
T2 = []  
for i in range(4):  
 triangles2 = []  
 corner1 = ballvectors2[i]  
 if i == 0:  
 springvecs2.append([[corner1,ballvectors[i]],[corner1,ballvectors[i+1]],[corner1,ballvectors[i+2]],[corner1,ballvectors[i+4]]])  
 triangles2.extend((corner1,ballvectors[i],ballvectors[i+1],ballvectors[i+2],ballvectors[i+4]))  
 elif i == 1:  
 springvecs2.append([[corner1,ballvectors[i]],[corner1,ballvectors[i-1]],[corner1,ballvectors[i+2]],[corner1,ballvectors[i+4]]])  
 triangles2.extend((corner1,ballvectors[i],ballvectors[i-1],ballvectors[i+2],ballvectors[i+4]))  
  
 elif i == 2:  
 springvecs2.append([[corner1,ballvectors[i]],[corner1,ballvectors[i-1]],[corner1,ballvectors[i+1]],[corner1,ballvectors[i+5]]])  
 triangles2.extend((corner1,ballvectors[i],ballvectors[i-1],ballvectors[i+1],ballvectors[i+5]))  
  
 elif i == 3:  
 springvecs2.append([[corner1,ballvectors[i]],[corner1,ballvectors[i-1]],[corner1,ballvectors[i-3]],[corner1,ballvectors[i+3]]])  
 triangles2.extend((corner1,ballvectors[i],ballvectors[i-1],ballvectors[i-3],ballvectors[i+3]))  
  
 triangles2list = []  
 for z in itertools.combinations(triangles2, 3):  
 triangles2list.append(z)  
  
 T2.append(list(range(len(triangles2list))))  
 for j in range(len(triangles2list)):  
 T2[i][j] = vp.triangle(v0= vp.vertex(pos=triangles2list[j][0]), v1=vp.vertex(pos=triangles2list[j][1] ), v2=vp.vertex(pos=triangles2list[j][2]), texture = "https://i.imgur.com/eQueRtf.jpg")  
  
  
  
  
  
  
  
  
spring2 = ['s\_1', 's\_2', 's\_3', 's\_4', 's\_5', 's\_6', 's\_7', 's\_8', 's\_9', 's\_10', 's\_11', 's\_12', 's\_13',  
 's\_14', 's\_15', 's\_16']  
  
for i in range(4):  
 for j in range(4):  
 position2 = springvecs2[i][j][1] - springvecs2[i][j][0]  
 spring2[j + i \* 4] = vp.cylinder(pos=springvecs2[i][j][0], axis=position2, length=vp.mag(position2),  
 radius=.002, color=vp.color.white)  
  
ballname3 = ['b13', 'b14', 'b15', 'b16']  
ballvectors3 = [vp.vector(-0.75, -1, -0.75), vp.vector(-0.75, -1, 1.75), vp.vector(1.75, -1, 1.75),  
 vp.vector(1.75, -1, -0.75)]  
for i in range(len(ballname3)):  
 ballname3[i] = vp.sphere(pos=ballvectors3[i], radius=0.01, color=vp.color.red, f\_k=vp.vector(0, 0, 0))  
  
springvecs3 = []  
for i in range(4):  
 corner1 = ballvectors3[i]  
 if i == 0:  
 springvecs3.append(  
 [[corner1, ballvectors2[i]], [corner1, ballvectors[i + 1]], [corner1, ballvectors[i]],  
 [corner1, ballvectors[i + 2]]])  
 elif i == 1:  
 springvecs3.append(  
 [[corner1, ballvectors2[i]], [corner1, ballvectors[i]], [corner1, ballvectors[i + 2]],  
 [corner1, ballvectors[i - 1]]])  
 elif i == 2:  
 springvecs3.append(  
 [[corner1, ballvectors2[i]], [corner1, ballvectors[i - 1]], [corner1, ballvectors[i + 1]],  
 [corner1, ballvectors[i]]])  
 elif i == 3:  
 springvecs3.append(  
 [[corner1, ballvectors2[i]], [corner1, ballvectors[i - 1]], [corner1, ballvectors[i - 3]],  
 [corner1, ballvectors[i]]])  
  
spring3 = ['s\_1', 's\_2', 's\_3', 's\_4', 's\_5', 's\_6', 's\_7', 's\_8', 's\_9', 's\_10', 's\_11', 's\_12', 's\_13',  
 's\_14', 's\_15', 's\_16']  
for i in range(4):  
 for j in range(4):  
 position3 = springvecs3[i][j][1] - springvecs3[i][j][0]  
 spring3[j + i \* 4] = vp.cylinder(pos=springvecs3[i][j][0], axis=position3, length=vp.mag(position3),  
 radius=.002, color=vp.color.white)  
  
ballnameC = ballname + ballname2 + ballname3  
  
OriginalCOM = (ballnameC[0].pos + ballnameC[1].pos + ballnameC[2].pos + ballnameC[3].pos + ballnameC[4].pos +  
 ballnameC[5].pos + ballnameC[6].pos + ballnameC[7].pos + ballnameC[8].pos + ballnameC[9].pos +  
 ballnameC[10].pos + ballnameC[11].pos + ballnameC[12].pos + ballnameC[13].pos + ballnameC[  
 14].pos + ballnameC[15].pos) / 16  
  
for i in range(len(ballnameC)):  
 ballnameC[i].velocity = vp.vector(0, 0, 0)  
  
  
  
springvecs3 = []  
T3 = []  
for i in range(4):  
 triangles3 = []  
 corner1 = ballvectors3[i]  
 if i == 0:  
 springvecs3.append([[corner1,ballvectors2[i]],[corner1,ballvectors[i+1]],[corner1,ballvectors[i]],[corner1,ballvectors[i+2]]])  
 triangles3.extend((corner1,ballvectors2[i],ballvectors[i+1],ballvectors[i],ballvectors[i+2]))  
  
 elif i == 1:  
 springvecs3.append([[corner1,ballvectors2[i]],[corner1,ballvectors[i]],[corner1,ballvectors[i+2]],[corner1,ballvectors[i-1]]])  
 triangles3.extend((corner1,ballvectors2[i],ballvectors[i],ballvectors[i+2],ballvectors[i-1]))  
  
 elif i == 2:  
 springvecs3.append([[corner1,ballvectors2[i]],[corner1,ballvectors[i-1]],[corner1,ballvectors[i+1]],[corner1,ballvectors[i]]])  
 triangles3.extend((corner1,ballvectors2[i],ballvectors[i-1],ballvectors[i+1],ballvectors[i]))  
  
 elif i == 3:  
 springvecs3.append([[corner1,ballvectors2[i]],[corner1,ballvectors[i-1]],[corner1,ballvectors[i-3]],[corner1,ballvectors[i]]])  
 triangles3.extend((corner1,ballvectors2[i],ballvectors[i-1],ballvectors[i-3],ballvectors[i]))  
  
 triangles3list = []  
 for z in itertools.combinations(triangles3, 3):  
 triangles3list.append(z)  
  
 T3.append(list(range(len(triangles3list))))  
 for j in range(len(triangles3list)):  
 T3[i][j] = vp.triangle(v0= vp.vertex(pos=triangles3list[j][0]), v1=vp.vertex(pos=triangles3list[j][1] ), v2=vp.vertex(pos=triangles3list[j][2]), texture = "https://i.imgur.com/eQueRtf.jpg")#"https://i.imgur.com/5FclvgA.jpg")  
  
  
  
  
# for i in range(len(ballname2)):  
# ballname2[i].velocity = vp.vector(0, 0, 0)  
  
F\_c = vp.vector(0, 1000, 0)  
  
L0 = np.zeros((16, 16))  
for i in range(16):  
 for j in range(16):  
 if i == j:  
 L0[j][i] = 0  
 else:  
 position = ballnameC[j].pos - ballnameC[i].pos  
 L0[j][i] = vp.mag(position)  
  
Repeated = []  
total\_indices = []  
for x in range(16):  
 for y in range(16):  
 value = L0[y][x]  
 if value not in Repeated:  
 Repeated.append(value)  
 indices = []  
 for n in range(16):  
 for m in range(16):  
 index = []  
 if L0[m][n] == value:  
 index.append(m)  
 index.append(n)  
 indices.append(index)  
 total\_indices.append(indices)  
# print(len(Repeated))  
  
del total\_indices[0]  
  
ballvectorsC = ballvectors + ballvectors2 + ballvectors3  
  
F = np.zeros((16, 3))  
ic, fnum = 0, 0  
c = 1  
t = 0.001  
  
  
  
while 1:  
 vp.rate(50)  
  
 L0rate = np.zeros((16, 16))  
 w = 10 \* np.pi  
  
 #print(total\_indices,"ASdfasdfasdf")  
 # asdfff = 0  
 for i in range(len(total\_indices)):  
 for z in total\_indices[i]:  
 L0rate[z[0]][z[1]] = L0[z[0]][z[1]] + pa1[i][0] \* sin(w \* t + pa1[i][1])  
 # asdfff += 1  
 # print(asdfff, len(total\_indices), len(total\_indices[10]),"ASdfasdfasdfsf")  
  
 for i in range(8):  
 ballvectors[i] = ballnameC[i].pos  
 # ballnameC[i].pos = ballname[i].pos  
#  
 springvecs = [[ballvectors[0], ballvectors[4]], [ballvectors[0], ballvectors[1]],  
 [ballvectors[0], ballvectors[2]],  
 [ballvectors[1], ballvectors[5]], [ballvectors[1], ballvectors[3]],  
 [ballvectors[2], ballvectors[6]],  
 [ballvectors[3], ballvectors[7]], [ballvectors[3], ballvectors[2]],  
 [ballvectors[4], ballvectors[5]],  
 [ballvectors[4], ballvectors[6]], [ballvectors[7], ballvectors[5]],  
 [ballvectors[7], ballvectors[6]]]  
  
 for i in range(12):  
 position = springvecs[i][1] - springvecs[i][0] # - L0[i]  
 spring[i].pos = springvecs[i][0]  
 spring[i].axis = position  
 spring[i].length = vp.mag(position)  
  
 triangles = []  
 for z in itertools.combinations(ballvectors, 3):  
 triangles.append(z)  
 for i in range(len(triangles)):  
 T[i].v0.pos = triangles[i][0]  
 T[i].v1.pos = triangles[i][1]  
 T[i].v2.pos = triangles[i][2]  
  
 for i in range(4):  
 ballvectors2[i] = ballnameC[i+8].pos  
 # ballnameC[i+4].pos = ballname2[i].pos  
  
  
 springvecs2 = []  
 for i in range(4):  
 triangles2 = []  
 corner1 = ballvectors2[i]  
 if i == 0:  
 springvecs2.append([[corner1, ballvectors[i]], [corner1, ballvectors[i + 1]], [corner1, ballvectors[i + 2]],  
 [corner1, ballvectors[i + 4]]])  
 triangles2.extend((corner1, ballvectors[i], ballvectors[i + 1], ballvectors[i + 2], ballvectors[i + 4]))  
 elif i == 1:  
 springvecs2.append([[corner1, ballvectors[i]], [corner1, ballvectors[i - 1]], [corner1, ballvectors[i + 2]],  
 [corner1, ballvectors[i + 4]]])  
 triangles2.extend((corner1, ballvectors[i], ballvectors[i - 1], ballvectors[i + 2], ballvectors[i + 4]))  
  
 elif i == 2:  
 springvecs2.append([[corner1, ballvectors[i]], [corner1, ballvectors[i - 1]], [corner1, ballvectors[i + 1]],  
 [corner1, ballvectors[i + 5]]])  
 triangles2.extend((corner1, ballvectors[i], ballvectors[i - 1], ballvectors[i + 1], ballvectors[i + 5]))  
  
 elif i == 3:  
 springvecs2.append([[corner1, ballvectors[i]], [corner1, ballvectors[i - 1]], [corner1, ballvectors[i - 3]],  
 [corner1, ballvectors[i + 3]]])  
 triangles2.extend((corner1, ballvectors[i], ballvectors[i - 1], ballvectors[i - 3], ballvectors[i + 3]))  
  
 triangles2list = []  
 for z in itertools.combinations(triangles2, 3):  
 triangles2list.append(z)  
  
 for k in range(len(triangles2list)):  
 T2[i][k].v0.pos = triangles2list[k][0]  
 T2[i][k].v1.pos = triangles2list[k][1]  
 T2[i][k].v2.pos = triangles2list[k][2]  
  
  
 for i in range(4):  
 for j in range(4):  
 position2 = springvecs2[i][j][1] - springvecs2[i][j][0]  
 spring2[j + i \* 4].pos = springvecs2[i][j][0]  
 spring2[j + i \* 4].axis = position2  
 spring2[j + i \* 4].length = vp.mag(position2)  
  
 for i in range(4):  
 ballvectors3[i] = ballnameC[i+12].pos  
# ballnameC[i+8].pos = ballname3[i].pos  
 springvecs3 = []  
 for i in range(4):  
 triangles3 = []  
 corner1 = ballvectors3[i]  
 if i == 0:  
 springvecs3.append([[corner1, ballvectors2[i]], [corner1, ballvectors[i + 1]], [corner1, ballvectors[i]],  
 [corner1, ballvectors[i + 2]]])  
 triangles3.extend((corner1, ballvectors2[i], ballvectors[i + 1], ballvectors[i], ballvectors[i + 2]))  
  
 elif i == 1:  
 springvecs3.append([[corner1, ballvectors2[i]], [corner1, ballvectors[i]], [corner1, ballvectors[i + 2]],  
 [corner1, ballvectors[i - 1]]])  
 triangles3.extend((corner1, ballvectors2[i], ballvectors[i], ballvectors[i + 2], ballvectors[i - 1]))  
  
 elif i == 2:  
 springvecs3.append(  
 [[corner1, ballvectors2[i]], [corner1, ballvectors[i - 1]], [corner1, ballvectors[i + 1]],  
 [corner1, ballvectors[i]]])  
 triangles3.extend((corner1, ballvectors2[i], ballvectors[i - 1], ballvectors[i + 1], ballvectors[i]))  
  
 elif i == 3:  
 springvecs3.append(  
 [[corner1, ballvectors2[i]], [corner1, ballvectors[i - 1]], [corner1, ballvectors[i - 3]],  
 [corner1, ballvectors[i]]])  
 triangles3.extend((corner1, ballvectors2[i], ballvectors[i - 1], ballvectors[i - 3], ballvectors[i]))  
  
 triangles3list = []  
 for z in itertools.combinations(triangles3, 3):  
 triangles3list.append(z)  
  
 for k in range(len(triangles3list)):  
 T3[i][k].v0.pos = triangles3list[k][0]  
 T3[i][k].v1.pos = triangles3list[k][1]  
 T3[i][k].v2.pos = triangles3list[k][2]  
  
 for i in range(4):  
 for j in range(4):  
 position3 = springvecs3[i][j][1] - springvecs3[i][j][0]  
 spring3[j + i \* 4].pos = springvecs3[i][j][0]  
 spring3[j + i \* 4].axis = position3  
 spring3[j + i \* 4].length = vp.mag(position3)  
  
 dampening = 0.9  
 #ballvectorsC = ballvectors + ballvectors2 + ballvectors3  
 # ballnameC = ballname + ballname2 + ballname3  
  
 F\_mat = np.zeros((16, 16))  
 F\_vec = []  
 F\_v = []  
 a = np.array(np.zeros((16, 16)))  
 for i in range(16):  
 for k in range(16):  
 if k == i:  
 L = 0  
 F\_mat[i][k] = 0  
 F\_vec.append(vp.vector(0, 0, 0))  
 else:  
 L = vp.mag(ballnameC[k].pos - ballnameC[i].pos) - L0rate[k][i]  
 F\_mat[i][k] = L \* k\_sp  
 pf0 = ballnameC[k].pos - ballnameC[i].pos  
 F\_vec.append(vp.norm(pf0) \* L \* k\_sp)  
  
 a = np.array(F\_vec).reshape(16, 16)  
 F = a.sum(axis=0)  
 # print(pa1, "asdfasdfasdfasdf")  
 for i in range(16):  
  
 F[i] = F[i] + g\_vector \* mass  
 if ballnameC[i].pos.y < floor.pos.y:  
 F\_N = ((floor.pos.y - ballnameC[i].pos.y) \*\* 2) \* 1000  
 F[i].y = 0.99 \* (F[i].y - F\_N)  
 mu = 1  
 F\_st = mu \* F\_N  
 F\_horiz = (F[i].x \*\* 2 + F[i].z \*\* 2) \*\* 0.5  
 v\_xz = (ballnameC[i].velocity.x \*\* 2 + ballnameC[i].velocity.z \*\* 2) \*\* 0.5  
 vx = ballnameC[i].velocity.x / v\_xz  
 vz = ballnameC[i].velocity.z / v\_xz  
 if F\_st < F\_horiz:  
 F[i].x += F\_horiz \* vx - F\_N \* vx  
 F[i].z += F\_horiz \* vz - F\_N \* vz  
 else:  
 F[i].x = F\_horiz \* vx  
 F[i].z = F\_horiz \* vz  
 ballnameC[i].velocity.x = 0  
 ballnameC[i].velocity.z = 0  
  
 for i in range(16):  
 ballnameC[i].velocity -= F[i] / mass \* dt  
 ballnameC[i].pos += ballnameC[i].velocity \* dt  
 t += 0.001  
 c += 1  
 if c == 2000:  
 break  
# COM = getCOM(ballnameC)  
# dvec = COM - OriginalCOM  
# dis = sqrt(dvec.x \* 2 + dvec.z \* 2)  
# total\_dis.append(dis)  
# dis\_index = np.argsort(total\_dis)  
# sorted\_dis = []  
# sorted\_pa1 = []  
# for i in range(20):  
# sorted\_dis.append(total\_dis[dis\_index[i]])  
# sorted\_pa1.append(pa1[dis\_index[i]])  
# good\_dis = sorted\_dis[-10:]  
# dots.append(good\_dis)  
# print(dots)  
# print('GOODDIS', good\_dis[-1])  
# # for i in good\_dis:  
# # if i <= 50:  
# # take.append(i)  
# best\_dis.append(good\_dis[-1])  
# pa1 = sorted\_pa1[-10:]  
# print('PA1END', len(pa1))  
  
# print(best\_dis)  
# print(pa1[-1])  
# evals = list(range(1, 3))  
  
# plt.plot(evals, best\_dis)  
# plt.xlabel('Evaluations')  
# plt.ylabel('Distance Moved')  
# plt.title('Evolutionary Algorithm')  
# plt.show()  
  
# plt.plot(evals, dots)  
# plt.xlabel('Evaluations')  
# plt.ylabel('Distance Moved')  
# plt.title('Evolutionary Algorithm Dot Plot')  
# plt.show()

3x3 Cube

import vpython as vp  
import itertools  
import random  
import numpy as np  
from math import \*  
import matplotlib.pyplot as plt  
  
# so first we create 10 parents randomly  
# then what we gonna do is to crossover the 10 parents each pair of parents would generate two children  
# then we pick the best 10 out of the 20 which would become the next generation  
scene = vp.canvas()  
# vp.display(width=100, height=100)  
floor = vp.box(pos=vp.vector(0, 0, 0), length=100, height=0.001, width=100, color=vp.color.blue)  
  
v = 0  
dt = 0.001  
mass = 0.1  
g = 9.81  
  
  
# k1 = 1000   
  
  
def getCOM(v):  
 COM = (v[0].pos + v[1].pos + v[2].pos + v[3].pos + v[4].pos + v[5].pos + v[6].pos + v[7].pos) / 8  
 return COM  
  
  
def mutation(r):  
 ran = random.randint(0, 9)  
 if ran > mutation\_rate:  
 x = random.randint(0, len(r) - 1)  
 y = random.randint(0, len(r) - 1)  
 # z = random.randint(0,27)  
 r[x], r[y] = r[y], r[x]  
 # r[z][0] = np.random.uniform(-0.2,0.2)  
 # r[z][1] = np.random.uniform(-np.pi,np.pi)  
 return r  
 else:  
 return r  
  
  
def Crossover(m, n):  
 M = []  
 N = []  
 for i in range(len(m)):  
 M.append(m[i])  
 N.append(n[i])  
 rand = random.randint(0, len(m) - 10)  
 index = list(range(rand, rand + 10))  
 exchangem = []  
 exchangen = []  
 for i in range(len(index)):  
 exchangem.append(M[index[i]])  
 exchangen.append(N[index[i]])  
 for j in range(len(index)):  
 M[index[j]] = exchangen[j]  
 N[index[j]] = exchangem[j]  
 return (M, N)  
  
  
dis1 = 0  
good\_dis = []  
best\_dis = []  
mutation\_rate = 7  
  
randdd = random.randint(0, 6)  
## Generate 10 parents  
pa1 = []  
for i in range(10):  
 pa = []  
 for i in range(28):  
 p = []  
 b = np.random.uniform(-0.2, 0.2)  
 c = np.random.uniform(-np.pi, np.pi)  
 k1 = np.random.uniform(1000, 6000)  
 p.append(b)  
 p.append(c)  
 p.append(k1)  
 pa.append(p)  
 pa1.append(pa)  
# print('PA1',len(pa1))  
  
dots = []  
for a in range(100):  
 for i in np.arange(0, 10, 2):  
 child1, child2 = Crossover(pa1[i], pa1[i + 1])  
 pa1.append(child1)  
 pa1.append(child2)  
  
 for i in range(10, 20):  
 pa1[i] = mutation(pa1[i])  
  
 # print('PA1after',pa1)  
  
 total\_dis = []  
 for h in range(20):  
 parameter = ['sine', 'cosine', 'air', 'hard k', 'soft k']  
  
ballname = ['b1', 'b2', 'b3', 'b4', 'b5', 'b6', 'b7', 'b8']  
ballvectors = [vp.vector(0, 0, 0), vp.vector(0, 1, 0), vp.vector(0, 0, 1), vp.vector(1, 0, 0), vp.vector(1, 1, 0),  
 vp.vector(0, 1, 1),  
 vp.vector(1, 0, 1), vp.vector(1, 1, 1)]  
  
COM = (ballvectors[0] + ballvectors[1] + ballvectors[2] + ballvectors[3] + ballvectors[4] + ballvectors[5] +  
 ballvectors[6] + ballvectors[7]) / 16  
  
springvecs = []  
for i in range(len(ballname)):  
 ballname[i] = vp.sphere(pos=ballvectors[i], radius=0.1, color=vp.color.red, velocity=vp.vector(0, 0, 0))  
for z in itertools.combinations(ballvectors, 2):  
 springvecs.append(z)  
  
spring = ['s1', 's2', 's3', 's4', 's5', 's6', 's7', 's8', 's9', 's10', 's11', 's12', 's13', 's14', 's15', 's16', 's17',  
 's18', 's19', 's20', 's21', 's22', 's23', 's24', 's25', 's26', 's27', 's28']  
  
for i in range(28):  
 position = springvecs[i][1] - springvecs[i][0]  
 spring[i] = vp.cylinder(pos=springvecs[i][0], axis=position, length=vp.mag(position), radius=.03,  
 color=vp.color.white)  
  
g\_vector = vp.vector(0, 9.81, 0)  
for i in range(len(ballname)):  
 ballname[i].velocity = vp.vector(0, 0, 0)  
  
  
def generateBalls(randnum):  
 if randnum == 0:  
 ballname1 = ['b11', 'b22', 'b33', 'b44', 'b55', 'b66', 'b77', 'b88']  
 ballvectors1 = [vp.vector(1, 0, 0), vp.vector(1, 1, 0), vp.vector(1, 0, 1), vp.vector(1, 1, 1),  
 vp.vector(2, 0, 0), vp.vector(2, 1, 0),  
 vp.vector(2, 0, 1), vp.vector(2, 1, 1)]  
 COM1 = (ballvectors1[0] + ballvectors1[1] + ballvectors1[2] + ballvectors1[3] + ballvectors1[4] + ballvectors1[  
 5] + ballvectors1[6] + ballvectors1[7]) / 8  
  
 springvecs1 = []  
 for i in range(len(ballname1)):  
 ballname1[i] = vp.sphere(pos=ballvectors1[i], radius=0.1, color=vp.color.red, velocity=vp.vector(0, 0, 0))  
 for z in itertools.combinations(ballvectors1, 2):  
 springvecs1.append(z)  
  
 spring1 = ['s1', 's2', 's3', 's4', 's5', 's6', 's7', 's8', 's9', 's10', 's11', 's12', 's13', 's14', 's15',  
 's16', 's17', 's18', 's19', 's20', 's21', 's22', 's23', 's24', 's25', 's26', 's27', 's28']  
  
 for i in range(28):  
 position1 = springvecs1[i][1] - springvecs1[i][0]  
 spring1[i] = vp.cylinder(pos=springvecs1[i][0], axis=position1, length=vp.mag(position1), radius=.03,  
 color=vp.color.white)  
  
 for i in range(len(ballname1)):  
 ballname1[i].velocity = vp.vector(0, 0, 0)  
  
 ##  
  
 ballname2 = ['b11', 'b22', 'b33', 'b44', 'b55', 'b66', 'b77', 'b88']  
 ballvectors2 = [vp.vector(0, 0, 1), vp.vector(0, 1, 1), vp.vector(1, 0, 1), vp.vector(1, 1, 1),  
 vp.vector(0, 0, 2), vp.vector(0, 1, 2),  
 vp.vector(1, 0, 2), vp.vector(1, 1, 2)]  
 COM2 = (ballvectors2[0] + ballvectors2[1] + ballvectors2[2] + ballvectors2[3] + ballvectors2[4] + ballvectors2[  
 5] + ballvectors2[6] + ballvectors2[7]) / 8  
  
 springvecs2 = []  
 for i in range(len(ballname2)):  
 ballname2[i] = vp.sphere(pos=ballvectors2[i], radius=0.1, color=vp.color.red, velocity=vp.vector(0, 0, 0))  
 for z in itertools.combinations(ballvectors2, 2):  
 springvecs2.append(z)  
  
 spring2 = ['s1', 's2', 's3', 's4', 's5', 's6', 's7', 's8', 's9', 's10', 's11', 's12', 's13', 's14', 's15',  
 's16', 's17', 's18', 's19', 's20', 's21', 's22', 's23', 's24', 's25', 's26', 's27', 's28']  
  
 for i in range(28):  
 position2 = springvecs2[i][1] - springvecs2[i][0]  
 spring2[i] = vp.cylinder(pos=springvecs2[i][0], axis=position2, length=vp.mag(position2), radius=.03,  
 color=vp.color.white)  
  
 for i in range(len(ballname1)):  
 ballname2[i].velocity = vp.vector(0, 0, 0)  
  
 ##   
  
 ballname3 = ['b11', 'b22', 'b33', 'b44', 'b55', 'b66', 'b77', 'b88']  
 ballvectors3 = [vp.vector(1, 0, 1), vp.vector(1, 1, 1), vp.vector(2, 1, 1), vp.vector(2, 0, 1),  
 vp.vector(1, 1, 2), vp.vector(1, 0, 2),  
 vp.vector(2, 1, 2), vp.vector(2, 0, 2)]  
 COM3 = (ballvectors3[0] + ballvectors3[1] + ballvectors3[2] + ballvectors3[3] + ballvectors3[4] + ballvectors3[  
 5] + ballvectors3[6] + ballvectors3[7]) / 8  
  
 springvecs3 = []  
 for i in range(len(ballname3)):  
 ballname3[i] = vp.sphere(pos=ballvectors3[i], radius=0.1, color=vp.color.red, velocity=vp.vector(0, 0, 0))  
 for z in itertools.combinations(ballvectors3, 2):  
 springvecs3.append(z)  
  
 spring3 = ['s1', 's2', 's3', 's4', 's5', 's6', 's7', 's8', 's9', 's10', 's11', 's12', 's13', 's14', 's15',  
 's16', 's17', 's18', 's19', 's20', 's21', 's22', 's23', 's24', 's25', 's26', 's27', 's28']  
  
 for i in range(28):  
 position3 = springvecs3[i][1] - springvecs3[i][0]  
 spring3[i] = vp.cylinder(pos=springvecs3[i][0], axis=position3, length=vp.mag(position3), radius=.03,  
 color=vp.color.white)  
  
 for i in range(len(ballname3)):  
 ballname3[i].velocity = vp.vector(0, 0, 0)  
  
 ##  
  
 ballname4 = ['b11', 'b22', 'b33', 'b44', 'b55', 'b66', 'b77', 'b88']  
 ballvectors4 = [vp.vector(0, 1, 0), vp.vector(1, 1, 0), vp.vector(0, 1, 1), vp.vector(1, 1, 1),  
 vp.vector(0, 2, 0), vp.vector(1, 2, 0),  
 vp.vector(0, 2, 1), vp.vector(1, 2, 1)]  
 COM4 = (ballvectors4[0] + ballvectors4[1] + ballvectors4[2] + ballvectors4[3] + ballvectors4[4] + ballvectors4[  
 5] + ballvectors4[6] + ballvectors4[7]) / 8  
  
 springvecs4 = []  
 for i in range(len(ballname4)):  
 ballname4[i] = vp.sphere(pos=ballvectors4[i], radius=0.1, color=vp.color.red, velocity=vp.vector(0, 0, 0))  
 for z in itertools.combinations(ballvectors4, 2):  
 springvecs4.append(z)  
  
 spring4 = ['s1', 's2', 's3', 's4', 's5', 's6', 's7', 's8', 's9', 's10', 's11', 's12', 's13', 's14', 's15',  
 's16', 's17', 's18', 's19', 's20', 's21', 's22', 's23', 's24', 's25', 's26', 's27', 's28']  
  
 for i in range(28):  
 position4 = springvecs4[i][1] - springvecs4[i][0]  
 spring4[i] = vp.cylinder(pos=springvecs4[i][0], axis=position4, length=vp.mag(position4), radius=.03,  
 color=vp.color.white)  
  
 for i in range(len(ballname4)):  
 ballname4[i].velocity = vp.vector(0, 0, 0)  
  
 ##   
  
 ballname5 = ['b11', 'b22', 'b33', 'b44', 'b55', 'b66', 'b77', 'b88']  
 ballvectors5 = [vp.vector(1, 1, 1), vp.vector(1, 2, 1), vp.vector(2, 1, 1), vp.vector(2, 2, 1),  
 vp.vector(1, 1, 2), vp.vector(1, 2, 2),  
 vp.vector(2, 1, 2), vp.vector(2, 2, 2)]  
 COM5 = (ballvectors5[0] + ballvectors5[1] + ballvectors5[2] + ballvectors5[3] + ballvectors5[4] + ballvectors5[  
 5] + ballvectors5[6] + ballvectors5[7]) / 8  
  
 springvecs5 = []  
 for i in range(len(ballname5)):  
 ballname5[i] = vp.sphere(pos=ballvectors5[i], radius=0.1, color=vp.color.red, velocity=vp.vector(0, 0, 0))  
 for z in itertools.combinations(ballvectors5, 2):  
 springvecs5.append(z)  
  
 spring5 = ['s1', 's2', 's3', 's4', 's5', 's6', 's7', 's8', 's9', 's10', 's11', 's12', 's13', 's14', 's15',  
 's16', 's17', 's18', 's19', 's20', 's21', 's22', 's23', 's24', 's25', 's26', 's27', 's28']  
  
 for i in range(28):  
 position5 = springvecs5[i][1] - springvecs5[i][0]  
 spring5[i] = vp.cylinder(pos=springvecs5[i][0], axis=position5, length=vp.mag(position5), radius=.03,  
 color=vp.color.white)  
  
 for i in range(len(ballname5)):  
 ballname5[i].velocity = vp.vector(0, 0, 0)  
  
 ##  
  
 ballname6 = ['b11', 'b22', 'b33', 'b44', 'b55', 'b66', 'b77', 'b88']  
 ballvectors6 = [vp.vector(1, 1, 0), vp.vector(1, 1, 1), vp.vector(1, 2, 0), vp.vector(1, 2, 1),  
 vp.vector(2, 1, 0), vp.vector(2, 1, 1),  
 vp.vector(2, 2, 0), vp.vector(2, 2, 1)]  
 COM6 = (ballvectors6[0] + ballvectors6[1] + ballvectors6[2] + ballvectors6[3] + ballvectors6[4] + ballvectors6[  
 5] + ballvectors6[6] + ballvectors6[7]) / 8  
  
 springvecs6 = []  
 for i in range(len(ballname6)):  
 ballname6[i] = vp.sphere(pos=ballvectors6[i], radius=0.1, color=vp.color.red, velocity=vp.vector(0, 0, 0))  
 for z in itertools.combinations(ballvectors6, 2):  
 springvecs6.append(z)  
  
 spring6 = ['s1', 's2', 's3', 's4', 's5', 's6', 's7', 's8', 's9', 's10', 's11', 's12', 's13', 's14', 's15',  
 's16', 's17', 's18', 's19', 's20', 's21', 's22', 's23', 's24', 's25', 's26', 's27', 's28']  
  
 for i in range(28):  
 position6 = springvecs6[i][1] - springvecs6[i][0]  
 spring6[i] = vp.cylinder(pos=springvecs6[i][0], axis=position6, length=vp.mag(position6), radius=.03,  
 color=vp.color.white)  
  
 for i in range(len(ballname6)):  
 ballname6[i].velocity = vp.vector(0, 0, 0)  
  
 ##   
  
 ballname7 = ['b11', 'b22', 'b33', 'b44', 'b55', 'b66', 'b77', 'b88']  
 ballvectors7 = [vp.vector(0, 1, 1), vp.vector(1, 1, 1), vp.vector(0, 1, 2), vp.vector(1, 1, 2),  
 vp.vector(0, 2, 1), vp.vector(1, 2, 1),  
 vp.vector(0, 2, 2), vp.vector(1, 2, 2)]  
 COM7 = (ballvectors7[0] + ballvectors7[1] + ballvectors7[2] + ballvectors7[3] + ballvectors7[4] + ballvectors7[  
 5] + ballvectors7[6] + ballvectors7[7]) / 8  
  
 springvecs7 = []  
 for i in range(len(ballname7)):  
 ballname7[i] = vp.sphere(pos=ballvectors7[i], radius=0.1, color=vp.color.red, velocity=vp.vector(0, 0, 0))  
 for z in itertools.combinations(ballvectors7, 2):  
 springvecs7.append(z)  
  
 spring7 = ['s1', 's2', 's3', 's4', 's5', 's6', 's7', 's8', 's9', 's10', 's11', 's12', 's13', 's14', 's15',  
 's16', 's17', 's18', 's19', 's20', 's21', 's22', 's23', 's24', 's25', 's26', 's27', 's28']  
  
 for i in range(28):  
 position7 = springvecs7[i][1] - springvecs7[i][0]  
 spring7[i] = vp.cylinder(pos=springvecs7[i][0], axis=position7, length=vp.mag(position7), radius=.03,  
 color=vp.color.white)  
  
 for i in range(len(ballname7)):  
 ballname7[i].velocity = vp.vector(0, 0, 0)  
  
 Finalcom = COM + COM1 + COM2 + COM3 + COM4 + COM5 + COM6 + COM7  
  
 elif randnum == 1:  
  
 ballname1 = ['b11', 'b22', 'b33', 'b44', 'b55', 'b66', 'b77', 'b88']  
 ballvectors1 = [vp.vector(1, 0, 0), vp.vector(1, 1, 0), vp.vector(1, 0, 1), vp.vector(1, 1, 1),  
 vp.vector(2, 0, 0), vp.vector(2, 1, 0),  
 vp.vector(2, 0, 1), vp.vector(2, 1, 1)]  
 COM1 = (ballvectors1[0] + ballvectors1[1] + ballvectors1[2] + ballvectors1[3] + ballvectors1[4] + ballvectors1[  
 5] + ballvectors1[6] + ballvectors1[7]) / 8  
  
 springvecs1 = []  
 for i in range(len(ballname1)):  
 ballname1[i] = vp.sphere(pos=ballvectors1[i], radius=0.1, color=vp.color.red, velocity=vp.vector(0, 0, 0))  
 for z in itertools.combinations(ballvectors1, 2):  
 springvecs1.append(z)  
  
 spring1 = ['s1', 's2', 's3', 's4', 's5', 's6', 's7', 's8', 's9', 's10', 's11', 's12', 's13', 's14', 's15',  
 's16', 's17', 's18', 's19', 's20', 's21', 's22', 's23', 's24', 's25', 's26', 's27', 's28']  
  
 for i in range(28):  
 position1 = springvecs1[i][1] - springvecs1[i][0]  
 spring1[i] = vp.cylinder(pos=springvecs1[i][0], axis=position1, length=vp.mag(position1), radius=.03,  
 color=vp.color.white)  
  
 for i in range(len(ballname1)):  
 ballname1[i].velocity = vp.vector(0, 0, 0)  
  
 ##  
  
 ballname2 = ['b11', 'b22', 'b33', 'b44', 'b55', 'b66', 'b77', 'b88']  
 ballvectors2 = [vp.vector(0, 0, 1), vp.vector(0, 1, 1), vp.vector(1, 0, 1), vp.vector(1, 1, 1),  
 vp.vector(0, 0, 2), vp.vector(0, 1, 2),  
 vp.vector(1, 0, 2), vp.vector(1, 1, 2)]  
 COM2 = (ballvectors2[0] + ballvectors2[1] + ballvectors2[2] + ballvectors2[3] + ballvectors2[4] + ballvectors2[  
 5] + ballvectors2[6] + ballvectors2[7]) / 8  
  
 springvecs2 = []  
 for i in range(len(ballname2)):  
 ballname2[i] = vp.sphere(pos=ballvectors2[i], radius=0.1, color=vp.color.red, velocity=vp.vector(0, 0, 0))  
 for z in itertools.combinations(ballvectors2, 2):  
 springvecs2.append(z)  
  
 spring2 = ['s1', 's2', 's3', 's4', 's5', 's6', 's7', 's8', 's9', 's10', 's11', 's12', 's13', 's14', 's15',  
 's16', 's17', 's18', 's19', 's20', 's21', 's22', 's23', 's24', 's25', 's26', 's27', 's28']  
  
 for i in range(28):  
 position2 = springvecs2[i][1] - springvecs2[i][0]  
 spring2[i] = vp.cylinder(pos=springvecs2[i][0], axis=position2, length=vp.mag(position2), radius=.03,  
 color=vp.color.white)  
  
 for i in range(len(ballname1)):  
 ballname2[i].velocity = vp.vector(0, 0, 0)  
  
 ##   
  
 ballname3 = ['b11', 'b22', 'b33', 'b44', 'b55', 'b66', 'b77', 'b88']  
 ballvectors3 = [vp.vector(1, 0, 1), vp.vector(1, 1, 1), vp.vector(2, 1, 1), vp.vector(2, 0, 1),  
 vp.vector(1, 1, 2), vp.vector(1, 0, 2),  
 vp.vector(2, 1, 2), vp.vector(2, 0, 2)]  
 COM3 = (ballvectors3[0] + ballvectors3[1] + ballvectors3[2] + ballvectors3[3] + ballvectors3[4] + ballvectors3[  
 5] + ballvectors3[6] + ballvectors3[7]) / 8  
  
 springvecs3 = []  
 for i in range(len(ballname3)):  
 ballname3[i] = vp.sphere(pos=ballvectors3[i], radius=0.1, color=vp.color.red, velocity=vp.vector(0, 0, 0))  
 for z in itertools.combinations(ballvectors3, 2):  
 springvecs3.append(z)  
  
 spring3 = ['s1', 's2', 's3', 's4', 's5', 's6', 's7', 's8', 's9', 's10', 's11', 's12', 's13', 's14', 's15',  
 's16', 's17', 's18', 's19', 's20', 's21', 's22', 's23', 's24', 's25', 's26', 's27', 's28']  
  
 for i in range(28):  
 position3 = springvecs3[i][1] - springvecs3[i][0]  
 spring3[i] = vp.cylinder(pos=springvecs3[i][0], axis=position3, length=vp.mag(position3), radius=.03,  
 color=vp.color.white)  
  
 for i in range(len(ballname3)):  
 ballname3[i].velocity = vp.vector(0, 0, 0)  
  
 ##  
  
 ballname4 = ['b11', 'b22', 'b33', 'b44', 'b55', 'b66', 'b77', 'b88']  
 ballvectors4 = [vp.vector(0, 1, 0), vp.vector(1, 1, 0), vp.vector(0, 1, 1), vp.vector(1, 1, 1),  
 vp.vector(0, 2, 0), vp.vector(1, 2, 0),  
 vp.vector(0, 2, 1), vp.vector(1, 2, 1)]  
 COM4 = (ballvectors4[0] + ballvectors4[1] + ballvectors4[2] + ballvectors4[3] + ballvectors4[4] + ballvectors4[  
 5] + ballvectors4[6] + ballvectors4[7]) / 8  
  
 springvecs4 = []  
 for i in range(len(ballname4)):  
 ballname4[i] = vp.sphere(pos=ballvectors4[i], radius=0.1, color=vp.color.red, velocity=vp.vector(0, 0, 0))  
 for z in itertools.combinations(ballvectors4, 2):  
 springvecs4.append(z)  
  
 spring4 = ['s1', 's2', 's3', 's4', 's5', 's6', 's7', 's8', 's9', 's10', 's11', 's12', 's13', 's14', 's15',  
 's16', 's17', 's18', 's19', 's20', 's21', 's22', 's23', 's24', 's25', 's26', 's27', 's28']  
  
 for i in range(28):  
 position4 = springvecs4[i][1] - springvecs4[i][0]  
 spring4[i] = vp.cylinder(pos=springvecs4[i][0], axis=position4, length=vp.mag(position4), radius=.03,  
 color=vp.color.white)  
  
 for i in range(len(ballname4)):  
 ballname4[i].velocity = vp.vector(0, 0, 0)  
  
 ##   
  
 ballname5 = ['b11', 'b22', 'b33', 'b44', 'b55', 'b66', 'b77', 'b88']  
 ballvectors5 = [vp.vector(1, 1, 1), vp.vector(1, 2, 1), vp.vector(2, 1, 1), vp.vector(2, 2, 1),  
 vp.vector(1, 1, 2), vp.vector(1, 2, 2),  
 vp.vector(2, 1, 2), vp.vector(2, 2, 2)]  
 COM5 = (ballvectors5[0] + ballvectors5[1] + ballvectors5[2] + ballvectors5[3] + ballvectors5[4] + ballvectors5[  
 5] + ballvectors5[6] + ballvectors5[7]) / 8  
  
 springvecs5 = []  
 for i in range(len(ballname5)):  
 ballname5[i] = vp.sphere(pos=ballvectors5[i], radius=0.1, color=vp.color.red, velocity=vp.vector(0, 0, 0))  
 for z in itertools.combinations(ballvectors5, 2):  
 springvecs5.append(z)  
  
 spring5 = ['s1', 's2', 's3', 's4', 's5', 's6', 's7', 's8', 's9', 's10', 's11', 's12', 's13', 's14', 's15',  
 's16', 's17', 's18', 's19', 's20', 's21', 's22', 's23', 's24', 's25', 's26', 's27', 's28']  
  
 for i in range(28):  
 position5 = springvecs5[i][1] - springvecs5[i][0]  
 spring5[i] = vp.cylinder(pos=springvecs5[i][0], axis=position5, length=vp.mag(position5), radius=.03,  
 color=vp.color.white)  
  
 for i in range(len(ballname5)):  
 ballname5[i].velocity = vp.vector(0, 0, 0)  
  
 ##  
  
 ballname6 = ['b11', 'b22', 'b33', 'b44', 'b55', 'b66', 'b77', 'b88']  
 ballvectors6 = [vp.vector(1, 1, 0), vp.vector(1, 1, 1), vp.vector(1, 2, 0), vp.vector(1, 2, 1),  
 vp.vector(2, 1, 0), vp.vector(2, 1, 1),  
 vp.vector(2, 2, 0), vp.vector(2, 2, 1)]  
 COM6 = (ballvectors6[0] + ballvectors6[1] + ballvectors6[2] + ballvectors6[3] + ballvectors6[4] + ballvectors6[  
 5] + ballvectors6[6] + ballvectors6[7]) / 8  
  
 springvecs6 = []  
 for i in range(len(ballname6)):  
 ballname6[i] = vp.sphere(pos=ballvectors6[i], radius=0.1, color=vp.color.red, velocity=vp.vector(0, 0, 0))  
 for z in itertools.combinations(ballvectors6, 2):  
 springvecs6.append(z)  
  
 spring6 = ['s1', 's2', 's3', 's4', 's5', 's6', 's7', 's8', 's9', 's10', 's11', 's12', 's13', 's14', 's15',  
 's16', 's17', 's18', 's19', 's20', 's21', 's22', 's23', 's24', 's25', 's26', 's27', 's28']  
  
 for i in range(28):  
 position6 = springvecs6[i][1] - springvecs6[i][0]  
 spring6[i] = vp.cylinder(pos=springvecs6[i][0], axis=position6, length=vp.mag(position6), radius=.03,  
 color=vp.color.white)  
  
 for i in range(len(ballname6)):  
 ballname6[i].velocity = vp.vector(0, 0, 0)  
  
 Finalcom = COM + COM1 + COM2 + COM3 + COM4 + COM5 + COM6  
  
 elif randnum == 2:  
  
 ballname1 = ['b11', 'b22', 'b33', 'b44', 'b55', 'b66', 'b77', 'b88']  
 ballvectors1 = [vp.vector(1, 0, 0), vp.vector(1, 1, 0), vp.vector(1, 0, 1), vp.vector(1, 1, 1),  
 vp.vector(2, 0, 0), vp.vector(2, 1, 0),  
 vp.vector(2, 0, 1), vp.vector(2, 1, 1)]  
 COM1 = (ballvectors1[0] + ballvectors1[1] + ballvectors1[2] + ballvectors1[3] + ballvectors1[4] + ballvectors1[  
 5] + ballvectors1[6] + ballvectors1[7]) / 8  
  
 springvecs1 = []  
 for i in range(len(ballname1)):  
 ballname1[i] = vp.sphere(pos=ballvectors1[i], radius=0.1, color=vp.color.red, velocity=vp.vector(0, 0, 0))  
 for z in itertools.combinations(ballvectors1, 2):  
 springvecs1.append(z)  
  
 spring1 = ['s1', 's2', 's3', 's4', 's5', 's6', 's7', 's8', 's9', 's10', 's11', 's12', 's13', 's14', 's15',  
 's16', 's17', 's18', 's19', 's20', 's21', 's22', 's23', 's24', 's25', 's26', 's27', 's28']  
  
 for i in range(28):  
 position1 = springvecs1[i][1] - springvecs1[i][0]  
 spring1[i] = vp.cylinder(pos=springvecs1[i][0], axis=position1, length=vp.mag(position1), radius=.03,  
 color=vp.color.white)  
  
 for i in range(len(ballname1)):  
 ballname1[i].velocity = vp.vector(0, 0, 0)  
  
 ##  
  
 ballname2 = ['b11', 'b22', 'b33', 'b44', 'b55', 'b66', 'b77', 'b88']  
 ballvectors2 = [vp.vector(0, 0, 1), vp.vector(0, 1, 1), vp.vector(1, 0, 1), vp.vector(1, 1, 1),  
 vp.vector(0, 0, 2), vp.vector(0, 1, 2),  
 vp.vector(1, 0, 2), vp.vector(1, 1, 2)]  
 COM2 = (ballvectors2[0] + ballvectors2[1] + ballvectors2[2] + ballvectors2[3] + ballvectors2[4] + ballvectors2[  
 5] + ballvectors2[6] + ballvectors2[7]) / 8  
  
 springvecs2 = []  
 for i in range(len(ballname2)):  
 ballname2[i] = vp.sphere(pos=ballvectors2[i], radius=0.1, color=vp.color.red, velocity=vp.vector(0, 0, 0))  
 for z in itertools.combinations(ballvectors2, 2):  
 springvecs2.append(z)  
  
 spring2 = ['s1', 's2', 's3', 's4', 's5', 's6', 's7', 's8', 's9', 's10', 's11', 's12', 's13', 's14', 's15',  
 's16', 's17', 's18', 's19', 's20', 's21', 's22', 's23', 's24', 's25', 's26', 's27', 's28']  
  
 for i in range(28):  
 position2 = springvecs2[i][1] - springvecs2[i][0]  
 spring2[i] = vp.cylinder(pos=springvecs2[i][0], axis=position2, length=vp.mag(position2), radius=.03,  
 color=vp.color.white)  
  
 for i in range(len(ballname1)):  
 ballname2[i].velocity = vp.vector(0, 0, 0)  
  
 ##   
  
 ballname3 = ['b11', 'b22', 'b33', 'b44', 'b55', 'b66', 'b77', 'b88']  
 ballvectors3 = [vp.vector(1, 0, 1), vp.vector(1, 1, 1), vp.vector(2, 1, 1), vp.vector(2, 0, 1),  
 vp.vector(1, 1, 2), vp.vector(1, 0, 2),  
 vp.vector(2, 1, 2), vp.vector(2, 0, 2)]  
 COM3 = (ballvectors3[0] + ballvectors3[1] + ballvectors3[2] + ballvectors3[3] + ballvectors3[4] + ballvectors3[  
 5] + ballvectors3[6] + ballvectors3[7]) / 8  
  
 springvecs3 = []  
 for i in range(len(ballname3)):  
 ballname3[i] = vp.sphere(pos=ballvectors3[i], radius=0.1, color=vp.color.red, velocity=vp.vector(0, 0, 0))  
 for z in itertools.combinations(ballvectors3, 2):  
 springvecs3.append(z)  
  
 spring3 = ['s1', 's2', 's3', 's4', 's5', 's6', 's7', 's8', 's9', 's10', 's11', 's12', 's13', 's14', 's15',  
 's16', 's17', 's18', 's19', 's20', 's21', 's22', 's23', 's24', 's25', 's26', 's27', 's28']  
  
 for i in range(28):  
 position3 = springvecs3[i][1] - springvecs3[i][0]  
 spring3[i] = vp.cylinder(pos=springvecs3[i][0], axis=position3, length=vp.mag(position3), radius=.03,  
 color=vp.color.white)  
  
 for i in range(len(ballname3)):  
 ballname3[i].velocity = vp.vector(0, 0, 0)  
  
 ##  
  
 ballname4 = ['b11', 'b22', 'b33', 'b44', 'b55', 'b66', 'b77', 'b88']  
 ballvectors4 = [vp.vector(0, 1, 0), vp.vector(1, 1, 0), vp.vector(0, 1, 1), vp.vector(1, 1, 1),  
 vp.vector(0, 2, 0), vp.vector(1, 2, 0),  
 vp.vector(0, 2, 1), vp.vector(1, 2, 1)]  
 COM4 = (ballvectors4[0] + ballvectors4[1] + ballvectors4[2] + ballvectors4[3] + ballvectors4[4] + ballvectors4[  
 5] + ballvectors4[6] + ballvectors4[7]) / 8  
  
 springvecs4 = []  
 for i in range(len(ballname4)):  
 ballname4[i] = vp.sphere(pos=ballvectors4[i], radius=0.1, color=vp.color.red, velocity=vp.vector(0, 0, 0))  
 for z in itertools.combinations(ballvectors4, 2):  
 springvecs4.append(z)  
  
 spring4 = ['s1', 's2', 's3', 's4', 's5', 's6', 's7', 's8', 's9', 's10', 's11', 's12', 's13', 's14', 's15',  
 's16', 's17', 's18', 's19', 's20', 's21', 's22', 's23', 's24', 's25', 's26', 's27', 's28']  
  
 for i in range(28):  
 position4 = springvecs4[i][1] - springvecs4[i][0]  
 spring4[i] = vp.cylinder(pos=springvecs4[i][0], axis=position4, length=vp.mag(position4), radius=.03,  
 color=vp.color.white)  
  
 for i in range(len(ballname4)):  
 ballname4[i].velocity = vp.vector(0, 0, 0)  
  
 ##   
  
 ballname5 = ['b11', 'b22', 'b33', 'b44', 'b55', 'b66', 'b77', 'b88']  
 ballvectors5 = [vp.vector(1, 1, 1), vp.vector(1, 2, 1), vp.vector(2, 1, 1), vp.vector(2, 2, 1),  
 vp.vector(1, 1, 2), vp.vector(1, 2, 2),  
 vp.vector(2, 1, 2), vp.vector(2, 2, 2)]  
 COM5 = (ballvectors5[0] + ballvectors5[1] + ballvectors5[2] + ballvectors5[3] + ballvectors5[4] + ballvectors5[  
 5] + ballvectors5[6] + ballvectors5[7]) / 8  
  
 springvecs5 = []  
 for i in range(len(ballname5)):  
 ballname5[i] = vp.sphere(pos=ballvectors5[i], radius=0.1, color=vp.color.red, velocity=vp.vector(0, 0, 0))  
 for z in itertools.combinations(ballvectors5, 2):  
 springvecs5.append(z)  
  
 spring5 = ['s1', 's2', 's3', 's4', 's5', 's6', 's7', 's8', 's9', 's10', 's11', 's12', 's13', 's14', 's15',  
 's16', 's17', 's18', 's19', 's20', 's21', 's22', 's23', 's24', 's25', 's26', 's27', 's28']  
  
 for i in range(28):  
 position5 = springvecs5[i][1] - springvecs5[i][0]  
 spring5[i] = vp.cylinder(pos=springvecs5[i][0], axis=position5, length=vp.mag(position5), radius=.03,  
 color=vp.color.white)  
  
 for i in range(len(ballname5)):  
 ballname5[i].velocity = vp.vector(0, 0, 0)  
  
 Finalcom = COM + COM1 + COM2 + COM3 + COM4 + COM5  
  
 elif randnum == 3:  
  
 ballname1 = ['b11', 'b22', 'b33', 'b44', 'b55', 'b66', 'b77', 'b88']  
 ballvectors1 = [vp.vector(1, 0, 0), vp.vector(1, 1, 0), vp.vector(1, 0, 1), vp.vector(1, 1, 1),  
 vp.vector(2, 0, 0), vp.vector(2, 1, 0),  
 vp.vector(2, 0, 1), vp.vector(2, 1, 1)]  
 COM1 = (ballvectors1[0] + ballvectors1[1] + ballvectors1[2] + ballvectors1[3] + ballvectors1[4] + ballvectors1[  
 5] + ballvectors1[6] + ballvectors1[7]) / 8  
  
 springvecs1 = []  
 for i in range(len(ballname1)):  
 ballname1[i] = vp.sphere(pos=ballvectors1[i], radius=0.1, color=vp.color.red, velocity=vp.vector(0, 0, 0))  
 for z in itertools.combinations(ballvectors1, 2):  
 springvecs1.append(z)  
  
 spring1 = ['s1', 's2', 's3', 's4', 's5', 's6', 's7', 's8', 's9', 's10', 's11', 's12', 's13', 's14', 's15',  
 's16', 's17', 's18', 's19', 's20', 's21', 's22', 's23', 's24', 's25', 's26', 's27', 's28']  
  
 for i in range(28):  
 position1 = springvecs1[i][1] - springvecs1[i][0]  
 spring1[i] = vp.cylinder(pos=springvecs1[i][0], axis=position1, length=vp.mag(position1), radius=.03,  
 color=vp.color.white)  
  
 for i in range(len(ballname1)):  
 ballname1[i].velocity = vp.vector(0, 0, 0)  
  
 ##  
  
 ballname2 = ['b11', 'b22', 'b33', 'b44', 'b55', 'b66', 'b77', 'b88']  
 ballvectors2 = [vp.vector(0, 0, 1), vp.vector(0, 1, 1), vp.vector(1, 0, 1), vp.vector(1, 1, 1),  
 vp.vector(0, 0, 2), vp.vector(0, 1, 2),  
 vp.vector(1, 0, 2), vp.vector(1, 1, 2)]  
 COM2 = (ballvectors2[0] + ballvectors2[1] + ballvectors2[2] + ballvectors2[3] + ballvectors2[4] + ballvectors2[  
 5] + ballvectors2[6] + ballvectors2[7]) / 8  
  
 springvecs2 = []  
 for i in range(len(ballname2)):  
 ballname2[i] = vp.sphere(pos=ballvectors2[i], radius=0.1, color=vp.color.red, velocity=vp.vector(0, 0, 0))  
 for z in itertools.combinations(ballvectors2, 2):  
 springvecs2.append(z)  
  
 spring2 = ['s1', 's2', 's3', 's4', 's5', 's6', 's7', 's8', 's9', 's10', 's11', 's12', 's13', 's14', 's15',  
 's16', 's17', 's18', 's19', 's20', 's21', 's22', 's23', 's24', 's25', 's26', 's27', 's28']  
  
 for i in range(28):  
 position2 = springvecs2[i][1] - springvecs2[i][0]  
 spring2[i] = vp.cylinder(pos=springvecs2[i][0], axis=position2, length=vp.mag(position2), radius=.03,  
 color=vp.color.white)  
  
 for i in range(len(ballname1)):  
 ballname2[i].velocity = vp.vector(0, 0, 0)  
  
 ##   
  
 ballname3 = ['b11', 'b22', 'b33', 'b44', 'b55', 'b66', 'b77', 'b88']  
 ballvectors3 = [vp.vector(1, 0, 1), vp.vector(1, 1, 1), vp.vector(2, 1, 1), vp.vector(2, 0, 1),  
 vp.vector(1, 1, 2), vp.vector(1, 0, 2),  
 vp.vector(2, 1, 2), vp.vector(2, 0, 2)]  
 COM3 = (ballvectors3[0] + ballvectors3[1] + ballvectors3[2] + ballvectors3[3] + ballvectors3[4] + ballvectors3[  
 5] + ballvectors3[6] + ballvectors3[7]) / 8  
  
 springvecs3 = []  
 for i in range(len(ballname3)):  
 ballname3[i] = vp.sphere(pos=ballvectors3[i], radius=0.1, color=vp.color.red, velocity=vp.vector(0, 0, 0))  
 for z in itertools.combinations(ballvectors3, 2):  
 springvecs3.append(z)  
  
 spring3 = ['s1', 's2', 's3', 's4', 's5', 's6', 's7', 's8', 's9', 's10', 's11', 's12', 's13', 's14', 's15',  
 's16', 's17', 's18', 's19', 's20', 's21', 's22', 's23', 's24', 's25', 's26', 's27', 's28']  
  
 for i in range(28):  
 position3 = springvecs3[i][1] - springvecs3[i][0]  
 spring3[i] = vp.cylinder(pos=springvecs3[i][0], axis=position3, length=vp.mag(position3), radius=.03,  
 color=vp.color.white)  
  
 for i in range(len(ballname3)):  
 ballname3[i].velocity = vp.vector(0, 0, 0)  
  
 ##  
  
 ballname4 = ['b11', 'b22', 'b33', 'b44', 'b55', 'b66', 'b77', 'b88']  
 ballvectors4 = [vp.vector(0, 1, 0), vp.vector(1, 1, 0), vp.vector(0, 1, 1), vp.vector(1, 1, 1),  
 vp.vector(0, 2, 0), vp.vector(1, 2, 0),  
 vp.vector(0, 2, 1), vp.vector(1, 2, 1)]  
 COM4 = (ballvectors4[0] + ballvectors4[1] + ballvectors4[2] + ballvectors4[3] + ballvectors4[4] + ballvectors4[  
 5] + ballvectors4[6] + ballvectors4[7]) / 8  
  
 springvecs4 = []  
 for i in range(len(ballname4)):  
 ballname4[i] = vp.sphere(pos=ballvectors4[i], radius=0.1, color=vp.color.red, velocity=vp.vector(0, 0, 0))  
 for z in itertools.combinations(ballvectors4, 2):  
 springvecs4.append(z)  
  
 spring4 = ['s1', 's2', 's3', 's4', 's5', 's6', 's7', 's8', 's9', 's10', 's11', 's12', 's13', 's14', 's15',  
 's16', 's17', 's18', 's19', 's20', 's21', 's22', 's23', 's24', 's25', 's26', 's27', 's28']  
  
 for i in range(28):  
 position4 = springvecs4[i][1] - springvecs4[i][0]  
 spring4[i] = vp.cylinder(pos=springvecs4[i][0], axis=position4, length=vp.mag(position4), radius=.03,  
 color=vp.color.white)  
  
 for i in range(len(ballname4)):  
 ballname4[i].velocity = vp.vector(0, 0, 0)  
  
 finalcom = COM + COM1 + COM2 + COM3 + COM4  
  
 elif randnum == 4:  
  
 ballname1 = ['b11', 'b22', 'b33', 'b44', 'b55', 'b66', 'b77', 'b88']  
 ballvectors1 = [vp.vector(1, 0, 0), vp.vector(1, 1, 0), vp.vector(1, 0, 1), vp.vector(1, 1, 1),  
 vp.vector(2, 0, 0), vp.vector(2, 1, 0),  
 vp.vector(2, 0, 1), vp.vector(2, 1, 1)]  
 COM1 = (ballvectors1[0] + ballvectors1[1] + ballvectors1[2] + ballvectors1[3] + ballvectors1[4] + ballvectors1[  
 5] + ballvectors1[6] + ballvectors1[7]) / 8  
  
 springvecs1 = []  
 for i in range(len(ballname1)):  
 ballname1[i] = vp.sphere(pos=ballvectors1[i], radius=0.1, color=vp.color.red, velocity=vp.vector(0, 0, 0))  
 for z in itertools.combinations(ballvectors1, 2):  
 springvecs1.append(z)  
  
 spring1 = ['s1', 's2', 's3', 's4', 's5', 's6', 's7', 's8', 's9', 's10', 's11', 's12', 's13', 's14', 's15',  
 's16', 's17', 's18', 's19', 's20', 's21', 's22', 's23', 's24', 's25', 's26', 's27', 's28']  
  
 for i in range(28):  
 position1 = springvecs1[i][1] - springvecs1[i][0]  
 spring1[i] = vp.cylinder(pos=springvecs1[i][0], axis=position1, length=vp.mag(position1), radius=.03,  
 color=vp.color.white)  
  
 for i in range(len(ballname1)):  
 ballname1[i].velocity = vp.vector(0, 0, 0)  
  
 ##  
  
 ballname2 = ['b11', 'b22', 'b33', 'b44', 'b55', 'b66', 'b77', 'b88']  
 ballvectors2 = [vp.vector(0, 0, 1), vp.vector(0, 1, 1), vp.vector(1, 0, 1), vp.vector(1, 1, 1),  
 vp.vector(0, 0, 2), vp.vector(0, 1, 2),  
 vp.vector(1, 0, 2), vp.vector(1, 1, 2)]  
 COM2 = (ballvectors2[0] + ballvectors2[1] + ballvectors2[2] + ballvectors2[3] + ballvectors2[4] + ballvectors2[  
 5] + ballvectors2[6] + ballvectors2[7]) / 8  
  
 springvecs2 = []  
 for i in range(len(ballname2)):  
 ballname2[i] = vp.sphere(pos=ballvectors2[i], radius=0.1, color=vp.color.red, velocity=vp.vector(0, 0, 0))  
 for z in itertools.combinations(ballvectors2, 2):  
 springvecs2.append(z)  
  
 spring2 = ['s1', 's2', 's3', 's4', 's5', 's6', 's7', 's8', 's9', 's10', 's11', 's12', 's13', 's14', 's15',  
 's16', 's17', 's18', 's19', 's20', 's21', 's22', 's23', 's24', 's25', 's26', 's27', 's28']  
  
 for i in range(28):  
 position2 = springvecs2[i][1] - springvecs2[i][0]  
 spring2[i] = vp.cylinder(pos=springvecs2[i][0], axis=position2, length=vp.mag(position2), radius=.03,  
 color=vp.color.white)  
  
 for i in range(len(ballname1)):  
 ballname2[i].velocity = vp.vector(0, 0, 0)  
  
 ##   
  
 ballname3 = ['b11', 'b22', 'b33', 'b44', 'b55', 'b66', 'b77', 'b88']  
 ballvectors3 = [vp.vector(1, 0, 1), vp.vector(1, 1, 1), vp.vector(2, 1, 1), vp.vector(2, 0, 1),  
 vp.vector(1, 1, 2), vp.vector(1, 0, 2),  
 vp.vector(2, 1, 2), vp.vector(2, 0, 2)]  
 COM3 = (ballvectors3[0] + ballvectors3[1] + ballvectors3[2] + ballvectors3[3] + ballvectors3[4] + ballvectors3[  
 5] + ballvectors3[6] + ballvectors3[7]) / 8  
  
 springvecs3 = []  
 for i in range(len(ballname3)):  
 ballname3[i] = vp.sphere(pos=ballvectors3[i], radius=0.1, color=vp.color.red, velocity=vp.vector(0, 0, 0))  
 for z in itertools.combinations(ballvectors3, 2):  
 springvecs3.append(z)  
  
 spring3 = ['s1', 's2', 's3', 's4', 's5', 's6', 's7', 's8', 's9', 's10', 's11', 's12', 's13', 's14', 's15',  
 's16', 's17', 's18', 's19', 's20', 's21', 's22', 's23', 's24', 's25', 's26', 's27', 's28']  
  
 for i in range(28):  
 position3 = springvecs3[i][1] - springvecs3[i][0]  
 spring3[i] = vp.cylinder(pos=springvecs3[i][0], axis=position3, length=vp.mag(position3), radius=.03,  
 color=vp.color.white)  
  
 for i in range(len(ballname3)):  
 ballname3[i].velocity = vp.vector(0, 0, 0)  
  
 Finalcom = COM + COM1 + COM2 + COM3  
  
 elif randnum == 5:  
  
 ballname1 = ['b11', 'b22', 'b33', 'b44', 'b55', 'b66', 'b77', 'b88']  
 ballvectors1 = [vp.vector(1, 0, 0), vp.vector(1, 1, 0), vp.vector(1, 0, 1), vp.vector(1, 1, 1),  
 vp.vector(2, 0, 0), vp.vector(2, 1, 0),  
 vp.vector(2, 0, 1), vp.vector(2, 1, 1)]  
 COM1 = (ballvectors1[0] + ballvectors1[1] + ballvectors1[2] + ballvectors1[3] + ballvectors1[4] + ballvectors1[  
 5] + ballvectors1[6] + ballvectors1[7]) / 8  
  
 springvecs1 = []  
 for i in range(len(ballname1)):  
 ballname1[i] = vp.sphere(pos=ballvectors1[i], radius=0.1, color=vp.color.red, velocity=vp.vector(0, 0, 0))  
 for z in itertools.combinations(ballvectors1, 2):  
 springvecs1.append(z)  
  
 spring1 = ['s1', 's2', 's3', 's4', 's5', 's6', 's7', 's8', 's9', 's10', 's11', 's12', 's13', 's14', 's15',  
 's16', 's17', 's18', 's19', 's20', 's21', 's22', 's23', 's24', 's25', 's26', 's27', 's28']  
  
 for i in range(28):  
 position1 = springvecs1[i][1] - springvecs1[i][0]  
 spring1[i] = vp.cylinder(pos=springvecs1[i][0], axis=position1, length=vp.mag(position1), radius=.03,  
 color=vp.color.white)  
  
 for i in range(len(ballname1)):  
 ballname1[i].velocity = vp.vector(0, 0, 0)  
  
 ##  
  
 ballname2 = ['b11', 'b22', 'b33', 'b44', 'b55', 'b66', 'b77', 'b88']  
 ballvectors2 = [vp.vector(0, 0, 1), vp.vector(0, 1, 1), vp.vector(1, 0, 1), vp.vector(1, 1, 1),  
 vp.vector(0, 0, 2), vp.vector(0, 1, 2),  
 vp.vector(1, 0, 2), vp.vector(1, 1, 2)]  
 COM2 = (ballvectors2[0] + ballvectors2[1] + ballvectors2[2] + ballvectors2[3] + ballvectors2[4] + ballvectors2[  
 5] + ballvectors2[6] + ballvectors2[7]) / 8  
  
 springvecs2 = []  
 for i in range(len(ballname2)):  
 ballname2[i] = vp.sphere(pos=ballvectors2[i], radius=0.1, color=vp.color.red, velocity=vp.vector(0, 0, 0))  
 for z in itertools.combinations(ballvectors2, 2):  
 springvecs2.append(z)  
  
 spring2 = ['s1', 's2', 's3', 's4', 's5', 's6', 's7', 's8', 's9', 's10', 's11', 's12', 's13', 's14', 's15',  
 's16', 's17', 's18', 's19', 's20', 's21', 's22', 's23', 's24', 's25', 's26', 's27', 's28']  
  
 for i in range(28):  
 position2 = springvecs2[i][1] - springvecs2[i][0]  
 spring2[i] = vp.cylinder(pos=springvecs2[i][0], axis=position2, length=vp.mag(position2), radius=.03,  
 color=vp.color.white)  
  
 for i in range(len(ballname1)):  
 ballname2[i].velocity = vp.vector(0, 0, 0)  
  
 Finalcom = COM + COM1 + COM2  
  
 elif randnum == 6:  
  
 ballname1 = ['b11', 'b22', 'b33', 'b44', 'b55', 'b66', 'b77', 'b88']  
 ballvectors1 = [vp.vector(1, 0, 0), vp.vector(1, 1, 0), vp.vector(1, 0, 1), vp.vector(1, 1, 1),  
 vp.vector(2, 0, 0), vp.vector(2, 1, 0),  
 vp.vector(2, 0, 1), vp.vector(2, 1, 1)]  
 COM1 = (ballvectors1[0] + ballvectors1[1] + ballvectors1[2] + ballvectors1[3] + ballvectors1[4] + ballvectors1[  
 5] + ballvectors1[6] + ballvectors1[7]) / 8  
  
 springvecs1 = []  
 for i in range(len(ballname1)):  
 ballname1[i] = vp.sphere(pos=ballvectors1[i], radius=0.1, color=vp.color.red, velocity=vp.vector(0, 0, 0))  
 for z in itertools.combinations(ballvectors1, 2):  
 springvecs1.append(z)  
  
 spring1 = ['s1', 's2', 's3', 's4', 's5', 's6', 's7', 's8', 's9', 's10', 's11', 's12', 's13', 's14', 's15',  
 's16', 's17', 's18', 's19', 's20', 's21', 's22', 's23', 's24', 's25', 's26', 's27', 's28']  
  
 for i in range(28):  
 position1 = springvecs1[i][1] - springvecs1[i][0]  
 spring1[i] = vp.cylinder(pos=springvecs1[i][0], axis=position1, length=vp.mag(position1), radius=.03,  
 color=vp.color.white)  
  
 for i in range(len(ballname1)):  
 ballname1[i].velocity = vp.vector(0, 0, 0)  
  
 Finalcom = COM + COM1  
  
 ballname = ['b1', 'b2', 'b3', 'b4', 'b5', 'b6', 'b7', 'b8']  
 ballvectors = [vp.vector(0, 0, 0), vp.vector(0, 1, 0), vp.vector(0, 0, 1), vp.vector(1, 0, 0),  
 vp.vector(1, 1, 0), vp.vector(0, 1, 1),  
 vp.vector(1, 0, 1), vp.vector(1, 1, 1)]  
  
 # for i in range(8):  
 # ballvectors[i] = ballvectors[i].rotate(angle=3.14/4, axis =vp.vector(1,1,1))  
 OriginalCOM = (ballvectors[0] + ballvectors[1] + ballvectors[2] + ballvectors[3] + ballvectors[4] + ballvectors[  
 5] + ballvectors[6] + ballvectors[7]) / 8  
  
 springvecs = []  
 for i in range(len(ballname)):  
 ballname[i] = vp.sphere(pos=ballvectors[i], radius=0.1, color=vp.color.red, velocity=vp.vector(0, 0, 0))  
 for z in itertools.combinations(ballvectors, 2):  
 springvecs.append(z)  
  
 spring = ['s1', 's2', 's3', 's4', 's5', 's6', 's7', 's8', 's9', 's10', 's11', 's12', 's13', 's14', 's15', 's16',  
 's17', 's18', 's19', 's20', 's21', 's22', 's23', 's24', 's25', 's26', 's27', 's28']  
  
 for i in range(28):  
 position = springvecs[i][1] - springvecs[i][0]  
 spring[i] = vp.cylinder(pos=springvecs[i][0], axis=position, length=vp.mag(position), radius=.03,  
 color=vp.color.white)  
  
 g\_vector = vp.vector(0, 9.81, 0)  
 for i in range(len(ballname)):  
 ballname[i].velocity = vp.vector(0, 0, 0)  
  
 F\_c = vp.vector(0, 1000, 0)  
  
 L0 = np.zeros((8, 8))  
 for i in range(8):  
 for j in range(8):  
 if i == j:  
 L0[j][i] = 0  
 else:  
 position = ballname[j].pos - ballname[i].pos  
 L0[j][i] = vp.mag(position)  
  
 Repeated = []  
 total\_indices = []  
 for x in range(8):  
 for y in range(8):  
 value = L0[y][x]  
 if value not in Repeated:  
 Repeated.append(value)  
 indices = []  
 for n in range(8):  
 for m in range(8):  
 index = []  
 if L0[m][n] == value:  
 index.append(m)  
 index.append(n)  
 indices.append(index)  
 total\_indices.append(indices)  
 # print(len(Repeated))   
  
 del total\_indices[0]  
  
 t = 0.001  
 # t1 = 0  
 # Time = []  
 # E\_H = []  
 # E\_S = []  
 # E\_K = []  
 c = 1  
 w = 10 \* np.pi  
 while True:  
 # vp.rate(200)  
  
 L0rate = np.zeros((8, 8))  
 w = 10 \* np.pi  
 for i in range(len(total\_indices)):  
 for z in total\_indices[i]:  
 L0rate[z[0]][z[1]] = L0[z[0]][z[1]] + pa1[h][i][0] \* sin(w \* t + pa1[h][i][1])  
  
 # L0rate = np.zeros((8,8))  
 # L0rate[0][1] = L0[0][1]+pa1[h][0][0]\*sin(w\*t+pa1[h][0][1])  
 # L0rate[1][0] = L0[1][0]+pa1[h][0][0]\*sin(w\*t+pa1[h][0][1])  
 # L0rate[0][2] = L0[0][2]+pa1[h][1][0]\*sin(w\*t+pa1[h][1][1])  
 # L0rate[2][0] = L0[2][0]+pa1[h][1][0]\*sin(w\*t+pa1[h][1][1])  
 # L0rate[0][3] = L0[0][3]+pa1[h][2][0]\*sin(w\*t+pa1[h][2][1])  
 # L0rate[3][0] = L0[3][0]+pa1[h][2][0]\*sin(w\*t+pa1[h][2][1])  
 # L0rate[0][4] = L0[0][4]+pa1[h][3][0]\*sin(w\*t+pa1[h][3][1])  
 # L0rate[4][0] = L0[4][0]+pa1[h][3][0]\*sin(w\*t+pa1[h][3][1])  
 # L0rate[0][5] = L0[0][5]+pa1[h][4][0]\*sin(w\*t+pa1[h][4][1])  
 # L0rate[5][0] = L0[5][0]+pa1[h][4][0]\*sin(w\*t+pa1[h][4][1])  
 # L0rate[0][6] = L0[0][6]+pa1[h][5][0]\*sin(w\*t+pa1[h][5][1])  
 # L0rate[6][0] = L0[6][0]+pa1[h][5][0]\*sin(w\*t+pa1[h][5][1])  
 # L0rate[0][7] = L0[0][7]+pa1[h][6][0]\*sin(w\*t+pa1[h][6][1])  
 # L0rate[7][0] = L0[7][0]+pa1[h][6][0]\*sin(w\*t+pa1[h][6][1])  
 # L0rate[1][2] = L0[1][2]+pa1[h][7][0]\*sin(w\*t+pa1[h][7][1])  
 # L0rate[2][1] = L0[2][1]+pa1[h][7][0]\*sin(w\*t+pa1[h][7][1])  
 # L0rate[1][3] = L0[1][3]+pa1[h][8][0]\*sin(w\*t+pa1[h][8][1])  
 # L0rate[3][1] = L0[3][1]+pa1[h][8][0]\*sin(w\*t+pa1[h][8][1])  
 # L0rate[1][4] = L0[1][4]+pa1[h][9][0]\*sin(w\*t+pa1[h][9][1])  
 # L0rate[4][1] = L0[4][1]+pa1[h][9][0]\*sin(w\*t+pa1[h][9][1])  
 # L0rate[1][5] = L0[1][5]+pa1[h][10][0]\*sin(w\*t+pa1[h][10][1])  
 # L0rate[5][1] = L0[5][1]+pa1[h][10][0]\*sin(w\*t+pa1[h][10][1])  
 # L0rate[1][6] = L0[1][6]+pa1[h][11][0]\*sin(w\*t+pa1[h][11][1])  
 # L0rate[6][1] = L0[6][1]+pa1[h][11][0]\*sin(w\*t+pa1[h][11][1])  
 # L0rate[1][7] = L0[1][7]+pa1[h][12][0]\*sin(w\*t+pa1[h][12][1])  
 # L0rate[7][1] = L0[7][1]+pa1[h][12][0]\*sin(w\*t+pa1[h][12][1])  
 # L0rate[2][3] = L0[2][3]+pa1[h][13][0]\*sin(w\*t+pa1[h][13][1])  
 # L0rate[3][2] = L0[3][2]+pa1[h][13][0]\*sin(w\*t+pa1[h][13][1])  
 # L0rate[2][4] = L0[2][4]+pa1[h][14][0]\*sin(w\*t+pa1[h][14][1])  
 # L0rate[4][2] = L0[4][2]+pa1[h][14][0]\*sin(w\*t+pa1[h][14][1])  
 # L0rate[2][5] = L0[2][5]+pa1[h][15][0]\*sin(w\*t+pa1[h][15][1])  
 # L0rate[5][2] = L0[5][2]+pa1[h][15][0]\*sin(w\*t+pa1[h][15][1])  
 # L0rate[2][6] = L0[2][6]+pa1[h][16][0]\*sin(w\*t+pa1[h][16][1])  
 # L0rate[6][2] = L0[6][2]+pa1[h][16][0]\*sin(w\*t+pa1[h][16][1])  
 # L0rate[2][7] = L0[2][7]+pa1[h][17][0]\*sin(w\*t+pa1[h][17][1])  
 # L0rate[7][2] = L0[7][2]+pa1[h][17][0]\*sin(w\*t+pa1[h][17][1])  
 # L0rate[3][4] = L0[3][4]+pa1[h][18][0]\*sin(w\*t+pa1[h][18][1])  
 # L0rate[4][3] = L0[4][3]+pa1[h][18][0]\*sin(w\*t+pa1[h][18][1])  
 # L0rate[3][5] = L0[3][5]+pa1[h][19][0]\*sin(w\*t+pa1[h][19][1])  
 # L0rate[5][3] = L0[5][3]+pa1[h][19][0]\*sin(w\*t+pa1[h][19][1])  
 # L0rate[3][6] = L0[3][6]+pa1[h][20][0]\*sin(w\*t+pa1[h][20][1])  
 # L0rate[6][3] = L0[6][3]+pa1[h][20][0]\*sin(w\*t+pa1[h][20][1])  
 # L0rate[3][7] = L0[3][7]+pa1[h][21][0]\*sin(w\*t+pa1[h][21][1])  
 # L0rate[7][3] = L0[7][3]+pa1[h][21][0]\*sin(w\*t+pa1[h][21][1])  
 # L0rate[4][5] = L0[4][5]+pa1[h][22][0]\*sin(w\*t+pa1[h][22][1])  
 # L0rate[5][4] = L0[5][4]+pa1[h][22][0]\*sin(w\*t+pa1[h][22][1])  
 # L0rate[4][6] = L0[4][6]+pa1[h][23][0]\*sin(w\*t+pa1[h][23][1])  
 # L0rate[6][4] = L0[6][4]+pa1[h][23][0]\*sin(w\*t+pa1[h][23][1])  
 # L0rate[4][7] = L0[4][7]+pa1[h][24][0]\*sin(w\*t+pa1[h][24][1])  
 # L0rate[7][4] = L0[7][4]+pa1[h][24][0]\*sin(w\*t+pa1[h][24][1])  
 # L0rate[5][6] = L0[5][6]+pa1[h][25][0]\*sin(w\*t+pa1[h][25][1])  
 # L0rate[6][5] = L0[6][5]+pa1[h][25][0]\*sin(w\*t+pa1[h][25][1])  
 # L0rate[5][7] = L0[5][7]+pa1[h][26][0]\*sin(w\*t+pa1[h][26][1])  
 # L0rate[7][5] = L0[7][5]+pa1[h][26][0]\*sin(w\*t+pa1[h][26][1])  
 # L0rate[6][7] = L0[6][7]+pa1[h][27][0]\*sin(w\*t+pa1[h][27][1])  
 # L0rate[7][6] = L0[7][6]+pa1[h][27][0]\*sin(w\*t+pa1[h][27][1])  
  
 ks = np.zeros((8, 8))  
 ks[0][1] = pa1[h][0][2]  
 ks[1][0] = pa1[h][0][2]  
 ks[0][2] = pa1[h][1][2]  
 ks[2][0] = pa1[h][1][2]  
 ks[0][3] = pa1[h][2][2]  
 ks[3][0] = pa1[h][2][2]  
 ks[0][4] = pa1[h][3][2]  
 ks[4][0] = pa1[h][3][2]  
 ks[0][5] = pa1[h][4][2]  
 ks[5][0] = pa1[h][4][2]  
 ks[0][6] = pa1[h][5][2]  
 ks[6][0] = pa1[h][5][2]  
 ks[0][7] = pa1[h][6][2]  
 ks[7][0] = pa1[h][6][2]  
 ks[1][2] = pa1[h][7][2]  
 ks[2][1] = pa1[h][7][2]  
 ks[1][3] = pa1[h][8][2]  
 ks[3][1] = pa1[h][8][2]  
 ks[1][4] = pa1[h][9][2]  
 ks[4][1] = pa1[h][9][2]  
 ks[1][5] = pa1[h][10][2]  
 ks[5][1] = pa1[h][10][2]  
 ks[1][6] = pa1[h][11][2]  
 ks[6][1] = pa1[h][11][2]  
 ks[1][7] = pa1[h][12][2]  
 ks[7][1] = pa1[h][12][2]  
 ks[2][3] = pa1[h][13][2]  
 ks[3][2] = pa1[h][13][2]  
 ks[2][4] = pa1[h][14][2]  
 ks[4][2] = pa1[h][14][2]  
 ks[2][5] = pa1[h][15][2]  
 ks[5][2] = pa1[h][15][2]  
 ks[2][6] = pa1[h][16][2]  
 ks[6][2] = pa1[h][16][2]  
 ks[2][7] = pa1[h][17][2]  
 ks[7][2] = pa1[h][17][2]  
 ks[3][4] = pa1[h][18][2]  
 ks[4][3] = pa1[h][18][2]  
 ks[3][5] = pa1[h][19][2]  
 ks[5][3] = pa1[h][19][2]  
 ks[3][6] = pa1[h][20][2]  
 ks[6][3] = pa1[h][20][2]  
 ks[3][7] = pa1[h][21][2]  
 ks[7][3] = pa1[h][21][2]  
 ks[4][5] = pa1[h][22][2]  
 ks[5][4] = pa1[h][22][2]  
 ks[4][6] = pa1[h][23][2]  
 ks[6][4] = pa1[h][23][2]  
 ks[4][7] = pa1[h][24][2]  
 ks[7][4] = pa1[h][24][2]  
 ks[5][6] = pa1[h][25][2]  
 ks[6][5] = pa1[h][25][2]  
 ks[5][7] = pa1[h][26][2]  
 ks[7][5] = pa1[h][26][2]  
 ks[6][7] = pa1[h][27][2]  
 ks[7][6] = pa1[h][27][2]  
 # print(L0rate)  
 t += 0.001  
 for i in range(8):  
 ballvectors[i] = ballname[i].pos  
 springvecs = []  
 for z in itertools.combinations(ballvectors, 2):  
 springvecs.append(z)  
 for i in range(28):  
 position = springvecs[i][1] - springvecs[i][0]  
 spring[i].pos = springvecs[i][0]  
 spring[i].axis = position  
 spring[i].length = vp.mag(position)  
  
 dampening = 0.99  
  
 F\_mat = np.zeros((8, 8))  
 F\_vec = []  
 F\_v = []  
 a = np.array(np.zeros((8, 8)))  
  
 for i in range(8):  
 for k in range(8):  
 if k == i:  
 L = 0  
 F\_mat[i][k] = 0  
 F\_vec.append(vp.vector(0, 0, 0))  
 else:  
 L = vp.mag(ballname[k].pos - ballname[i].pos) - L0rate[k][i]  
 # E\_s.append(1/2\*k\_sp\*L\*\*2)  
 F\_mat[i][k] = L \* ks[k][i]  
 pf0 = ballname[k].pos - ballname[i].pos  
 # a[i,k] = vp.norm(pf0)\*L\*k\_sp  
 F\_vec.append(vp.norm(pf0) \* L \* ks[k][i])  
 # E\_S.append(sum(E\_s)/2)  
 a = np.array(F\_vec).reshape(8, 8)  
 F = a.sum(axis=0)  
  
 for i in range(8):  
 F[i] = F[i] + g\_vector \* mass  
 if ballname[i].pos.y < floor.pos.y:  
 F\_N = ((floor.pos.y - ballname[i].pos.y) \*\* 2) \* 800  
 F[i].y = F[i].y - F\_N  
 mu = 1  
 F\_st = mu \* F\_N  
 F\_horiz = (F[i].x \*\* 2 + F[i].z \*\* 2) \*\* 0.5  
 v\_xz = (ballname[i].velocity.x \*\* 2 + ballname[i].velocity.z \*\* 2) \*\* 0.5  
 vx = ballname[i].velocity.x / v\_xz  
 vz = ballname[i].velocity.z / v\_xz  
 if F\_st < F\_horiz:  
 F[i].x += F\_horiz \* vx - F\_N \* vx  
 F[i].z += F\_horiz \* vz - F\_N \* vz  
 else:  
 F[i].x = F\_horiz \* vx  
 F[i].z = F\_horiz \* vz  
 ballname[i].velocity.x = 0  
 ballname[i].velocity.z = 0  
  
 for i in range(8):  
 ballname[i].velocity -= (F[i] / mass \* dt) \* dampening  
 ballname[i].pos += ballname[i].velocity \* dt  
  
 c += 1  
 if c == 6000:  
 break  
  
 # Calculating COM  
 COM = getCOM(ballname)  
 dvec = COM - OriginalCOM  
 dis = sqrt(dvec.x \*\* 2 + dvec.z \*\* 2)  
 # print(dis)  
 total\_dis.append(dis)  
 dis\_index = np.argsort(total\_dis)  
 sorted\_dis = []  
 sorted\_pa1 = []  
 for i in range(20):  
 sorted\_dis.append(total\_dis[dis\_index[i]])  
 sorted\_pa1.append(pa1[dis\_index[i]])  
 good\_dis = sorted\_dis[-10:]  
 dots.append(good\_dis)  
 print('GOODDIS', good\_dis[-1])  
 best\_dis.append(good\_dis[-1])  
 pa1 = sorted\_pa1[-10:]  
 print('PA1END', len(pa1))  
  
  
# if dis > dis1:   
# dis1 = dis  
# good\_dis.append(dis1)  
  
print(best\_dis)  
print(pa1[-1])  
evals = list(range(1, 101))  
  
plt.plot(evals, best\_dis)  
plt.xlabel('Evaluations')  
plt.ylabel('Distance Moved')  
plt.title('Evolutionary Algorithm')  
plt.show()  
  
plt.plot(evals, dots)  
plt.xlabel('Evaluations')  
plt.ylabel('Distance Moved')  
plt.title('Evolutionary Algorithm Dot Plot')  
plt.show()