THION Theophile

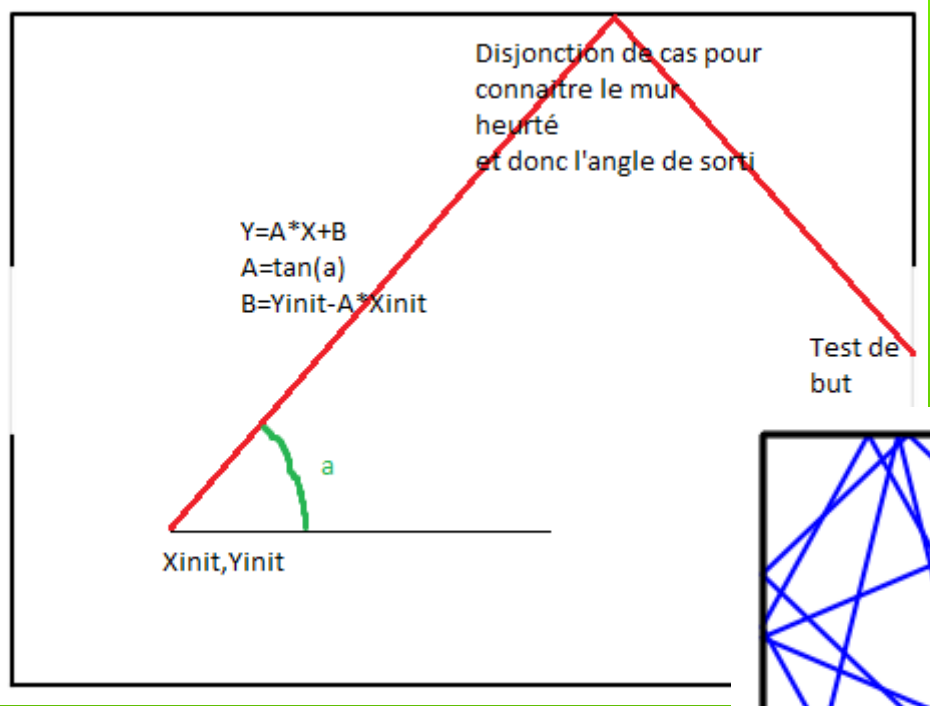
Mouvement d’une balle de baby-foot

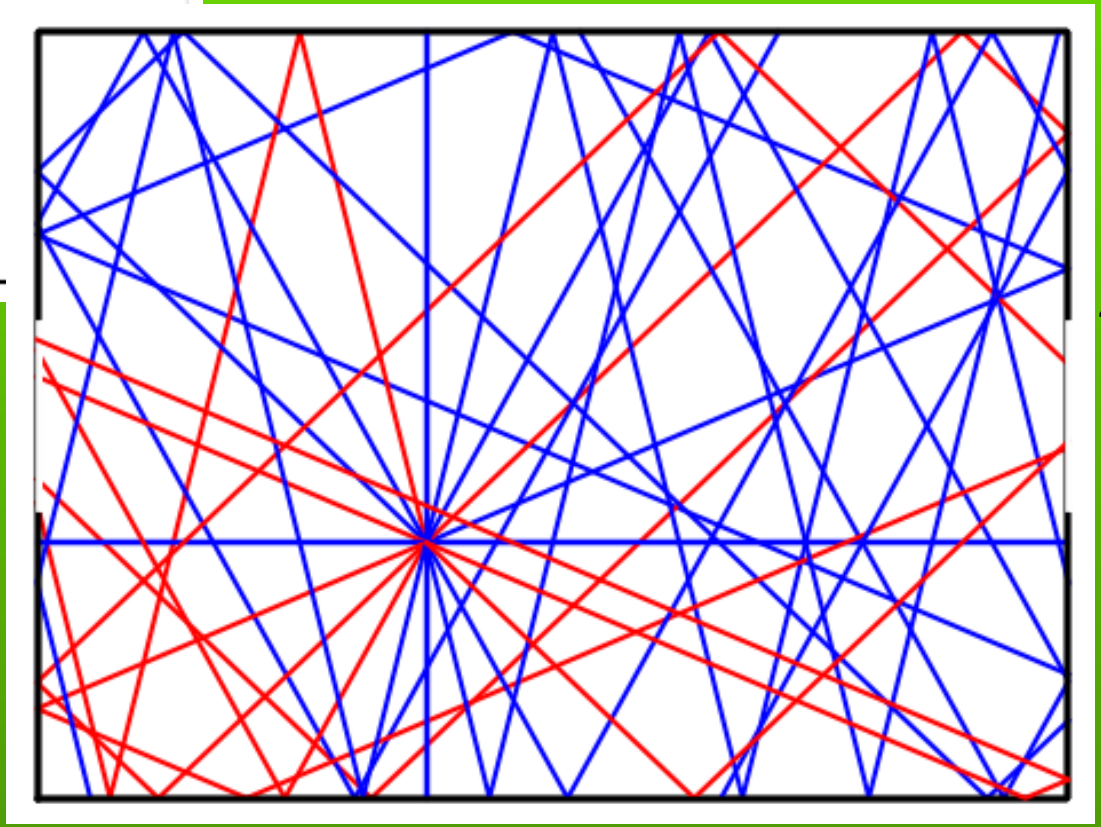
Plan

I. Étude statistiques
    simulation sans joueurs
    simulation avec joueurs

II. Étude mécanique
    frottement
    chocs

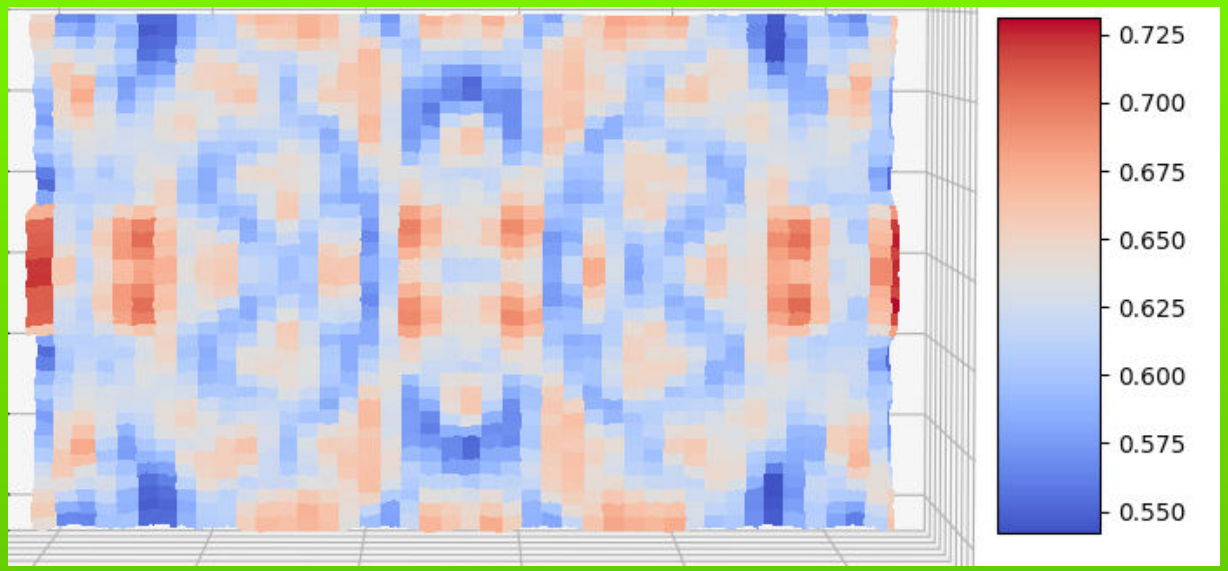
Principe du lancer de rayon



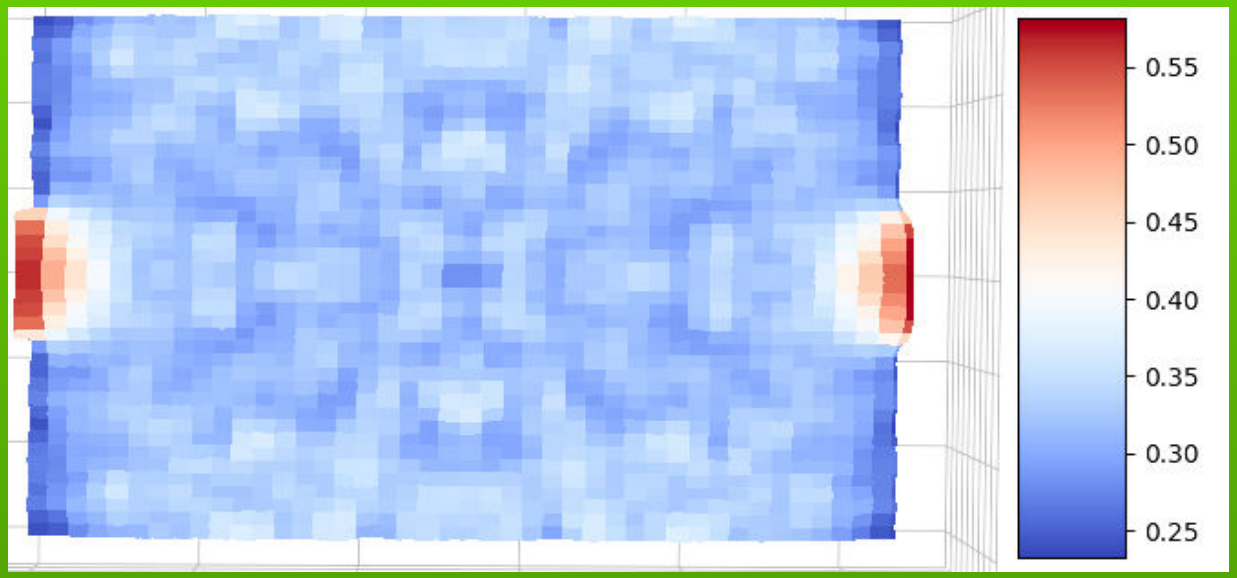


Lancer de
40 rayons

Approximation sans joueur

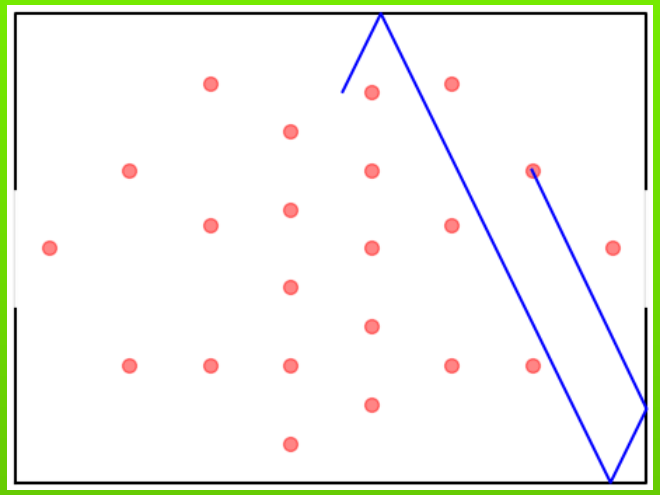


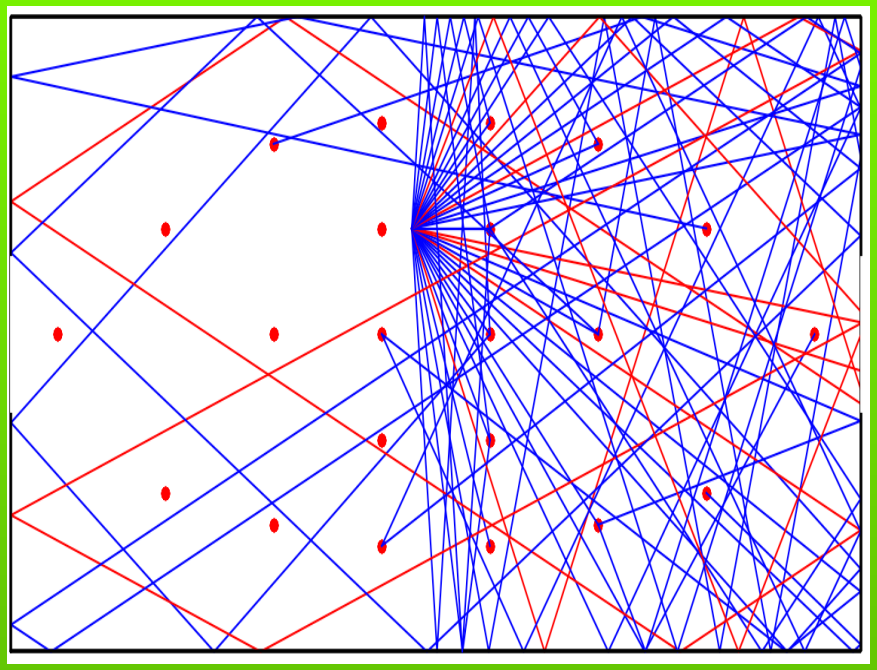
-120x207 points
-Arrêt à 10 rebond
-Lancer de 100 rayons



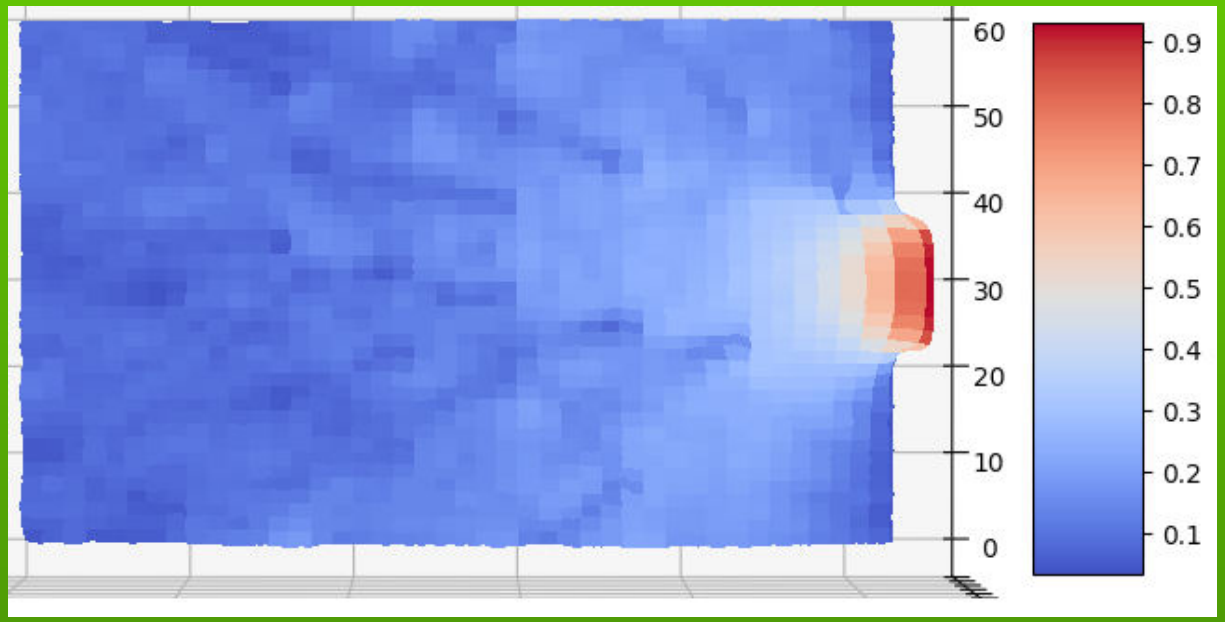
-120x207 points
-Arrêt à 3 rebond
-Lancer de 100 rayons

Approximation avec joueur





Lancer de
40 rayons

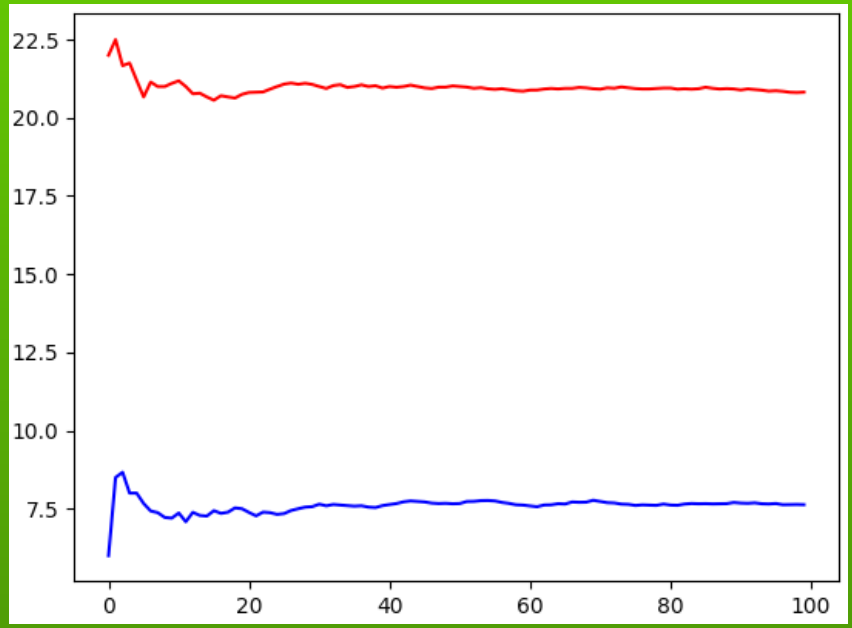


Lancer de 100 rayons sur
120x207 pts
3 rebonds max

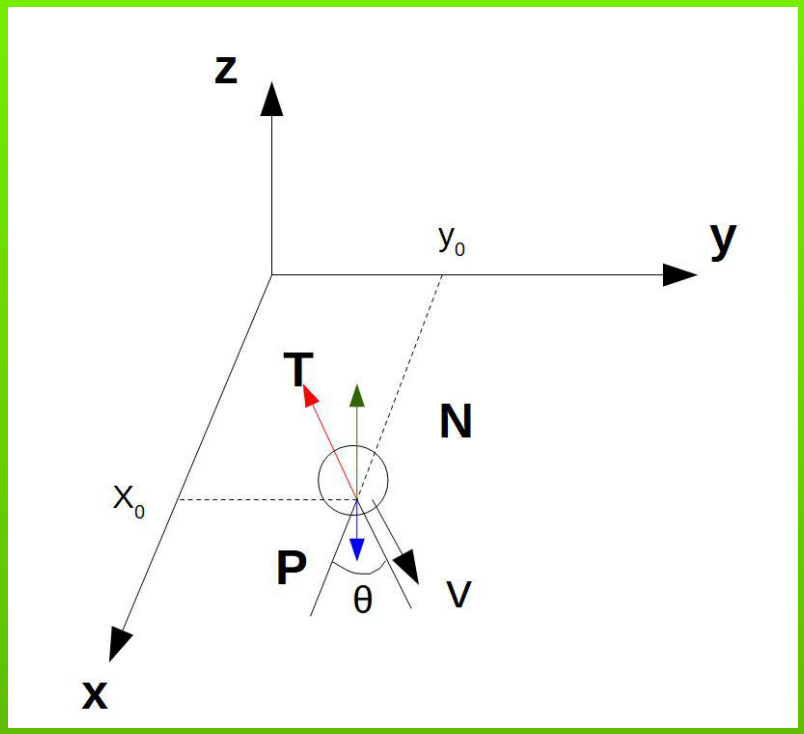
Résultat pour chaque joueur

Lancer de 200 rayons pour 1000 positions de joueurs différentes

En rouge : attaquant centre
En bleu : goal



Etude physique



$$\vec{N} = N \,.\, \vec{z}$$

$$\vec{P} = -P \,.\, \vec{z}$$

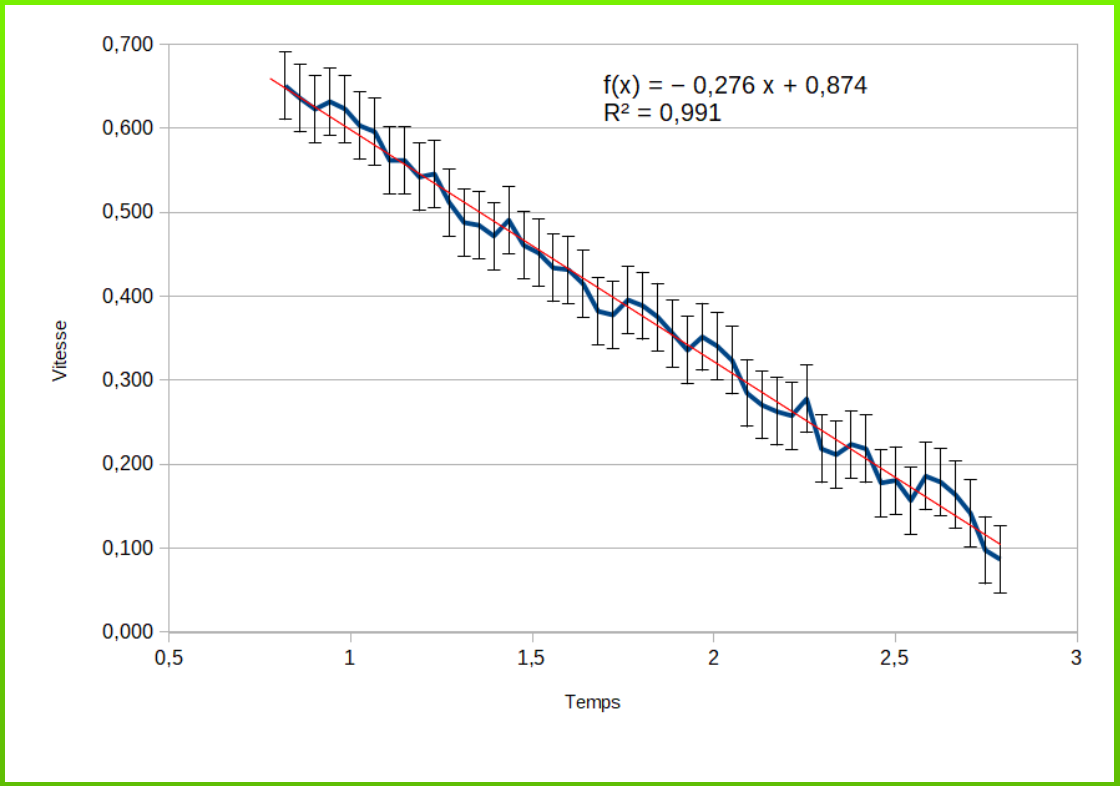
$$\vec{T} = -T \,.\, \cos(\theta) \,.\, \vec{x} - T \,.\, \sin(\theta) \,.\, \vec{y}$$

$$\vec{v} = v \,.\, \cos(\theta) \,.\, \vec{x} + v \,.\, \sin(\theta) \,.\, \vec{y}$$

$$x(t) = -\left(\frac{1}{2}\right) fg\cos(\theta) t^2 + v\cos(\theta) t + x_0$$

$$y(t) = -\left(\frac{1}{2}\right) fg\sin(\theta) t^2 + v\sin(\theta) t + y_0$$

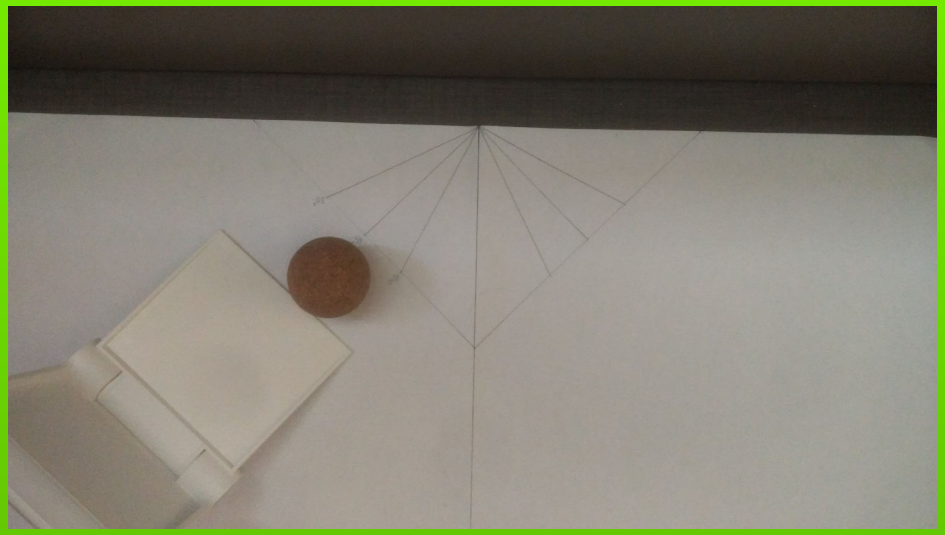
Etude coefficient de frottement



Etude sur tapis de billard

	Coefficient de frottement
Balle/Béton	0,01
Balle/Bois de Baby	0,01
Balle/Tapis billard	0,03

Etude des choc



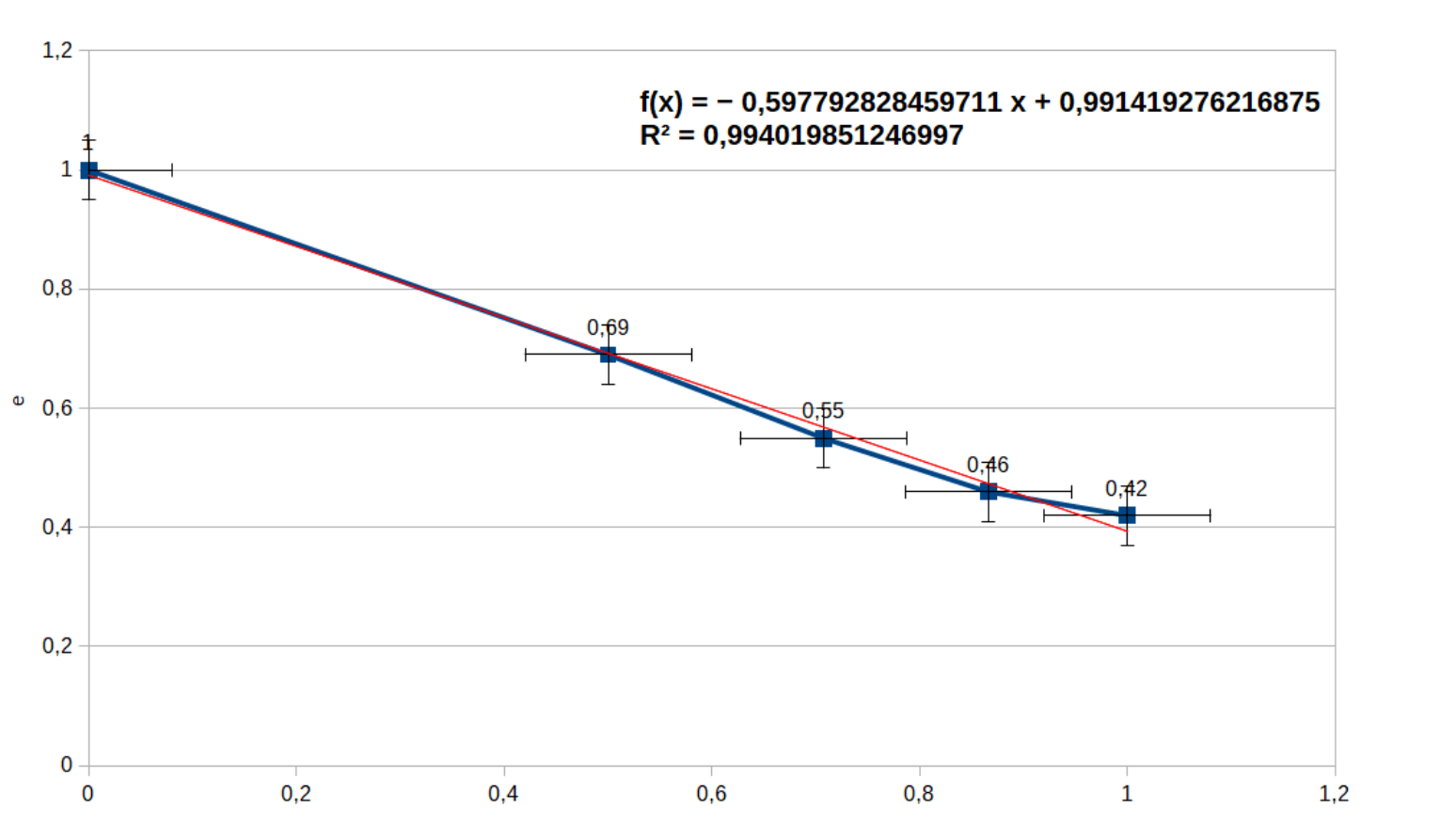
Coefficient de restitution :

$$e = \frac{Vitesse\ après\ collision}{Vitesse\ avant\ collision}$$

Angle (en radians)	0	pi/6	pi/4	pi/3	pi/2
e	1	0,62	0,55	0,49	0,42

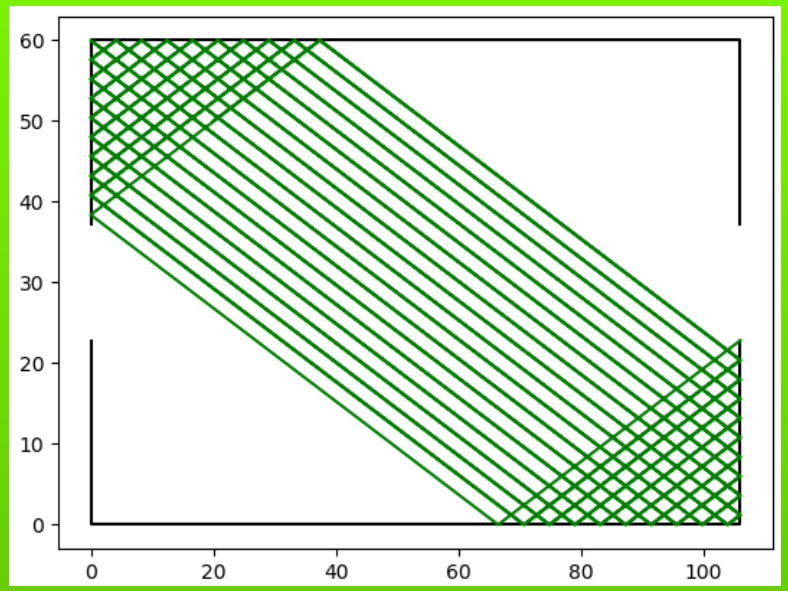
Etude des choc

$$e = f(\sin(\theta))$$

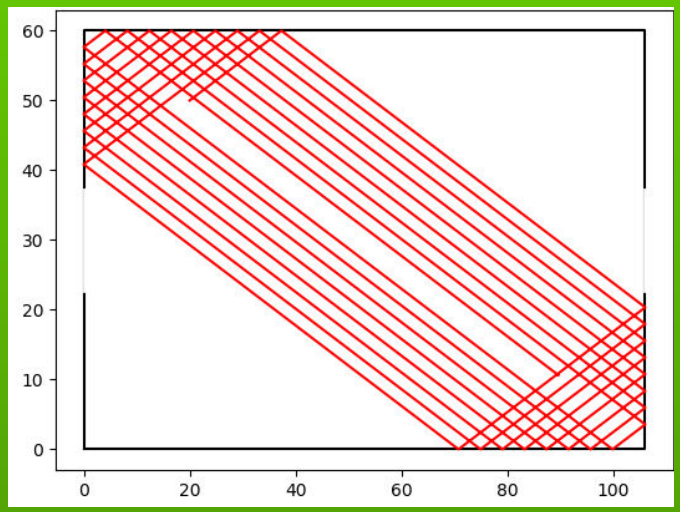


$$Vf = Vi \times (-\lambda \times \sin(\theta) + 1)$$

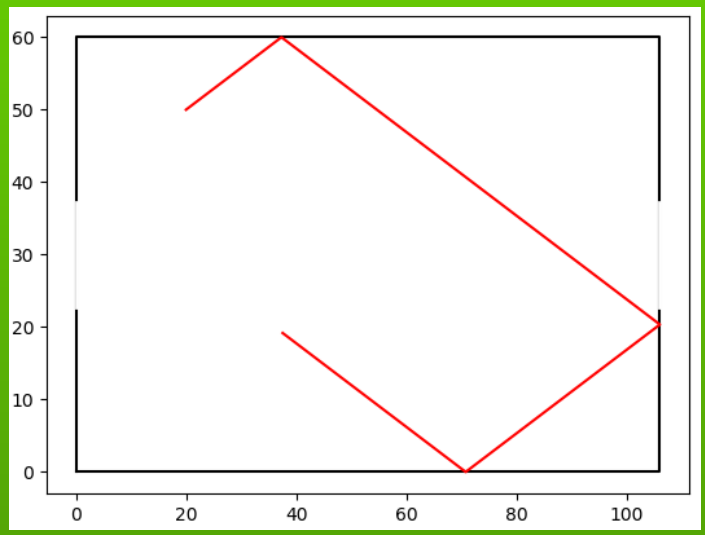
Simulation



Simulation sans frottement et e=1



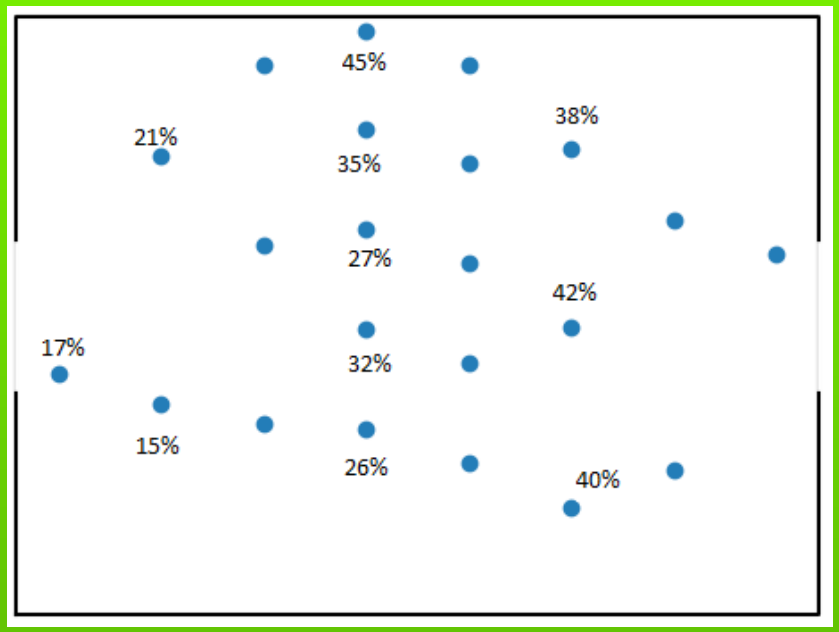
Simulation avec f=0,02 et e=1



Simulation avec f=0,02 et en prenant compte des chocs

Conclusion

Cas intéressants :



Lois de rebond : $$Vf = Vi \times \left(-\lambda \times \sin\left(\theta\right) + 1\right)$$

Importance des rebonds par rapport aux frottements dans le ralentissement

Debut algo1

```
# 106 x 60 cm entre les bars 13.5 , entre les joueurs defense 25 , entre j attaque 18, entre j milieu 10 , 14.5 largeur des cages

from random import *
from math import *
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt
from matplotlib import cm
from matplotlib.ticker import LinearLocator, FormatStrFormatter
import numpy as np



largeurbut=14.5
xmin=0
xmax=106
ymin=0
ymax=60


#ecrire la trajectoire de la balle sous la forme d'une fonction affine

def pointderebond (x,y,angle):
    global largeurbut
    global xmin
    global xmax
    global ymin
    global ymax
    fction=[tan(angle),y-tan(angle)*x]
    #test but

    #but à droite
    if xmax*fction[0]+fction[1]<(ymax/2)+(largeurbut/2) and xmax*fction[0]+fction[1]>(ymax/2)-(largeurbut/2) and cos(angle)>0:

        return([xmax,xmax*fction[0]+fction[1],0,True])

    #but à gauche
    elif xmin*fction[0]+fction[1]<(ymax/2)+(largeurbut/2) and xmin*fction[0]+fction[1]>(ymax/2)-(largeurbut/2) and cos(angle)<0:
        global but
        but=True
        return([xmin,xmin*fction[0]+fction[1],0,True])
```

```
41
42
43      #la balle tape le bord de droite
44      elif xmax*fction[0]+fction[1] < ymax and xmax*fction[0]+fction[1] > ymin and
    cos(angle) > 0 :
45          return([xmax,xmax*fction[0]+fction[1],pi-angle,False])
46      #gauche
47      elif xmin*fction[0]+fction[1] < ymax and xmin*fction[0]+fction[1] > ymin and
    cos(angle)<0 :
48          return([xmin,xmin*fction[0]+fction[1],pi-angle,False])
49      #haut
50      elif fction[0]!=0 and (ymax-fction[1])/fction[0] >xmin and (ymax-
    fction[1])/fction[0] < xmax and sin(angle)>0:
51          return([(ymax-fction[1])/fction[0],ymax,-angle,False])
52      #bas
53      elif fction[0]!=0 and (ymin-fction[1])/fction[0] >xmin and (ymin-
    fction[1])/fction[0] <xmax and sin(angle)<0:
54          return([(ymin-fction[1])/fction[0],ymin,-angle,False])
55
56
57  def trajectoire (xinit,yinit , angle ):
58      if xinit<xmin or xinit>xmax or yinit<ymin or yinit >ymax :
59          return ("la balle n'est pas dans le terrain")
60      stop=False
61      nbrebond=0
62      x=xinit
63      y=yinit
64      a=angle
65      X=[x]
66      Y=[y]
67      while not(stop) :
68
69          liste = pointderebond (x,y,a)
70          X.append(liste[0])
71          Y.append(liste[1])
72          if liste[3]==True :
73              # plt.plot([0,106,106,0,0],[0,0,60,60,0],c='black')
74              # plt.plot([0,0],[22.75,37.25],'w-',lw=2)
75              # plt.plot([106,106],[22.75,37.25],'w-',lw=2)
76              # plt.plot(X,Y,color='red')
77              # plt.show()
78              return([nbrebond,liste[0],liste[1],liste[3]])
79          else:
80              nbrebond = nbrebond +1
81              x=liste[0]
```

```
        else:
            nbrebond = nbrebond +1
            x=liste[0]
            y=liste[1]
            a=liste[2]
        if nbrebond > 3 :
            # plt.plot([0,106,106,0,0],[0,0,60,60,0],c='black')
            # plt.plot([0,0],[22.75,37.25],'w-',lw=2)
            # plt.plot([106,106],[22.75,37.25],'w-',lw=2)
            # plt.plot(X,Y,color='blue')
            # plt.show()
            return([nbrebond,liste[0],liste[1],liste[3]])


def testpoint (xinit,yinit,nbdetest):
    angleinter=2*pi/nbdetest
    angle=0
    nbdereussite=0
    for i in range (0,nbdetest) :
        liste=trajectoire(xinit,yinit,angle)
        if liste [3]==True:
            nbdereussite=nbdereussite+1
        angle=angle+angleinter
    return (nbdereussite)



def touslespoints (nbdex , nbdey ):
    retour=[]
    pasx=xmax/nbdex
    pasy=ymax/nbdey
    for i in range (0,nbdex):
        for j in range (0,nbdey):
            nbdereussite=testpoint(i*pasx+0.00000000001,j*pasy+0.00000000000001,10
0)
            retour.append([i*pasx,j*pasy,nbdereussite])
    return(retour)


def affichagematrice (nbdex,nbdey):
```

```
118
119 def affichagematrice (nbdex,nbdey):
120
121     matrice=touslespoints(nbdex,nbdey)
122     longueur=len(matrice)
123     print(matrice)
124     X=[]
125     Y=[]
126     Z=[]
127     for i in range (0,longueur):
128         X.append(matrice[i][0])
129         Y.append(matrice[i][1])
130         Z.append(matrice[i][2])
131     print(X)
132     print(Y)
133     print(Z)
134     plt.scatter(X,Y,s=50,c=Z,alpha=0.5)
135     plt.show()
136
137
138
139
140
141
142 #fonctionne
143 def figure():
144     fig = plt.figure()
145     ax = fig.gca(projection='3d')
146
147     # Make data.
148     X = np.arange(0.0000001, 106, 0.5)
149     Y = np.arange(0.0000001, 60, 0.5)
150     X, Y = np.meshgrid(X, Y)
151     Z=np.zeros((len(X),len(X[0])))
152
153
154     for i in range (len(X)):
155         for j in range (len(X[i])):
156
157             print(int((Y[i][j]/60)*100),"%")
158
159             Z[i][j]=testpoint(X[i][j],Y[i][j],100)/100
160
161     print(Z)
162     # Plot the surface.
```

```
    fig = plt.figure()
    ax = fig.gca(projection='3d')

    # Make data.
    X = np.arange(0.0000001, 106, 0.5)
    Y = np.arange(0.0000001, 60, 0.5)
    X, Y = np.meshgrid(X, Y)
    Z=np.zeros((len(X),len(X[0])))


    for i in range (len(X)):
        for j in range (len(X[i])):

            print(int((Y[i][j]/60)*100),"%")

            Z[i][j]=testpoint(X[i][j],Y[i][j],100)/100

    print(Z)
    # Plot the surface.
    surf = ax.plot_surface(X, Y, Z,cmap=cm.RdBu_r,linewidth=0,antialiased=False)

    # Customize the z axis.
    ax.set_zlim(0, 1)
    ax.zaxis.set_major_locator(LinearLocator(10))
    ax.zaxis.set_major_formatter(FormatStrFormatter('%.02f'))

    # Add a color bar which maps values to colors.
    fig.colorbar(surf, shrink=0.5, aspect=5)

    plt.show()
```

Fin Algo1

Debut algo2

```
# 106 x 60 cm entre les bars 13.5 , entre les joueurs defense 25 , entre j attaque
18, entre j milieu 10 , 14.5 largeur des cages

from random import *
from math import *
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt
from matplotlib import cm
from matplotlib.ticker import LinearLocator, FormatStrFormatter
import numpy as np


#creation d'une matrice aleatoire des joueurs de babyfoot

largeurbut=14.5
xmin=0
xmax=106
ymin=0
ymax=60
entrebar=13.5
entredef=25
entreatt=18
entremil=10
largeurjoueur=1

positionbars=[]
a=random()
a=a*30+15
positionbars.append(a)
a=random()
a=a*35
positionbars.append(a)
a=random()
a=a*24
positionbars.append(a)
a=random()
a=a*20
positionbars.append(a)
a=random()
a=a*20
positionbars.append(a)
a=random()
a=a*24
positionbars.append(a)
a=random()
```

```
a=a*24
positionbars.append(a)
a=random()
a=a*35
positionbars.append(a)
a=random()
a=a*30+15
positionbars.append(a)


premierebar=5.75
matricejoueurs=[]
matricejoueurs.append([premierebar,positionbars[0],"gauche"])
for i in range (0,2):
    matricejoueurs.append([premierebar +
entrebar,positionbars[1]+i*entredef,"gauche"])
for i in range (0,3):
    matricejoueurs.append([premierebar +
entrebar*2,positionbars[2]+i*entreatt,"droite"])
for i in range (0,5):
    matricejoueurs.append([premierebar+entrebar*3,positionbars[3]+i*entremil,"gauc
he"])
for i in range (0,5):
    matricejoueurs.append([premierebar+entrebar*4,positionbars[4]+i*entremil,"droi
te"])
for i in range (0,3):
    matricejoueurs.append([premierebar+entrebar*5,positionbars[5]+i*entreatt,"gauc
he"])
for i in range (0,2):
    matricejoueurs.append([premierebar+entrebar*6,positionbars[6]+i*entredef,"droi
te"])
matricejoueurs.append([premierebar+entrebar*7,positionbars[7],"droite"])




def pointderebond (x,y,angle):
    global largeurbut
    global xmin
    global xmax
    global ymin
    global ymax
    global matricejoueurs
    global largeurjoueur
```

```
    global largeurjoueur
    fction=[tan(angle),y-tan(angle)*x]
    X,Y=x,y
    #test rencontre joueur
    for i in range (len(matricejoueurs)):
        x=matricejoueurs[i][0]
        y=matricejoueurs[i][1]
        if fction[0]*x+fction[1] <= y + largeurjoueur and fction[0]*x+fction[1] >= y - largeurjoueur and (X-x)*cos(angle)<=0 :
            return([x,y,0,"arreté"])
        if fction[0]==0 and fction[1]<= x + largeurjoueur and fction[1]>= x - largeurjoueur:
            return([x,y,0,"arreté"])
        if fction[0]!=0 :
            if (y-fction[1])/fction[0] <= x + largeurjoueur and (y-fction[1])/fction[0] >= x - largeurjoueur and (X-x)*cos(angle)<=0 :
                return([x,y,0,"arreté"])



    #test but
    #but à droite
    if xmax*fction[0]+fction[1]<(ymax/2)+(largeurbut/2) and xmax*fction[0]+fction[1]>(ymax/2)-(largeurbut/2) and cos(angle)>0:

        return([xmax,xmax*fction[0]+fction[1],0,True])

    #but à gauche
    # elif xmin*fction[0]+fction[1]<(ymax/2)+(largeurbut/2) and xmin*fction[0]+fction[1]>(ymax/2)-(largeurbut/2) and cos(angle)<0:
    #     global but
    #     but=True
    #     return([xmin,xmin*fction[0]+fction[1],0,True])

    #la balle tape le bord de droite
    elif xmax*fction[0]+fction[1] < ymax and xmax*fction[0]+fction[1] > ymin and cos(angle) > 0 :
        return([xmax,xmax*fction[0]+fction[1],pi-angle,False])
    #gauche
    elif xmin*fction[0]+fction[1] < ymax and xmin*fction[0]+fction[1] > ymin and cos(angle)<0 :
        return([xmin,xmin*fction[0]+fction[1],pi-angle,False])
    #haut
    elif fction[0]!=0 and (ymax-fction[1])/fction[0] >xmin and (ymax-fction[1])/fction[0] < xmax and sin(angle)>0:
```

```
    elif fction[0]!=0 and (ymax-fction[1])/fction[0] >xmin and (ymax-
fction[1])/fction[0] < xmax and sin(angle)>0:
        return([(ymax-fction[1])/fction[0],ymax,-angle,False])
    #bas
    elif fction[0]!=0 and (ymin-fction[1])/fction[0] >xmin and (ymin-
fction[1])/fction[0] <xmax and sin(angle)<0:
        return([(ymin-fction[1])/fction[0],ymin,-angle,False])



def trajectoire (xinit,yinit , angle ):
    global matricejoueurs
    if xinit<xmin or xinit>xmax or yinit<ymin or yinit >ymax :
        return ("la balle n'est pas dans le terrain")
    stop=False
    nbrebond=0
    x=xinit
    y=yinit
    a=angle
    X=[x]
    Y=[y]
    affichage=True
    while not(stop) :

        liste = pointderebond (x,y,a)
        X.append(liste[0])
        Y.append(liste[1])

        if affichage and (liste[3] =="arreté" or liste[3]==True or nbrebond > 3) :
            plt.plot([0,106,106,0,0],[0,0,60,60,0],c='black')
            plt.plot([0,0],[22.75,37.25],'w-',lw=2)
            plt.plot([106,106],[22.75,37.25],'w-',lw=2)
            plt.plot(X,Y,color='blue')
            x=[]
            y=[]
            for i in range (0,len(matricejoueurs)):
                x.append(matricejoueurs[i][0])
                y.append(matricejoueurs[i][1])
            plt.scatter(x,y,s=50,c='red',alpha=0.5)
            plt.show()


        if liste[3] =="arreté" :
            return([nbrebond,liste[0],liste[1],False])
```

```
155
156
157         if liste[3] =="arreté" :
158             return([nbrebond,liste[0],liste[1],False])
159
160         if liste[3]==True :
161             return([nbrebond,liste[0],liste[1],liste[3]])
162
163         else:
164             nbrebond = nbrebond +1
165             x=liste[0]
166             y=liste[1]
167             a=liste[2]
168         if nbrebond > 10 :
169             return([nbrebond,liste[0],liste[1],liste[3]])
170
171
172
173
174 def testpoint (xinit,yinit,nbdetest):
175     angleinter=pi/nbdetest
176     angle=-pi/2
177     nbdereussite=0
178     for i in range (0,nbdetest) :
179         liste=trajectoire(xinit,yinit,angle)
180         if liste [3]==True:
181             nbdereussite=nbdereussite+1
182         angle=angle+angleinter
183     return (nbdereussite)
184
185
186
187 def figure():
188     fig = plt.figure()
189     ax = fig.gca(projection='3d')
190
191     # Make data.
192     X = np.arange(0.0000001, 106, 0.5)
193     Y = np.arange(0.0000001, 60, 0.5)
194     X, Y = np.meshgrid(X, Y)
195     Z=np.zeros((len(X),len(X[0])))
196
197
198     for i in range (len(X)):
199         for j in range (len(X[i])):
```

```
197
198     for i in range (len(X)):
199         for j in range (len(X[i])):
200
201             print(int((Y[i][j]/60)*100),"%")
202             Z[i][j]=testpoint(X[i][j],Y[i][j],100)/100
203
204     print(Z)
205     # Plot the surface.
206     surf = ax.plot_surface(X, Y, Z,cmap=cm.coolwarm,linewidth=0,antialiased=False)
207
208     # Customize the z axis.
209     ax.set_zlim(0, 1)
210     ax.zaxis.set_major_locator(LinearLocator(10))
211     ax.zaxis.set_major_formatter(FormatStrFormatter('%.02f'))
212
213     # Add a color bar which maps values to colors.
214     fig.colorbar(surf, shrink=0.5, aspect=5)
215
216     plt.show()
217
218
219
```

Fin algo2

Debut algo avec les lois

```
def rebond(vf,xf,yf,angf,bord):
        # vi=vf*(-0.455*(angf%(2*pi))+1)
    print(bord)
    print('angle avant rebond',angf)
    print('vitesse avant',vf)

    if (bord=="haut" and cos(angf)>0)or (bord=="bas" and cos(angf)<0):
        tetavoulu=angf%(pi)
        angi=-angf
    elif (bord=="haut" and cos(angf)<0)or (bord=="bas" and cos(angf)>0):
        tetavoulu=pi-(angf%(pi))
        angi=-angf
    elif bord=="droite" or bord=="gauche":
        tetavoulu=(pi/2)- (abs(angf)%(pi))
        angi=pi-angf
    vi=vf*(-0.455*tetavoulu+1)
    print('tetavoulu',tetavoulu)
    print('vitesse apres',vi)
    return(vi,xf,yf,angi)
```

```
45  def trajectoire(vi,xi,yi,angi):
46      global largeurbut
47      global xmin
48      global xmax
49      global ymin
50      global ymax
51      global g
52      global f
53      affichage=True
54      but=False
55      dparcouru=0
56      fction=[tan(angi),yi-xi*tan(angi)]
57      X=[xi]
58      Y=[yi]
59  #     determination du mur d'en face
60      #droite
61      if xmax*fction[0]+fction[1] < ymax and xmax*fction[0]+fction[1] > ymin and
    cos(angi) > 0 :
62          mur="droite"
63          ptface=[xmax,fction[0]*xmax+fction[1]]
64      #gauche
65      elif xmin*fction[0]+fction[1] < ymax and xmin*fction[0]+fction[1] > ymin and
    cos(angi)<0 :
66          mur="gauche"
67          ptface=[xmin,fction[0]*xmin+fction[1]]
68      #haut
69      elif fction[0]!=0 and (ymax-fction[1])/fction[0] >xmin and (ymax-
    fction[1])/fction[0] < xmax and sin(angi)>0:
70          mur="haut"
71          ptface=[(ymax-fction[1])/fction[0],ymax]
72      #bas
73      elif fction[0]!=0 and (ymin-fction[1])/fction[0] >xmin and (ymin-
    fction[1])/fction[0] <xmax and sin(angi)<0:
74          mur="bas"
75          ptface=[(ymin-fction[1])/fction[0],ymin]
76
77      xf=ptface[0]
78      yf=ptface[1]
79      dmax=(vi**2)/(2*f*g)
80      atteindbord= dmax> sqrt(((yf-yi)**2)+(xf-xi)**2)
81      d=min(dmax,sqrt(((yf-yi)**2)+(xf-xi)**2))
82      dparcouru=dparcouru+d
83      vf=sqrt((vi**2) - 2*d*f*g)
84
85
```

```
85
86      while atteindbord :
87          if affichage :
88              X.append(xf)
89              Y.append(yf)
90
91          vi,xi,yi,angi=rebond(vf,xf,yf,angi,mur)
92
93          fction=[tan(angi),yi-xi*tan(angi)]
94
95          if xmax*fction[0]+fction[1] < ymax and xmax*fction[0]+fction[1] > ymin and
    cos(angi) > 0 :
96              mur="droite"
97              ptface=[xmax,fction[0]*xmax+fction[1]]
98          #gauche
99          elif xmin*fction[0]+fction[1] < ymax and xmin*fction[0]+fction[1] > ymin
    and cos(angi)<0 :
100             mur="gauche"
101             ptface=[xmin,fction[0]*xmin+fction[1]]
102         #haut
103         elif fction[0]!=0 and (ymax-fction[1])/fction[0] >xmin and (ymax-
    fction[1])/fction[0] < xmax and sin(angi)>0:
104             mur="haut"
105             ptface=[(ymax-fction[1])/fction[0],ymax]
106         #bas
107         elif fction[0]!=0 and (ymin-fction[1])/fction[0] >xmin and (ymin-
    fction[1])/fction[0] <xmax and sin(angi)<0:
108             mur="bas"
109             ptface=[(ymin-fction[1])/fction[0],ymin]
110
111         xf=ptface[0]
112         yf=ptface[1]
113         dmax=(vi**2)/(2*f*g)
114         atteindbord= dmax> sqrt(((yf-yi)**2)+(xf-xi)**2)
115         d=min(dmax,sqrt(((yf-yi)**2)+(xf-xi)**2))
116         dparcouru=dparcouru+d
117         #test but
118         if (mur=='droite' or mur=='gauche') and yf>(ymax/2)-(largeurbut/2) and
    yf<(ymax/2)+(largeurbut/2) and atteindbord :
119             X.append(xf)
120             Y.append(yf)
121             but=True
122             break
123
```

```
            break


        vf=sqrt(abs((vi**2) - 2*d*f*g))


    if but :
        xarret=xf
        yarret=yf
        if affichage :
                plt.plot([0,106,106,0,0],[0,0,60,60,0],c='black')
                plt.plot([0,0],[22.75,37.25],'w-',lw=2)
                plt.plot([106,106],[22.75,37.25],'w-',lw=2)
                plt.plot(X,Y,color='red')
                plt.show()
        return(xarret,yarret,dparcouru,but)



    else :
        xarret=dmax*cos(angi)+xi
        yarret=dmax*sin(angi)+yi
        X.append(xarret)
        Y.append(yarret)
        if affichage :
                plt.plot([0,106,106,0,0],[0,0,60,60,0],c='black')
                plt.plot([0,0],[22.75,37.25],'w-',lw=2)
                plt.plot([106,106],[22.75,37.25],'w-',lw=2)
                plt.plot(X,Y,color='blue')
                # x=[]
                # y=[]
                # for i in range (0,len(matricejoueurs)):
                #     x.append(matricejoueurs[i][0])
                #     y.append(matricejoueurs[i][1])
                # plt.scatter(x,y,s=50,c='red',alpha=0.5)
                plt.show()

        return(xarret,yarret,dparcouru,but)
```