Taller 4 - CoppeliaSim y Python

April 5, 2024

Taller 4 - CoppeliaSim y Python

Requiere que los archivos 'sim.py', 'simConst.py', 'remoteapi.dll' estén alojados en la misma carpeta que este cuaderno de Jupyter

Instrucciones

1. Abra la escena robotPlanar.ttt 2. Seleccionando la base del robot desde la jerarquía de escena, presione botón derecho del mouse y agregue un archivo de script mediante Add -> Associated child script -> Non threaded. Aparecerá un pequeño ícono de documento junto al nombre del robot en la escena de jerarquía. 3. En el script es posible incluir código de programación, escritos en lenguaje LUA. Para nuestro caso, todo el código que requeriremos es habilitar el API remoto, asignando un puerto de comunicación. En la función sysCall_init() agregue la siguiente línea:

simRemoteApi.start(19999) Proceda a continuación con las actividades:

[1]:
```
import sim
import numpy as np
```

0.0.1 1. Establecer la conexión

Utilizaremos las funciones del API Remoto de VREP. Para más detalles refiérase a la documentación de la librería: http://www.coppeliarobotics.com/helpFiles/en/remoteApiFunctionsPython.htm

[2]:
```
def connect(port):
# Establece la conexión a VREP
# port debe coincidir con el puerto de conexión en VREP
# retorna el número de cliente o -1 si no puede establecer conexión
    sim.simxFinish(-1) # just in case, close all opened connections
    clientID=sim.simxStart('127.0.0.1',port,True,True,2000,5) # Conectarse
    if clientID == 0: print("conectado a", port)
    else: print("no se pudo conectar")
    return clientID
```

[4]:
```
# Conectarse al servidor de VREP
# *** ejecutar cada vez que se reinicia la simulación ***
clientID = connect(19999)
```

```
conectado a 19999
```

2. Obtener los manejadores (handlers)

Un manejador (handler) es un número identificador que asigna VREP para cada uno de los elementos de la escena. A través de su manejador se puede hacer referencia a un elemento en específico de la escena.

[5]:
```
# Obtenemos el manejador para el dummy
returnCode,handle=sim.simxGetObjectHandle(clientID,'Dummy',sim.
 ↪simx_opmode_blocking)
dummy = handle
print(dummy)
```

```
20
```

[6]:
```
# A partir de su manejador podemos accionar sobre el objeto,
# por ejemplo, obtener su posición
returnCode,pos=sim.simxGetObjectPosition(clientID, dummy, -1, sim.
 ↪simx_opmode_blocking)
print(pos)
```

```
[0.3890935182571411, 0.29496702551841736, 1.64858786344535e-11]
```

[7]:
```
# Obtenemos los manejadores para cada una de las articulaciones del robot
ret,joint1=sim.simxGetObjectHandle(clientID,'joint1',sim.simx_opmode_blocking)
ret,joint2=sim.simxGetObjectHandle(clientID,'joint2',sim.simx_opmode_blocking)
print(joint1, joint2)
```

```
16 18
```

3. ¿Cuál es la posición de las articulaciones?
   Utilizando los manejadores, podemos obtener información de los elementos.

[15]:
```
# leemos la posición de joint1, en radianes.
returnCode, pos1 = sim.simxGetJointPosition(clientID, joint1, sim.
 ↪simx_opmode_blocking)
print(pos1)
```

```
4236.90869140625
```

[21]:
```
returnCode, pos2 = sim.simxGetJointPosition(clientID, joint2, sim.
 ↪simx_opmode_blocking)
print(pos2)
```

```
6059.37744140625
```

4. … y movemos el robot
   Utilizando los manejadores, podemos enviar parámetros a los elementos.

[29]:
```
# enviamos la posición de joint1, en radianes.
q1 = -30 * np.pi/180
returnCode = sim.simxSetJointTargetPosition(clientID, joint1, q1, sim.
 ↪simx_opmode_oneshot)
print(returnCode)
```

0

[32]:
```
q2 = 30 * np.pi/180
returnCode = sim.simxSetJointTargetPosition(clientID, joint2, q2, sim.
  ↪simx_opmode_oneshot)
print(returnCode)
```

0

[39]:
```
# Ponemos todo el código junto
# conectamos
clientID = connect(19999)
# obtenemos los manejadores
returnCode,handle=sim.simxGetObjectHandle(clientID,'Dummy',sim.
  ↪simx_opmode_blocking)
dummy = handle
ret,joint1=sim.simxGetObjectHandle(clientID,'joint1',sim.simx_opmode_blocking)
ret,joint2=sim.simxGetObjectHandle(clientID,'joint2',sim.simx_opmode_blocking)
#movemos
q1 = -90 * np.pi/180
returnCode = sim.simxSetJointTargetPosition(clientID, joint1, q1, sim.
  ↪simx_opmode_oneshot)
q2 = 0 * np.pi/180
returnCode = sim.simxSetJointTargetPosition(clientID, joint2, q2, sim.
  ↪simx_opmode_oneshot)
```

conectado a 19999

[37]:
```
returnCode,pos=sim.simxGetObjectPosition(clientID, dummy, -1, sim.
  ↪simx_opmode_blocking)
print(pos)
```

[-2.064197524020983e-08, -0.5, 1.3172660878746356e-11]

[ ]: