


STRIKE MACHINE



Alexandre LAURON

n° 42491

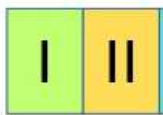
2023/2024

A close-up, low-angle shot of a robotic arm on a bowling lane. The arm is white and black, with a green bowling ball visible in the background. The lane is made of light-colored wood. In the background, there are blurred lights and structures, suggesting a bowling alley setting.

**Comment régler
un bras robotisé
pour
réussir un strike
à tous les coups ?**

S O M M A I R E

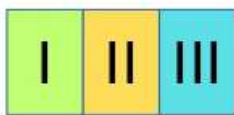
I – ANGLE D'ATTAQUE



S O M M A I R E

I – ANGLE D'ATTAQUE

II – TRAJECTOIRE D'UNE BOULE HOMOGÈNE



S O M M A I R E

I – ANGLE D'ATTAQUE

II – TRAJECTOIRE D'UNE BOULE HOMOGENE

III – TRAJECTOIRE D'UNE BOULE HETEROGENE



S O M M A I R E

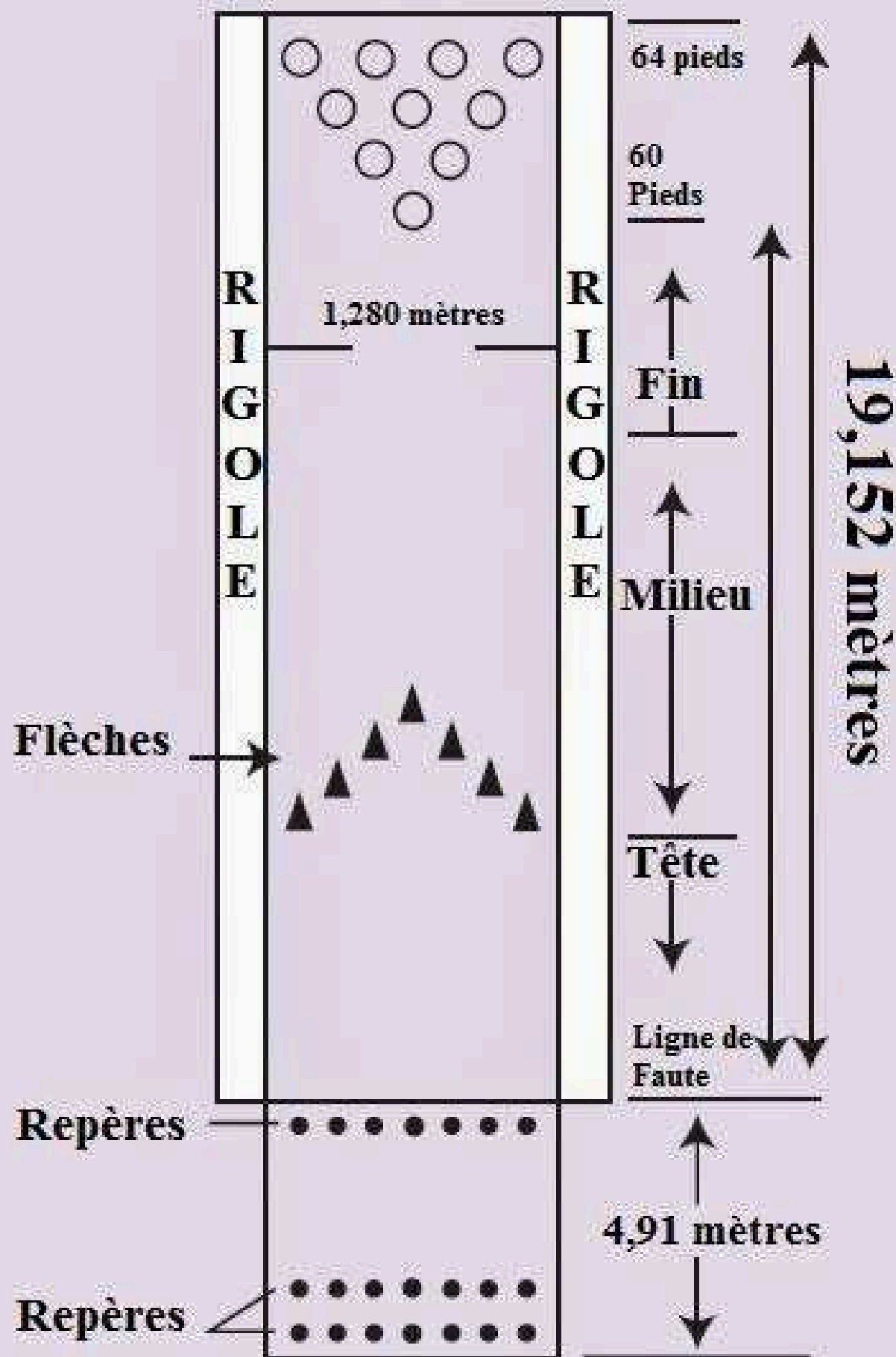
I – ANGLE D'ATTAQUE

II – TRAJECTOIRE D'UNE BOULE HOMOGENE

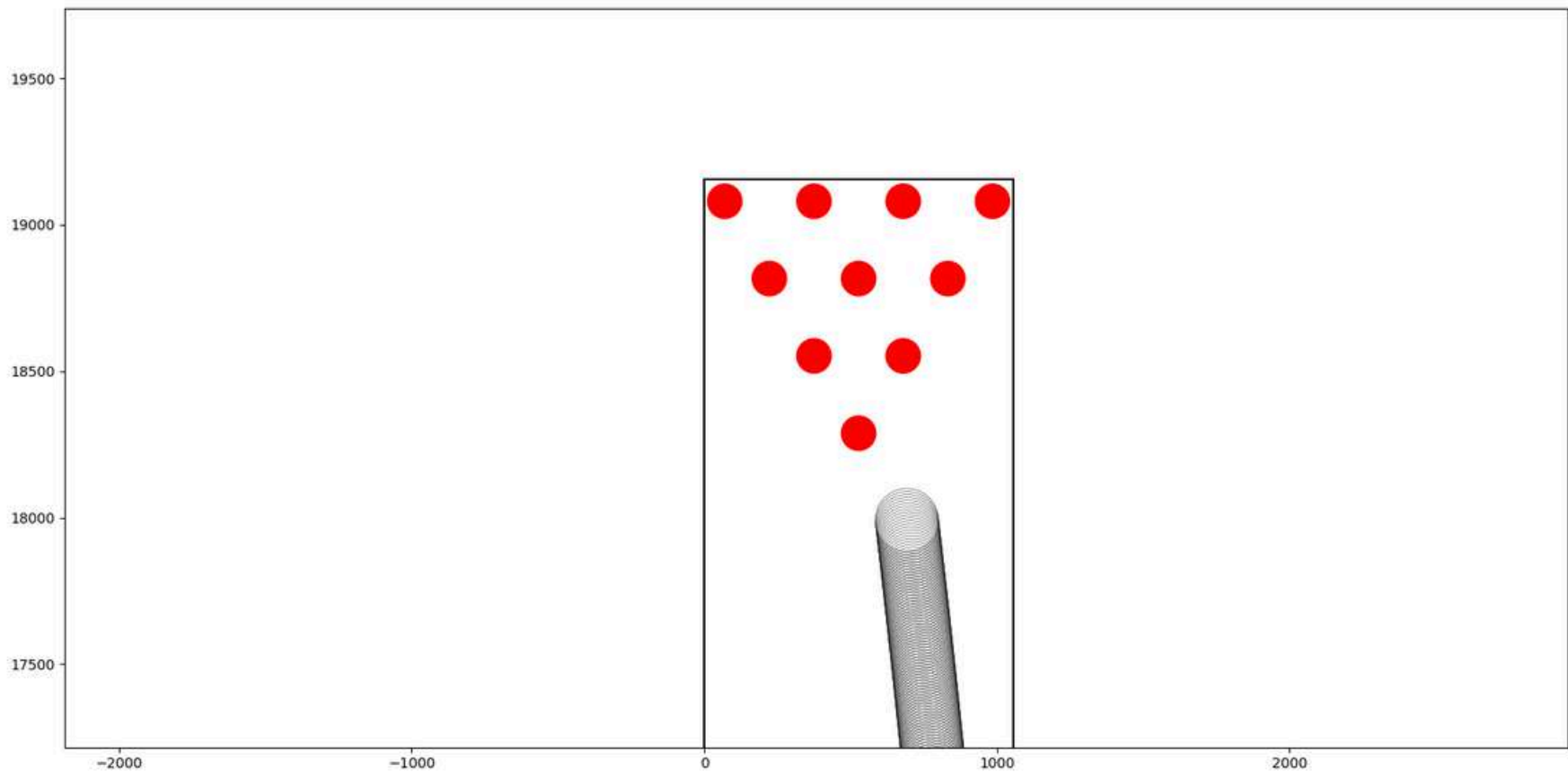
III – TRAJECTOIRE D'UNE BOULE HETEROGENE

IV – HUILAGE ET NOMBRE DE QUILLES

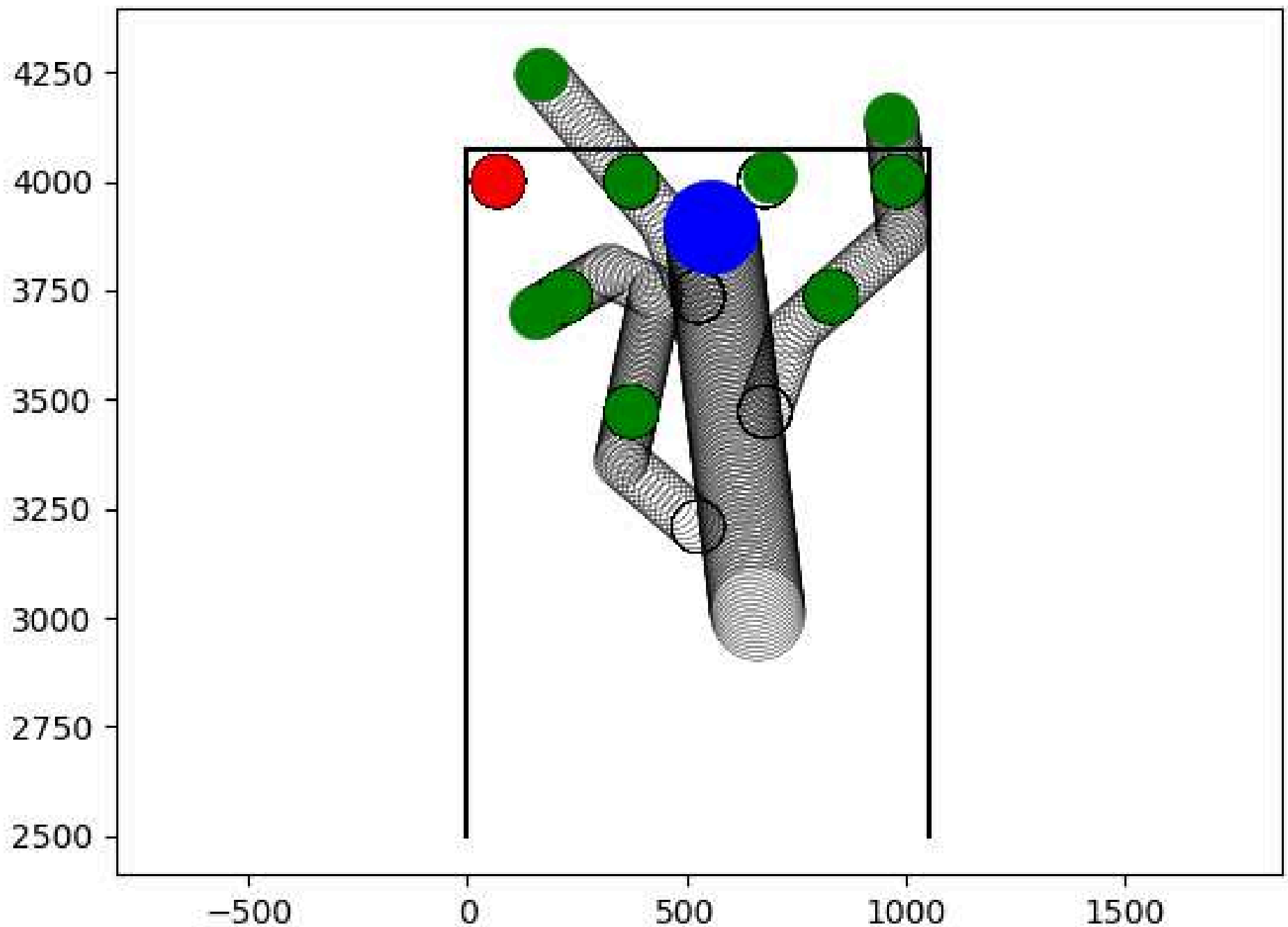
LE JEU



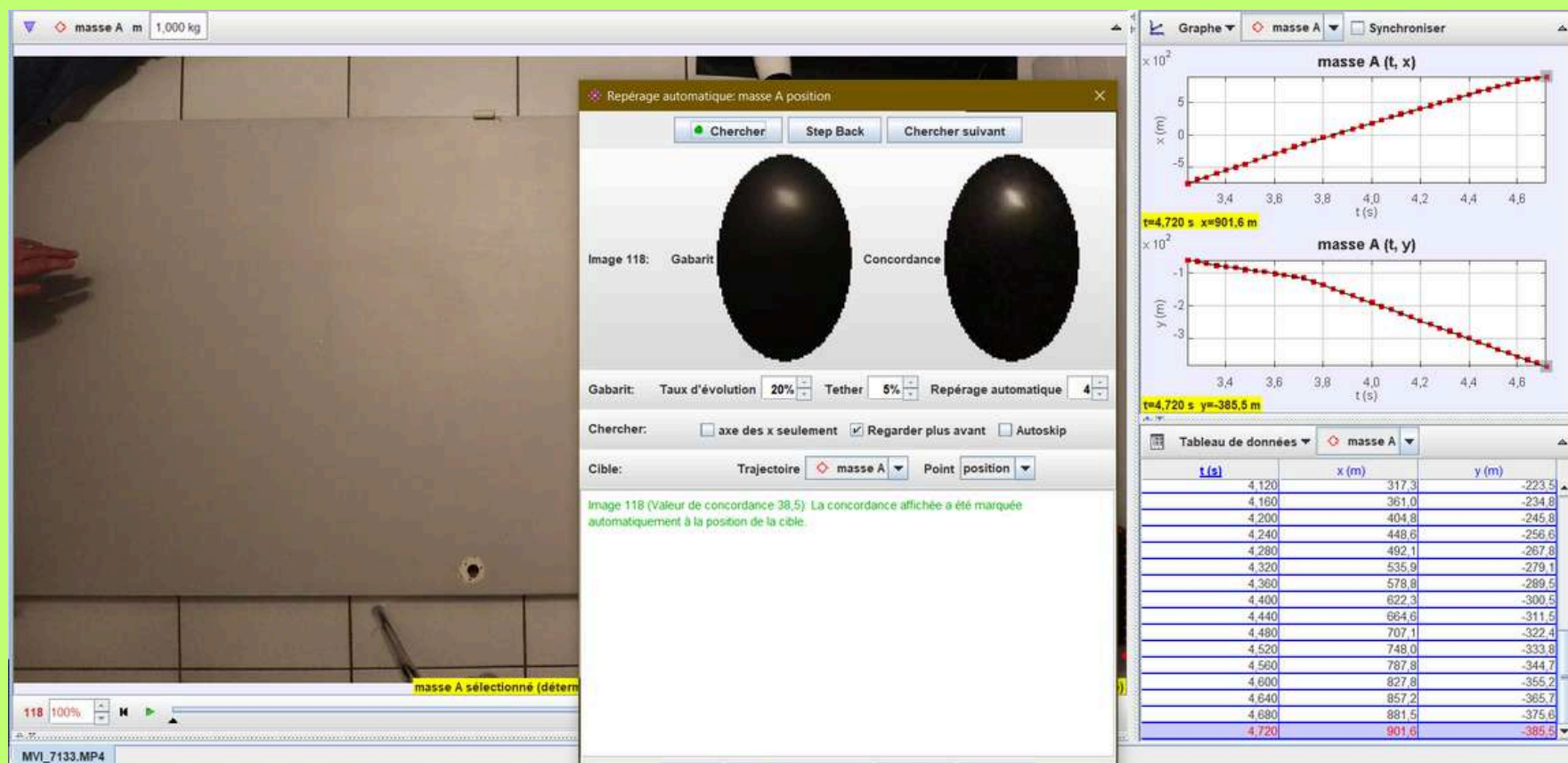
Le meilleur angle d'arrivée



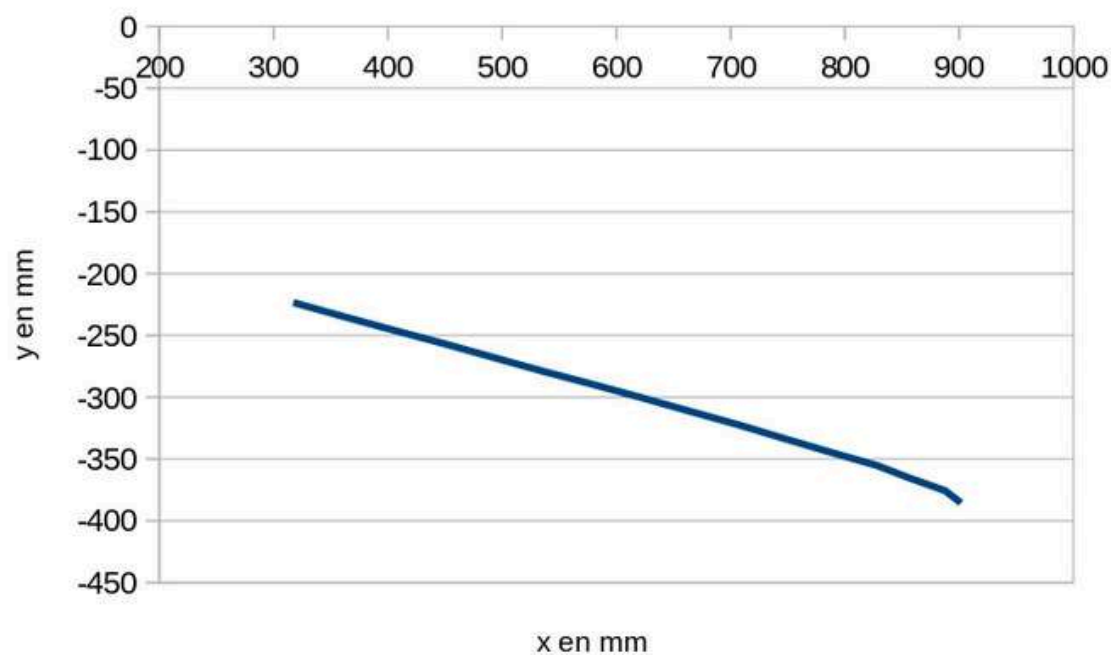
Simulations informatiques



Etude du choc boule - quille



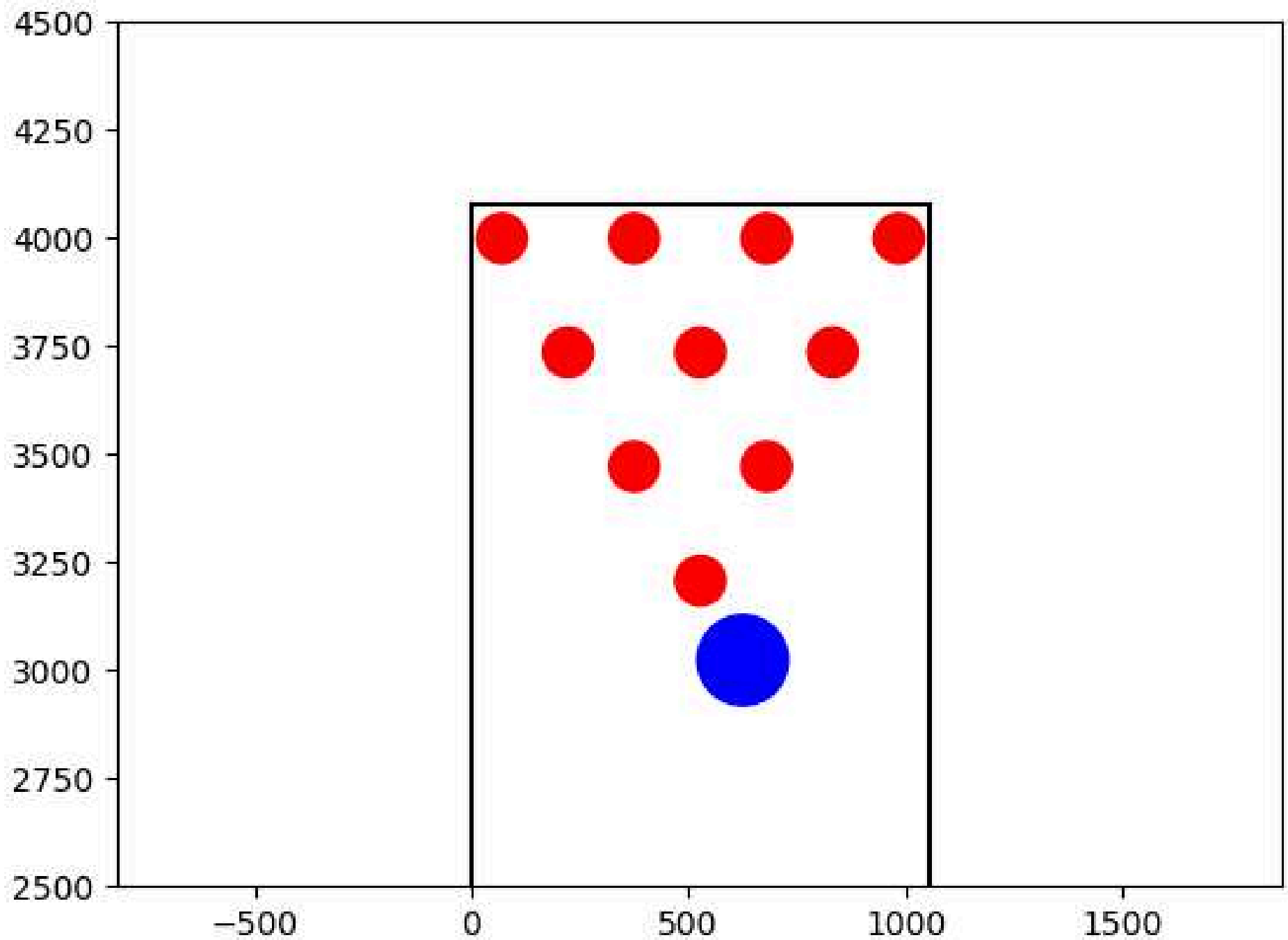
Trajectoire de la boule lors du choc avec une quille

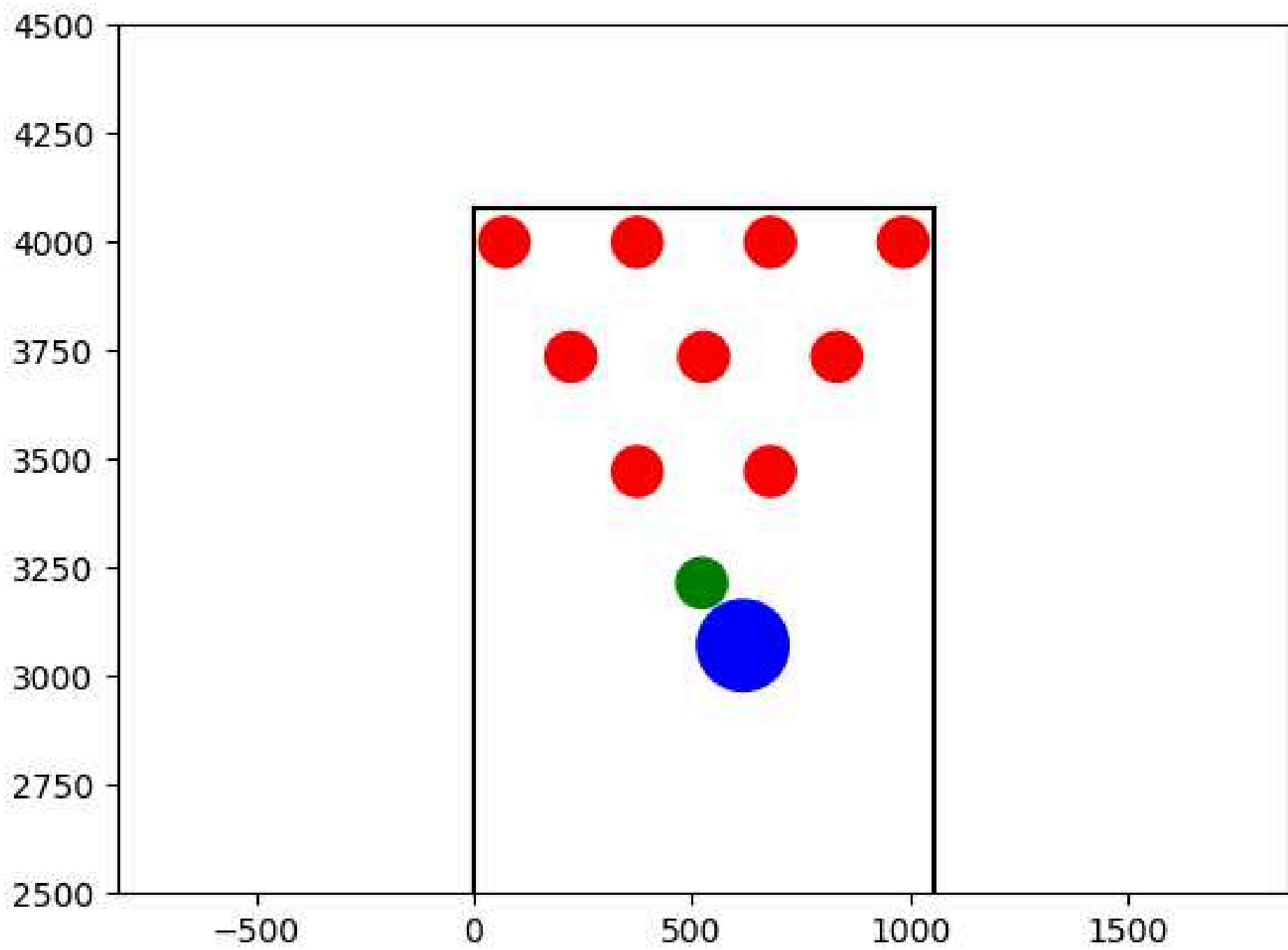


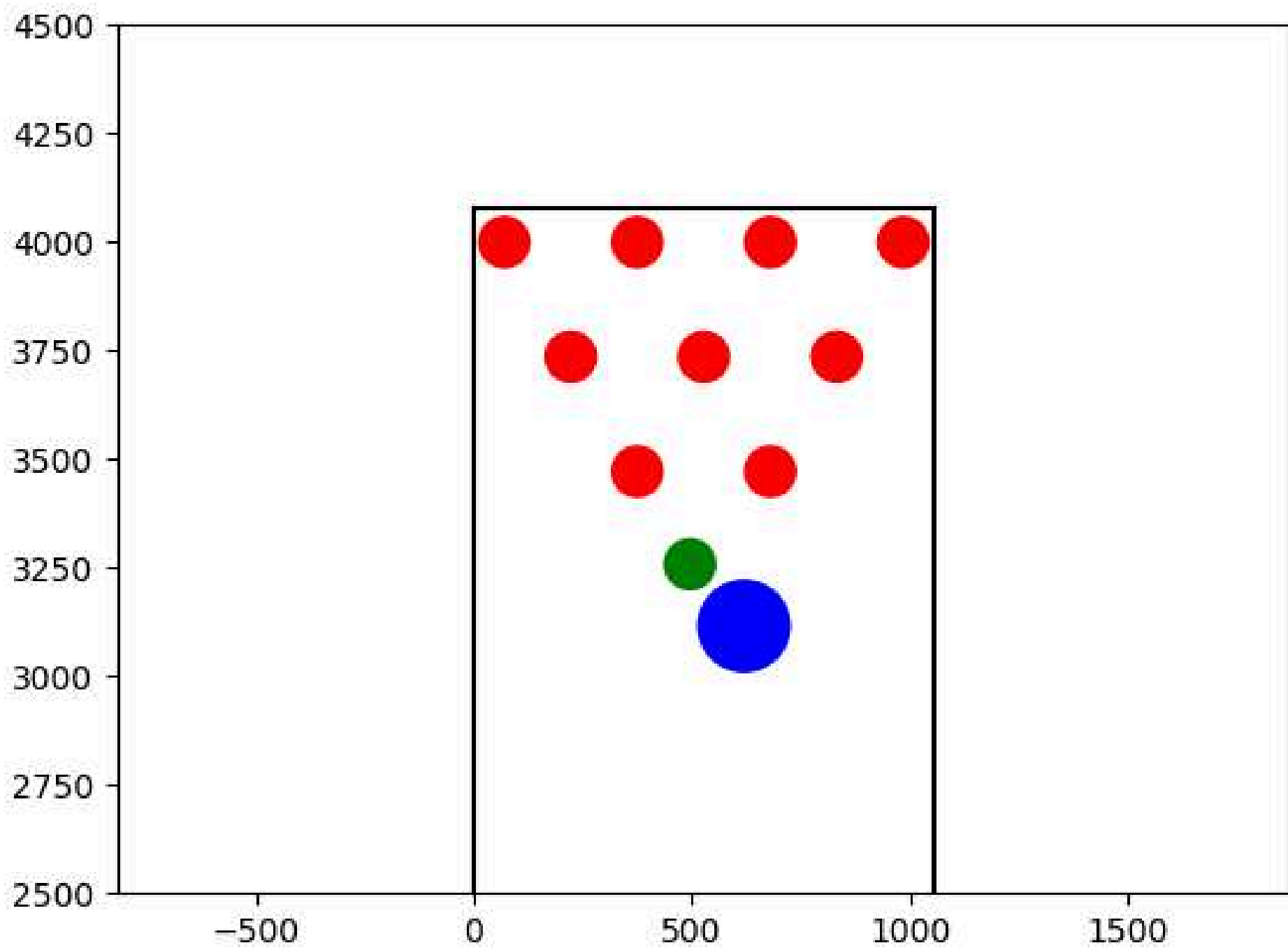
L'angle d'attaque idéal

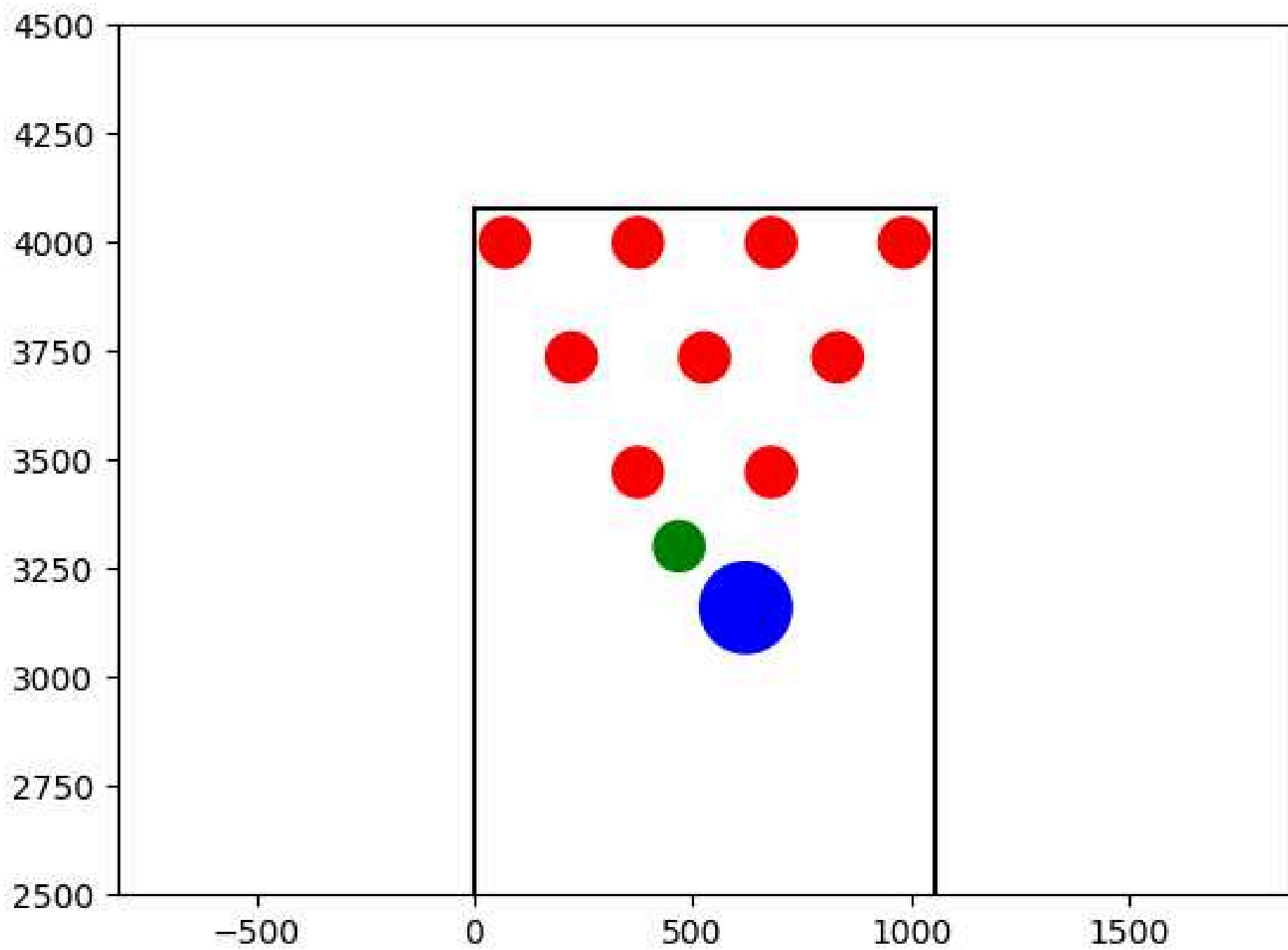
valeur de q	cumul quilles	angle	angle
pente -q/100	Tombées/150 tirs	en radians	en degrés
0	1277	0	0
1	1271	0,01	0,57
2	1264	0,02	1,14
3	1259	0,03	1,71
4	1267	0,04	2,29
5	1277	0,05	2,86
6	1276	0,06	3,43
7	1297	0,07	4
8	1293	0,08	4,57
9	1290	0,09	5,14
10	1293	0,1	5,71
11	1314	0,11	6,27
12	1310	0,12	6,84
13	1302	0,13	7,41
14	1302	0,14	7,96
15	1290	0,15	8,53
16	1293	0,16	9,09
17	1311	0,17	9,64
18	1294	0,18	10,2
19	1286	0,19	10,75

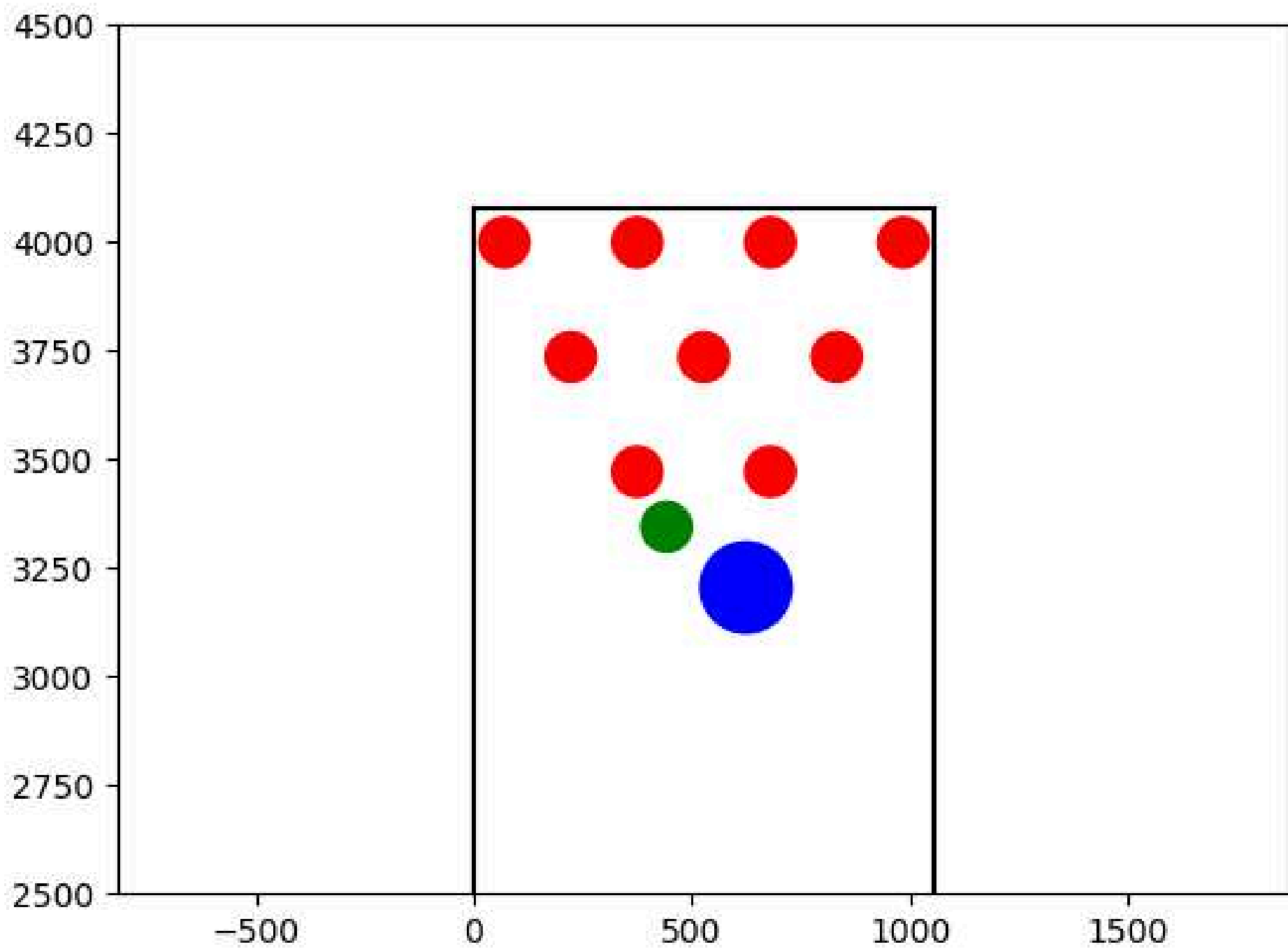
Simulation du strike - approche à 6°

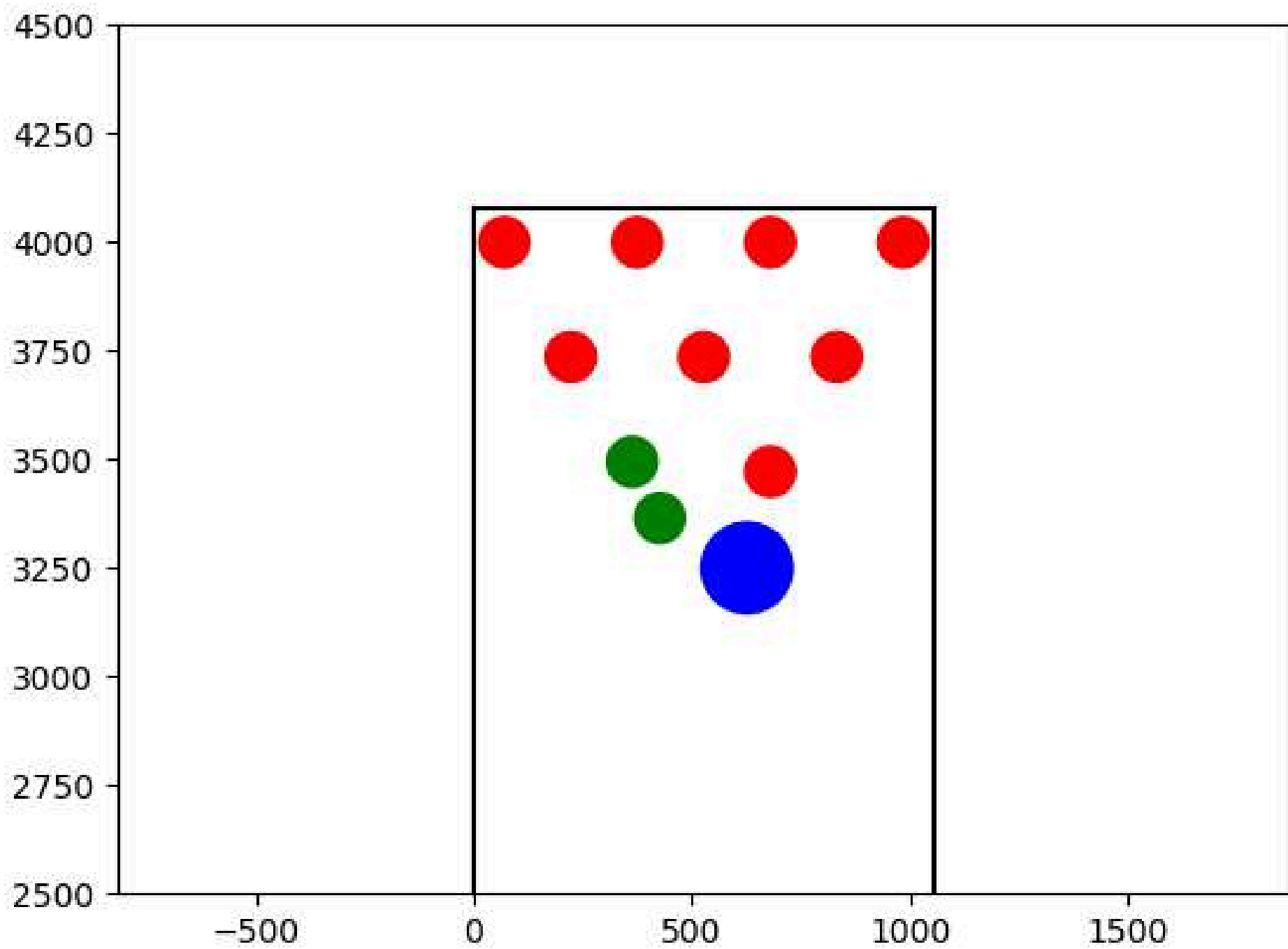


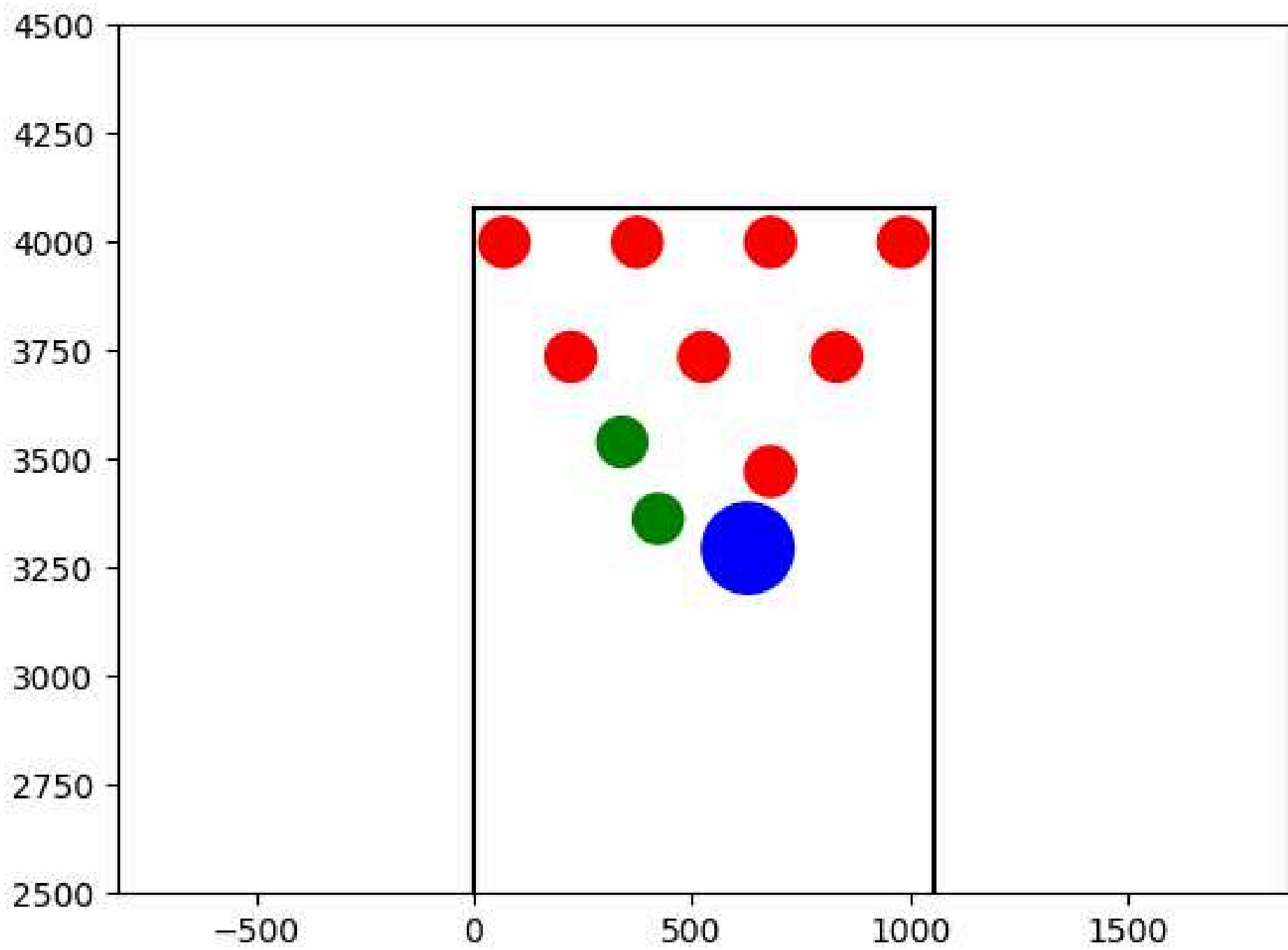


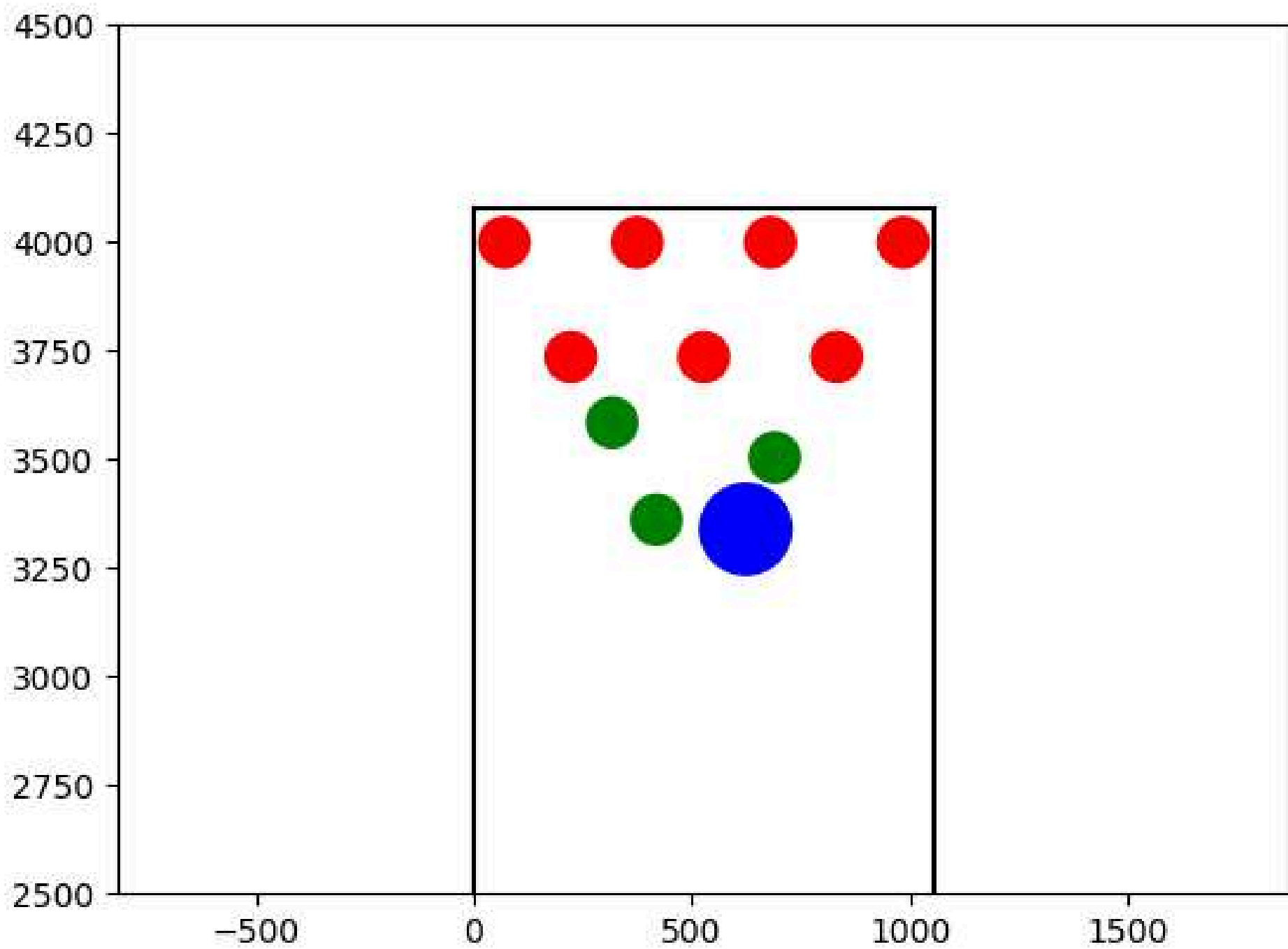


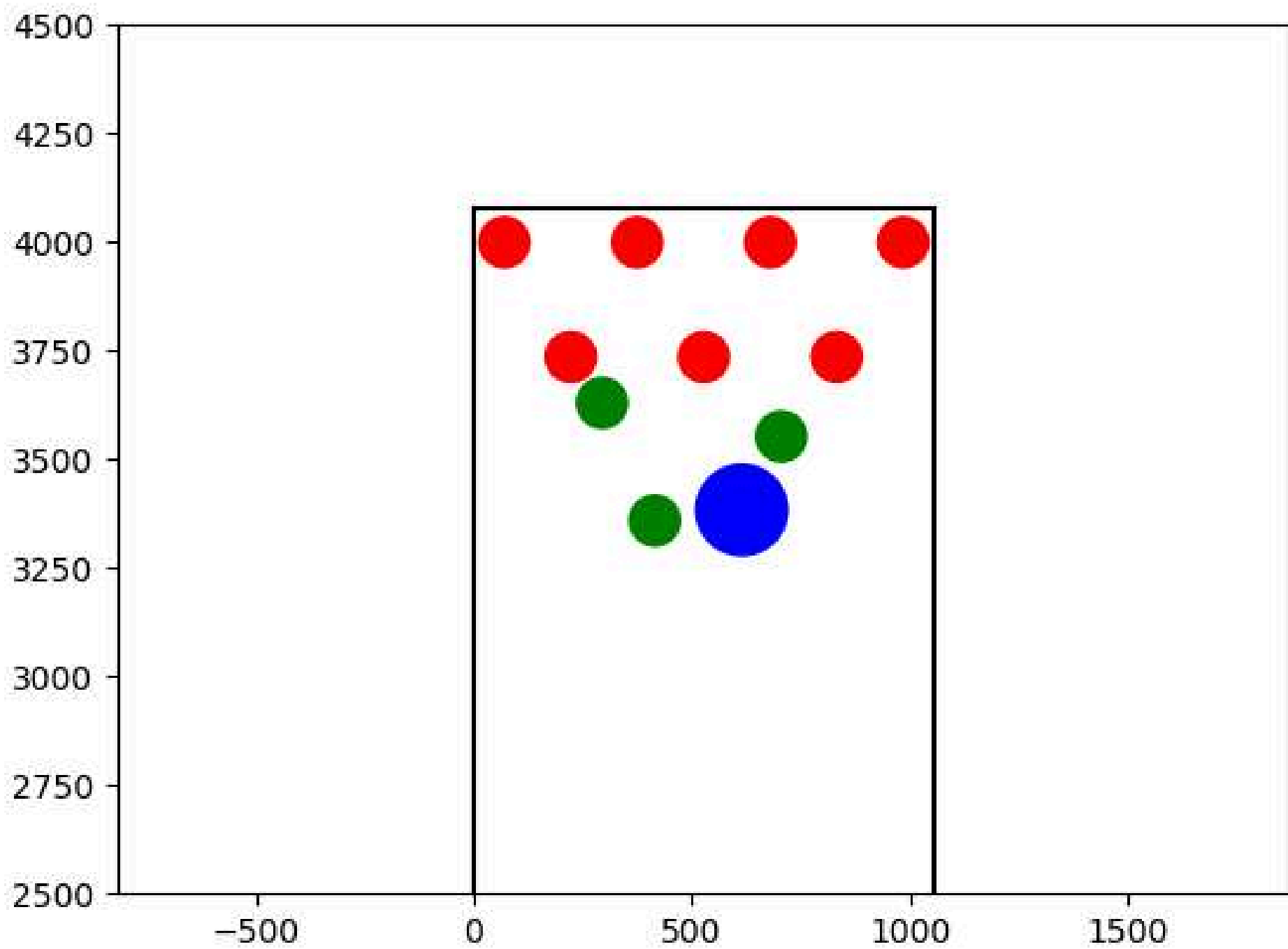


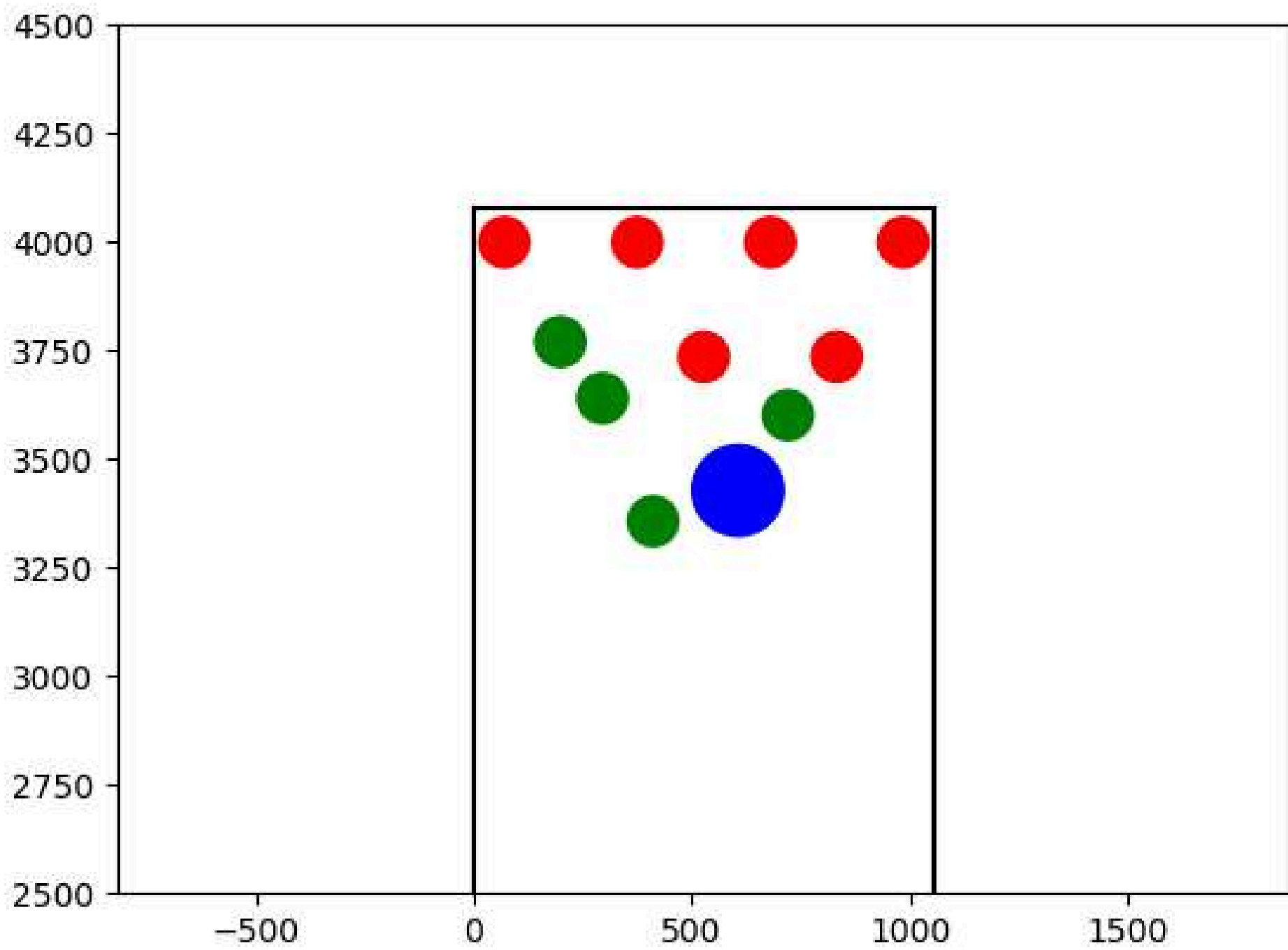


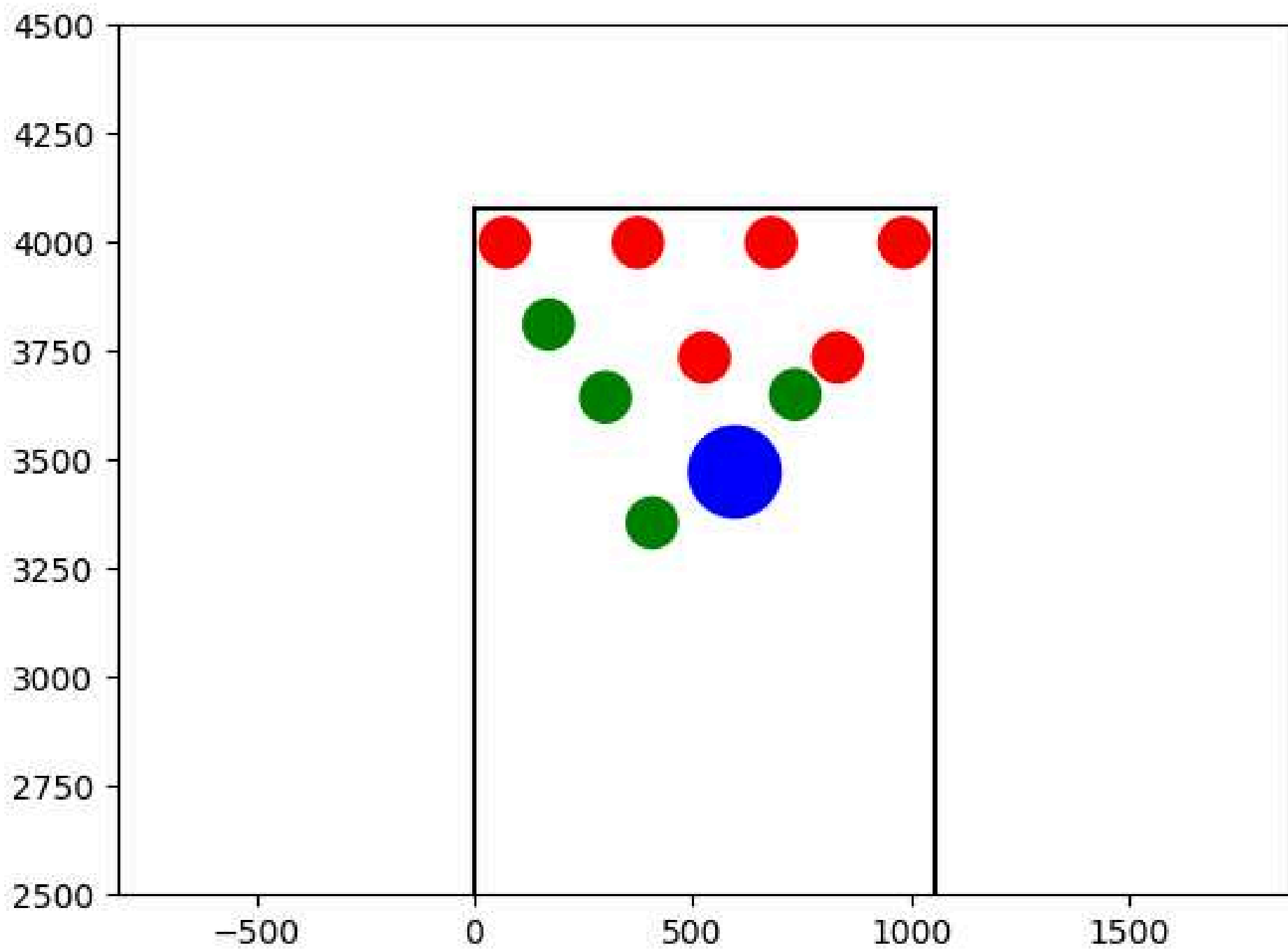


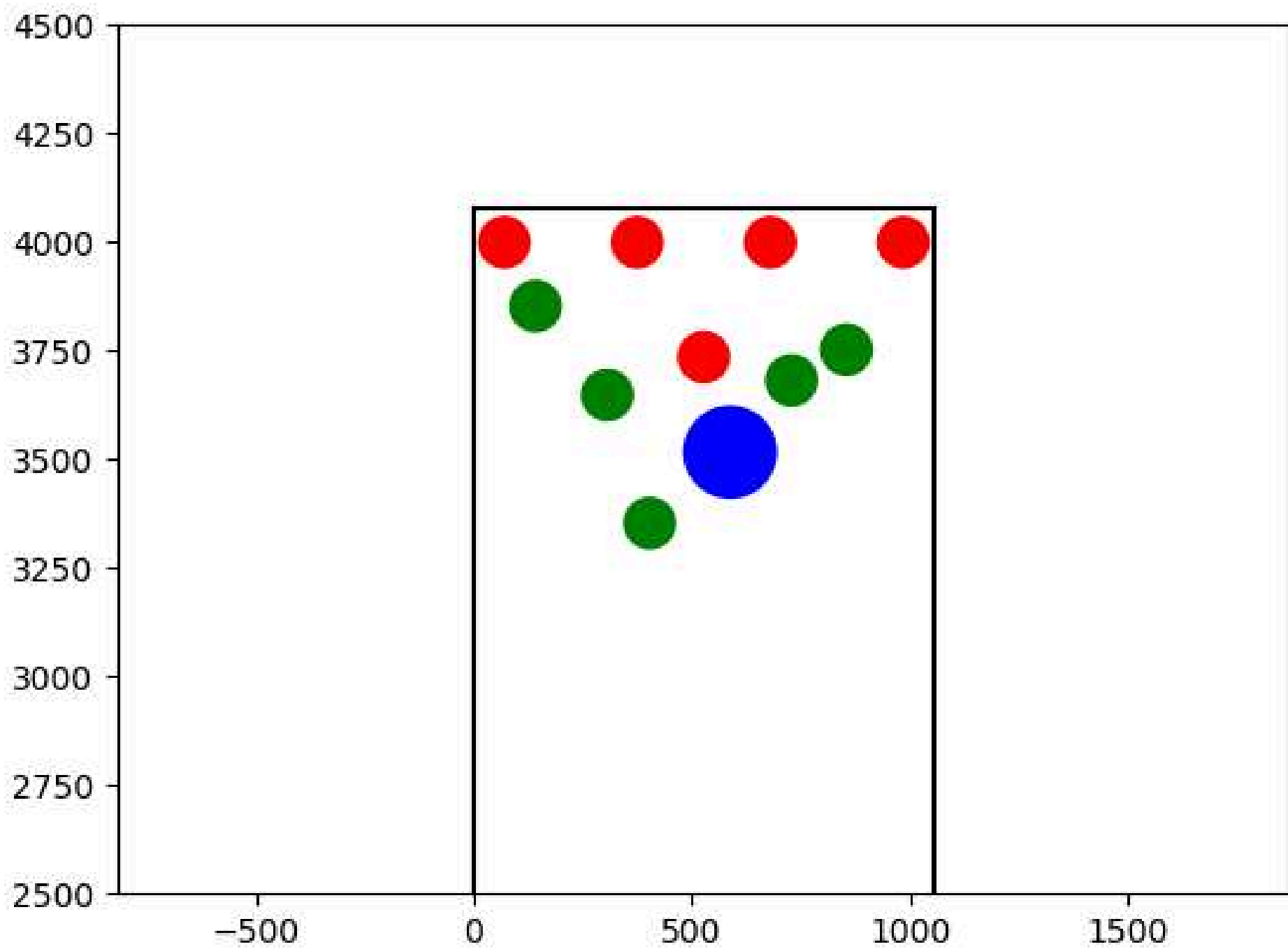


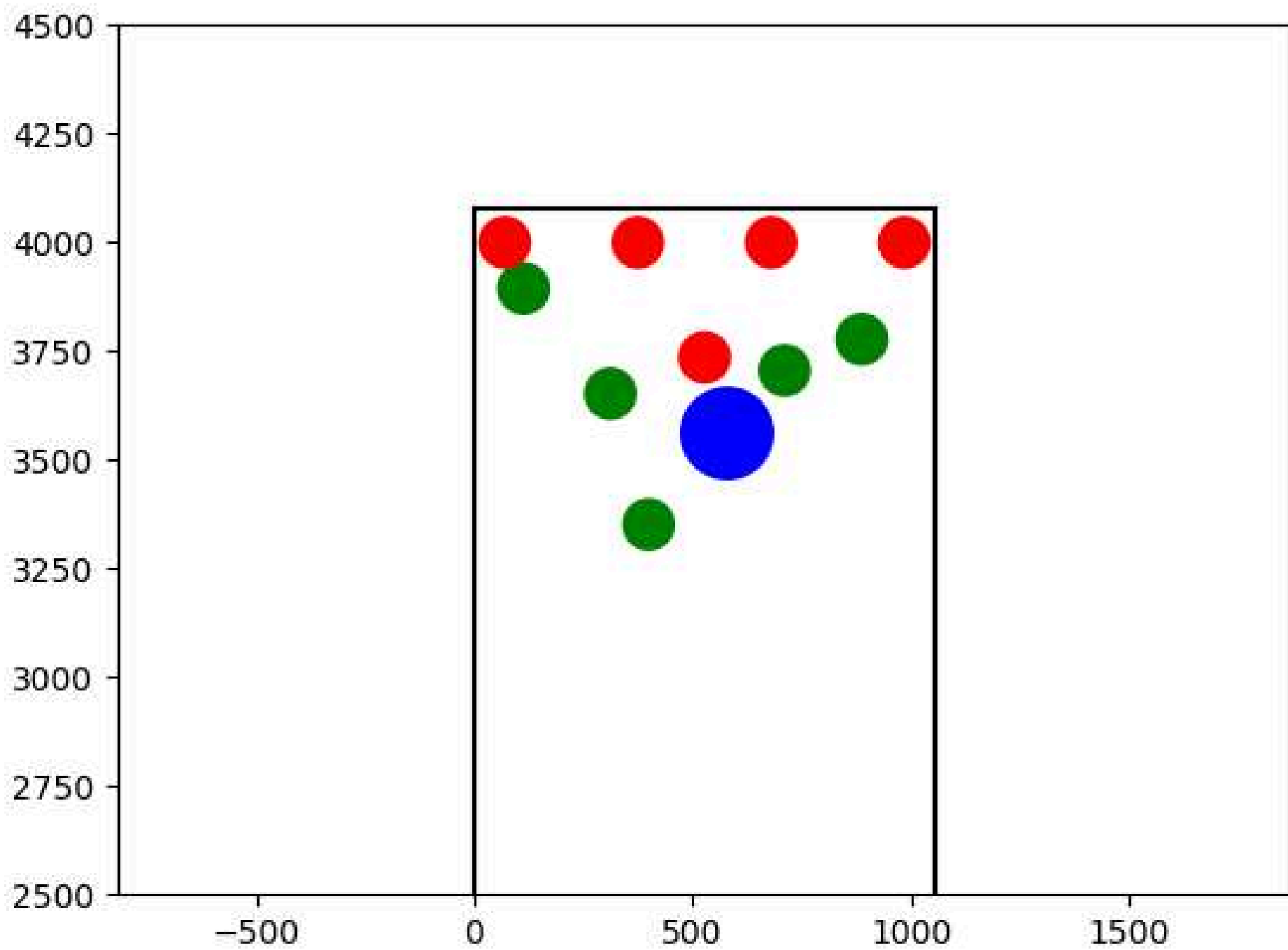


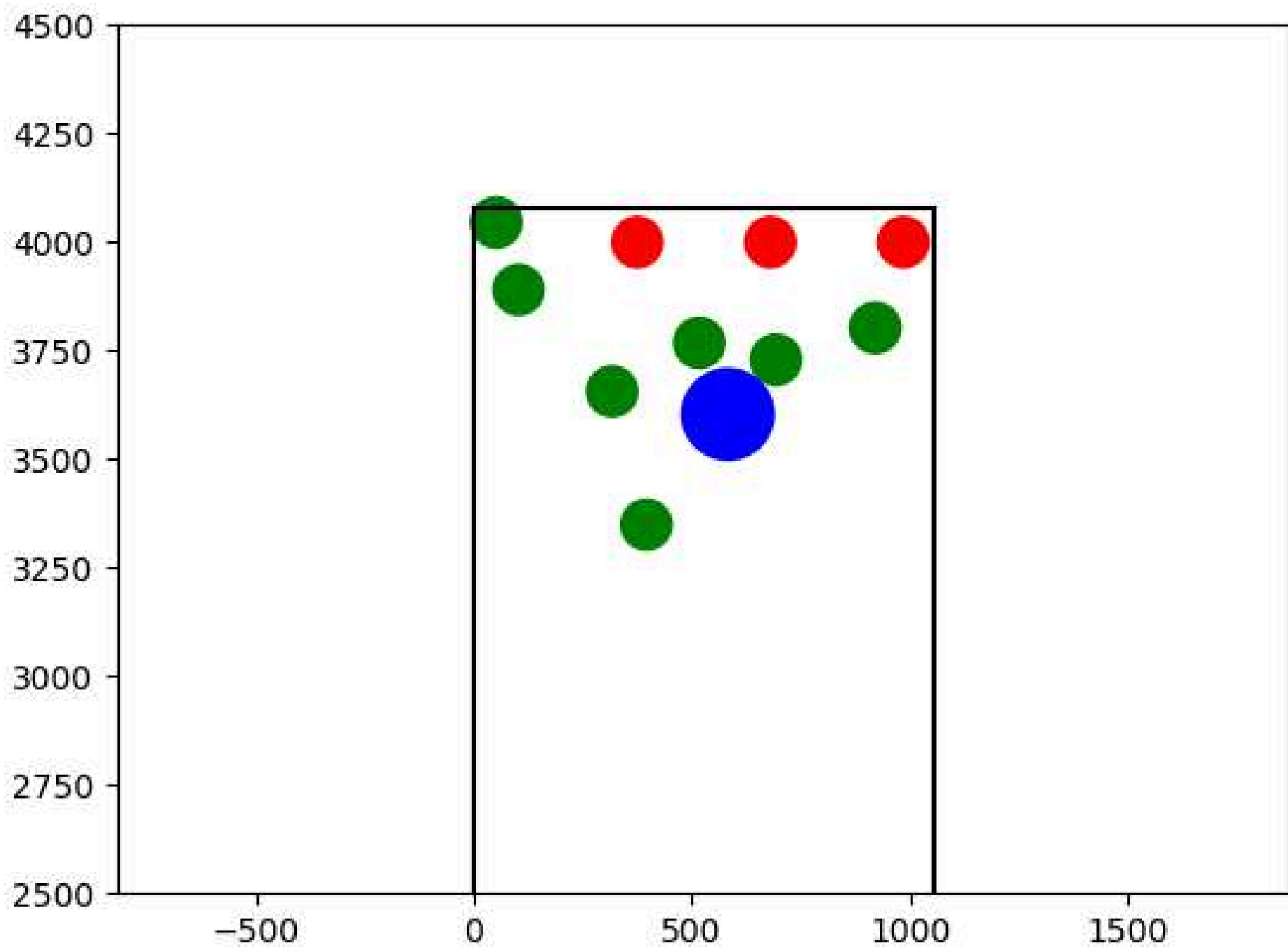


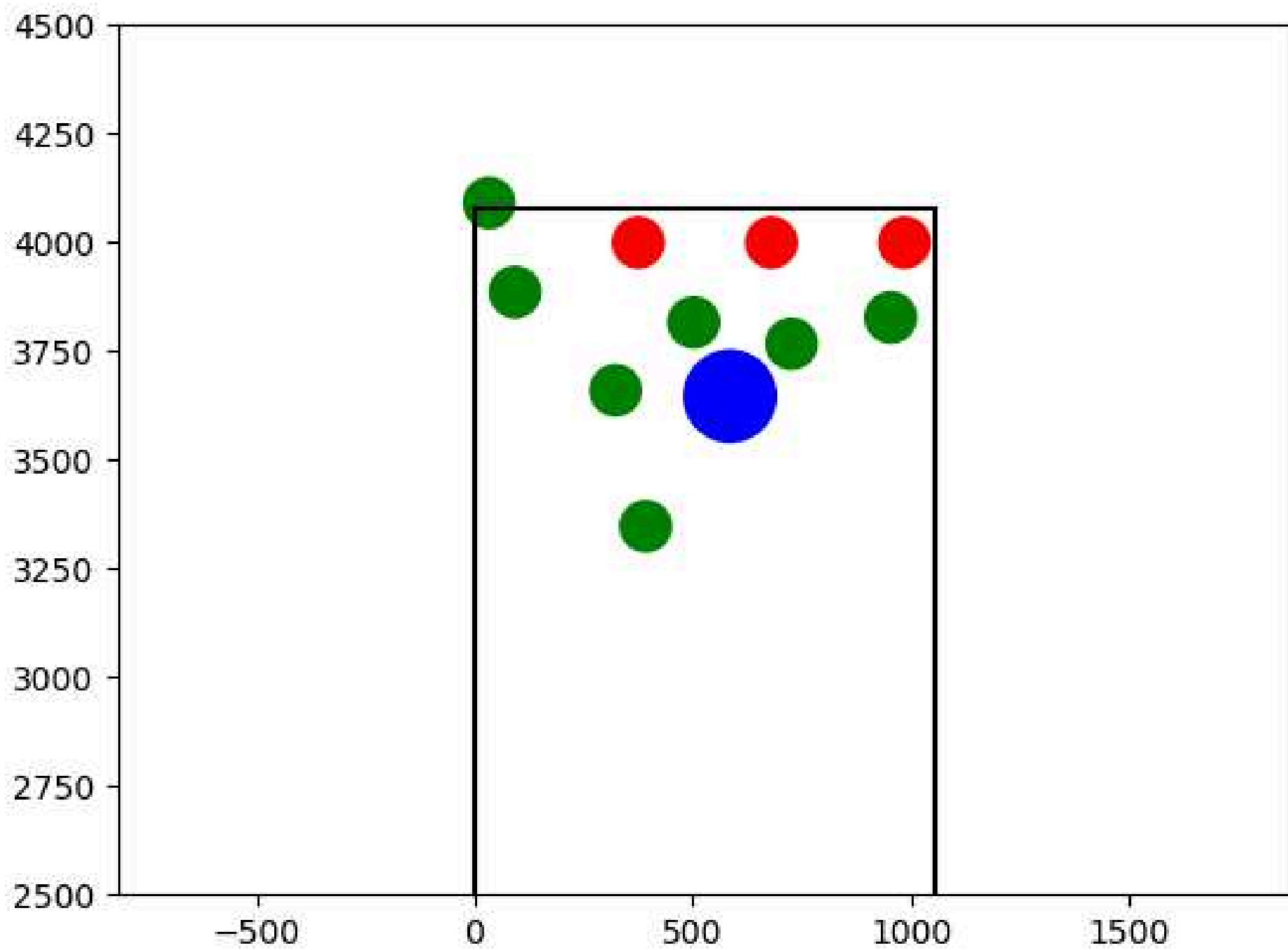


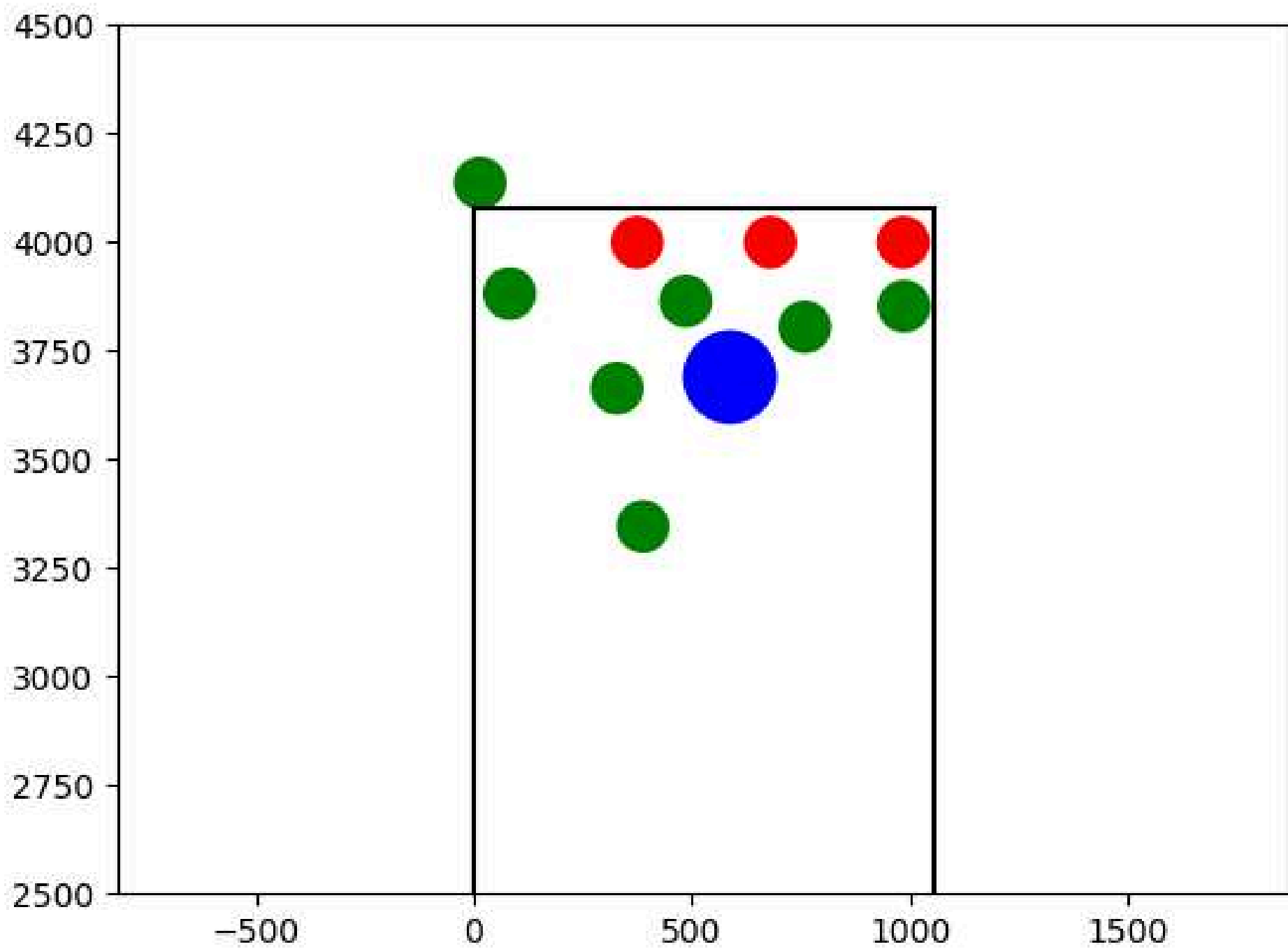


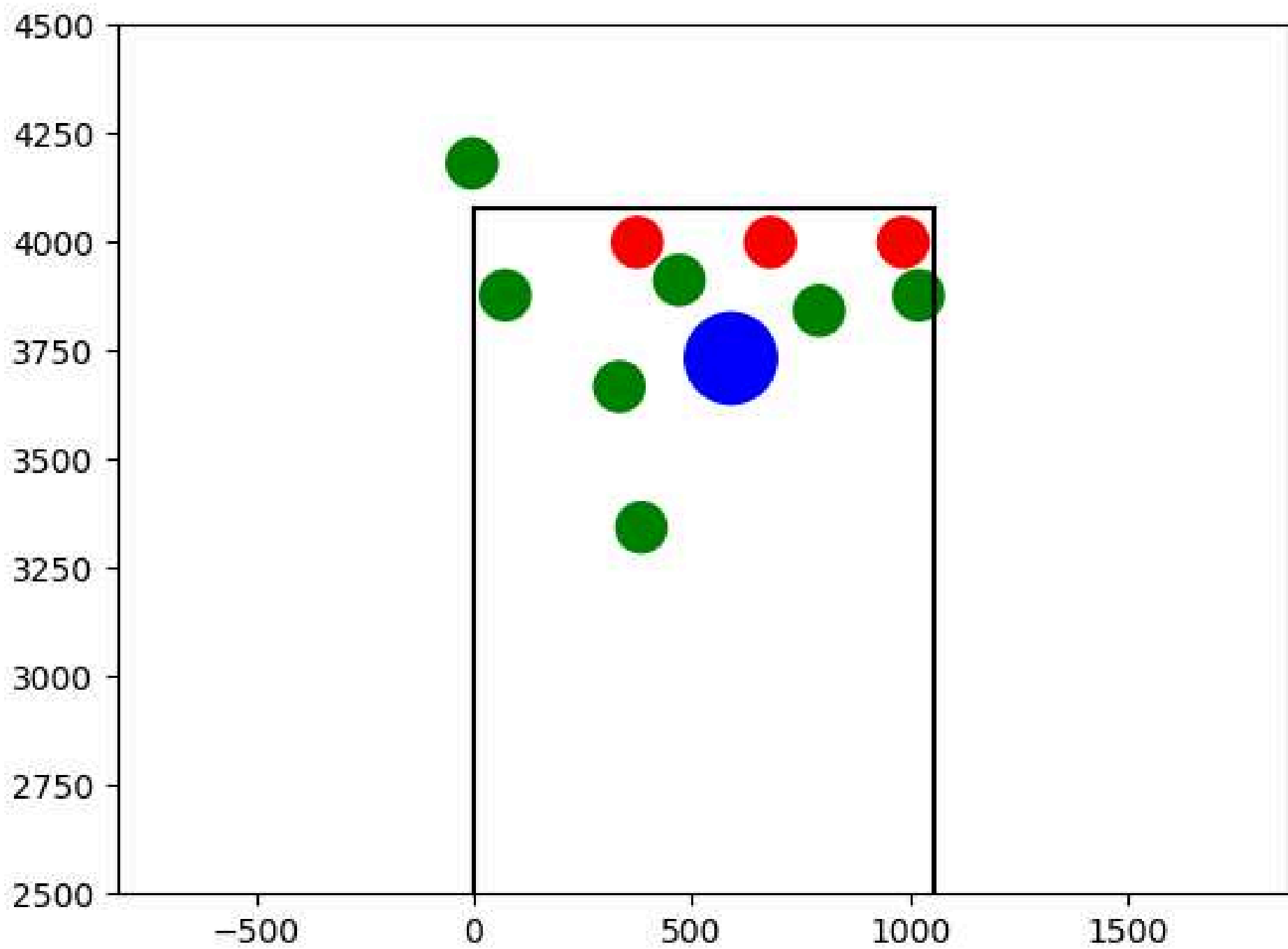


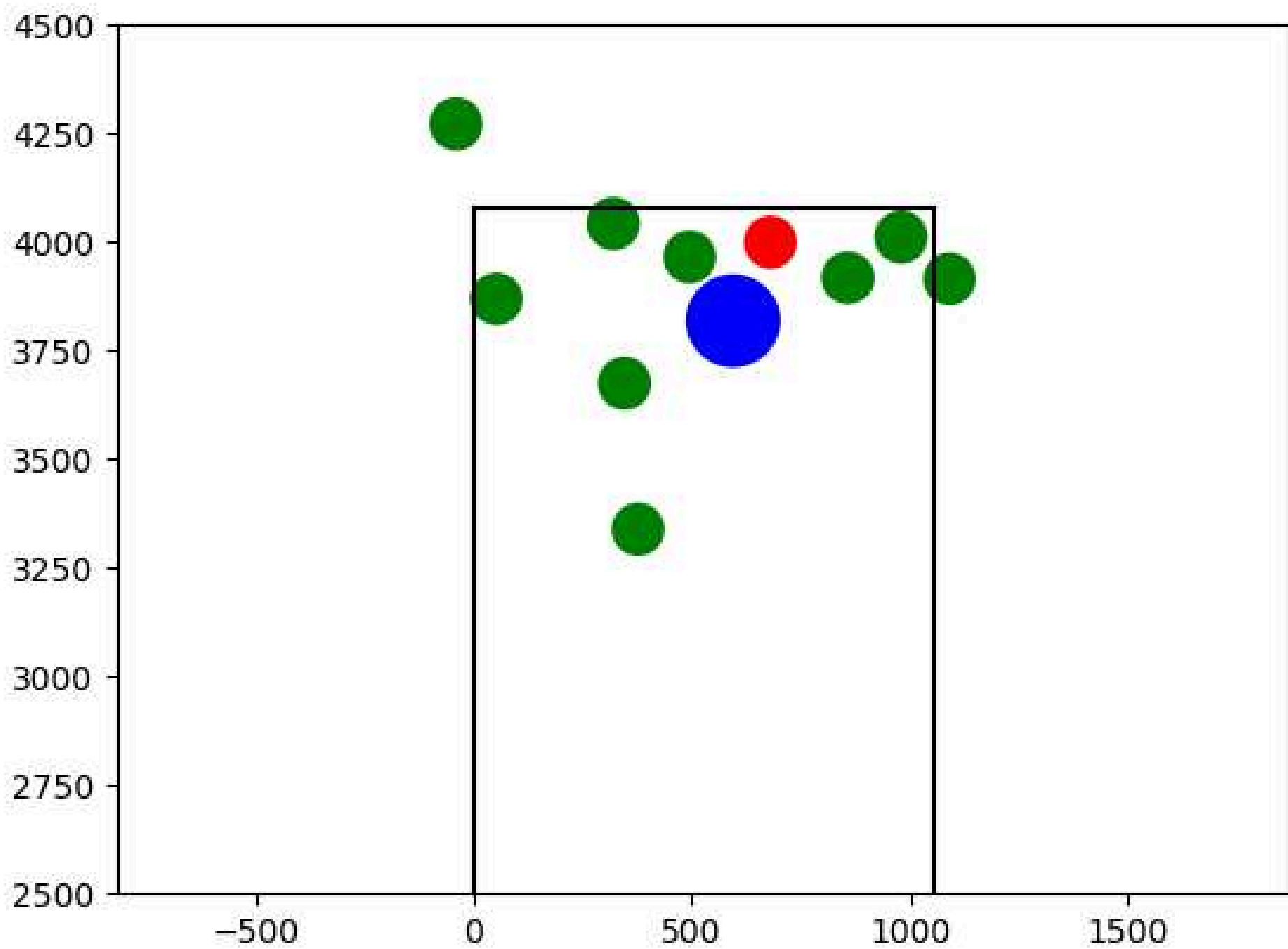


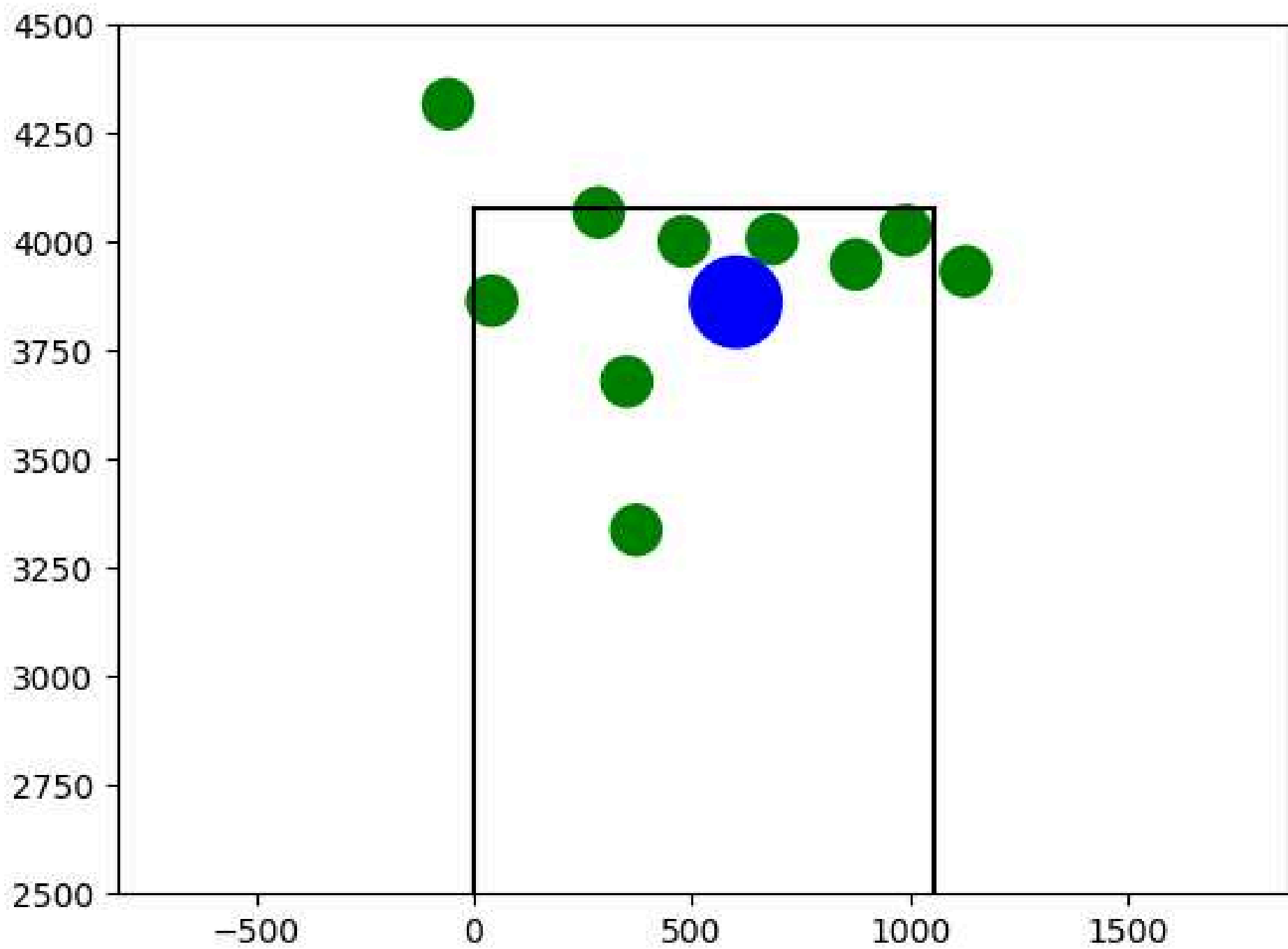


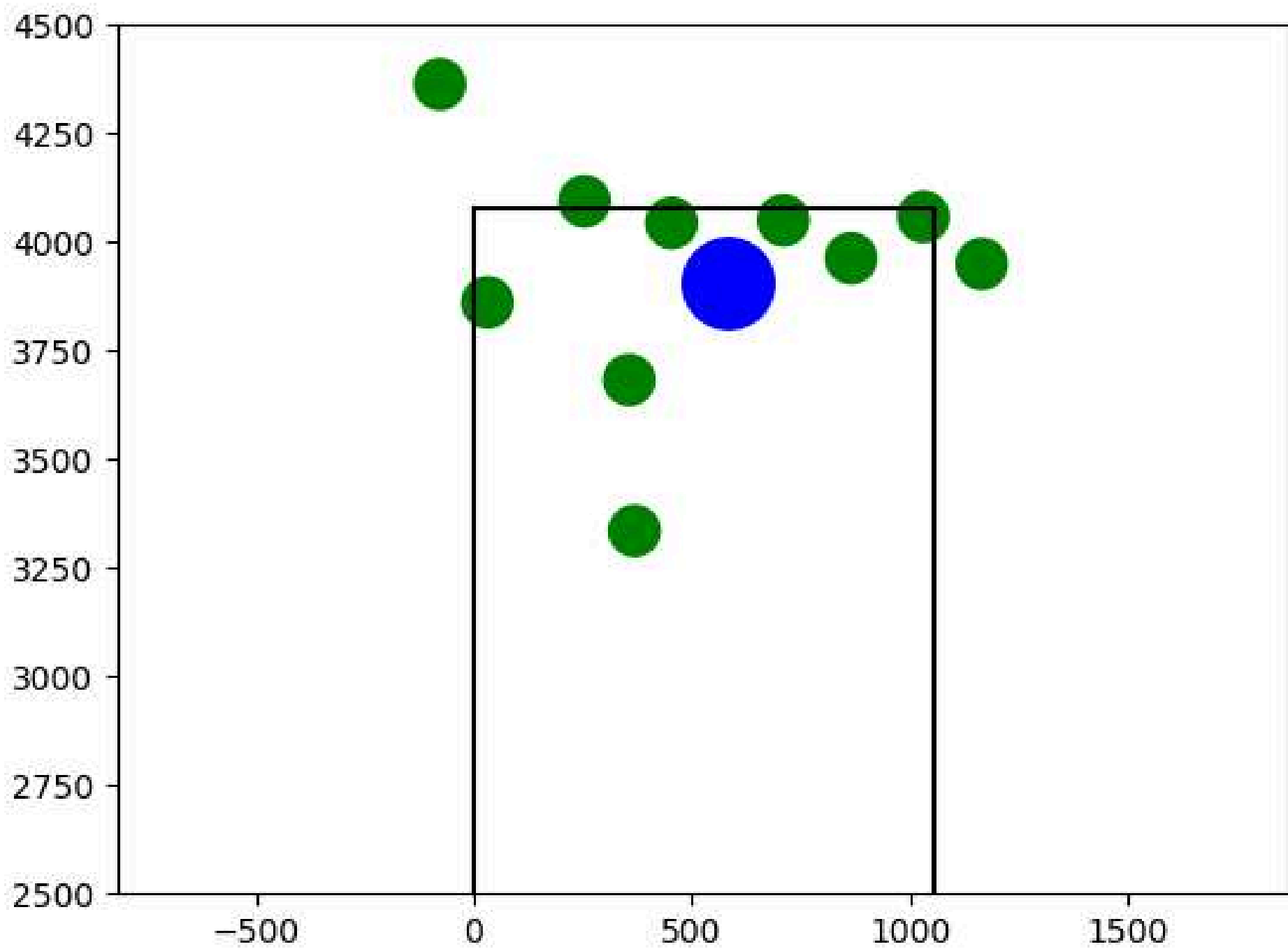


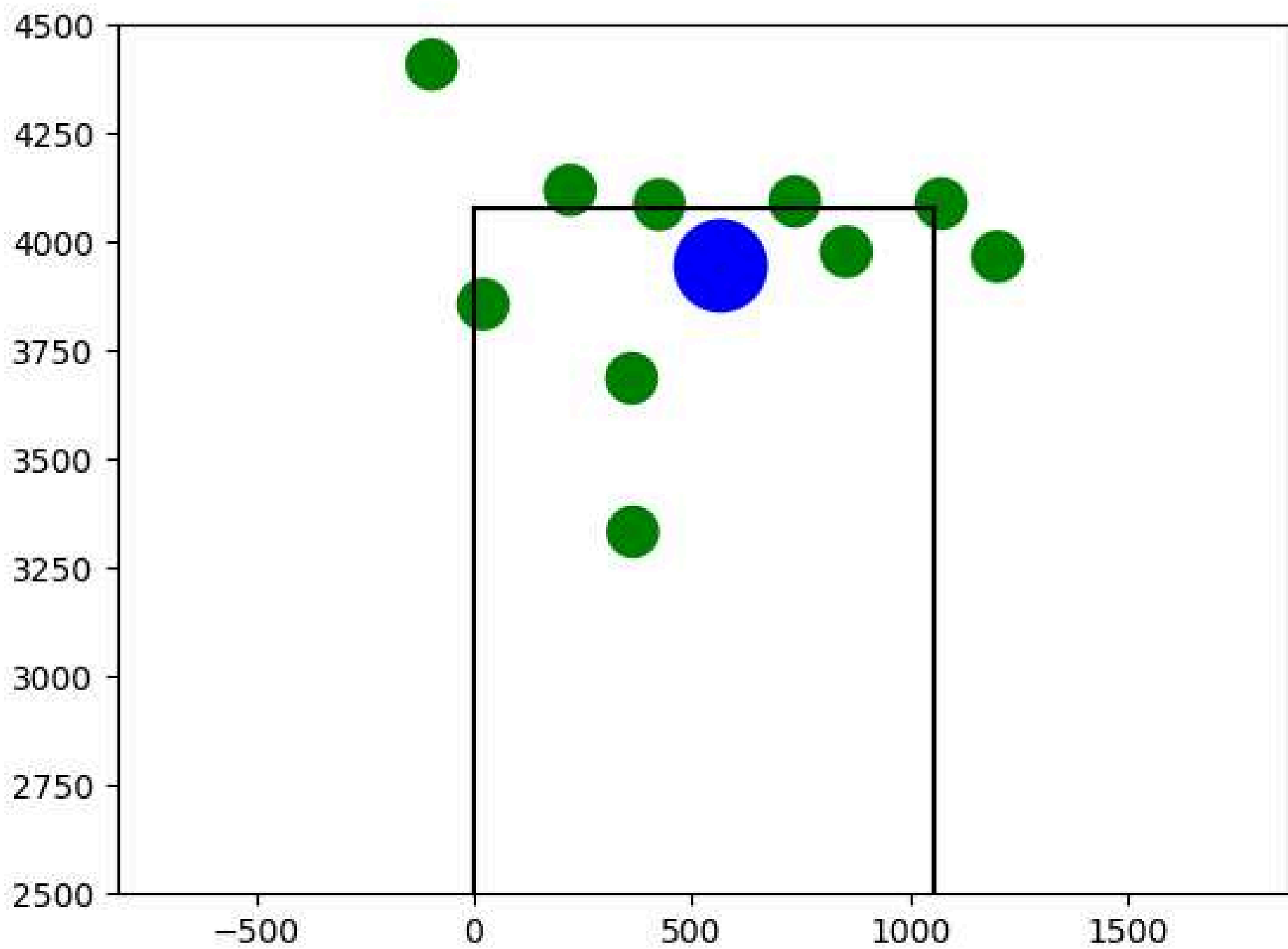








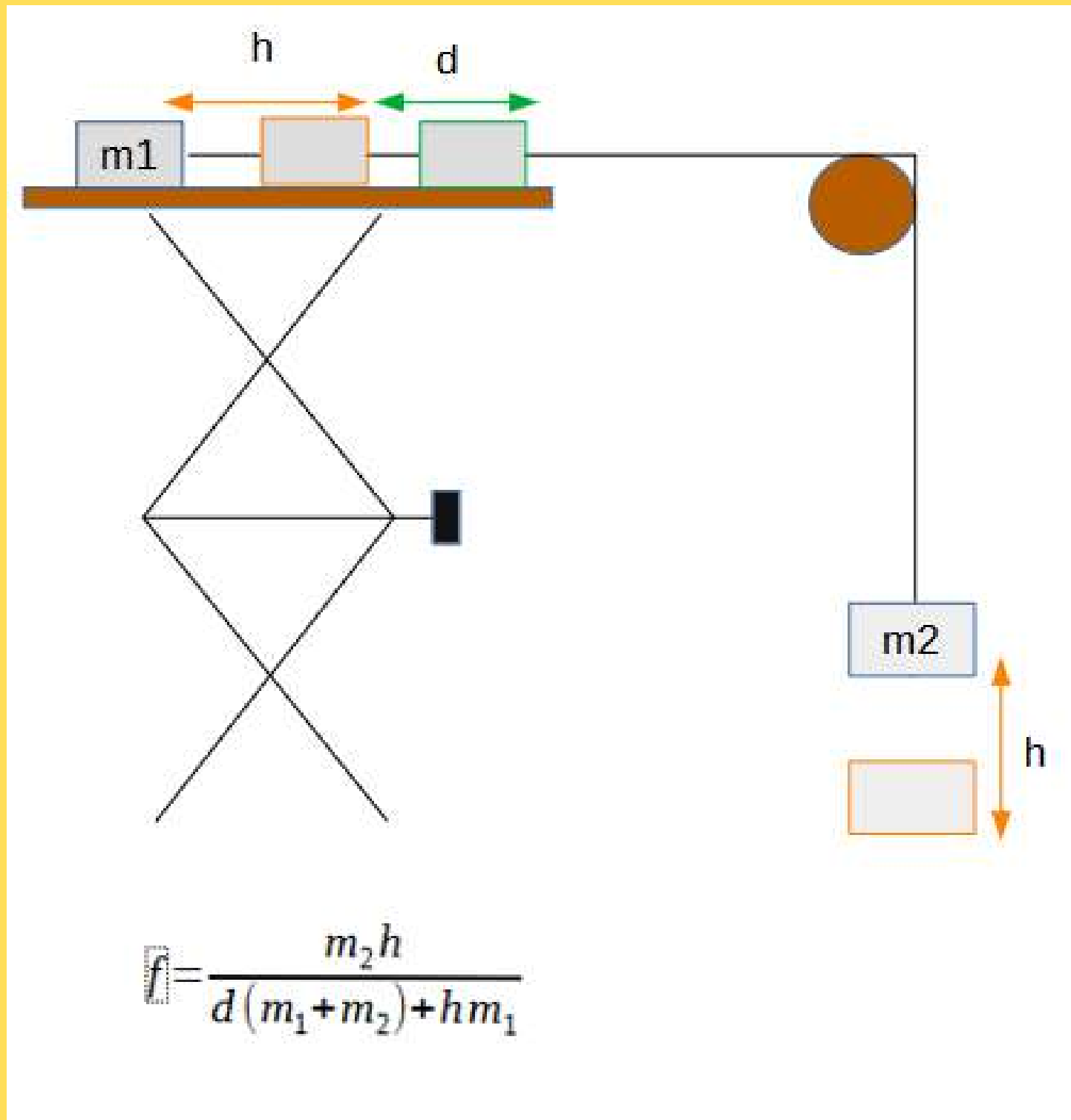




le geste professionnel



Premières mesures en labo





Valeur obtenue:
 $F=0.347\pm0,001$

Modèle de Contensou pour une boule homogène

$$I \frac{d\omega}{dt} = M \quad , \quad m\ddot{\mathbf{r}} = \mathbf{F}$$

$$F = F_0 \frac{3\pi v}{8\epsilon |\omega_z| + 3\pi v}$$

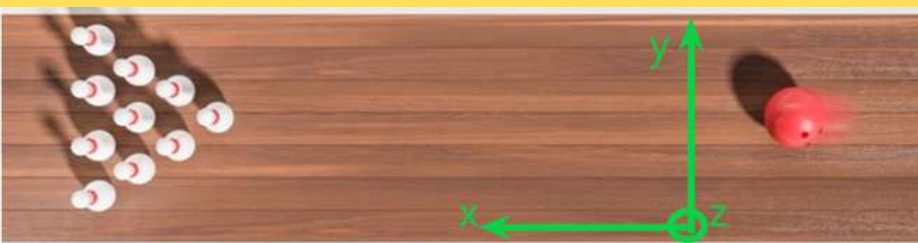
$$M = M_0 \frac{16\epsilon \omega_z}{16\epsilon |\omega_z| + 15\pi v}$$

$$\dot{\omega}_y = \frac{F_0}{I} \frac{3\pi v_x R}{8\epsilon |\omega_z| + 3\pi v}$$

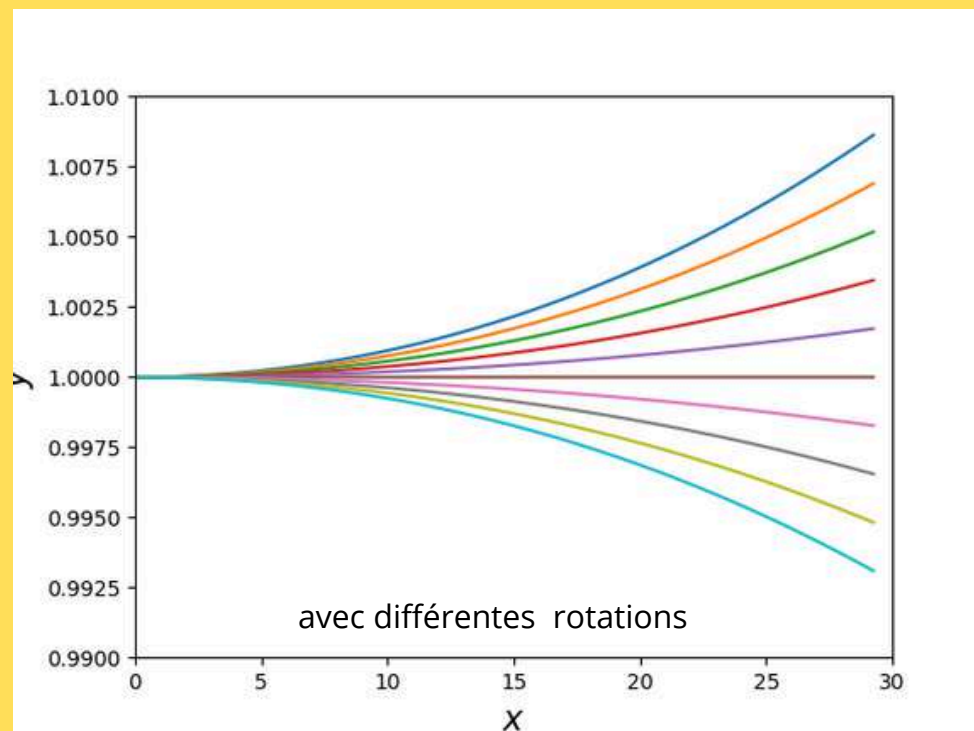
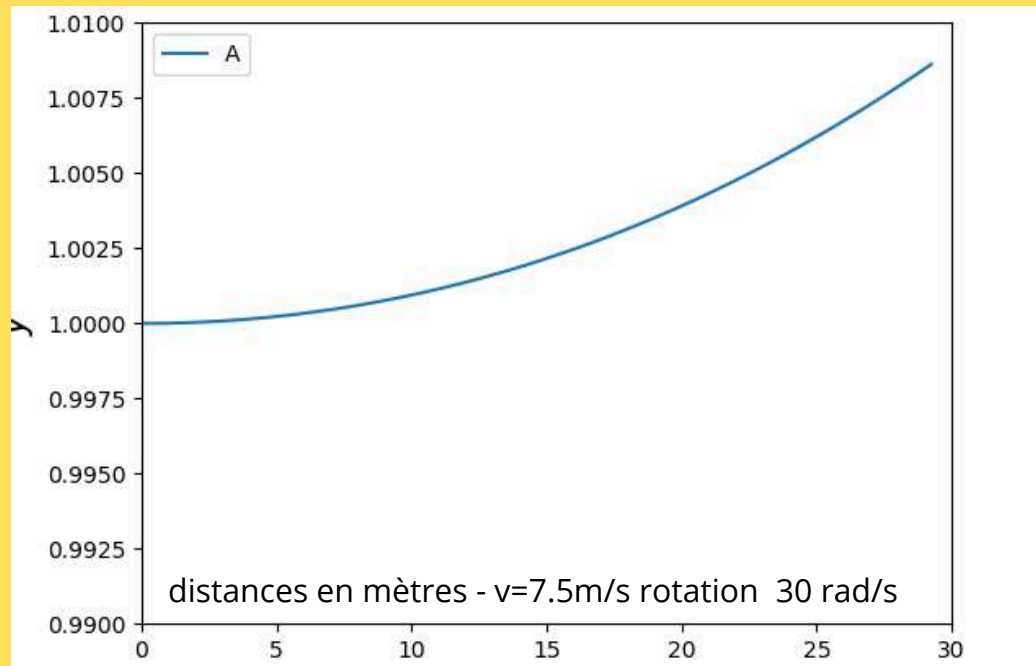
$$\dot{\omega}_z = \frac{-M_0}{I} \frac{16\epsilon \omega_z}{16\epsilon |\omega_z| + 15\pi v}$$

$$\ddot{x} = \frac{-F_0}{m} \frac{3\pi v_x}{8\epsilon |\omega_z| + 3\pi v}$$

$$\ddot{y} = \frac{-F_0}{m} \frac{3\pi v_y}{8\epsilon |\omega_z| + 3\pi v}$$



Trajectoire obtenue

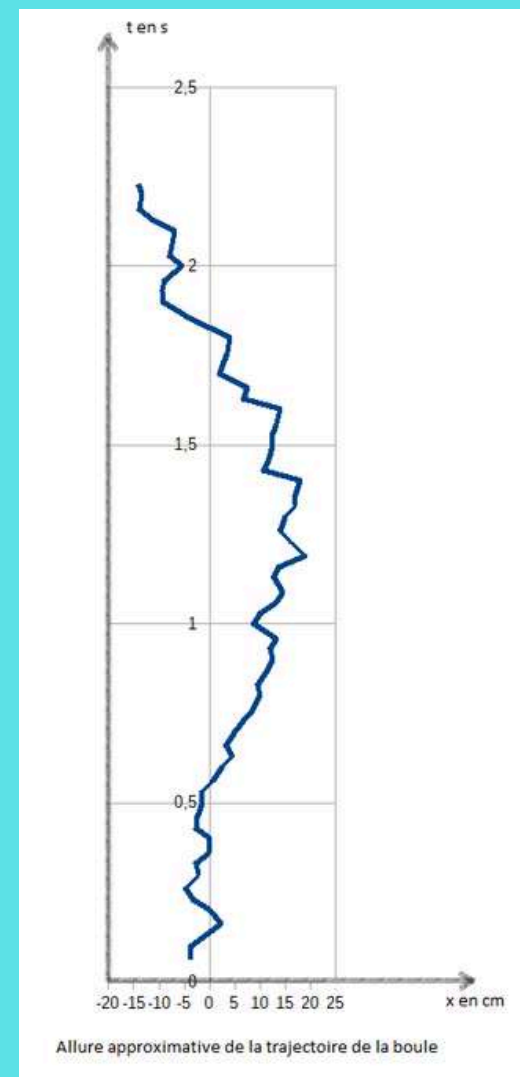
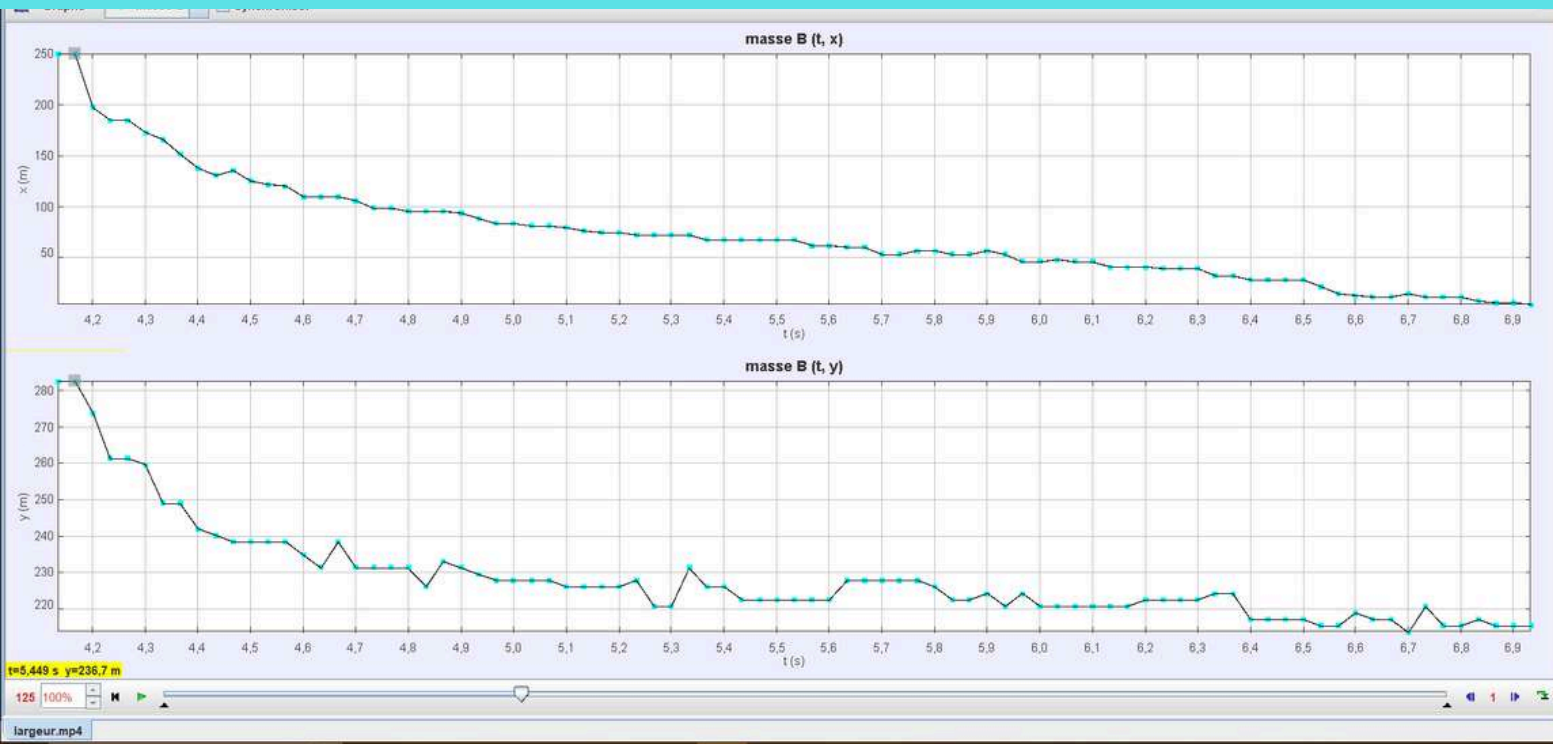


L'avantage des boules hétérogènes

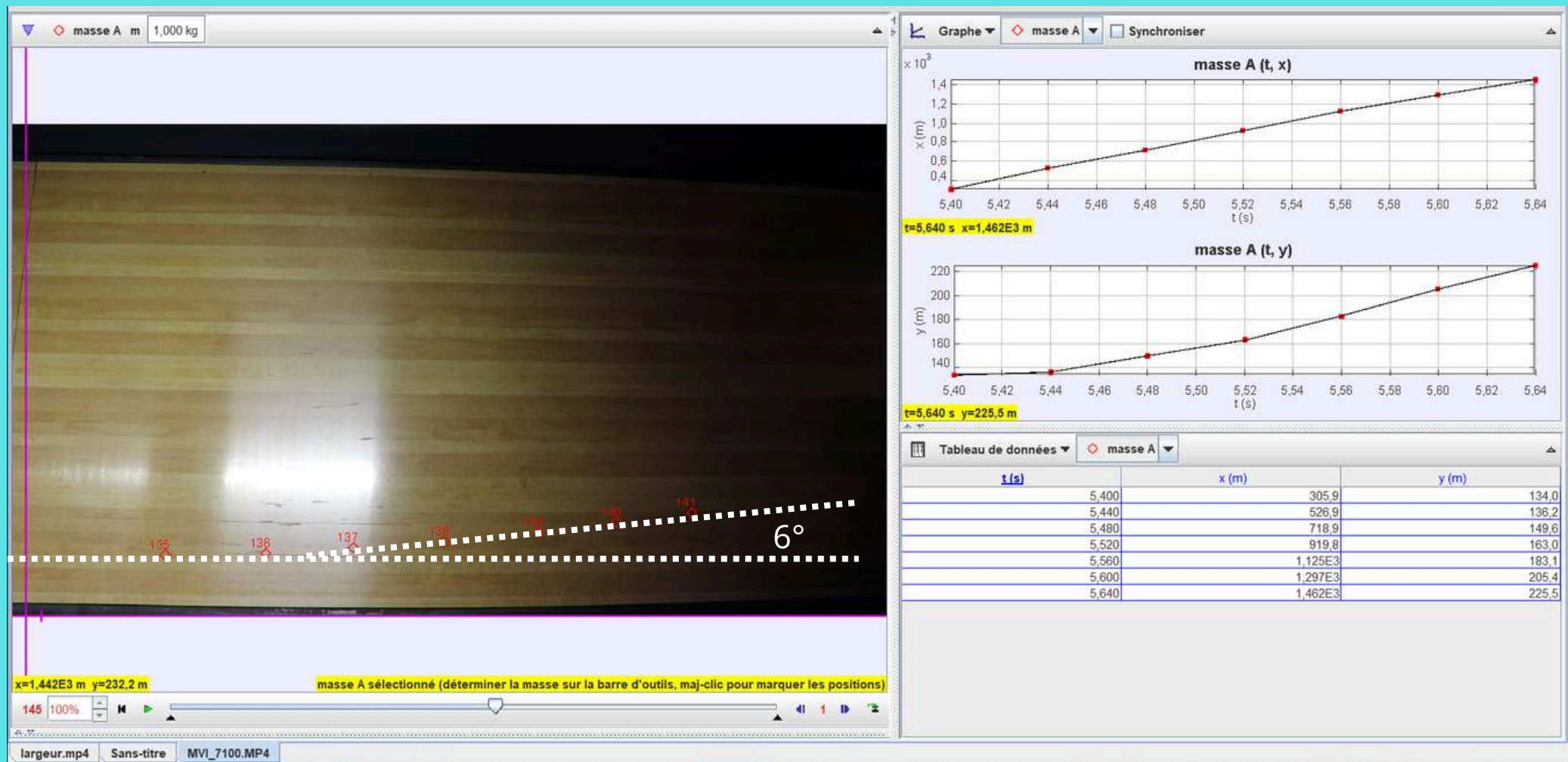


<https://www.sciencesculture.com/post/bowling>

Etude de la vidéo



Mesure du hook en fin de piste



MODELISATION DE FROHLICH

Mouvement avec glissement

$$\frac{d(I\vec{\omega})}{dt} = (\vec{r}_{\Delta} + \vec{R}_{con}) \wedge \vec{F}_{con} \quad , \quad M\ddot{\vec{r}} = \vec{F}_{con} + \vec{F}_g$$

$$I\vec{\alpha} + \vec{\omega} \wedge (I\vec{\omega}) = M(\vec{r}_{\Delta} + \vec{R}_{con}) \wedge (g + a_{\alpha} + a_{\omega}) \vec{\mu}$$

$$M\ddot{x} = M(g + [\vec{r}_{\Delta} \wedge \vec{\alpha}]_z + [(\vec{\omega} \wedge \vec{r}_{\Delta}) \wedge \omega]_z) \mu_x$$

$$M\ddot{y} = M(g + [\vec{r}_{\Delta} \wedge \vec{\alpha}]_z + [(\vec{\omega} \wedge \vec{r}_{\Delta}) \wedge \omega]_z) \mu_y$$



MODELISATION DE FROHLICH

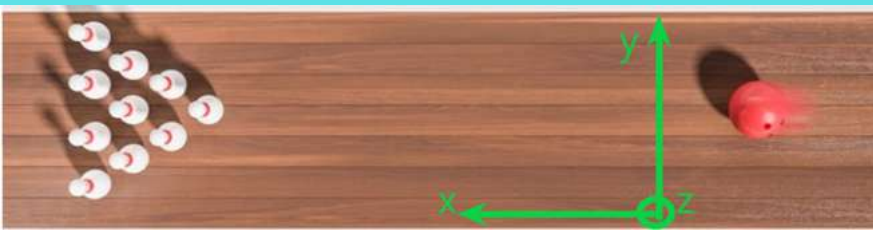
Roulement sans glissement

$$\frac{d(I\vec{\omega})}{dt} = (\vec{r}_{\Delta} + \vec{R}_{con}) \wedge \vec{F}_{con} \quad , \quad M\ddot{\vec{r}} = \vec{F}_{con} + \vec{F}_g$$

L'absence de glissement donne $\dot{\vec{r}}_{cb} = \vec{R}_{con} \wedge \vec{\alpha}$ où $\dot{\vec{r}}_{cb}$ est la position du centre de la boule.

$$(I + I_{roll})\vec{\alpha} = (I\vec{\omega}) \wedge \vec{\omega}$$

$$M\ddot{\vec{r}} = (\vec{r}_{\Delta} + \vec{R}_{con}) \wedge \vec{\alpha} + (\vec{r}_{\Delta} + \vec{\omega}) \wedge \vec{\omega}$$



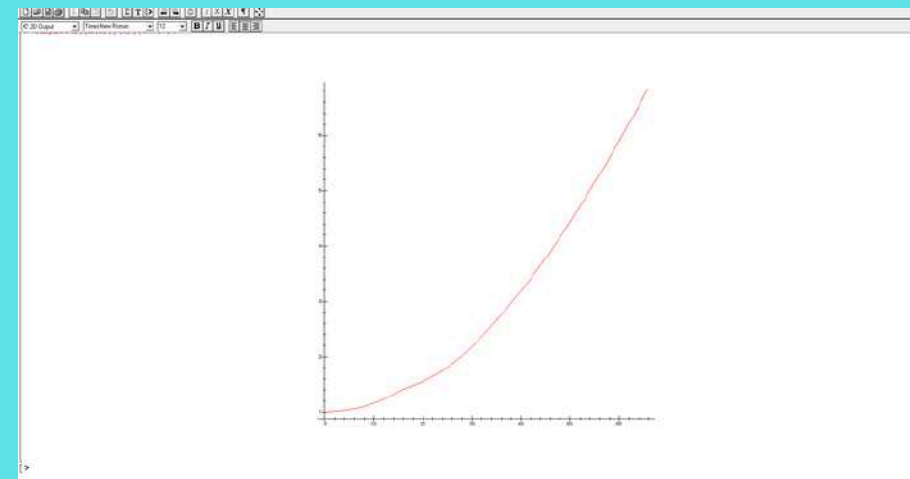
Traitement par un logiciel de calcul

```
Maple V Release 4 - [placement.mws]
File Edit View Insert Format Options Window Help
[Icons]
C 2D Output [Free New Roman] f2 [B I U] [E]
> s:=solve(equations, {x(t), y(t), t1(t), t2(t), t3(t)}, type=numeric);
> points:=[0,0,0,0,0,0,0,0,0,0,0];
> for k from 1 to 12 do
  points[k]:= [op(2,s(k/10)[2]),op(2,s(k/10)[4])];
od;

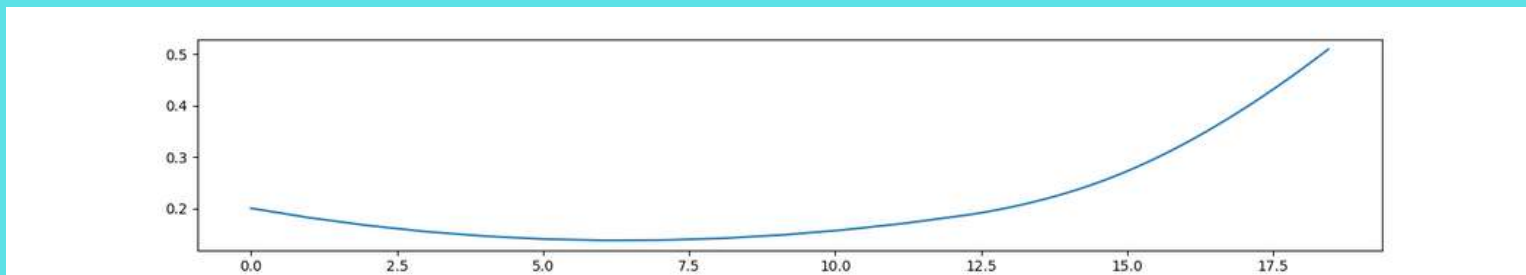
point1:= [1.001641694317371, 4716416943173710]
point2:= [2.006555755787239, 4465557557872387]
point3:= [3.014699912819012, 4246999128190119]
point4:= [4.025942234023984, 4059422340239817]
point5:= [5.040007492935660, 3900074929356560]
point6:= [6.056702010619109, 3767020106191035]
point7:= [7.0774318576023596, 3674318576023596]
point8:= [8.10250301579203, 362503015791964]
point9:= [9.12896730885450, 358967308854434]
point10:= [10.16110692981619, 3611069298161816]
point11:= [11.20073521003705, 3707352100370403]
point12:= [12.24524488026974, 3852448802697295]

> points;
[[1.001641694317371, 4716416943173710], [2.006555755787239, 4465557557872387], [3.014699912819012, 4246999128190119], [4.025942234023984, 4059422340239817], [5.040007492935660, 3900074929356560], [6.056702010619109, 3767020106191035], [7.0774318576023596, 3674318576023596], [8.10250301579203, 362503015791964], [9.12896730885450, 358967308854434], [10.16110692981619, 3611069298161816], [11.20073521003705, 3707352100370403], [12.24524488026974, 3852448802697295]]
> s(1.2);
[1.2, x(r) = 12.24524488026974,  $\frac{\partial}{\partial t}x(r) = 10.457196660505264$ ,  $y(r) = 3852448802697295$ ,  $\frac{\partial}{\partial t}y(r) = 15719666050526172$ ,  $t1(r) = 3.942166848567116$ ,  $\frac{\partial}{\partial t}t1(r) = 7.420901689629666$ ,  $t2(r) = -19.88804426899264$ ,  $\frac{\partial}{\partial t}t2(r) = -19.61208343784657$ ,  $t3(r) = 3473396707485343$ ,  $\frac{\partial}{\partial t}t3(r) = 17.7619199540561$ ]
> with(plots):
> odoplot(s, {x(t), y(t)}, 0..6 1;
```

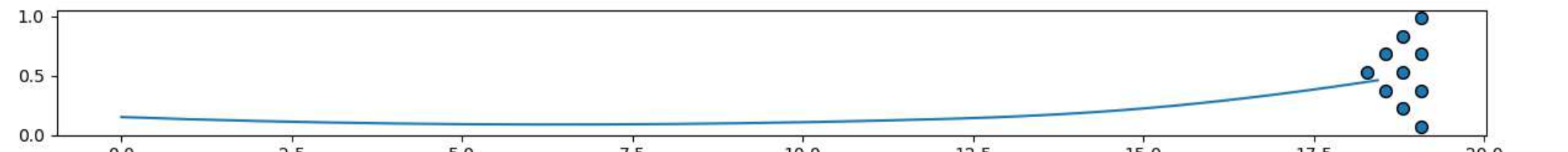
résolution avec Maple



trajectoire obtenue
zone sans glissement



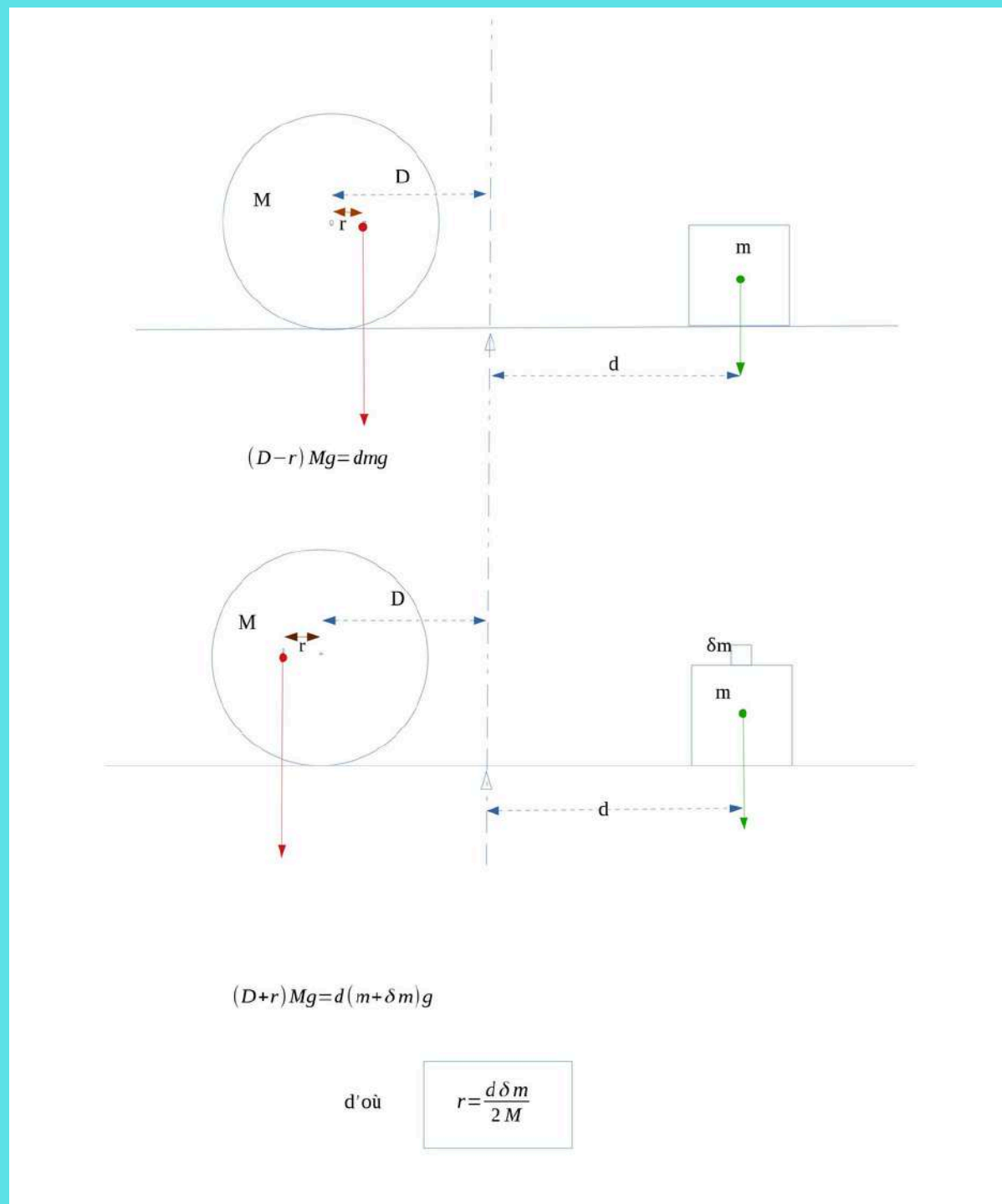
trajectoires recollées



résultat final à l'échelle - angle de 7° obtenu

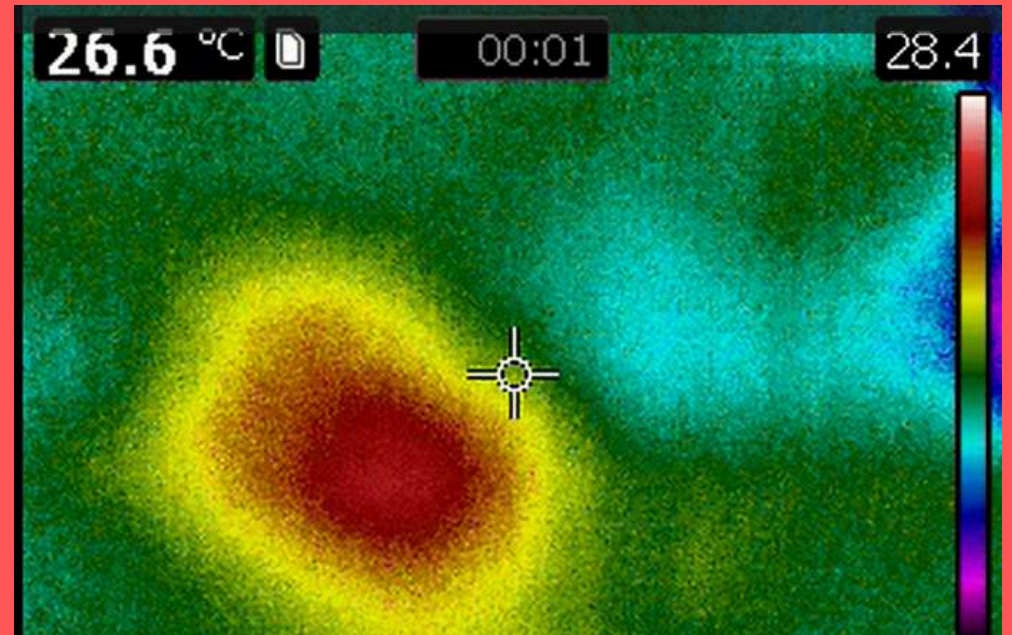
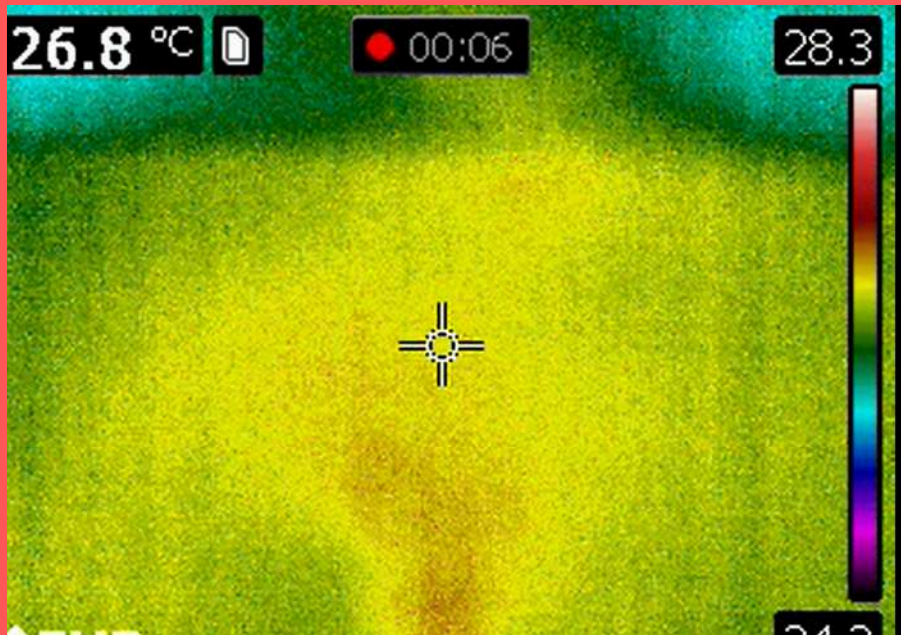


Mesure du décalage du centre de masse

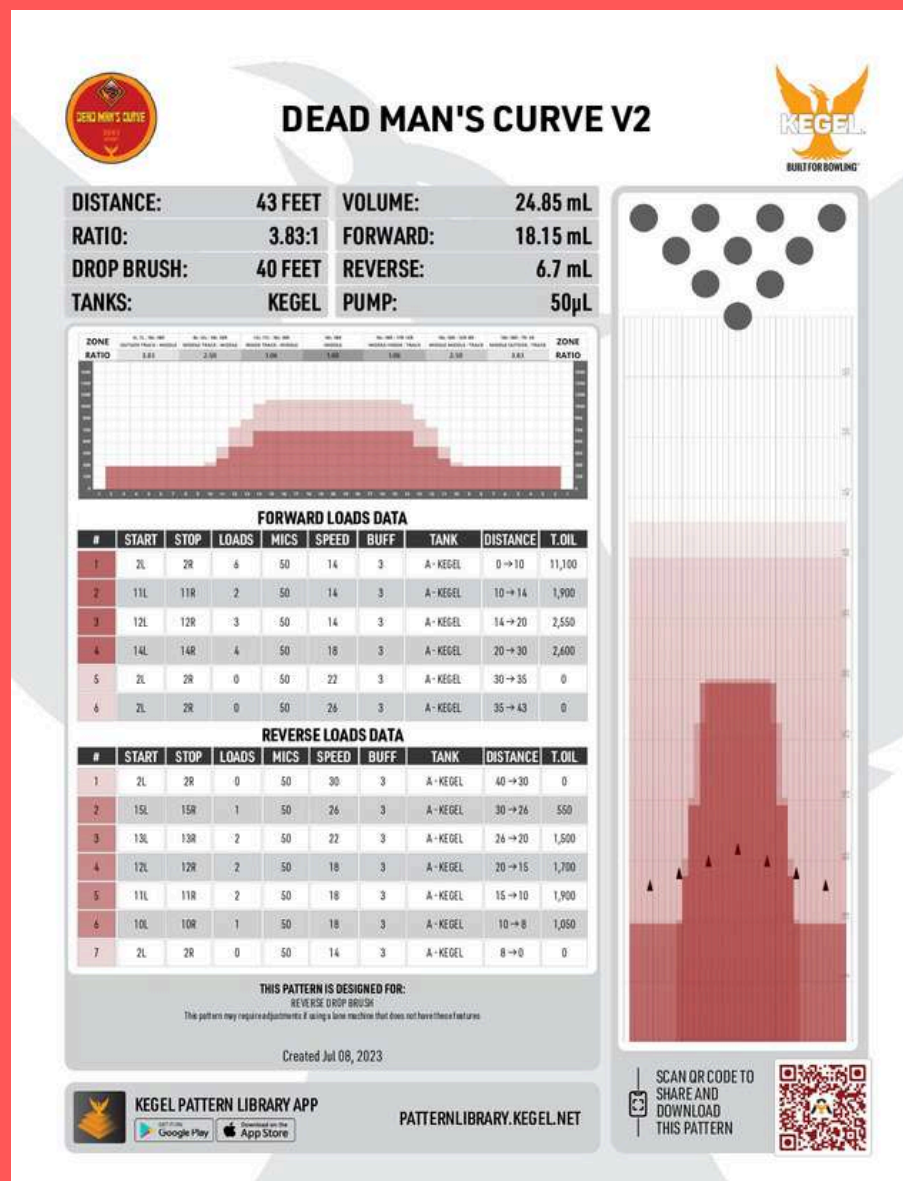
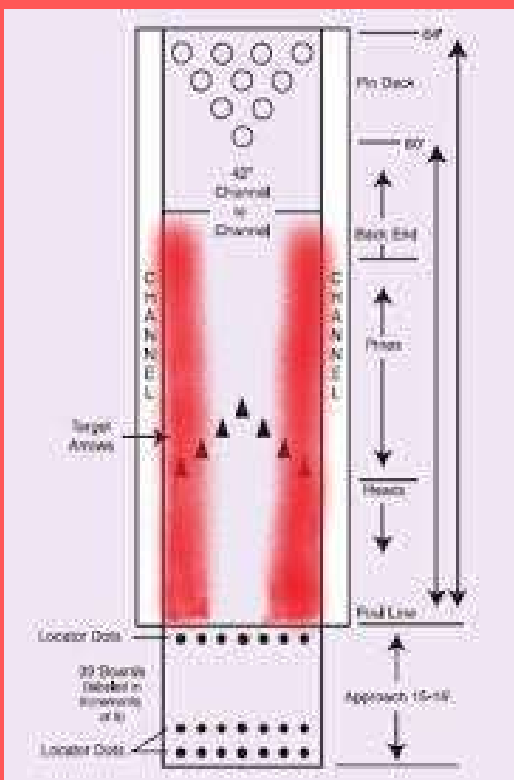


en pratique, r vaut environ 2cm

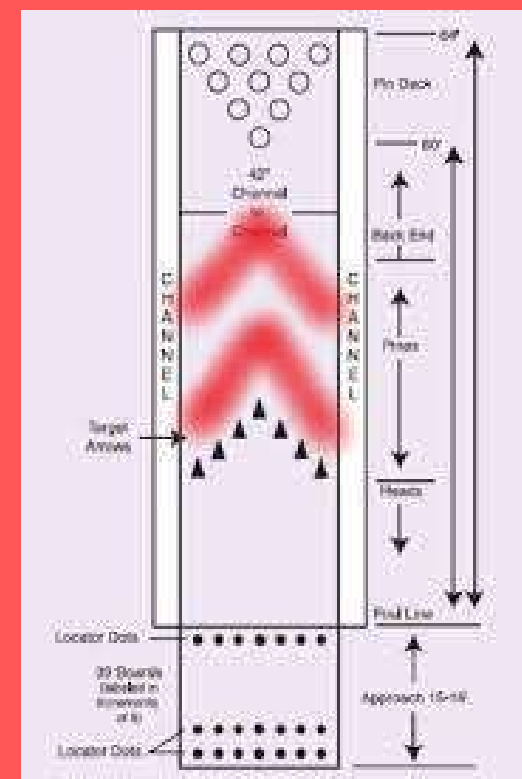
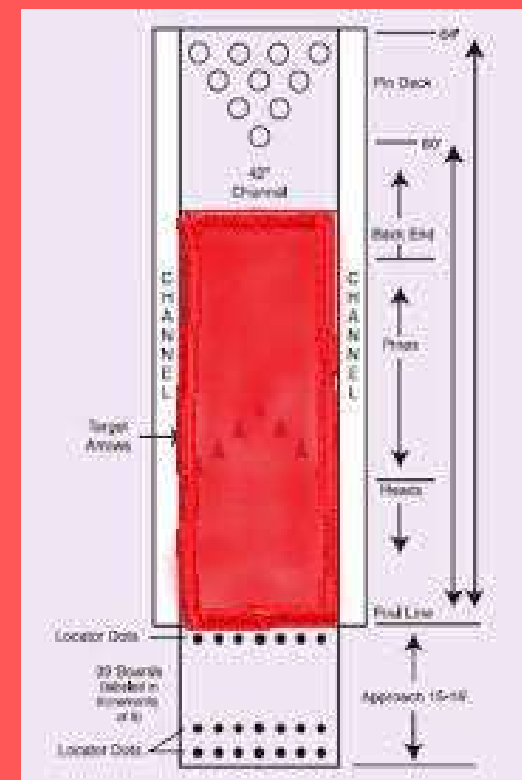
Nécessité du huilage



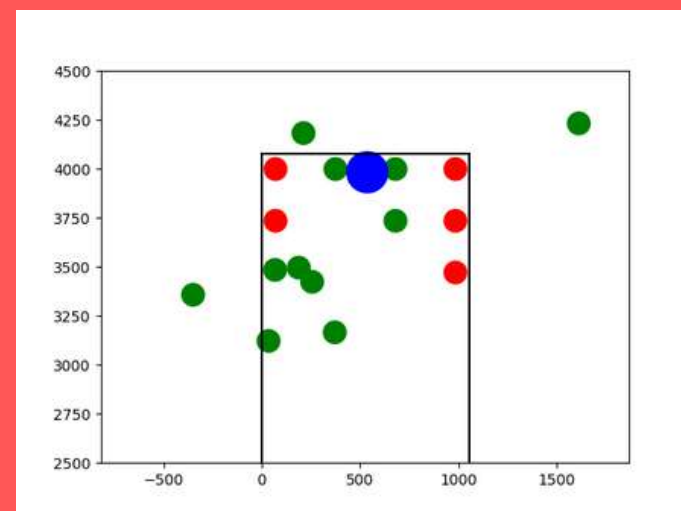
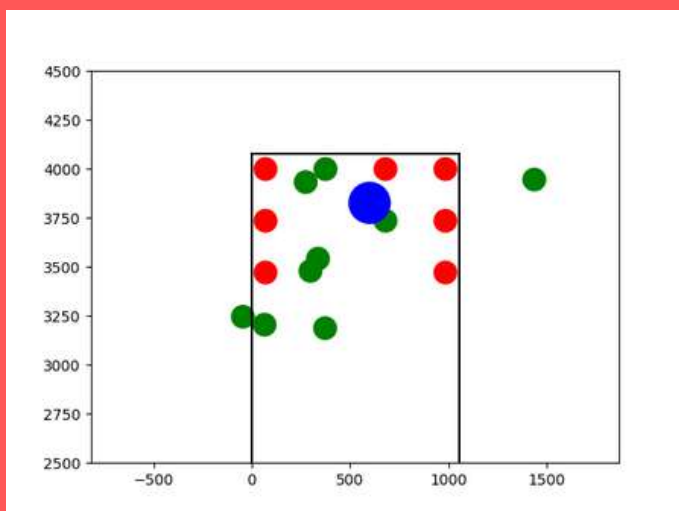
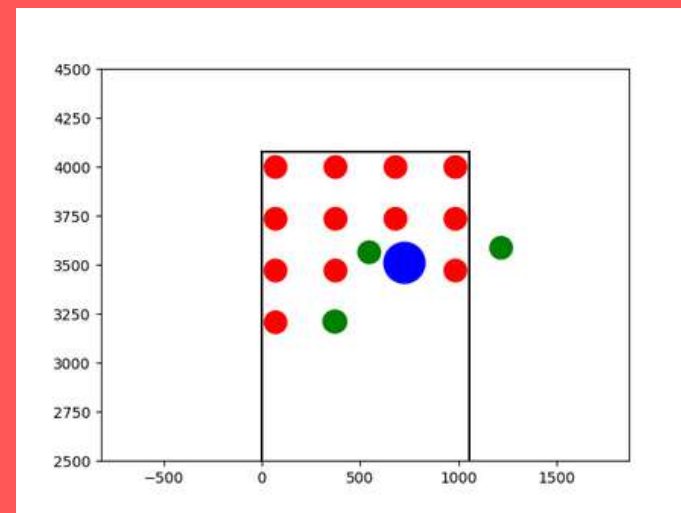
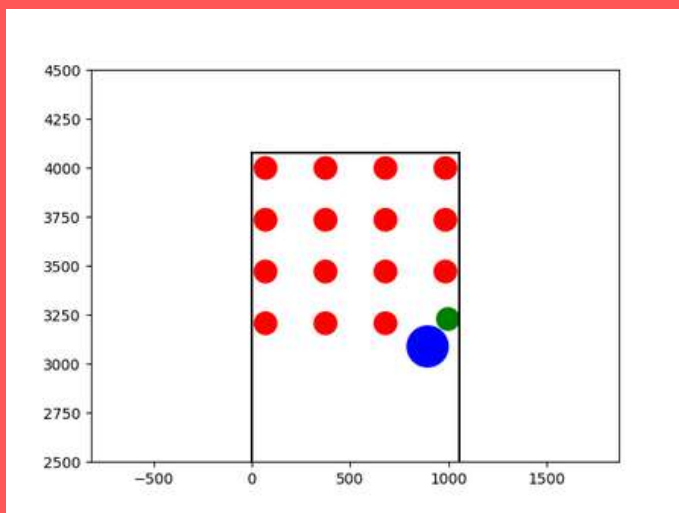
Quel huilage pour corser le jeu ?

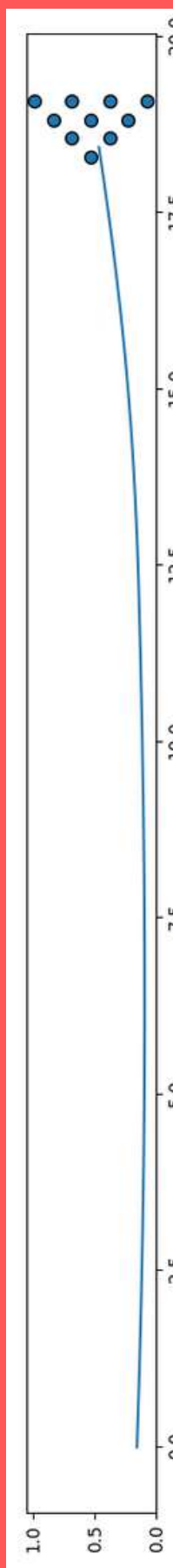


<https://patternlibrary.kegel.net/pattern/fec2cae3-d052-ec11-8c62-000d3a5afd36>



Plus de quilles ?





$$x_0 \simeq 12,24 \text{ m}$$

$$y_0 \simeq 0,38 \text{ m}$$

$$v_x \simeq 10,5 \text{ m s}^{-1}$$

$$v_y \simeq 0,15 \text{ m s}^{-1}$$

$$\dot{\theta}_1(0) \simeq 7,4 \text{ rad s}^{-1}$$

$$\dot{\theta}_2(0) \simeq -40 \text{ rad s}^{-1}$$

$$\dot{\theta}_3(0) \simeq 18 \text{ rad s}^{-1}$$

https://www.gamaniak.com/video/bras-robot-lance-boule-bowling#google_vignette




```

# C:\Users\phlau\Dropbox\Mon PC (DESKTOP-20E6RHT)\Desktop\Alexandre\TIPE
bowling\programmes, tracés et photos présentation\tous les scripts TIPE bowling.py
001|
002| from math import *
003|
004| #définition des fonctions de tracé:
005|
006|
007| import matplotlib.pyplot as plt
008| from matplotlib.patches import Rectangle
009|
010| def disk(center, rad, **kwargs):
011|     circ=plt.Circle(center, rad, lw=0, **kwargs)
012|     plt.gca().add_patch(circ)
013|
014| def diskvide(center, rad, **kwargs):
015|     circ=plt.Circle(center, rad, lw=0.3, fill=False)
016|     plt.gca().add_patch(circ)
017|
018| def line(A, B, **kwargs):
019|     plt.plot([A[0], B[0]], [A[1], B[1]], **kwargs)
020|
021|
022|
023| bouge=[] # cette liste vide au départ se remplira des numéros des quilles
heurtées
024|
025|
026| #la fonction suivante renvoie la liste à l'instant t des chocs entre boule et
quille ou entre deux quilles
027| def choc(Q,B):
028|     bq=[] #liste des chocs boule-quille initialisée
029|     qq=[] #liste des chocs quille-quille initialisée
030|
031|     for q in Q: #on parcourt la liste des quilles
032|         if ((q[1]-B[0])**2+(q[2]-B[1])**2)<170**2: #on teste si la quille est en
choc avec la boule (condition sur la distance entre les centres)
033|             bq.append(q[0]) # s'il y a choc, on le recense dans la liste bq
034|             if q[0] not in bouge: # si la quille en question n'avait jamais été
heurtée,
035|                 bouge.append(q[0]) # on la rajoute à la liste des numéros des
quilles heurtées
036|         for q in Q: #on recommence en cherchant les chocs entre deux quilles
037|             for p in Q:
038|                 if q!=p: #o écarte le cas où les quilles sont identiques
039|                     if ((q[1]-p[1])**2+(q[2]-p[2])**2)<122**2:
040|                         qq.append([q[0],p[0]])
041|                         if q[0] not in bouge:
042|                             bouge.append(q[0])
043|
044|     return(bq,qq) # la fonction choc renvoie la liste des quilles heurtées par la
boule et celles heurtées par une autre quille
045|
046|
047|
048| # la fonction suivante modifie les vitesses des quilles après leur choc et fait
avancer pas à pas la boule et les quilles en mouvement
049| def mouvement(Q,B):
050|     boule,quille=choc(Q,B)# recherche de chocs
051|     if len(quille)!=0:# s'il y a choc entre deux quilles alors modification des
vitesses
052|         for couple in quille:
053|             n=min(couple[0],couple[1])#la quille de petit numéro heurte celle à
gros numéro (les quilles de devant ont des petits numéros et heurtent les autres de
numéro plus élevé)
054|             m=max(couple[0],couple[1])
055|             o1=[Q[n][1],Q[n][2]] #vecteur position de la quille heurtante
056|             o1=[Q[m][1],Q[m][2]] #vecteur position de la quille heurtée

```

```

057|         v1=[Q[n][3],Q[n][4]] #vecteur vitesse de la quille heurtante
058|         v2=[Q[m][3],Q[m][4]] #vecteur vitesse de la quille heurtée (en
général nul)
059|         v=[v1[0]+v2[0],v1[1]+v2[1]] #au cas où les deux quilles bougent,
c'est le vecteur somme qui interviendra à la place de v1
060|         o1o2=[Q[m][1]-Q[n][1],Q[m][2]-Q[n][2]] #direction de la droite des
centres= future direction de la vitesse de la quille heurtée
061|         N=[Q[m][2]-Q[n][2],-Q[m][1]+Q[n][1]] #direction normale à la droite
des centres=future direction de la quille heurtante (les quilles ont la même masse
donc les vitesses après le choc seront orthogonales)
062|         alpha=(o1o2[0]*v[0]+o1o2[1]*v[1])/(o1o2[0]**2+o1o2[1]**2)
#coefficient issu de la conservation qté de mvt
063|
064|         beta=(o1o2[1]*v[0]-o1o2[0]*v[1])/(o1o2[0]**2+o1o2[1]**2)
#coefficient issu de la conservation qté de mvt
065|         Q[m][3],Q[m][4]=alpha*o1o2[0],alpha*o1o2[1] #nouvelle vitesse de la
quille heurtée
066|
067|         Q[n][3],Q[n][4]=beta*N[0],beta*N[1] #nouvelle vitesse de la quille
heurtante
068|
069|
070|
071|     if len(boule)!=0: #si on a choc entre boule et quille
072|         for i in range(len(boule)): #on parcourt la liste des quilles heurtées
par la boule (en général une seule)
073|             k=boule[i] # la quille portera le numéro k
074|
075|             o2=[Q[k][1],Q[k][2]] # vecteur position de la quille
076|             o1=[B[0],B[1]] #vecteur position de la boule
077|             v2=[Q[k][3],Q[k][4]] #vecteur vitesse de la quille (en général nul)
078|             v1=[B[2],B[3]] #vecteur vitesse de la boule
079|
080|             o1o2=[Q[k][1]-B[0],Q[k][2]-B[1]] # la droite des centres, futur
vecteur vitesse de la quille
081|
082|             w1=[-(Q[k][1]-B[0])/(abs(Q[k][1]-B[0])+0.001)*B[3],B[2]] # w1 est un
vecteur orthogonal à la droite des centres et dirigé en sens opposé à la quille
heurtée, ce vecteur viendra légèrement détourner la boule de sa trajectoire sous
l'effet du choc
083|             cosalpha2=abs((v1[0]*o1o2[0]+v1[1]*o1o2[1])/
sqrt((v1[0]**2+v1[1]**2)*(o1o2[0]**2+o1o2[1]**2))) # cosinus de l'angle de déviation
de la quille
084|             cosalpha1=1/0.99-0.44*cosalpha2/10 # cosinus de l'angle de déviation
de la boule
085|             sinalpha1=sqrt(abs(1-cosalpha1**2)) # le sinus correspondant
086|
087|
088|             Q[k][3],Q[k][4]=o1o2[0],o1o2[1] # nouvelle vitesse de la quille
089|             B[2]=(B[2]+sinalpha1*w1[0]/2)*0.98 #nouvelle abscisse de vitesse de
la boule; le coefficient 0.98 traduit le ralentissement; la division par deux reflète
un impact faible sur la boule d'après les constatations par l'expérience
090|             B[3]=(B[3]+sinalpha1*w1[1]/2)*0.98 # même chose pour l'ordonnée
091|
092|
093|
094|     #calcul des nouvelles positions à l'instant t+dt, on tient compte des chocs
éventuels et des nouvelles vitesses, on fait avancer boules et quilles y compris
celles qui n'ont pas subi de nouveaux chocs, les quilles non heurtées restent
immobiles
095|     for i in range(1,11):
096|         Q[i][1],Q[i][2]=Q[i][1]+0.05*Q[i][3],Q[i][2]+Q[i][4]*0.05 # le
coefficient 0.05 permet d'avoir un tracé point par point plus précis et que les
quilles ne se superposent pas
097|
098|         B[0],B[1]=B[0]+B[2]*0.8,B[1]+B[3]*0.8 #la vitesse est ralentie pour la
précision du tracé, la vitesse absolue n'a aucune importance dans le mouvement à
étudier, on a intérêt à travailler avec des vitesses lentes, tant que la boule finit

```

```

sa course
099|
100|
101|
102|
103|
104| #liste des quilles au départ, la quille 0 est virtuelle et ne sert à rien, juste
à ce que Q[1] désigne bien la première quille, sans décalage
105| Q=[[0,0,0,0,0],[1,527.5,3208,0,0],[2,375,3472,0,0],[3,680,3472,0,0],
[4,222.5,3736,0,0],[5,527.5,3736,0,0],[6,832.5,3736,0,0],[7,70,4000,0,0],
[8,375,4000,0,0],[9,680,4000,0,0],[10,985,4000,0,0]]
106| B=[630,3000,-2,10] # le choix de -2/10 correspond à un angle de 10 degrés, pas
optimal pour le strike mais plus visuel pour les déviations
107|
108| for i in range(50): #on réalise un cliché des positions 50 fois tous les 3
passages de boucle pour avoir un "film"
109|     plt.figure(i)
110|     for k in range(3):
111|         mouvement(Q,B)
112|         # for i in range(1,11):
113|         #     if i in bouge:
114|         #         diskvide((Q[i][1],Q[i][2]), 61, color="green")#les cercles
vides servent à visualiser la trajectoire
115|         #     else:
116|         #         diskvide((Q[i][1],Q[i][2]), 61, color="red")
117|
118|         ##     diskvide((B[0],B[1]),108,color="blue")
119|
120|
121|
122|     # dessin de la piste
123|     line((0,4076), (1055, 4076), color="black")
124|     line((0,4076), (0, 2500), color="black")
125|     line((1055,4076), (1055, 2500), color="black")
126|
127|
128|     #dessin des positions finales des quilles et de la boule
129|
130|     for i in range(1,11): #on parcourt la liste des quilles
131|         if i in bouge:
132|             disk((Q[i][1],Q[i][2]), 61, color="green") #les quilles heurtées sont
dessinées en vert
133|         else:
134|             disk((Q[i][1],Q[i][2]), 61, color="red") #les autres en rouge
135|
136|         disk((B[0],B[1]),108,color="blue") #la boule est en bleu
137|
138|     # affichage final du résultat
139|
140|     plt.axis('equal') # on choisit un repère orthonormé pour mieux visualiser
les trajectoires
141|     plt.xlim([-50, 1100]) #on définit la fenêtre de représentation: seule la fin
de la piste sera dessinée pour ne pas tout écraser
142|     plt.ylim([2500, 4500])
143|
144|     plt.savefig("fonctions_auxiliaires_matplotlib.png")
145|     plt.show()
146|
147|
148|
149|
#

```

```

150|
151| #Recherche du meilleur angle d'attaque:
152|
153|
154|

```

```

155| for q in range(20,30): #on va tester 30 angles différents d'approche, entre 0 et
-15 degrés environ
156|     compteur=0 #on initialise un compteur de quilles tombées
157|     strikes=0 #on initialise le nombre de strikes
158|     for p in range(150): #Pour chaque angle, on teste les 100 abscisses
initiales correspondant à l'espace entre les quilles 1 et 3
159|         Q=[[0,0,0,0,0],[1,527.5,3208,0,0],[2,375,3472,0,0],[3,680,3472,0,0],
[4,222.5,3736,0,0],[5,527.5,3736,0,0],[6,832.5,3736,0,0],[7,70,4000,0,0],
[8,375,4000,0,0],[9,680,4000,0,0],[10,985,4000,0,0]]
160|
161|         B=[550+p,3000,-q/10,10] #la position initiale de la boule et la direction
de sa vitesse dépendent de p et de q
162|         bouge=[] #on initialise le nombre de quilles qui vont tomber
163|
164|
165|         for k in range(1500): #on fait avancer le mouvement 2500 fois, ce qui
suffit à faire arriver la boule en fin de piste et à toutes les quilles de tomber
166|             mouvement(Q,B)
167|             compteur+=len(bouge) #on cumule les boules tombées
168|             if len(bouge)==10:
169|                 strikes+=1
170|             print(q,compteur,strikes) #pour chaque angle on affiche le résultat obtenu
171|
172| #résultats:
173|
174| 0 1277 29
175| 1 1271 27
176| 2 1264 28
177| 3 1259 26
178| 4 1267 29
179| 5 1277 31
180| 6 1276 38
181| 7 1297 43
182| 8 1293 34
183| 9 1290 35
184| 10 1293 36
185| 11 1314 45
186| 12 1310 45
187| 13 1302 42
188| 14 1302 43
189| 15 1290 37
190| 16 1293 43
191| 17 1311 45
192| 18 1294 43
193| 19 1286 44
194|
195| #meilleur angle pour le total de points pour q=-11 soit environ 6.3 degrés
196|
197| 20 1278 43
198| 21 1290 48
199| 22 1290 50
200| 23 1270 47
201| 24 1261 43
202| 25 1256 36
203| 26 1256 36
204| 27 1253 35
205| 28 1258 34
206| 29 1267 40
207|
208|
209| #pour des angles plus grands, plus de strikes mais en moyenne moins de points
210|
211|
212|
213|
#
214| #Résolution numérique trajectoire boule homogène modèle de friction de Contensou:

```

```

215|
216|
217| from __future__ import division
218| from scipy import *
219| from pylab import *
220| from scipy.integrate import odeint          # Module de résolution des équations
différentielles
221|
222| R=0.11  #rayon de la boule
223| f=0.2   #coefficient de frottement
224| m=6     #masse de la boule
225| eps=0.01 #rayon de la zone circulaire de contact boule-piste
226| N=5     #réaction normale de la piste
227| F0=f*N  #force de friction
228| M0=3*pi*N*f*eps/16 #moment de la force de friction
229| I0=2*m*R**2/5    #moment d'inertie de la boule homogène
230|
231|
232| def deriv(syst, t):
233|     [x,y,vx,vy,om2,om3] = syst          #variables: x,y position, vx,vy
vitesse, om2,om3 vitesses de rotation selon Oy et Oz
234|     dxdt=vx                             #première équa diff
235|     dydt=vy                             #deuxième
236|     dom2dt=F0*3*pi*R*(vx-R*om2)/(I0*(8*eps*abs(om3)+3*pi*sqrt((vx-R*om2)**2+
(vy+R*om3)**2))) #troisième
237|     dom3dt=M0*16*eps*om3/(I0*(16*eps*abs(om3)+15*pi*sqrt((vx-R*om2)**2+
(vy+R*om3)**2))) #quatrième
238|     dvxdt=-F0*3*pi*(vx-R*om2)/(m*(8*eps*abs(om3)+3*pi*sqrt((vx-R*om2)**2+
(vy+R*om3)**2))) #cinquième
239|     dvydt=-F0*3*pi*(vy+R*om3)/(m*(8*eps*abs(om3)+3*pi*sqrt((vx-R*om2)**2+
(vy+R*om3)**2))) #sixième
240|     return [dxdt,dydt,dvxdt,dvydt,dom2dt,dom3dt]          #retour des dérivées du
système
241|
242|
243| # Paramètres d'intégration
244| start = 0
245| end = 3
246| numsteps = 2500
247| t = linspace(start,end,numsteps) #2500 valeurs du temps également réparties dans
les 3 premières secondes du mouvement
248|
249|
250| for k in range(10):
251|     # Conditions initiales et résolution
252|     x0, y0 = 0, 1
253|     vx0, vy0 = 10, 0
254|     om20=0.1+k    #on teste différentes vitesses angulaires (peu concluant pour
om2)
255|     om30=-1
256|
257|
258|     syst_CI=array([x0,y0,vx0,vy0,om20,om30])    # Tableau des CI
259|     Sols=odeint(deriv,syst_CI,t)                # Résolution numérique des équations
différentielles- méthode de Runge-Kutta par défaut
260|
261|     # Récupération des solutions
262|     [x,y,vx,vy,om2,om3] = Sols . T              # Décomposition du tableau des
solutions : Affectation avec transposition
263|
264|
265|     # Graphiques des solutions
266|     plot(x, y, ms=6, mfc='w', mec='b', label=u"A")    # Solution numérique
267|
268|
269|
270| xlim(0, 30)                                     # Limites de l'axe des abscisses
271| xlabel(r'$x$', fontsize=16)                     # Label de l'axe des abscisses

```

```

272| ylim(0.99,1.01)                                # Limites de l'axe des ordonnées
273| ylabel(r'$y$', fontsize=16)                     # Label de l'axe des ordonnées
274|
275| show()
276|
277|
278|
279|
280| for k in range(10):
281|     # Conditions initiales et résolution
282|     x0, y0 = 0, 1
283|     vx0, vy0 = 10, 0
284|     om20=0.1
285|     om30=-1+0.2*k #on recommence avec la vitesse angulaire om3 (beaucoup plus
influente sur la trajectoire)
286|
287|
288|     syst_CI=array([x0,y0,vx0,vy0,om20,om30])      # Tableau des CI
289|     Sols=odeint(deriv,syst_CI,t)                  # Résolution numérique des équations
différentielles- méthode de Runge-Kutta par défaut
290|
291|     # Récupération des solutions
292|     [x,y,vx,vy,om2,om3] = Sols . T                # Décomposition du tableau des
solutions : Affectation avec transposition
293|
294|
295|     # Graphiques des solutions
296|     plot(x, y, ms=6, mfc='w', mec='b', label=u"A") # Solution numérique
297|
298|
299|
300| xlim(0, 30)                                       # Limites de l'axe des abscisses
301| xlabel(r'$x$', fontsize=16)                       # Label de l'axe des abscisses
302| ylim(0.99,1.01)                                # Limites de l'axe des ordonnées
303| ylabel(r'$y$', fontsize=16)                     # Label de l'axe des ordonnées
304|
305| show()
306|
307|
308|
309|
310|
311|
#

```

```

312| #Tracé de la trajectoire avec boule hétérogène. Les coordonnées des points
proviennent de la résolution des équations différentielles par maple:
313|
314|
315|
316|

```

```

317| B=[[0,0.5],[1.001602032834511, .4816020328345113], [2.006407823484468,
.4664078234844686], [3.014414466529971, .4544144665299725], [4.025612026196056,
.4456120261960570], [5.039978311192988, .4399783111929894], [6.057477878543340,
.4374778785433405], [7.078068812150534, .4380688121505352], [8.101712729749659,
.4417127297496594], [9.128378983556459, .4483789835564597], [10.15804370120953,
.4580437012095338], [11.19070339707804, .4707033970780439], [12.22642282855373,
.4864228285537384], [12.33368306989644, .4881722118144287, .1759675519e-1],
[12.43733199886193, .4900479426572986, .1859550009e-1], [12.54094624484968,
.4920260446085142, .1958538180e-1], [12.64452513230326, .4941056807279069,
.2056961911e-1], [12.74806789153637, .4962863615284470, .2155175662e-1],
[12.85157370339331, .4985679755480089, .2253559875e-1], [12.95504174866296,
.5009508126422429, .2352513021e-1], [13.05847126150652, .5034355786414262,
.2452442356e-1], [13.16186158590386, .5060234001042443, .2553753484e-1],
[13.26521223383407, .5087158180859940, .2656838999e-1], [13.36852294360287,
.5115147701371016, .2762066521e-1], [13.47179373644676, .5144225601682667,
.2869766644e-1], [13.57502496932289, .5174418163525516, .2980221334e-1],
[13.67821738168001, .5205754378540832, .3093653454e-1], [13.78137213404085,

```

```

.5238265318297685, .3210218058e-1],[13.88449083643614, .5271983427811573,
.3329996063e-1],[13.98757556512066, .5306941768679321, .3452990761e-1],
[14.09062886655084, .5343173241663147, .3579127457e-1],[14.19365374826521,
.5380709820150503, .3708256282e-1],[14.29665365701656, .5419581825146602,
.3840157984e-1],[14.39963244518352, .5459817269396848, .3974552307e-1],
[14.50259432706740, .5501441293263104, .4111108353e-1],[14.60554382710345,
.5544475708701350, .4249456274e-1],[14.7084857225182, .5588938660842229,
.4389199579e-1],[14.81142498087945, .5634844409989411, .4529927411e-1],
[14.91436670032027, .5682203230936655, .4671226237e-1],[15.01731604504598,
.5731021421783132, .4812690541e-1],[15.12027818704152, .5781301411096277,
.4953932240e-1],[15.22325824960676, .5833041950322195, .5094588677e-1],
[15.32626125544212, .5886238377602313, .5234329182e-1],[15.42929207955409,
.5940882939344688, .5372860233e-1],[15.53235540725970, .5996965156697205,
.5509929319e-1],[15.63545569738913, .6054472225159010, .5645327632e-1],
[15.73859715068670, .6113389436664604, .5778891675e-1],[15.84178368338890,
.6173700614359521, .5910503865e-1],[15.94501890600500, .6235388550808314,
.6040092169e-1],[16.04830610742636, .6298435440469019, .6167628749e-1],
[16.15164824462439, .6362823296960674, .6293127588e-1],[16.25504793833887,
.6428534345066905, .6416641018e-1],[16.35850747527696, .6495551376778047,
.6538255111e-1],[16.46202881740484, .6563858060268942, .6658083906e-1],
[16.56561361888499, .6633439190864188, .6776262563e-1],[16.66926325106372,
.6704280874052931, .6892939578e-1],[16.77297883562808, .6776370632677814,
.7008268383e-1],[16.87676128562862, .6849697433568279, .7122398689e-1],
[16.98061135352599, .6924251632944042, .7235468061e-1],[17.08452968481089,
.7000024844505860, .7347594226e-1],[17.18851687513578, .7077009738735405,
.7458868584e-1],[17.29257352836515, .7155199785967995, .7569351326e-1],
[17.39670031258306, .7234588958766327, .7679068392e-1],[17.50089801096441,
.7315171410661874, .7788010362e-1],[17.60516756456511, .7396941148330047,
.7896133211e-1],[17.70951010452243, .7479891712860277, .8003360683e-1],
[17.81392697184414, .7564015883312195, .8109587977e-1],[17.91841972382907,
.7649305412672643, .8214686360e-1],[18.02299012709975, .7735750803120505,
.8318508321e-1],[18.12764013813068, .7823341124567914, .8420892948e-1],
[18.23237187292309, .7912063878043026, .8521671251e-1],[18.33718756803684,
.8001904903723557, .8620671234e-1],[18.44208953550759, .8092848332299664,
.8717722605e-1]]]
318|
319| abs,ord=[],[] #préparation de la liste des abscisses et ordonnées, séparées pour
matplotlib
320|
321| for x in B:
322|     abs.append(x[0])
323|     ord.append(x[1]-0.35)
324|
325|
326| import matplotlib.pyplot as plt
327| fig, ax = plt.subplots(figsize=(15, 2.7))
328| ax.scatter(18.288, .5275, s=50, facecolor='C0', edgecolor='k') #tracé des 10
boules en leurs positions précises
329| ax.scatter(18.552,.375, s=50, facecolor='C0', edgecolor='k')
330| ax.scatter(18.552,.68, s=50, facecolor='C0', edgecolor='k')
331| ax.scatter(18.816, .2225, s=50, facecolor='C0', edgecolor='k')
332| ax.scatter(18.816,.5275, s=50, facecolor='C0', edgecolor='k')
333| ax.scatter(18.816, .8325, s=50, facecolor='C0', edgecolor='k')
334| ax.scatter(19.08,.07, s=50, facecolor='C0', edgecolor='k')
335| ax.scatter(19.08, .375, s=50, facecolor='C0', edgecolor='k')
336| ax.scatter(19.08, .68, s=50, facecolor='C0', edgecolor='k')
337| ax.scatter(19.08, .985, s=50, facecolor='C0', edgecolor='k')
338|
339|
340| ax.set_ylim(0, 1.05)
341| ax.plot(abs, ord) #tracé de la trajectoire de la boule
342|
343| plt.show()
344|
345|
346|
347|
348| #

```



```

349| # reprise du problème avec 16 boules en carré
350|
351|
352|
353|
354| B=[930,3000,-4,10]
355|
356| #liste des quilles au départ, la quille 0 est virtuelle et ne sert à rien, juste
à ce que Q[1] désigne bien la première quille, sans décalage
357| Q=[[0,0,0,0,0],[1,70,3208,0,0],[2,375,3208,0,0],[3,680,3208,0,0],
[4,985,3208,0,0],[5,70,3472,0,0],[6,375,3472,0,0],[7,680,3472,0,0],[8,985,3472,0,0],
[9,70,3736,0,0],[10,375,3736,0,0],[11,680,3736,0,0],[12,985,3736,0,0],
[13,70,4000,0,0],[14,375,4000,0,0],[15,680,4000,0,0],[16,985,4000,0,0]]
358|
359|
360|
361|
362| for i in range(15):
363|     plt.figure(i)
364|
365|
366|     for k in range(10):
367|         mouvement(Q,B)
368|
369|
370|
371|     # dessin de la piste
372|     line((0,4076), (1055, 4076), color="black")
373|     line((0,4076), (0, 2500), color="black")
374|     line((1055,4076), (1055, 2500), color="black")
375|
376|
377|     #dessin des positions finales des quilles et de la boule
378|
379|     for i in range(1,17):
380|         if i in bouge:
381|             disk((Q[i][1],Q[i][2]), 61, color="green")
382|         else:
383|             disk((Q[i][1],Q[i][2]), 61, color="red")
384|
385|     disk((B[0],B[1]),108,color="blue")
386|
387|     # affichage final du résultat
388|
389|     plt.axis('equal')
390|     plt.xlim([-50, 1100])
391|     plt.ylim([2500, 4500])
392|     # plt.axis('off')
393|     plt.savefig("fonctions_auxiliaires_matplotlib.png")
394|     plt.show()
395|
396|
397|
398|
399| for i in range(180):
400|
401|     mouvement(Q,B)
402|     for i in range(1,16):
403|         if i in bouge:
404|             diskvide((Q[i][1],Q[i][2]), 61, color="green")#les cercles vides
servent à visualiser la trajectoire
405|         else:
406|             diskvide((Q[i][1],Q[i][2]), 61, color="red")
407|
408|     diskvide((B[0],B[1]),108,color="blue")
409|
410|
411|

```



```

412| # dessin de la piste
413| line((0,4076), (1055, 4076), color="black")
414| line((0,4076), (0, 2500), color="black")
415| line((1055,4076), (1055, 2500), color="black")
416|
417|
418|
419|
420| #dessin des positions finales des quilles et de la boule
421|
422| for i in range(1,17):
423|     if i in bouge:
424|         disk((Q[i][1],Q[i][2]), 61, color="green")
425|     else:
426|         disk((Q[i][1],Q[i][2]), 61, color="red")
427|
428| disk((B[0],B[1]),108,color="blue")
429|
430| # affichage final du résultat
431|
432| plt.axis('equal')
433| plt.xlim([-50, 1100])
434| plt.ylim([2500, 4500])
435| # plt.axis('off')
436| plt.savefig("fonctions_auxiliaires_matplotlib.png")
437| plt.show()
438|
439|
440|
441|
442|
443|
444|
445|
446|
447|
448|
449|
450|
451|
452|
453|
454|
455|
456|
457|

```

```

> with(linalg):
Warning, new definition for norm
Warning, new definition for trace
> J:=matrix(3,3,[0.03,0,0,0,0.05,0,0,0,0.03]);a:=0.02;R:=0.11;M:=6
:
> r1:=matrix(3,3,[1,0,0,0,cos(t1(t)), -sin(t1(t)),0,sin(t1(t)),cos(
t1(t))]):
> r2:=matrix(3,3,[cos(t2(t)),0,-sin(t2(t)),0,1,0,sin(t2(t)),0,cos(
t2(t))]):
> r3:=matrix(3,3,[cos(t3(t)), -sin(t3(t)),0,sin(t3(t)),cos(t3(t)),0
,0,0,1]):
> It:=map(simplify,evalm(r1^(-1)&*r2^(-1)&*r3^(-1)&*J&*r3&*r2&*r1+
diag(R**2,R**2,0))):
> r:=matrix(3,1,[0,a,0]):
> rd:=map(simplify,evalm(r1^(-1)&*r2^(-1)&*r3^(-1)&*r)):
> om:=matrix(3,1,[D(t1)(t),D(t2)(t),D(t3)(t)]):
> alpha:=(D@D)(t1)(t),(D@D)(t2)(t),(D@D)(t3)(t)):
> w:=map(simplify,evalm(It&*alpha)):
> f:=map(simplify,crossprod([w[1,1],w[2,1],w[3,1]],[om[1,1],om[2,1
],om[3,1]])):
> gauche:=evalm(It&*alpha):
> aAlpha:=evalm(crossprod([rd[1,1],rd[2,1],rd[3,1]-R],[alpha[1],al
pha[2],alpha[3]])):
> aAlphaZ:=simplify(aAlpha[3]):
> aOm:=crossprod(crossprod([om[1,1],om[2,1],om[3,1]],[rd[1,1],rd[2
,1],rd[3,1]]),[om[1,1],om[2,1],om[3,1]]):
> aOmZ:=simplify(aOm[3]):
> mu:=[0.2,0.2,1]:
> force:=map(simplify,evalm(crossprod([M*rd[1,1],M*rd[2,1],M*rd[3,
1]-M*R],9.8*mu+aOmZ*mu))):
> equations:={ (D@D)(x)(t)-aAlphaZ*mu[1]/M=simplify((9.8+aOmZ)*mu[1
])/M, (D@D)(y)(t)-aAlphaZ*mu[2]/M=simplify((9.8+aOmZ)*mu[2])/M, si
mplify(gauche[1])=simplify(force[1]+f[1]), simplify(gauche[2])=si
mplify(force[2]+f[2]), simplify(gauche[3])=simplify(force[3]+f[3]
), x(0)=0,y(0)=0.5,D(x)(0)=10,D(y)(0)=-0.3,t1(0)=0,t2(0)=0,t3(0)=
0,D(t1)(0)=0,D(t2)(0)=0,D(t3)(0)=0}:
> s:=dsolve(equations,{x(t),y(t),t1(t),t2(t),t3(t)},type=numeric):
> points:=[0,0,0,0,0,0,0,0,0,0,0,0]:
> for k from 1 to 12 do
points[k]:=[op(2,s(k/10)[2]),op(2,s(k/10)[4])]
od:
> force:=evalm(crossprod([rd[1,1],rd[2,1],rd[3,1]-R],[alpha[1],alp
ha[2],alpha[3]])+(om[1,1]**2+om[2,1]**2+om[3,1]**2)*[rd[1,1],rd[
2,1],rd[3,1]]-(om[1,1]*rd[1,1]+om[2,1]*rd[2,1]+om[3,1]*rd[3,1])
*[om[1,1],om[2,1],om[3,1]]):
> equations:={ (D@D)(x)(t)=simplify(force[1])/M, (D@D)(y)(t)=simplif
y(force[2])/M, gauche[1]=f[1], gauche[2]=f[2], gauche[3]=f[3], x(0)=

```

