

UNIVERSIDAD DE PANAMÁ  
FACULTAD DE INFORMÁTICA, ELECTRÓNICA Y COMUNICACIÓN  
INGENIERÍA MECATRÓNICA  
ROBÓTICA

PARCIAL 2 – INFORME PROYECTO PICK AND PLACE CON COPPELIA SIM

INTEGRANTES:

ELÍAS CHEN	8-964-1289
ANA S. MUÑOZ C.	EC-35-286

PROFESOR:

YARIEN MORENO

PANAMÁ

5 DE DICIEMBRE DE 2022

## INTRODUCCIÓN

Anteriormente conocido como V-REP, Coppelia Sim es un simulador de robótica cuyo entorno de desarrollo está basado en una arquitectura de control distribuido, lo que le brinda versatilidad y lo convierte en un simulador ideal para aplicaciones multi – robot. Este es utilizado para el desarrollo de algoritmos, simulación de automatización de fábrica, creación y verificación de prototipos, robótica educacional, verificación de doble seguridad, entre otros. (Marras, 2016)

En las industrias, las actividades pick and place son las más comunes, estas consisten en recoger un producto y colocarlo en otro lugar. Sin embargo, a pesar de ser tareas sencillas, también son repetitivas y consumen demasiado tiempo, por lo que se ha optado por automatizar estos procesos con robots, llevando a minimizar tiempo y errores. (Mecalux, S.A., 2020)

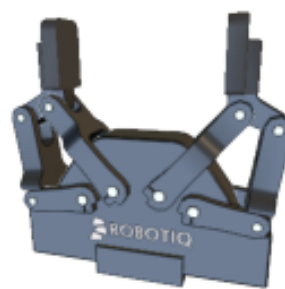
El proyecto detallado a continuación demuestra como se lleva a cabo una aplicación sencilla de pick and place con un robot UR3 y el software Coppelia Sim en su versión EDU 4.0.0.

## COMPONENTES

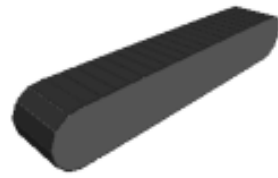
- Robot UR3 (*disponible en la carpeta robots | non-mobile*)
- Gripper ROBOTIQ 85 (*disponible en la carpeta components | grippers*)
- Generic conveyor (belt) (*disponible en la carpeta equipment | conveyors*)
- Customizable table (*disponible en la carpeta furniture | tables*)
- 2 cubos (lados de 5cm)



UR3.ttm



ROBOTIQ 85.ttm



generic conveyor  
(belt).ttm



customizable table.ttm

## PROCEDIMIENTO

Primero se añadió el brazo robótico UR3 a la escena (Ilustración 1), este se posicionó en el centro, para ello, en la pestaña Position se colocó 0 a todas las coordenadas (Ilustración 2).

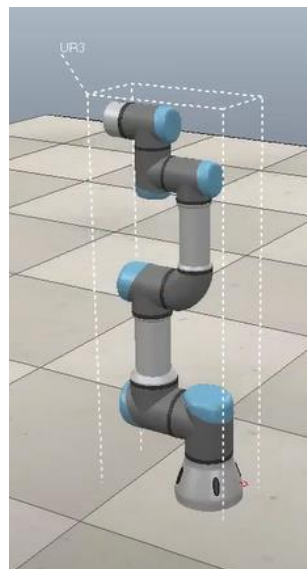


Ilustración 1

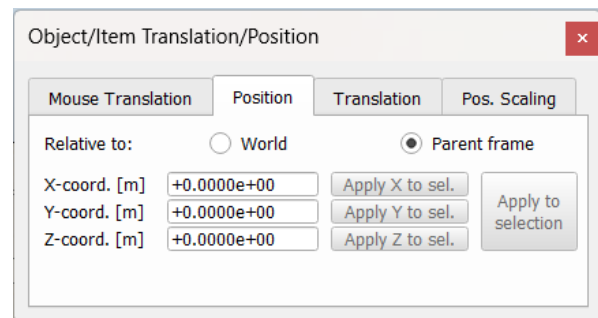


Ilustración 2

Una vez se ubicó el brazo robótico, se añadió el gripper ROBOTIQ 85 a la escena. (Ilustración 3)



Ilustración 3

Para que la garra quedé relacionada con el robot, se colocó dentro de la jerarquía del UR3, en el último link del robot, ya que este pertenece al extremo del brazo. En este caso, el último link es el link 7 (Ilustración 4).

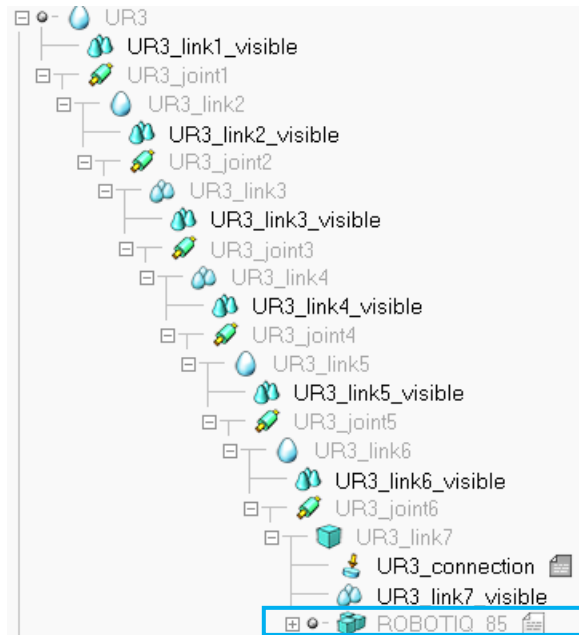


Ilustración 4

Se cambió la posición del gripper con respecto al **Parent frame**, para ello se rotó la pieza en x, y, z hasta que quedara ubicada en el extremo del brazo robótico.

Para que el gripper no se separará del brazo al iniciar la simulación, se modificaron sus propiedades, quitando la selección de **body is Dynamic**. (Ilustración 5)

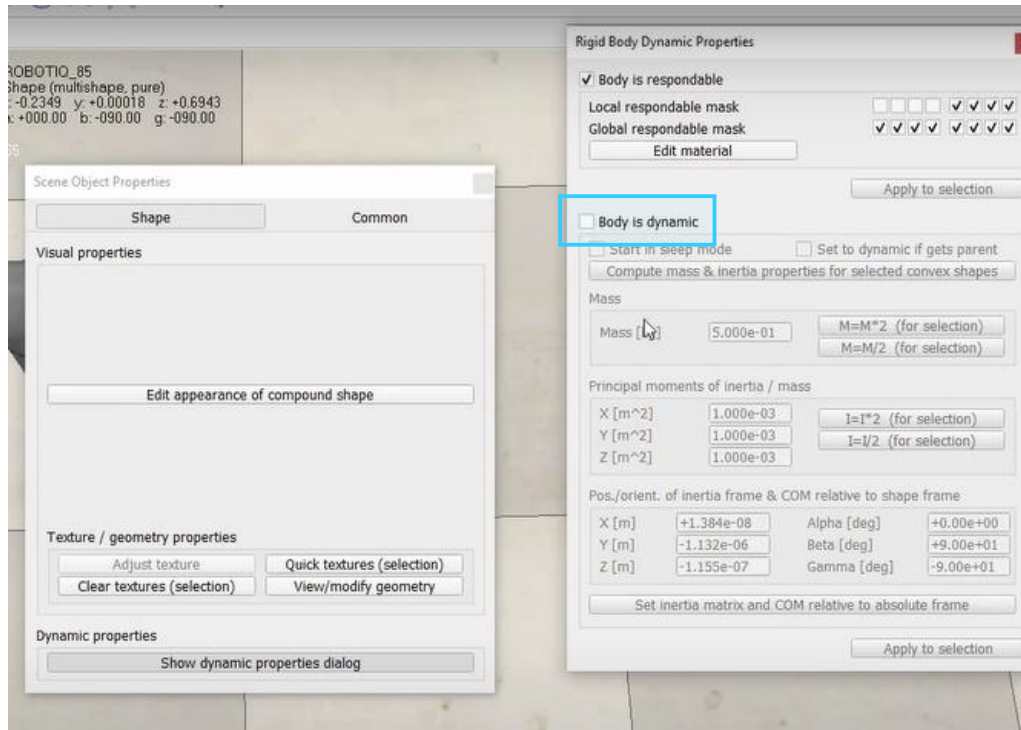


Ilustración 5

Se añadió la cinta transportadora y se ubicó al frente del robot, con respecto al eje z. (Ilustración 6)

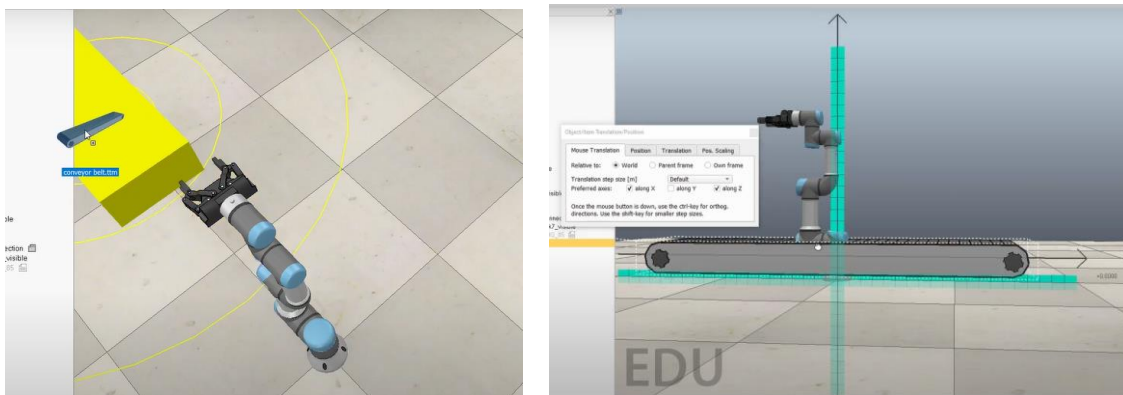


Ilustración 6

Se añadió una mesa, posicionándola a la izquierda del robot. El tamaño de esta se ajustó con las barras, estas se dejaron en lo mínimo permitido por el simulador. (Ilustración 7)

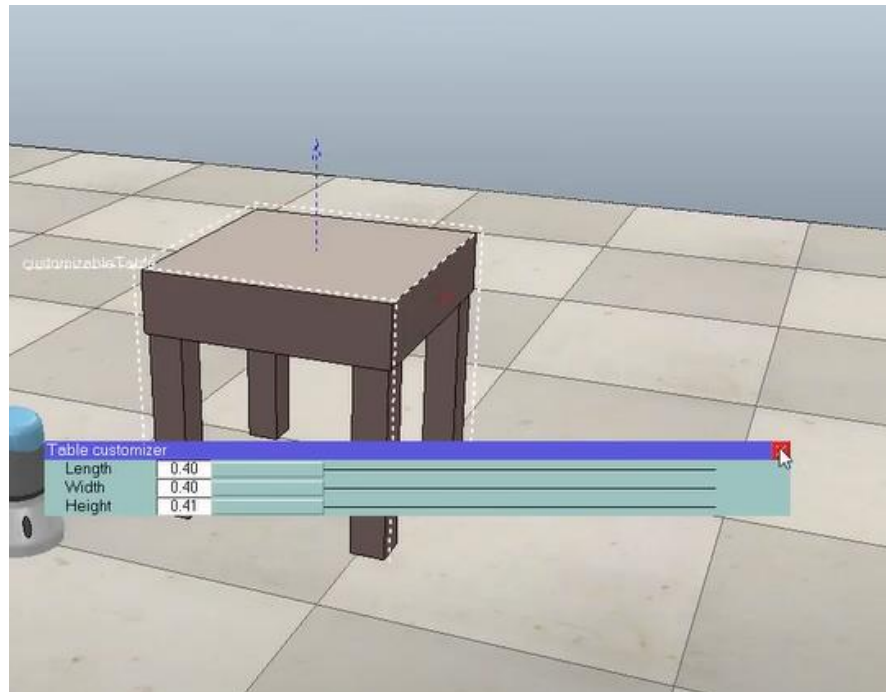


Ilustración 7

Se añadieron 2 cubos (Ilustración 8) y se posicionaron sobre la cinta transportadora. A cada uno se le dio el tamaño de 5 cm por lado para que la garra pueda agarrarlos sin inconvenientes. (Ilustración 9)

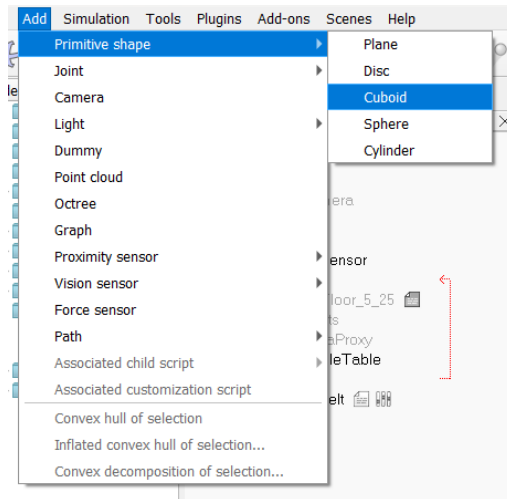


Ilustración 8

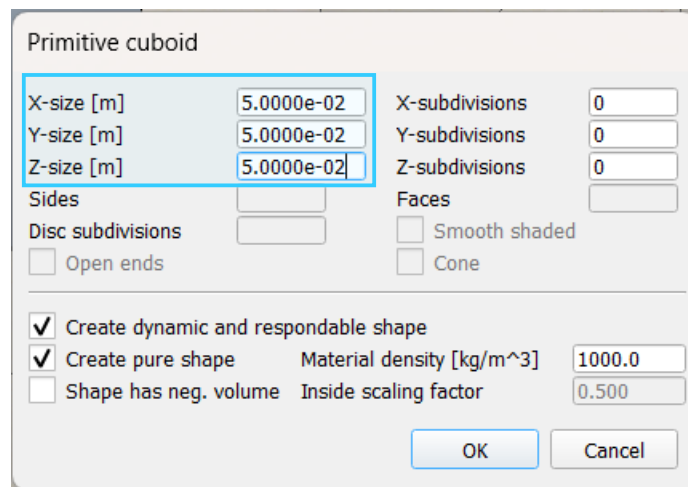


Ilustración 9

El color de cada uno se cambió en la pestaña Properties, en la opción Adjust color: ambient/diffuse component. (Ilustración 10)

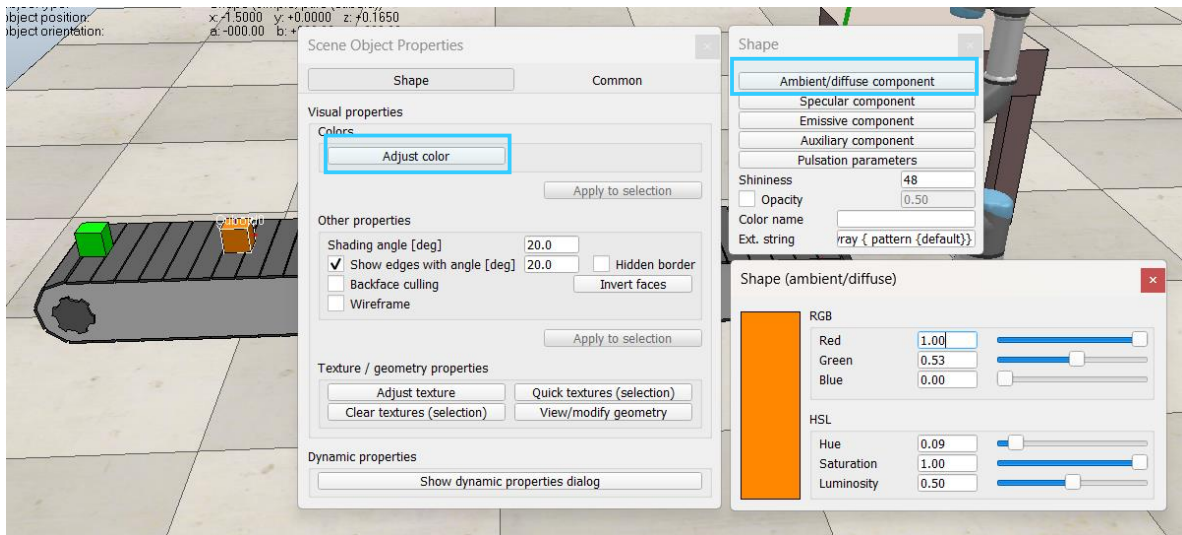


Ilustración 10

Para que los cubos puedan ser detectados por el sensor de proximidad, se activó la propiedad **Detectable**, la misma se encuentra en la pestaña Common. (Ilustración 11)



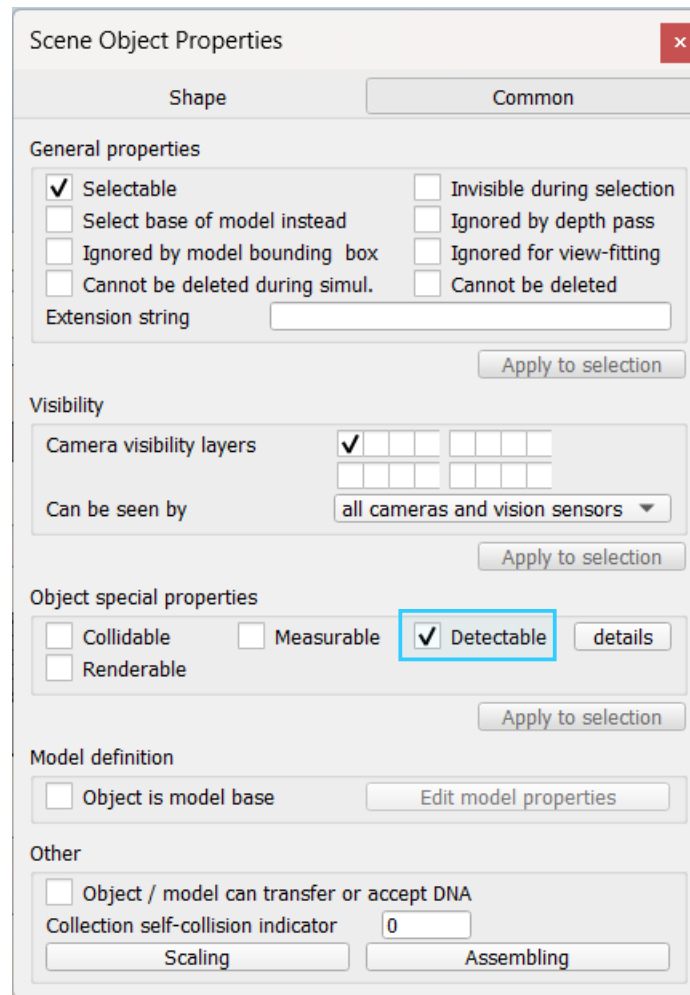


Ilustración 11

Se agregó un sensor de proximidad tipo ray (Ilustración 12) y se lo colocó al final de la cinta transportadora (Ilustración 13).

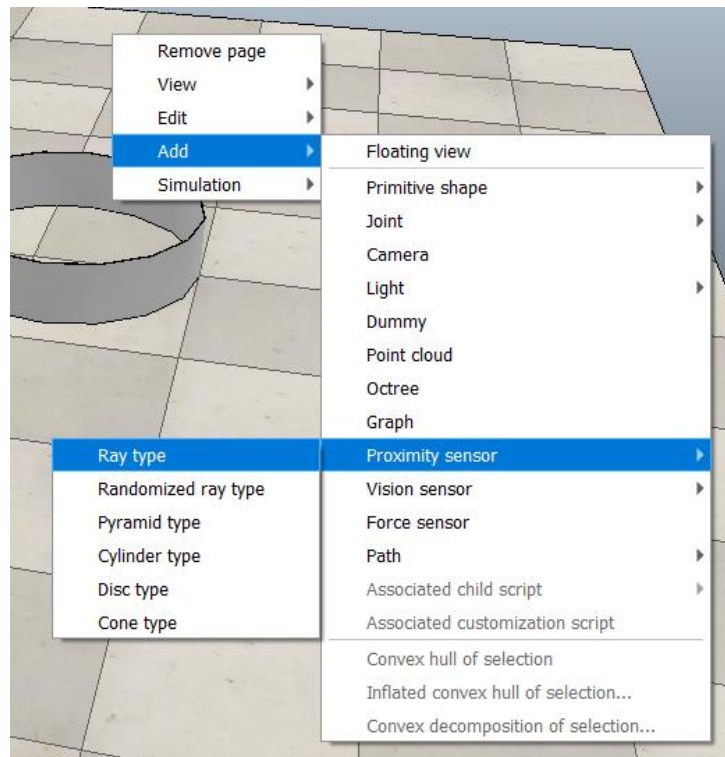


Ilustración 12

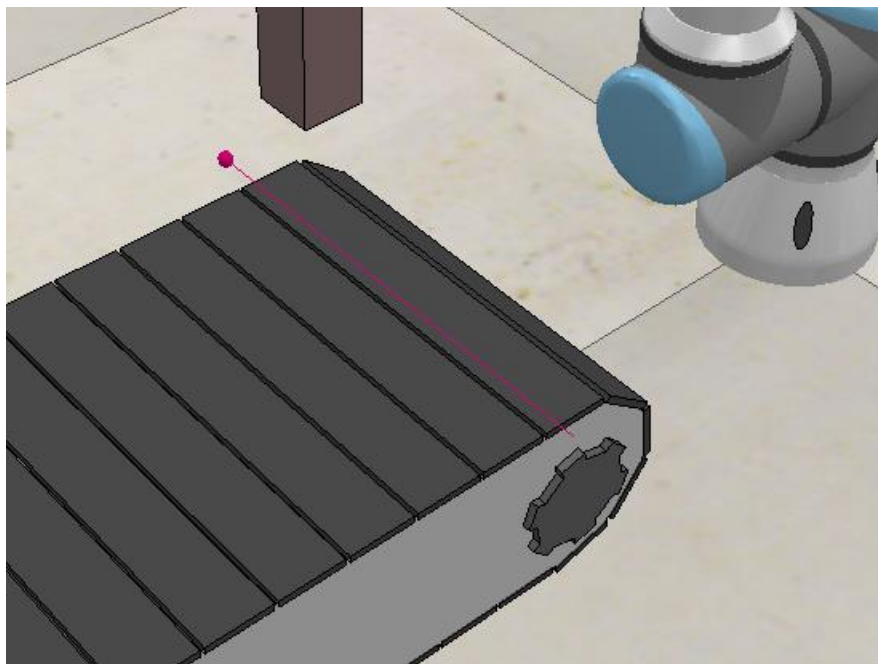


Ilustración 13

Los elementos quedarían posicionados como aparece en la ilustración 14.

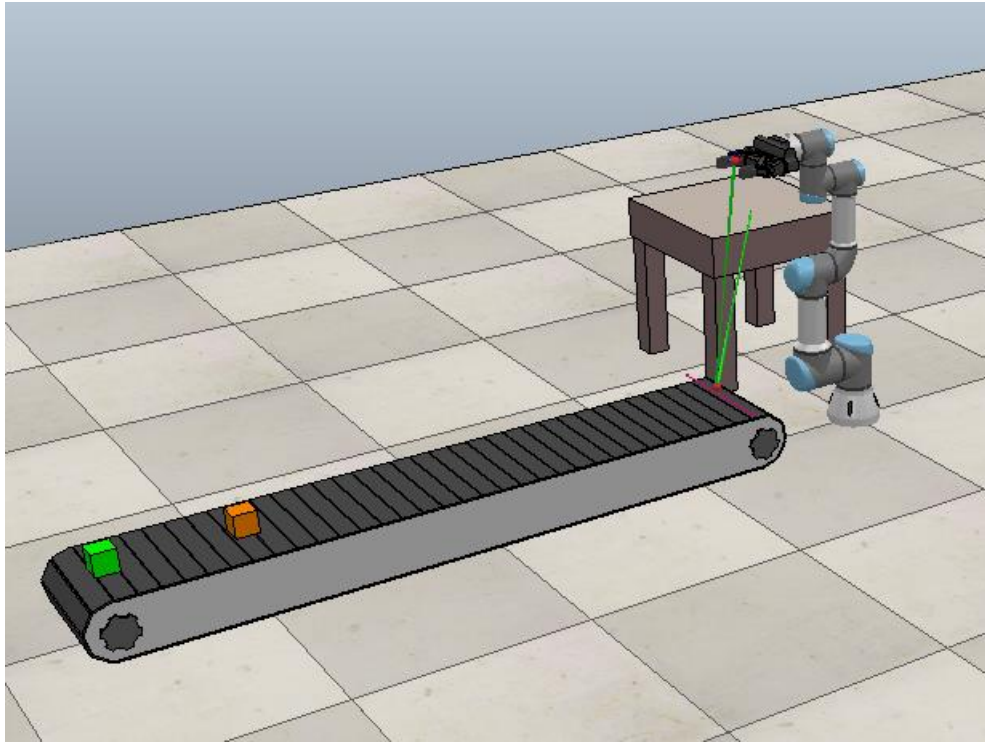


Ilustración 14

Se agregó un dummy y se cambió su nombre a Tip. (Ilustración 15)

Un dummy es un punto de orientación, en este caso nos ayudará a guiar la garra hacia el objetivo. (Coppelia Robotics)

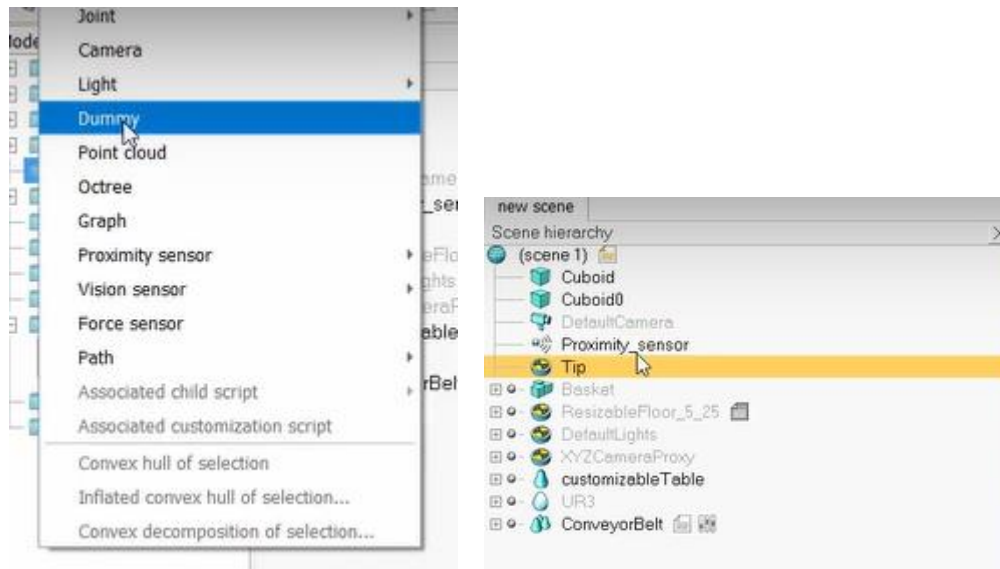


Ilustración 15

Se colocó el dummy Tip dentro de la jerarquía de la garra. (Ilustración 16)

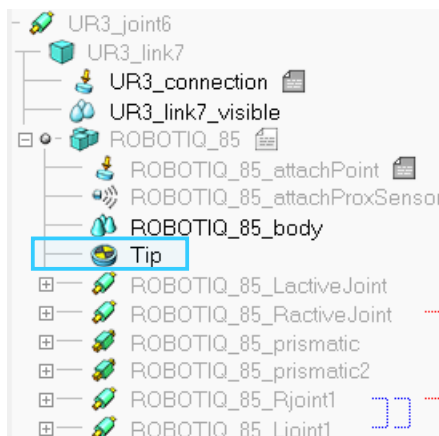


Ilustración 16

Se posicionó el dummy Tip en el centro de la garra (Ilustración 18), para ello cambiamos su posición con respecto al Parent Frame (Ilustración 17).

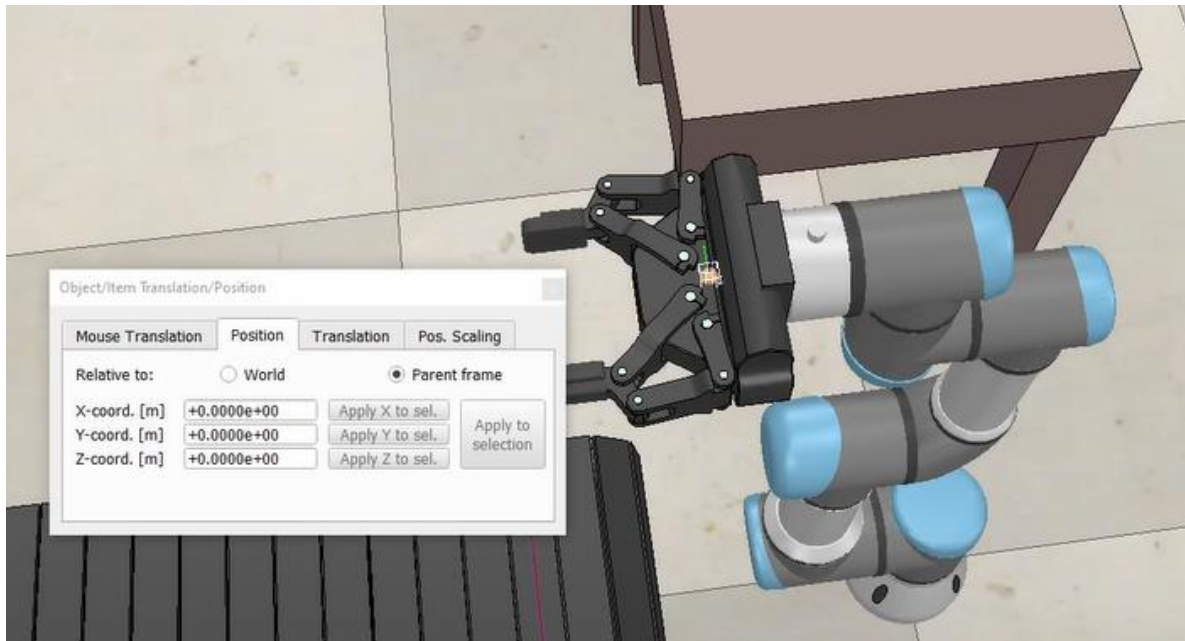


Ilustración 17

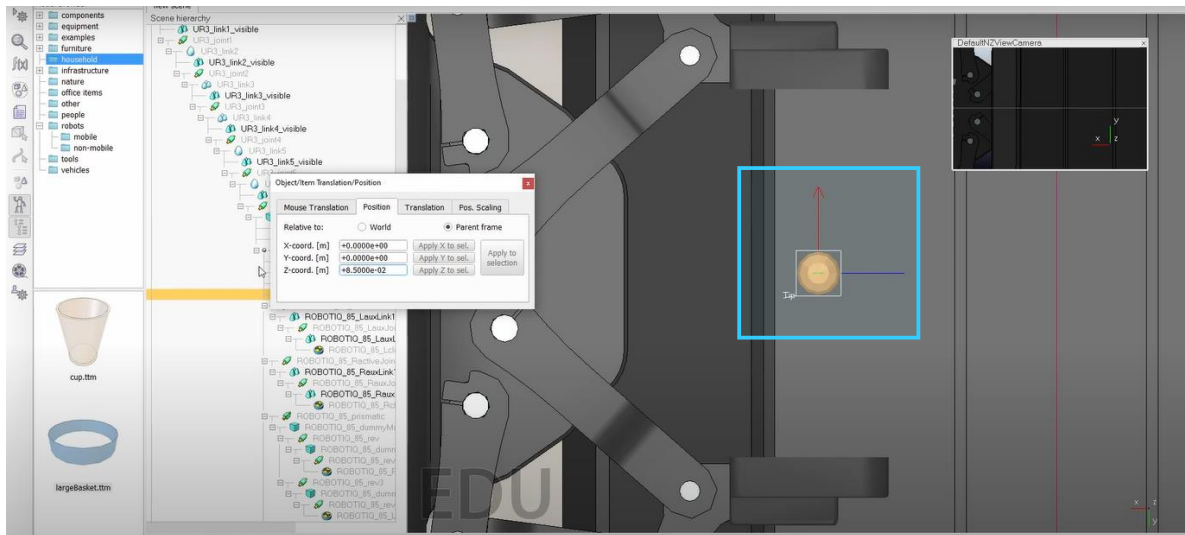


Ilustración 18

Se agregó otro dummy, este se llamaría Target. (Ilustración 19)

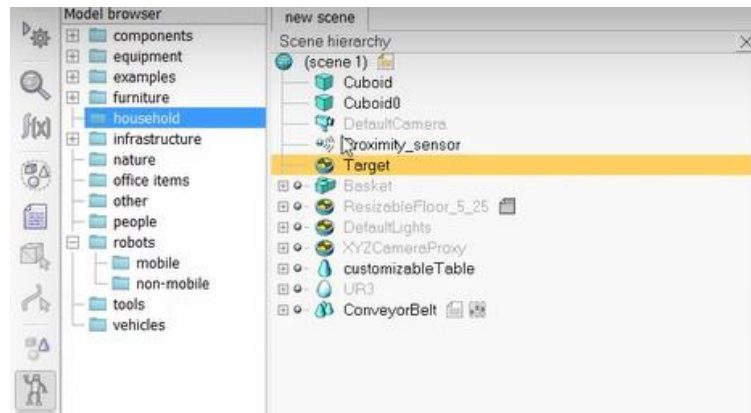


Ilustración 19

Para crear un enlace entre los dummy Target y Tip, se modificaron las propiedades del primero, colocando los parámetros de la siguiente forma:

- Linked dummy: Tip
- Link type: IK, tip-target

(ver ilustración 20)

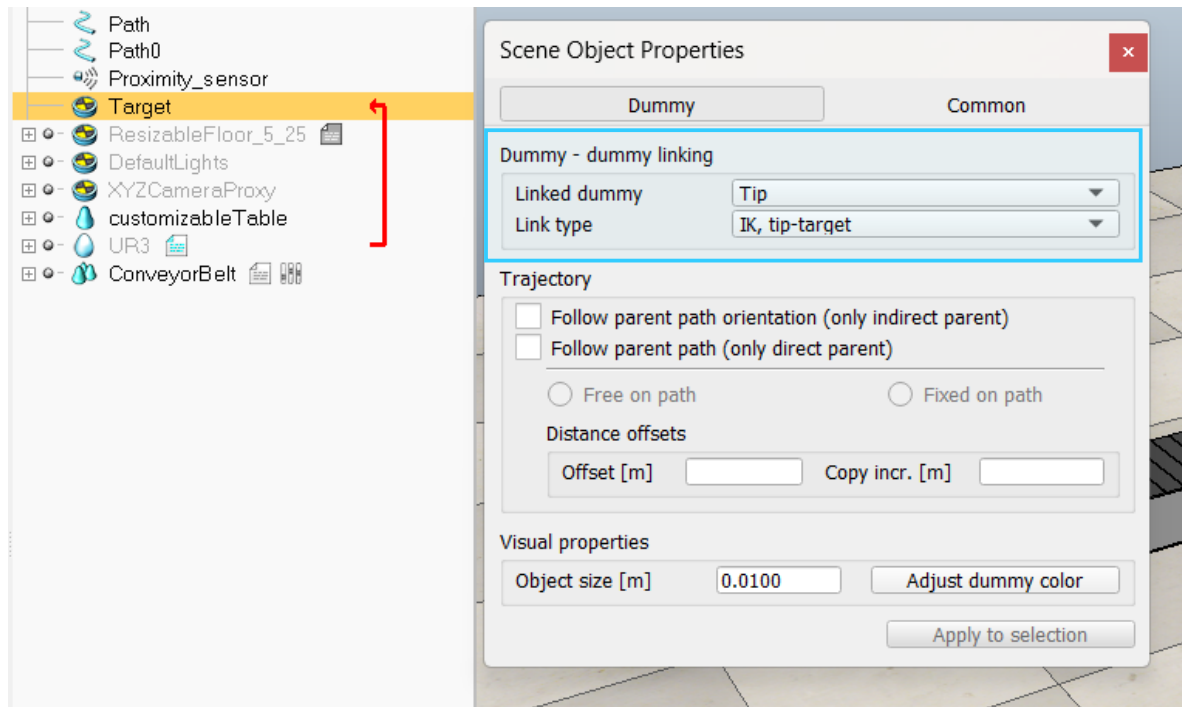


Ilustración 20

Luego entramos en la herramienta ***Calculation Modules***. (Ilustración 21)



Ilustración 21

El simulador nos abrió la siguiente ventana emergente, donde se seleccionó la pestaña ***Kinematics***. (Ilustración 22)

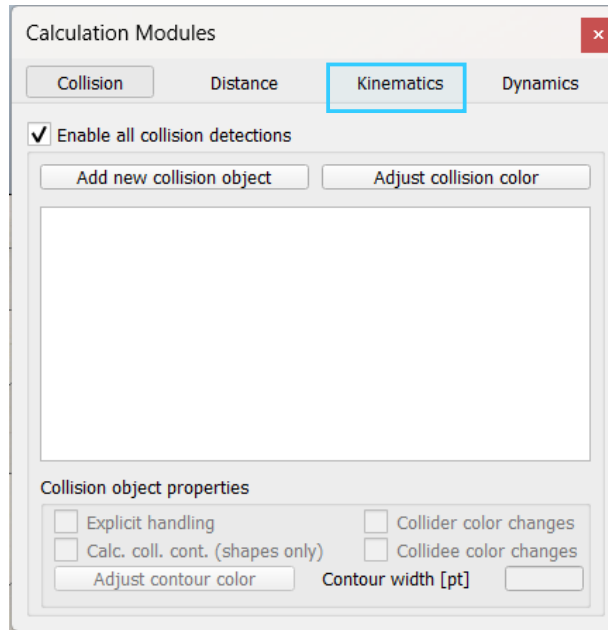


Ilustración 22

Se dio clic en la opción **Add new IK group**, se creó automáticamente un grupo que modificaremos dando clic en **Edit IK elements**. (Ilustración 23)

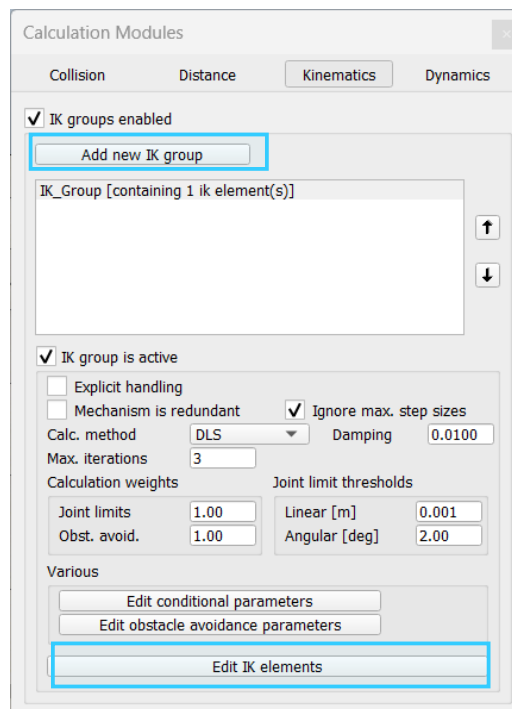


Ilustración 23



Se abrió la siguiente ventana emergente mostrada en la ilustración 24.

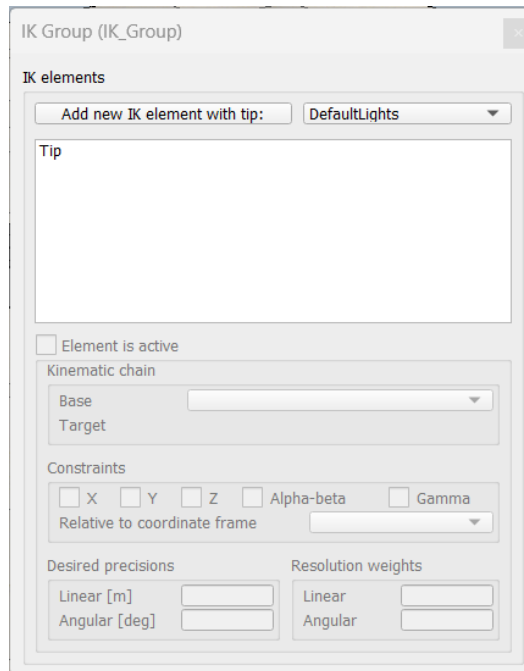


Ilustración 24

En las opciones para el elemento IK, se seleccionó Tip. (Ilustración 25)

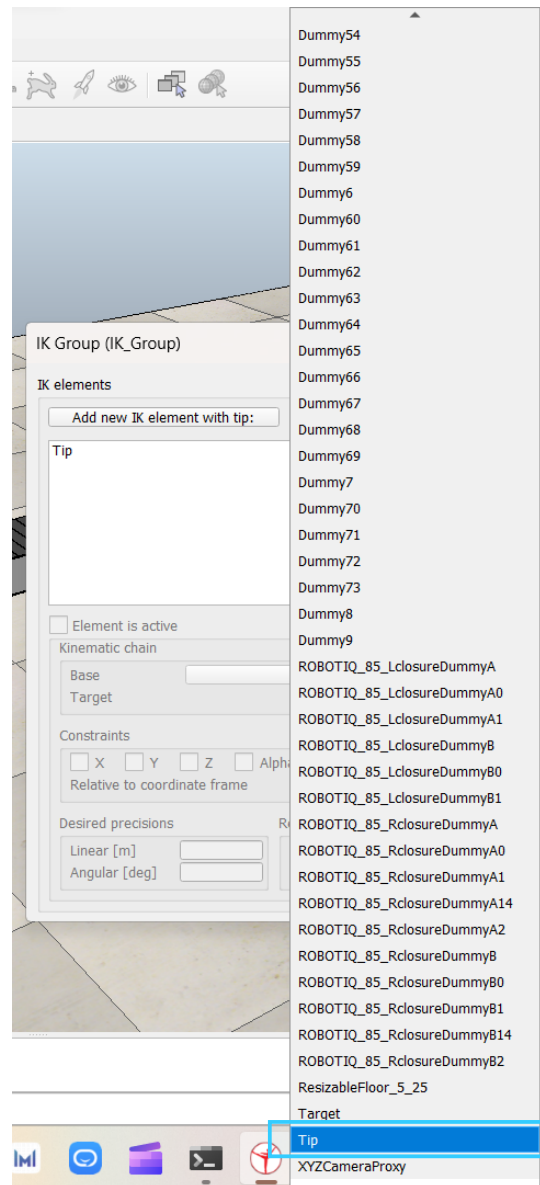


Ilustración 25

Una vez creado el elemento IK Tip, se cerraron las ventanas emergentes.

## CONFIGURACIÓN DEL PATH

El path será el camino que debe seguir el brazo para recoger el objeto y dejarlo en el lugar final.

Primero se agregó un segment path. (Ilustración 26)

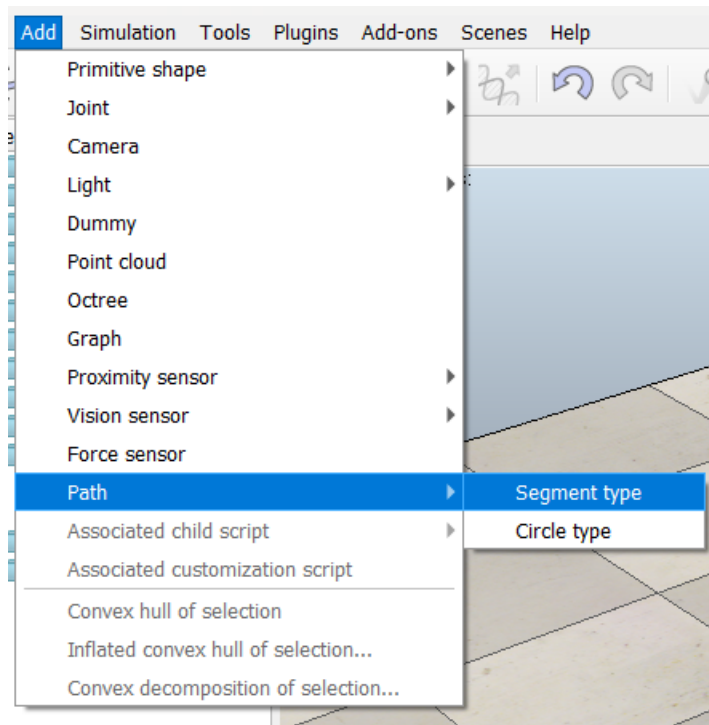


Ilustración 26

Para asignar los puntos de inicio y fin, entramos en la pestaña Path edition.  
(Ilustración 27)

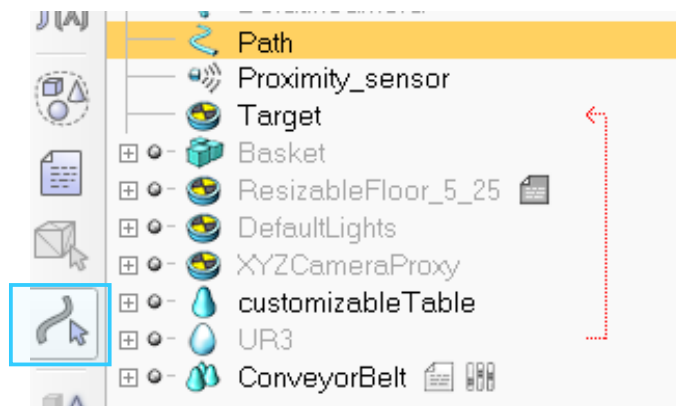


Ilustración 27

El entorno se verá como se muestra en la ilustración 28.

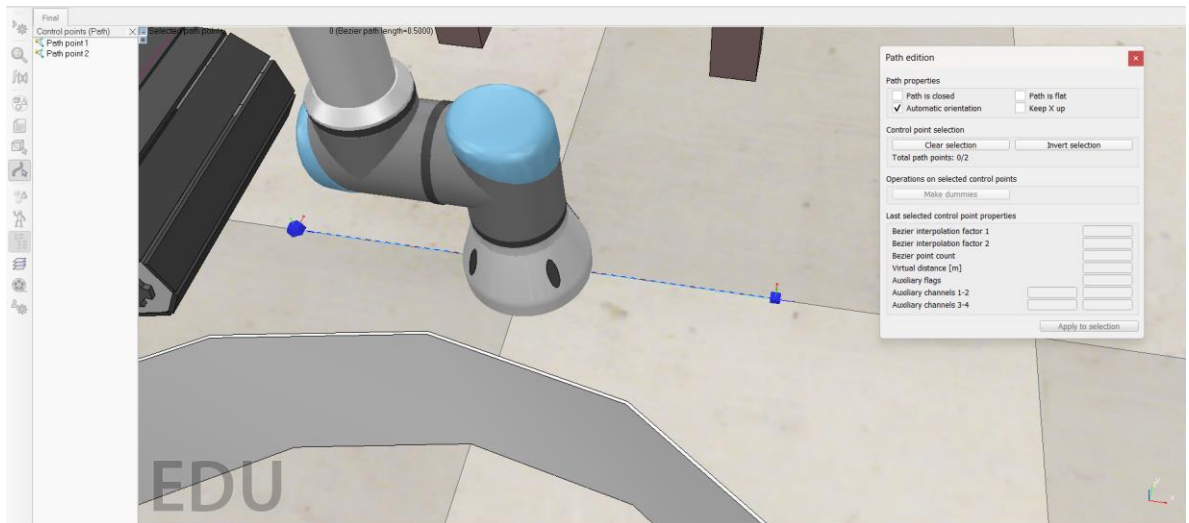


Ilustración 28

Se desactivo la opción **Automatic orientation**. (Ilustración 29)

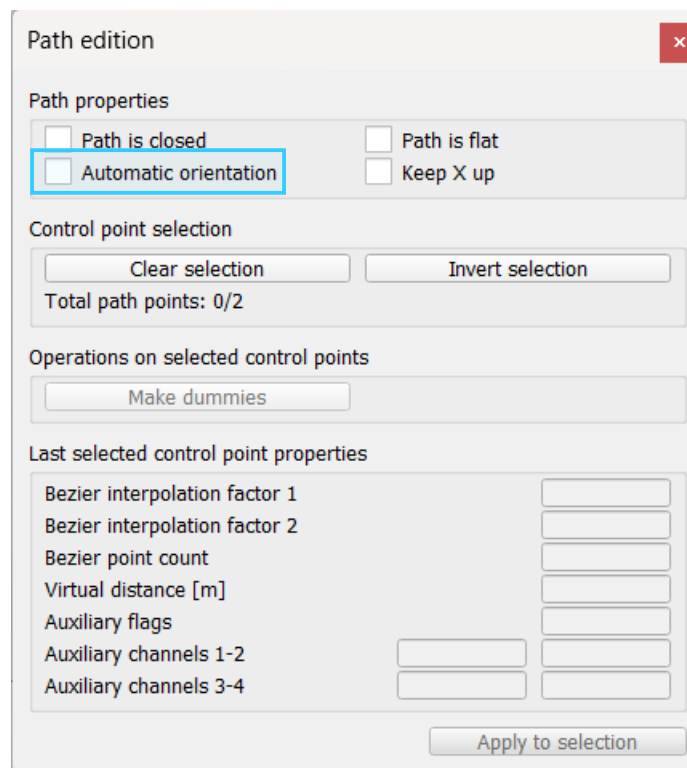


Ilustración 29

Se cerraron las pestañas y se guardaron los cambios.

Se dio doble clic sobre el icono de path para poder acceder a sus propiedades. Luego se dio clic en la opción **Show path shaping dialog** y activamos la opción **Path shaping enabled**. (Ilustración 30)

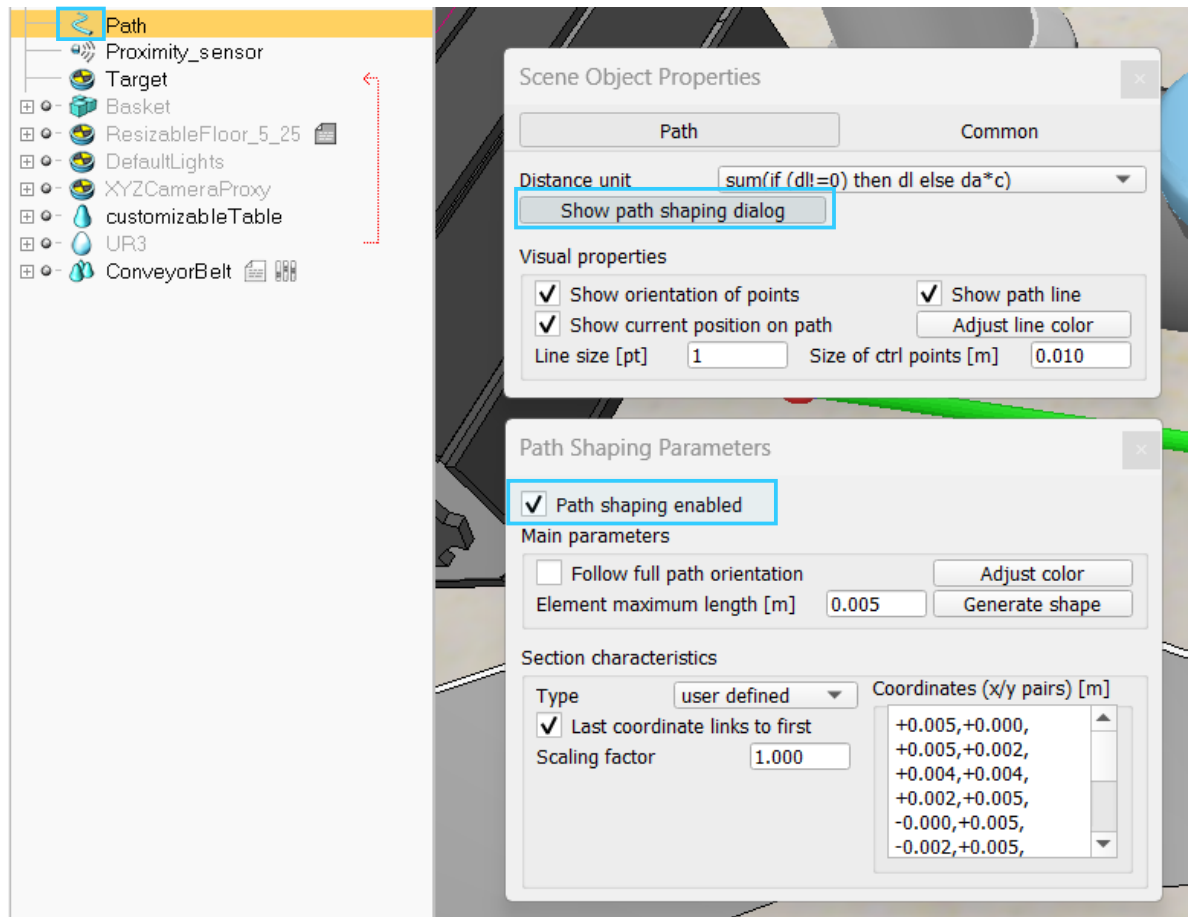


Ilustración 30

Se cerraron las ventanas emergentes y se seleccionó el dummy Target para poder copiar sus coordenadas. (Ilustración 31)

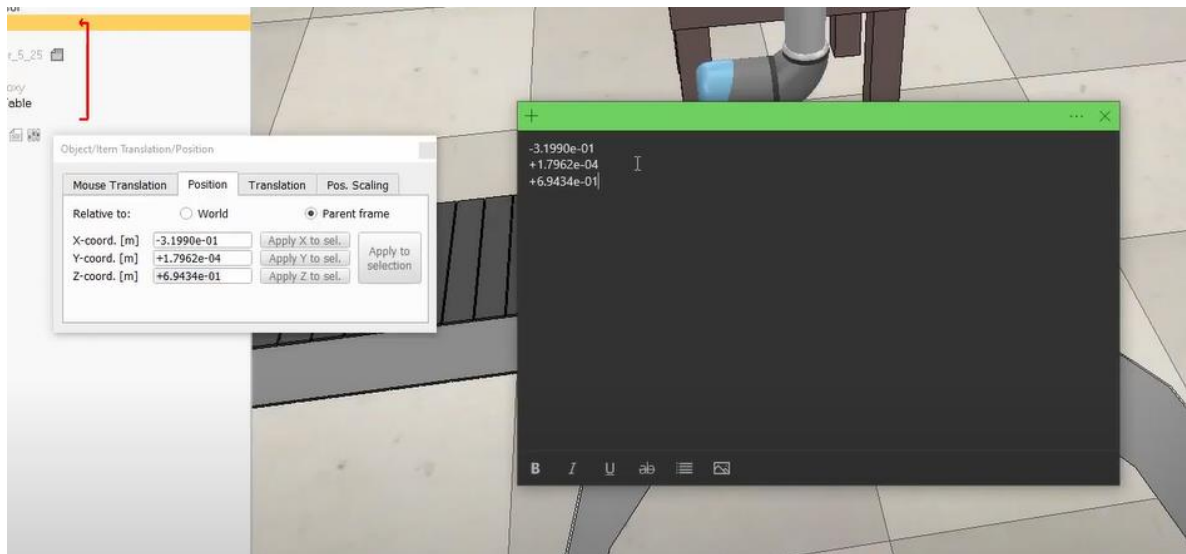


Ilustración 31

Se volvió a la pestaña **Path edition**, y se editó el primer punto del path way colocando como punto 1 las coordenadas que se copiaron anteriormente. (Ilustración 32)

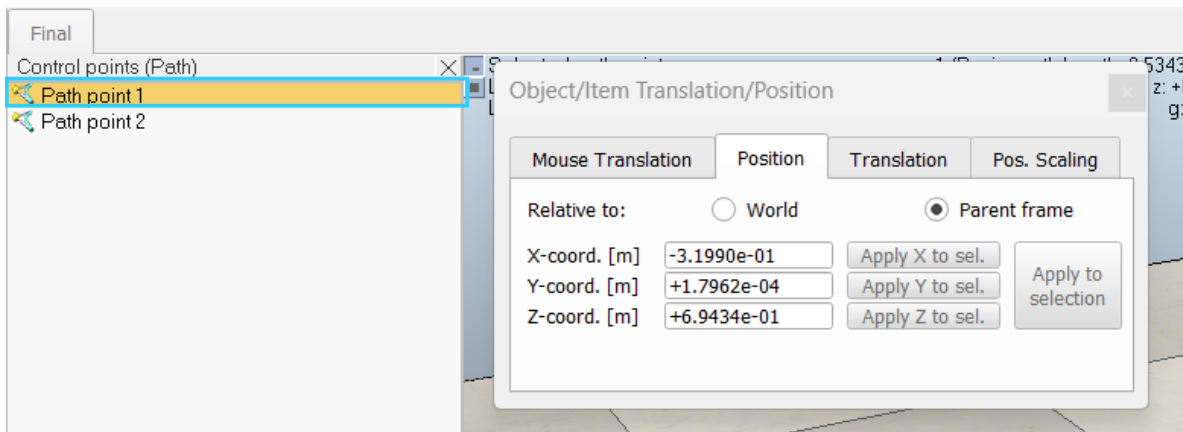


Ilustración 32

Una vez hemos colocado el primer punto, realizamos la misma operación con el punto 2 (Ilustración 34). Se debe aclarar que las coordenadas que se copian son las del cubo cuando ha llegado al sensor de proximidad, para poder obtenerlas se da

inicio a la simulación y cuando el cubo llegue a la posición deseada, detenemos la simulación. (Ilustración 33)

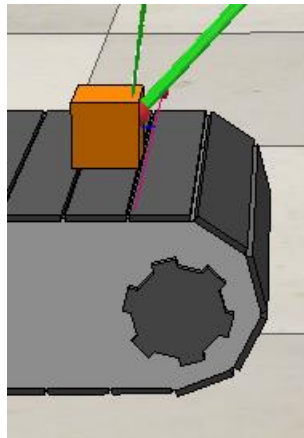


Ilustración 33

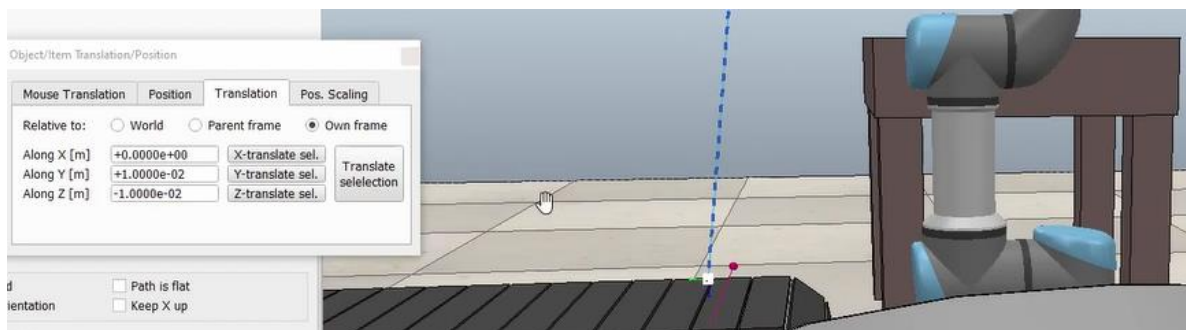
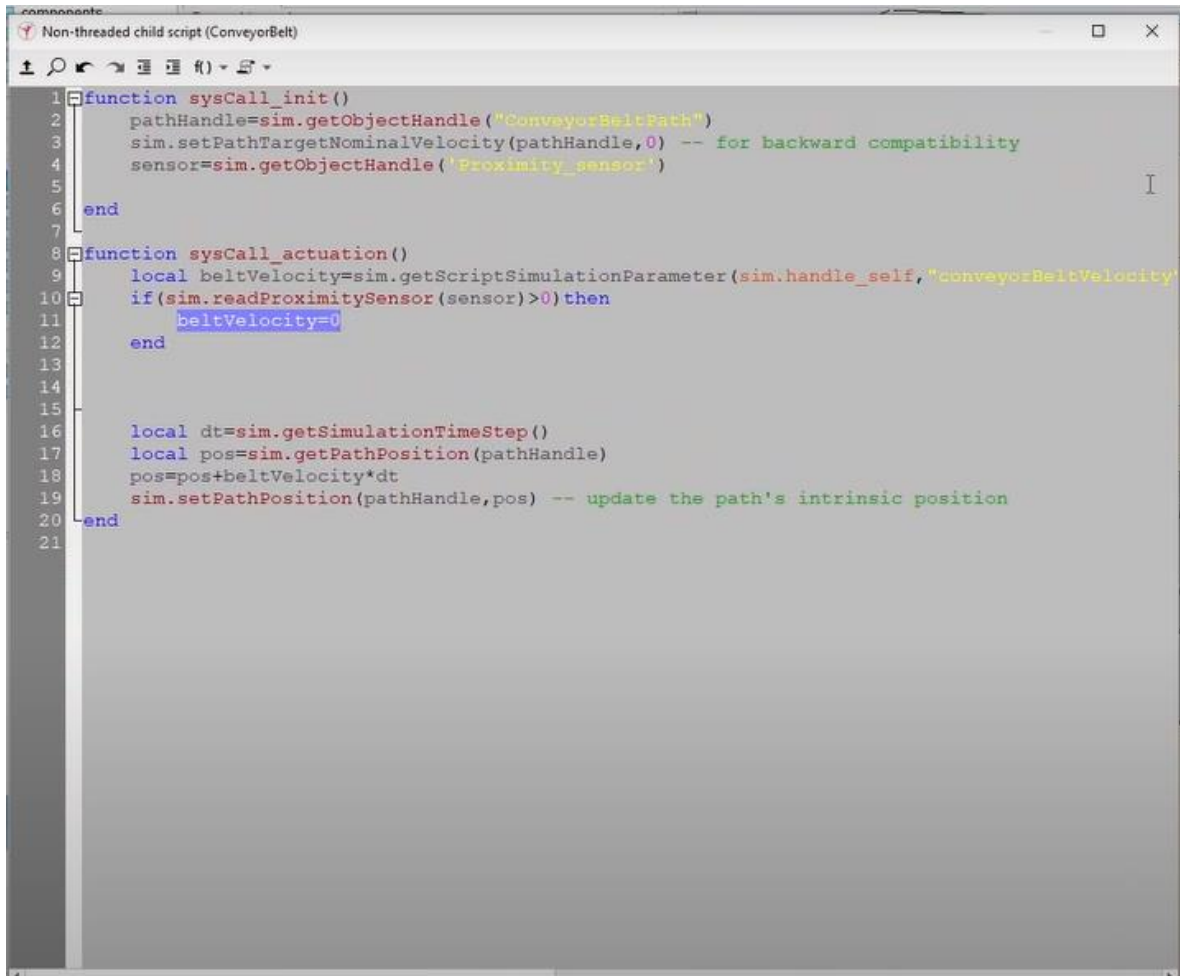


Ilustración 34

## PROGRAMACIÓN

### CONVEYOR BELT.



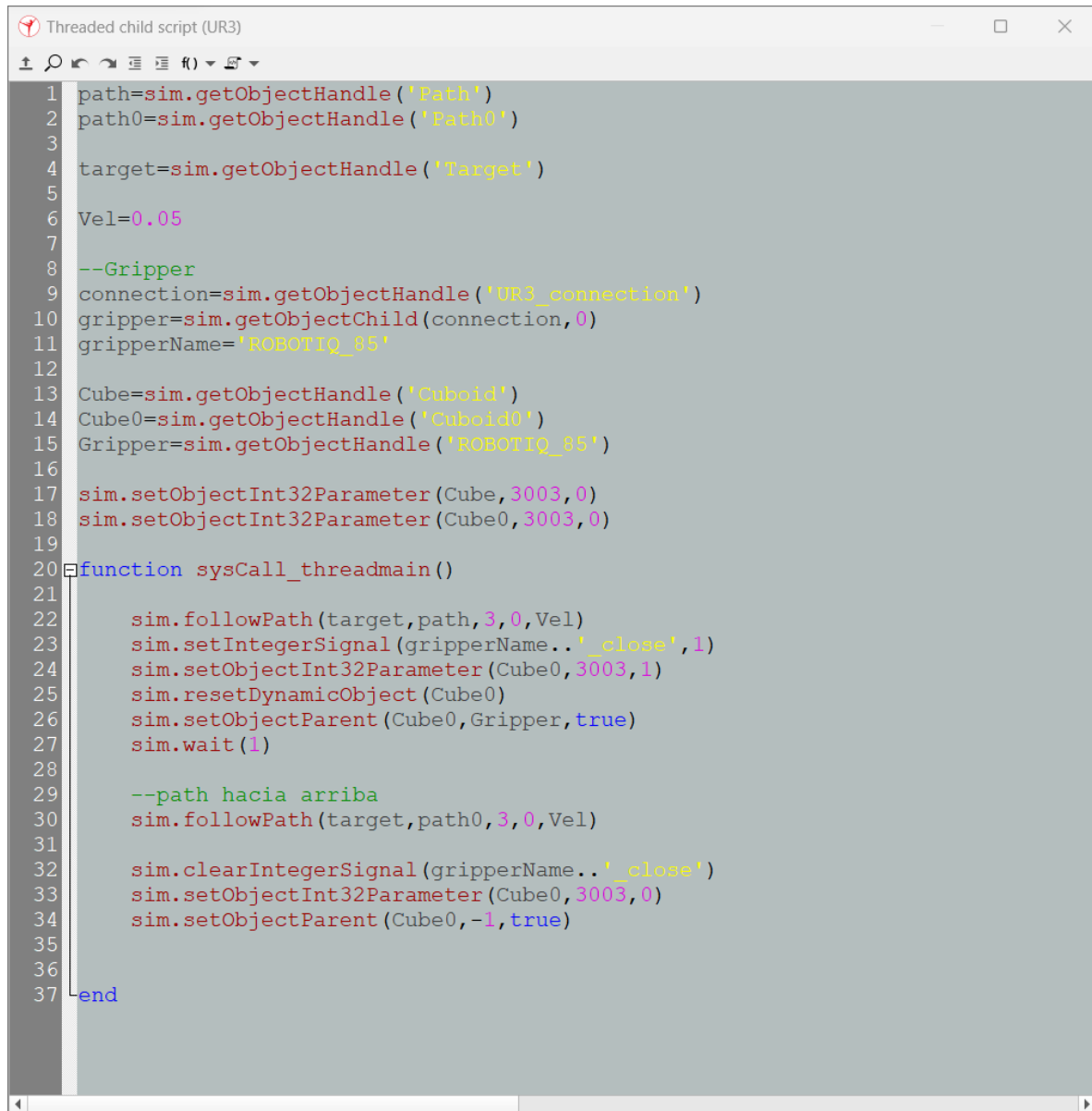
```
Non-threaded child script (ConveyorBelt)

1 function sysCall_init()
2     pathHandle=sim.getObjectHandle("ConveyorBeltPath")
3     sim.setPathTargetNominalVelocity(pathHandle,0) -- for backward compatibility
4     sensor=sim.getObjectHandle('Proximity_sensor')
5
6 end
7
8 function sysCall_actuation()
9     local beltVelocity=sim.getScriptSimulationParameter(sim.handle_self,"conveyorBeltVelocity")
10    if(sim.readProximitySensor(sensor)>0) then
11        beltVelocity=0
12    end
13
14
15
16    local dt=sim.getSimulationTimeStep()
17    local pos=sim.getPathPosition(pathHandle)
18    pos=pos+beltVelocity*dt
19    sim.setPathPosition(pathHandle,pos) -- update the path's intrinsic position
20 end
21
```

Ilustración 35. (Montiel, 2021)



## UR3



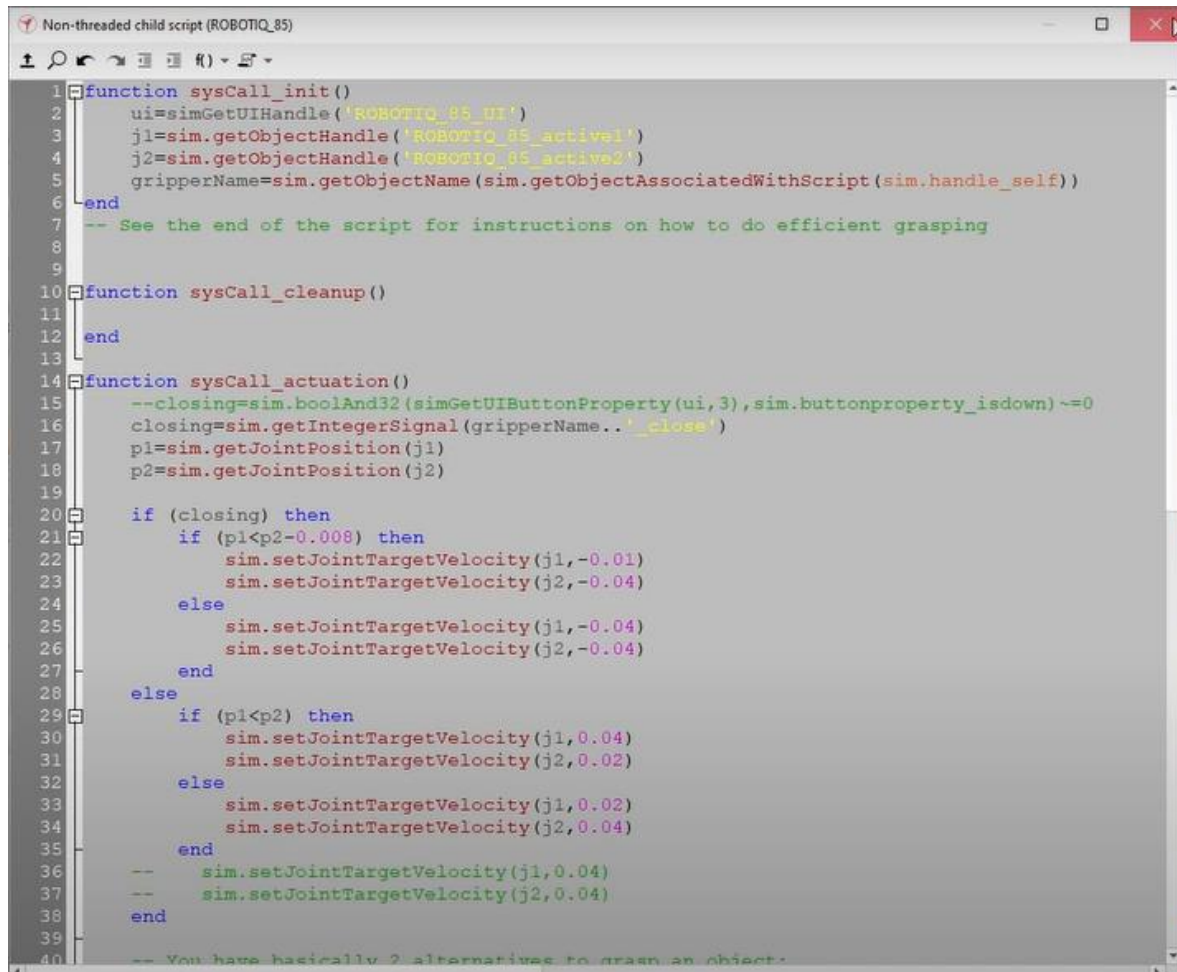
```
1 path=sim.getObjectHandle('Path')
2 path0=sim.getObjectHandle('Path0')
3
4 target=sim.getObjectHandle('Target')
5
6 Vel=0.05
7
8 --Gripper
9 connection=sim.getObjectHandle('UR3_connection')
10 gripper=sim.getObjectChild(connection,0)
11 gripperName='ROBOTIQ_85'
12
13 Cube=sim.getObjectHandle('Cuboid')
14 Cube0=sim.getObjectHandle('Cuboid0')
15 Gripper=sim.getObjectHandle('ROBOTIQ_85')
16
17 sim.setObjectInt32Parameter(Cube,3003,0)
18 sim.setObjectInt32Parameter(Cube0,3003,0)
19
20 function sysCall_threadmain()
21
22     sim.followPath(target,path,3,0,Vel)
23     sim.setIntegerSignal(gripperName..'_close',1)
24     sim.setObjectInt32Parameter(Cube0,3003,1)
25     sim.resetDynamicObject(Cube0)
26     sim.setObjectParent(Cube0,Gripper,true)
27     sim.wait(1)
28
29     --path hacia arriba
30     sim.followPath(target,path0,3,0,Vel)
31
32     sim.clearIntegerSignal(gripperName..'_close')
33     sim.setObjectInt32Parameter(Cube0,3003,0)
34     sim.setObjectParent(Cube0,-1,true)
35
36
37 end
```

Ilustración 36. (Montiel, 2021)

### Observaciones:

- Para que el cubo no se suelte a mitad del camino se establece una relación donde se establece el cubo se convierte en objeto estático a partir del momento en que la garra lo sujeta.

## GRIPPER



```
1 function sysCall_init()
2     ui=simGetUIHandle('ROBOTIQ_85_HI')
3     j1=sim.getObjectHandle('ROBOTIQ_85_active1')
4     j2=sim.getObjectHandle('ROBOTIQ_85_active2')
5     gripperName=sim.getObjectAssociatedWithScript(sim.handle_self())
6 end
7 -- See the end of the script for instructions on how to do efficient grasping
8
9
10 function sysCall_cleanup()
11
12 end
13
14 function sysCall_actuation()
15     --closing=sim.boolAnd32(simGetUIButtonProperty(ui,3),sim.buttonproperty_isdown)~=0
16     closing=sim.getIntegerSignal(gripperName..'close')
17     p1=sim.getJointPosition(j1)
18     p2=sim.getJointPosition(j2)
19
20     if (closing) then
21         if (p1<p2-0.008) then
22             sim.setJointTargetVelocity(j1,-0.01)
23             sim.setJointTargetVelocity(j2,-0.04)
24         else
25             sim.setJointTargetVelocity(j1,-0.04)
26             sim.setJointTargetVelocity(j2,-0.04)
27         end
28     else
29         if (p1<p2) then
30             sim.setJointTargetVelocity(j1,0.04)
31             sim.setJointTargetVelocity(j2,0.02)
32         else
33             sim.setJointTargetVelocity(j1,0.02)
34             sim.setJointTargetVelocity(j2,0.04)
35         end
36     -- sim.setJointTargetVelocity(j1,0.04)
37     -- sim.setJointTargetVelocity(j2,0.04)
38 end
39
40 -- You have basically 2 alternatives to grasp an object:
```

Ilustración 37. (Montiel, 2021)

## CONCLUSIÓN

Al desarrollar este proyecto pudimos poner en práctica los conocimientos base adquiridos del simulador Coppelia e implementación del Robot UR3, sin embargo, ante este proyecto nuestros conocimientos de programación se basaron en guías previamente instruidas. De igual forma aprendimos sobre programación a objeto y estructurar bases en el simulador Coppelia.

Podemos concluir que nuestra experiencia con el simulador Coppelia servirá de instructivos para futuros estudiantes que estén cursando la carrera.

## REFERENCIAS

Coppelia Robotics. (s.f.). *Scene objects*. Obtenido de CoppeliaSim User Manual:  
<https://coppeliarobotics.com/helpFiles/en/objects.htm>

Marras, J. J. (14 de Marzo de 2016). *Simuladores: Coppelia Robotics V-REP (Virtual Experimentation Platform)*. Obtenido de Romerobots Blog:  
<https://jjromeromarras.wordpress.com/2016/03/14/simuladores-coppelia-robotics-v-rep-virtual-experimentation-platform/>

Mecalux, S.A. (7 de Mayo de 2020). *El 'pick and place': los robots no se cansan de repetir movimientos*. Obtenido de MECALUX:  
<https://www.mecalux.com.mx/blog/pick-and-place>

Montiel, A. A. (Dirección). (2021). *Coppelia pick and place* [Película].