```
# C:\Users\charl\Desktop\FINAL_TIPE.py

# TIPE MABIT 2025

import matplotlib.pyplot as plt
import matplotlib.animation as animation
import requests as r
import threading as th
import numpy as np

"""
X = [x,y,z] # position 3D
A = [ax,ay,az] # postion angulaire
W = [wx,wy,wz] # vitesse angulaire
"""

def requete(IPPORT,souhait=["gyr_time","gyrX","gyrY","gyrZ"]):
    demande,reponse = "",[]
    for k in souhait: demande+=k+"&"  # création de la requête
    requete = "http://"+IPPORT+"/get?"+demande
    try:
        data = r.get(url=requete).json() # envoi et retour de la requête
        for k in souhait:
            reponse += [data["buffer"][k]["buffer"][0]] # formalisation des données
    except Exception as e: pass # print(e) # affiche erreur
    return reponse

def integrale(L,Lt,V_init=0): return V_init + np.trapezoid(L,x=Lt) # méthode des trapèze

def decomp_vecteurs(point_liaison,norme,a1,a2): # création du point suivant à partir du précédent et d'un changement de base (sphériques - capteur - à cartésiennes - affichage)
    x = point_liaison.X[0] + norme*np.cos(a1)*np.cos(a2)
    y = point_liaison.X[1] + norme*np.cos(a1)*np.sin(a2)
    z = point_liaison.X[2] + norme*np.sin(a1)
    return [x,y,z]

def rotations(theta,phi):
    Rtheta = np.array([[1,0,0],[0,np.cos(theta),-np.sin(theta)],[0,np.sin(theta),np.cos(theta)]])
    P = np.array([[0,1,0],[-1,0,0],[0,0,1]]) # matrice de passage
    Rphi = np.array([[1,0,0],[0,np.cos(phi),-np.sin(phi)],[0,np.sin(phi),np.cos(phi)]])
    P_1 = np.array([[0,-1,0],[1,0,0],[0,0,1]]) # inverse de P
    return Rtheta @ P_1 @ Rphi @ P # produit de matrices

def rotations(theta,phi):
    Rtheta = np.array([[1,0,0],[0,np.cos(theta),-np.sin(theta)],[0,np.sin(theta),np.cos(theta)]])
    Rphi = np.array([[1,0,0],[0,np.cos(phi),-np.sin(phi)],[0,np.sin(phi),np.cos(phi)]])
    return Rtheta @ Rphi

def dissymetrie(ag,ad): return round(abs(ag - ad),2) # valeur affichée

def epsilon(valeur,n=3):
    if abs(valeur) <= 10**(-n): return 0.
    return valeur

# def moy_gliss(valeur,L): pass

def pres(c,data): ax.plot3D(*[[],[],[]],c,label=data) # affichage légende matplotlib

class cpt:
    def __init__(self,IP):
        self.IP = IP
        self.t,self.A = 0.,[0.,0.,0.]
        def mesures(self):
            Lt,LW = [self.t],[[0.,0.,0.]]
            try:
                for h in ["stop","clear","start"]: r.get(url="http://"+self.IP+"/control?cmd="+h) # gestion début phyphox
                while True:
                    R = requete(self.IP)
                    if R[3] != None:
                        Lt.append(R[0]),LW.append(R[-3:])
                        self.A = [integrale([LW[-2][coord],LW[-1][coord]],Lt[-2:],self.A[coord]) for coord in range(3)] # intégration
                        # self.A = epsilon(self.A)
                        # self.A = moy_gliss(self.A,LA) # position
                        # self.W = moy_gliss(self.W,LW) # vitesse
            except Exception as e: pass # print(e) # affiche erreur
        th.Thread(target=mesures,args=(self,)).start()

    def pos(self):
        """ renvoie la position angulaire"""
        return self.A

class sim:
    def __init__(self):
        self.t,self.A = 0.,[0.,0.,0.]
```

```
082|
083| class pt:
084|     def __init__(self,X,col="ko"):
085|         self.X = X
086|         self.col = col
087|
088|     def afh(self):
089|         """ affiche sur un matplotlib 3D son point """
090|         ax.plot3D(*self.X,self.col)
091|
092| class sld:
093|     def __init__(self,ptA,ptB,col="k-"):
094|         self.ptA,self.ptB = ptA,ptB
095|         self.col = col
096|
097|     def afh(self,afh_pt=False):
098|         """ affiche sur un matplotlib 3D son solide, avec son 2nd point associé si besoin """
099|         if afh_pt: self.ptB.afh()
100|         ax.plot3D(*[[self.ptA.X[k],self.ptB.X[k]] for k in range(3)],self.col)
101|
102| class rep:
103|     def __init__(self,pt,A=np.array([0,0,0])):
104|         self.X,self.A = pt.X,A
105|
106|     def afh(self):
107|         """ affiche sur un matplotlib 3D un repère en son point"""
108|         M = rotations(self.A[0],self.A[1])
109|         e = 0.15
110|         ax_x = M @ np.array([1, 0, 0]) * e
111|         ax_y = M @ np.array([0, 1, 0]) * e
112|         ax_z = M @ np.array([0, 0, 1]) * e
113|
114|         ax.quiver(*self.X, *ax_x, color='r')
115|         ax.quiver(*self.X, *ax_y, color='g')
116|         ax.quiver(*self.X, *ax_z, color='b')
117|
118| # mensurations (Charles)
119| cm = 10**(-2)
120|
121| pied = 28*cm
122| basjambe = 44*cm
123| hautjambe = 45*cm
124|
125| bassin = 30*cm
126| jambedos = 14*cm
127| dos = 50*cm
128| tete = 30*cm
129|
130| epaules = 34*cm
131| hautbras = 34*cm
132| avantbras = 30*cm
133| main = 19*cm
134|
135| # squelette initial
136| bas_dos = pt([0,0,0],"ro")
137| haut_dos = pt([bas_dos.X[0],bas_dos.X[1],bas_dos.X[2]+dos])
138| colonne = sld(bas_dos,haut_dos)
139|
140|
141| epaule_g,epaule_d = pt([haut_dos.X[0]-epaules/2,haut_dos.X[1],haut_dos.X[2]],"yo"),pt([haut_dos.X[0]
+epaules/2,haut_dos.X[1],haut_dos.X[2]],"yo")
142| larg_epaules = sld(epaule_g,epaule_d)
143| face = pt([haut_dos.X[0],haut_dos.X[1],haut_dos.X[2]+tete],"go")
144| cou = sld(haut_dos,face)
145| coude_g,coude_d = pt([epaule_g.X[0],epaule_g.X[1],epaule_g.X[2]-
hautbras]),pt([epaule_d.X[0],epaule_d.X[1],epaule_d.X[2]-hautbras])
146| bras_h_g,bras_h_d = sld(epaule_g,coude_g),sld(epaule_d,coude_d)
147| poignet_g,poignet_d = pt([coude_g.X[0],coude_g.X[1],coude_g.X[2]-
avantbras]),pt([coude_d.X[0],coude_d.X[1],coude_d.X[2]-avantbras])
148| bras_b_g,bras_b_d = sld(coude_g,poignet_g),sld(coude_d,poignet_d)
149| doigt_g,doigt_d = pt([poignet_g.X[0],poignet_g.X[1],poignet_g.X[2]-
main],"go"),pt([poignet_d.X[0],poignet_d.X[1],poignet_d.X[2]-main],"go")
150| main_g,main_d = sld(poignet_g,doigt_g),sld(poignet_d,doigt_d)
151|
152| haut_jambe_g,haut_jambe_d = pt([bas_dos.X[0]-bassin/2,bas_dos.X[1],bas_dos.X[2]]),pt([bas_dos.X[0]
+bassin/2,bas_dos.X[1],bas_dos.X[2]])
153| larg_bassin = sld(haut_jambe_g,haut_jambe_d)
154| genou_g,genou_d = pt([haut_jambe_g.X[0],haut_jambe_g.X[1],haut_jambe_g.X[2]-
hautjambe]),pt([haut_jambe_d.X[0],haut_jambe_d.X[1],haut_jambe_d.X[2]-hautjambe])
155| jamb_h_g,jamb_h_d = sld(haut_jambe_g,genou_g),sld(haut_jambe_d,genou_d)
156| cheville_g,cheville_d = pt([genou_g.X[0],genou_g.X[1],genou_g.X[2]-
basjambe]),pt([genou_d.X[0],genou_d.X[1],genou_d.X[2]-basjambe])
157| tibia_g,tibia_d = sld(genou_g,cheville_g),sld(genou_d,cheville_d)
158| plante_g,plante_d = pt([cheville_g.X[0],cheville_g.X[1]-
pied,cheville_g.X[2]],"go"),pt([cheville_d.X[0],cheville_d.X[1]-pied,cheville_d.X[2]],"go")
159| yep_g,yep_d = sld(cheville_g,plante_g),sld(cheville_d,plante_d)
```

```
160|
161| # guide
162| epaule_g_bis = pt([haut_dos.X[0]-epaules/2,haut_dos.X[1],haut_dos.X[2]],"yo")
163| coude_g_bis = pt([epaule_g_bis.X[0],epaule_g_bis.X[1],epaule_g_bis.X[2]-hautbras])
164| bras_h_g_bis = sld(epaule_g_bis,coude_g_bis,col="r-")
165|
166| # repères
167| R1 = rep(bas_dos)
168| R2 = rep(haut_dos)
169| R3 = rep(epaule_g)
170| R4 = rep(epaule_d)
171| R5 = rep(face)
172| R6 = rep(coude_g)
173| R7 = rep(coude_d)
174| R8 = rep(poignet_g)
175| R9 = rep(poignet_d)
176| R10 = rep(doigt_g)
177| R11 = rep(doigt_d)
178| R12 = rep(haut_jambe_g)
179| R13 = rep(haut_jambe_d)
180| R14 = rep(genou_g)
181| R15 = rep(genou_d)
182| R16 = rep(cheville_g)
183| R17 = rep(cheville_d)
184| R18 = rep(plante_g)
185| R19 = rep(plante_d)
186|
187| # capteurs
188| sousres = "192.168.1." # sous-réseau
189| C3 = cpt(sousres+"15")
190| C4 = cpt(sousres+"13:8080")
191| C2 = cpt(sousres+"158")
192| C1 = cpt(sousres+"205")
193| C5 = cpt(sousres+"41:8080")
194| C6 = cpt(sousres+"53:8080")
195| C7 = cpt(sousres+"53:8080")
196| C8 = cpt(sousres+"53:8080")
197| C9 = cpt(sousres+"53:8080")
198| C10 = cpt(sousres+"53:8080")
199| C11 = cpt(sousres+"53:8080")
200| C12 = cpt(sousres+"53:8080")
201| C13 = cpt(sousres+"53:8080")
202| C14 = cpt(sousres+"53:8080")
203|
204| copie = C4 # pour la symsim
205|
206| fig = plt.figure()
207| ax = fig.add_subplot(projection='3d') # figure 3D matplotlib
208|
209| def animate(i):
210|     ax.clear(),plt.grid(False),plt.axis('off') # vide
211|     ax.set_xlim(-1,1),ax.set_ylim(-1,1),ax.set_zlim(-1,1) # cadre d'affichage
212|     ax.plot3D([(-1)**(n+1) for n in range(200)],[n/100 -1 for n in range(200) ],[-1 for k in
range(200)],"y-",label="sol") # sol
213|
214|     pres("k-","corps")
215|     # squelette temps réel
216|     haut_dos.X = decomp_vecteurs(bas_dos,dos,C1.A[0]+np.pi/2,C1.A[1]+np.pi/2)
217|     colonne.ptB=haut_dos
218|     epaule_g.X,epaule_d.X = [haut_dos.X[0]-epaules/2,haut_dos.X[1],haut_dos.X[2]],[haut_dos.X[0]+epaules/
2,haut_dos.X[1],haut_dos.X[2]]
219|     larg_epaules.ptA,larg_epaules.ptB=epaule_g,epaule_d
220|     face.X = decomp_vecteurs(haut_dos,tete,C2.A[0]+np.pi/2,C2.A[1]+np.pi/2)
221|     cou.ptA,cou.ptB = haut_dos,face
222|     coude_g.X,coude_d.X = decomp_vecteurs(epaule_g,hautbras,C3.A[0]-np.pi/2,C3.A[1]+np.pi/
2),decomp_vecteurs(epaule_d,hautbras,C4.A[0]-np.pi/2,C4.A[1]+np.pi/2)
223|     bras_h_g.ptA,bras_h_g.ptB = epaule_g,coude_g
224|     bras_h_d.ptA,bras_h_d.ptB = epaule_d,coude_d
225|     poignet_g.X,poignet_d.X = decomp_vecteurs(coude_g,avantbras,C5.A[0]-np.pi/
2,C5.A[1]),decomp_vecteurs(coude_d,avantbras,C6.A[0]-np.pi/2,C6.A[1])
226|     bras_b_g.ptA,bras_b_g.ptB = coude_g,poignet_g
227|     bras_b_d.ptA,bras_b_d.ptB = coude_d,poignet_d
228|     doigt_g.X,doigt_d.X = decomp_vecteurs(poignet_g,main,C7.A[0]-np.pi/
2,C7.A[1]),decomp_vecteurs(poignet_d,main,C8.A[0]-np.pi/2,C8.A[1])
229|     main_g.ptA,main_g.ptB = poignet_g,doigt_g
230|     main_d.ptA,main_d.ptB = poignet_d,doigt_d
231|     haut_jambe_g.X,haut_jambe_d.X = [bas_dos.X[0]-bassin/2,bas_dos.X[1],bas_dos.X[2]],[bas_dos.X[0]
+bassin/2,bas_dos.X[1],bas_dos.X[2]]
232|     larg_bassin.ptA,larg_bassin.ptB = haut_jambe_g,haut_jambe_d
233|     genou_g.X,genou_d.X = decomp_vecteurs(haut_jambe_g,hautjambe,C9.A[0]-np.pi/
2,C9.A[1]),decomp_vecteurs(haut_jambe_d,hautjambe,C10.A[0]-np.pi/2,C10.A[1])
234|     jamb_h_g.ptA,jamb_h_g.ptB = haut_jambe_g,genou_g
235|     jamb_h_d.ptA,jamb_h_d.ptB = haut_jambe_d,genou_d
236|     cheville_g.X,cheville_d.X = decomp_vecteurs(genou_g,basjambe,C11.A[0]-np.pi/
2,C11.A[1]),decomp_vecteurs(genou_d,basjambe,C12.A[0]-np.pi/2,C12.A[1])
237|
```

```
    tibia_g.ptA,tibia_g.ptB = genou_g,cheville_g
    tibia_d.ptA,tibia_d.ptB = genou_d,cheville_d
    plante_g.X,plante_d.X = decomp_vecteurs(cheville_g,pied,C13.A[0],C13.A[1]-np.pi/
2),decomp_vecteurs(cheville_d,pied,C14.A[0],C14.A[1]-np.pi/2)

    yep_g.ptA,yep_g.ptB = cheville_g,plante_g
    yep_d.ptA,yep_d.ptB = cheville_d,plante_d


    epaule_g_bis.X = [haut_dos.X[0]-epaules/2,haut_dos.X[1],haut_dos.X[2]]

    coude_g_bis.X = decomp_vecteurs(epaule_g_bis,hautbras,-copie.A[0]-np.pi/2,copie.A[1]+np.pi/2)

    bras_h_g_bis.ptA,bras_h_g_bis.ptB = epaule_g_bis,coude_g_bis

    # maj rep
    #R1.X,R1.A = bas_dos.X,C1.A
    #R2.X = haut_dos.X
    R3.X,R4.X  = epaule_g.X,epaule_d.X
    R3.A,R4.A = C3.A,C4.A
    # R5.X = face.X
    # R6.X,R7.X = coude_g.X,coude_d.X
    # R8.X,R9.X = poignet_g.X,poignet_d.X
    # R10.X,R11.X = doigt_g.X,doigt_d.X
    # R12.X,R13.X  = haut_jambe_g.X,haut_jambe_d.X
    # R14.X,R15.X  = genou_g.X,genou_d.X
    # R16.X,R17.X= cheville_g.X,cheville_d.X
    # R18.X,R19.X = plante_g.X,plante_d.X
    pres("yo",str(np.around(np.array(R3.A)+np.array(R4.A),4)))
    # Affichage des membres
    #bas_dos.afh(),haut_dos.afh()#
    #R1.afh(),R2.afh()
    colonne.afh()
    epaule_g.afh(),epaule_d.afh()#,
    R3.afh(),R4.afh()
    larg_epaules.afh()
    # face.afh()#,R5.afh()
    cou.afh()
    # coude_g.afh(),coude_d.afh()#,R6.afh(),R7.afh()
    bras_h_g.afh(),bras_h_d.afh()
    bras_h_g_bis.afh()
    # poignet_g.afh(),poignet_d.afh()#,R8.afh(),R9.afh()
    bras_b_g.afh(),bras_b_d.afh()
    # doigt_g.afh(),doigt_d.afh()#,R10.afh(),R11.afh()
    main_g.afh(),main_d.afh()
    # haut_jambe_g.afh(),haut_jambe_d.afh()#,R12.afh(),R13.afh()
    larg_bassin.afh()
    # genou_g.afh(),genou_d.afh()#,R14.afh(),R15.afh()
    jamb_h_g.afh(),jamb_h_d.afh()
    # cheville_g.afh(),cheville_d.afh()#,R16.afh(),R17.afh()
    tibia_g.afh(),tibia_d.afh()
    # plante_g.afh(),plante_d.afh()#,R18.afh(),R19.afh()
    yep_g.afh(),yep_d.afh()
    plt.legend()

ani = animation.FuncAnimation(fig,animate,interval=50,cache_frame_data=False)
plt.show()
```