Come ho modificato un threading molto impegnativo per il processore in uno che ne ha drasticamente ridotto l'utilizzo

Carlo Luviner

Quanto segue é in visualizzazione 3D: palco 10x8x6 con 4 teste mobili in americana frontale, le teste mobili con movimento TILT devono inseguirsi (stessi movimenti partendo da posizioni iniziali diverse); le figure sottostanti sono parti del programma che svolgono esattamente le stesse funzioni ma lo script di sinistra impegna molto il processore del mio iMac mentre quello di destra risulta molto meno impegnativo e con movimenti piú fluidi

```
alpha_ = 0.3
color_1 = '#ff0'

pz1 = []
name1 = []
passi1 = [0,1,2,3,4,5,4,3,2,1]
start = [0,2,4,6]
passi1_0 = passi1[start[0]:] + passi1[:start[0]]
passi1_1 = passi1[start[1]:] + passi1[:start[1]]
passi1_2 = passi1[start[2]:] + passi1[:start[2]]
passi1_3 = passi1[start[3]:] + passi1[:start[3]]
passi = [passi1_0,passi1_1,passi1_2,passi1_3]
time1=0.35
for j1 in range(4):
    pz_ = f'ax.plot_surface(x+vx+2*{j1}, y, z+(vz-h),
color=color_1, alpha=alpha_)'
    pz1.append(pz_)
    name1_ = f'pz_1_{j1}'
    name1.append(name1_)

figure.canvas.draw()
def move_(val):

    while True:

      for j in passi[val]:
         y =  y_ + 0.3*j*(h-z)
         exec(f'{name1[val]}={pz1[val]}')

         figure.canvas.draw()
         time.sleep(time1)


         exec(f'{name1[val]}.remove()')



threads = [threading.Thread(target=move_, args=(i,),daemon=True)
for i in range(4)]

for thread in threads:

    thread.start()
```

```
alpha_ = 0.3
color_1 = '#ff0'
pz1 = []
name1 = []
j_ = ['j0','j1','j2','j3']
j_y = ['j2','j3','j0','j1']
time1=0.2
for j1_ in range(4):
    pz_ = f'ax.plot_surface(x+2+2*jj, y+0.2*(h-z)*k{j1_}, z, color=color_1, alpha=alpha_)'
    pz1.append(pz_)
    name1_ = f'pz_1_{j1_}'
    name1.append(name1_)

dx = 0 # delta x per il PAN

dd_pos = dx-np.cos(u)
dd_neg = -dx-np.cos(u)

figure.canvas.draw()

def shift_(ar, nn):

    nn = nn % len(ar)
    arr = ar[nn:] + ar[:nn]
    return arr

passi = [0,1,2,3,4,5,4,3,2,1]
kk_ = [0,2,4,6]
kkk = []

for k0 in kk_:
    kkk.append(shift_(passi,k0))

def move_():

  while True:
    for k0,k1,k2,k3 in zip(kkk[0],kkk[1],kkk[2],kkk[3]):

        for jj in range(4):
             #print(kkk[jj])

             exec(f'{name1[jj]}={pz1[jj]}')

        figure.canvas.draw()
        time.sleep(time1)

        for jjj in range (len(pz1)):
            exec(f'{name1[jjj]}.remove()')

th = threading.Thread(target= move_, daemon=True)
th.start()
```