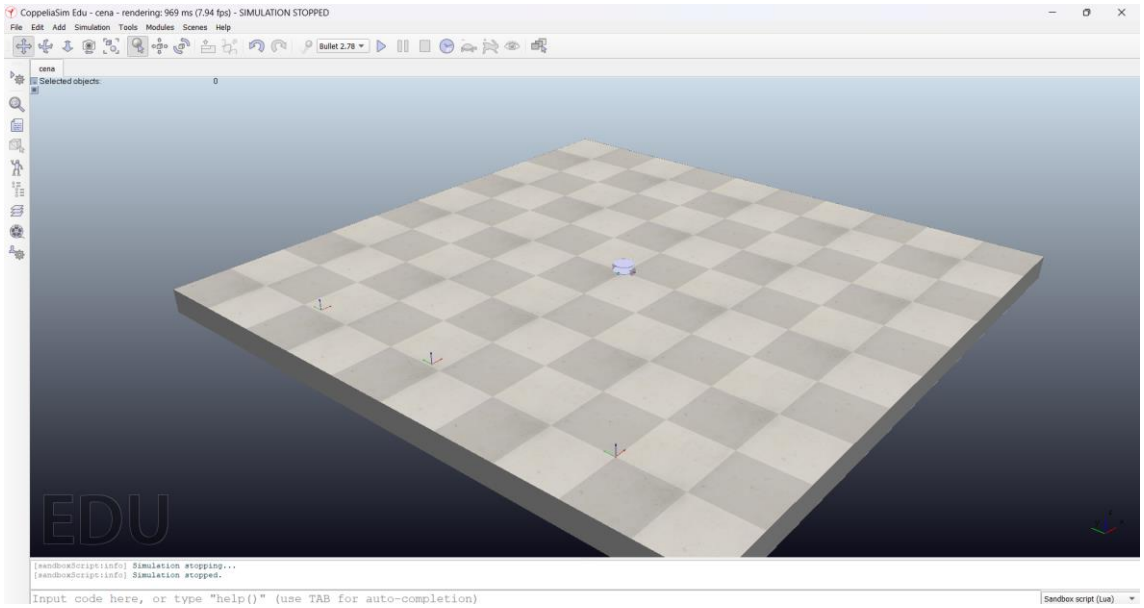
Relatório - Atividade 4

Na atividade 4 foi proposto que fizéssemos um robô Go To Goal, na qual ele chegasse em um ponto final. Nesse caso, o modelo do robô foi o mesmo desenvolvido na atividade passada, porém, agora codificado para que seguisse até um ponto específico.



Foram colocados 3 pontos na cena. O robô segue inicialmente para o Ponto 1, até chegar, sucessivamente ao Ponto 3. Logo após, a simulação para e o robô volta para o seu ponto inicial. O código que utilizamos foi:

```
from coppeliasim_zmqremoteapi_client import RemoteAPIClient
import numpy as np
import time

client = RemoteAPIClient()
sim = client.getObject("sim")


def normalizeAngle(angle):
    return np.mod(angle+np.pi, 2*np.pi) - np.pi


handle_robo = sim.getObject("/Chassi")


roda_direita = sim.getObject('/Chassi/direita')
roda_esquerda = sim.getObject('/Chassi/esquerda')
```

```
L = 0.381
r = 0.0975
maxv = 1.5
maxw = np.deg2rad(45)


handle_goal = sim.getObject("/PONTO1")
pos_goal = sim.getObjectPosition(handle_goal,-1)
theta_goal = sim.getObjectOrientation(handle_goal,-1)[2]


rho = np.inf

sim.startSimulation()
chegada = 0

while chegada < 3:

    pos_robo = sim.getObjectPosition(handle_robo,-1)
    theta_robo = sim.getObjectOrientation(handle_robo,-1)[2]


    dt_x = pos_goal[0] - pos_robo[0]
    dt_y = pos_goal[1] - pos_robo[1]
    theta = theta_goal - theta_robo
    theta = normalizeAngle(theta)

    rho = np.sqrt((dt_x**2) + (dt_y**2))
    alpha = (-theta_robo) + np.arctan2(dt_y,dt_x)
    beta = theta_goal - np.arctan2(dt_y,dt_x)
    alpha = normalizeAngle(alpha)
    beta = normalizeAngle(beta)


    kr = -0.4
    ka = 0.8
    kb = 0.15


    if np.abs(alpha) > np.deg2rad(90):
        alpha = normalizeAngle(alpha-np.pi)
        beta = normalizeAngle(beta-np.pi)

        v = -(kr * rho)
    else:
        v = kr * rho
    w = ka * alpha + kb * beta
```

```
    v = np.clip(v, -maxv, maxv)
    w = np.clip(w, -maxw, maxw)


    wl = ((2 * v) - (w * L)) / (2 * r)
    wr = ((2 * v) + (w * L)) / (2 * r)



    vel_esq = sim.setJointTargetVelocity(roda_esquerda, wl)
    vel_dir = sim.setJointTargetVelocity(roda_direita, wr)
    if (rho < 0.05):
        chegada = chegada + 1
        if chegada == 1:
            handle_goal = sim.getObject("/PONTO2")
            pos_goal = sim.getObjectPosition(handle_goal,-1)
            theta_goal = sim.getObjectOrientation(handle_goal,-1)[2]
        if chegada == 2:
            handle_goal = sim.getObject("/PONTO3")
            pos_goal = sim.getObjectPosition(handle_goal,-1)
            theta_goal = sim.getObjectOrientation(handle_goal,-1)[2]


vel_esq = sim.setJointTargetVelocity(roda_esquerda, 0)
vel_dir = sim.setJointTargetVelocity(roda_direita, 0)

sim.stopSimulation()
```