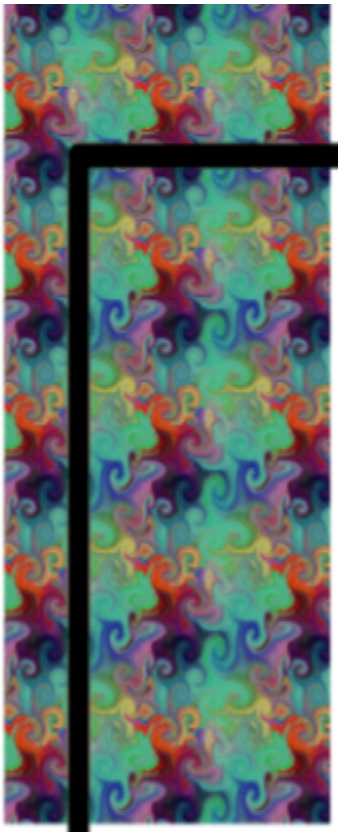


Actividad 1:
Robot Sencillo de Coppelia

CE Inteligencia Artificial y Big Data

Modelos de Inteligencia Artificial

2024/2025

Daniel Marín López

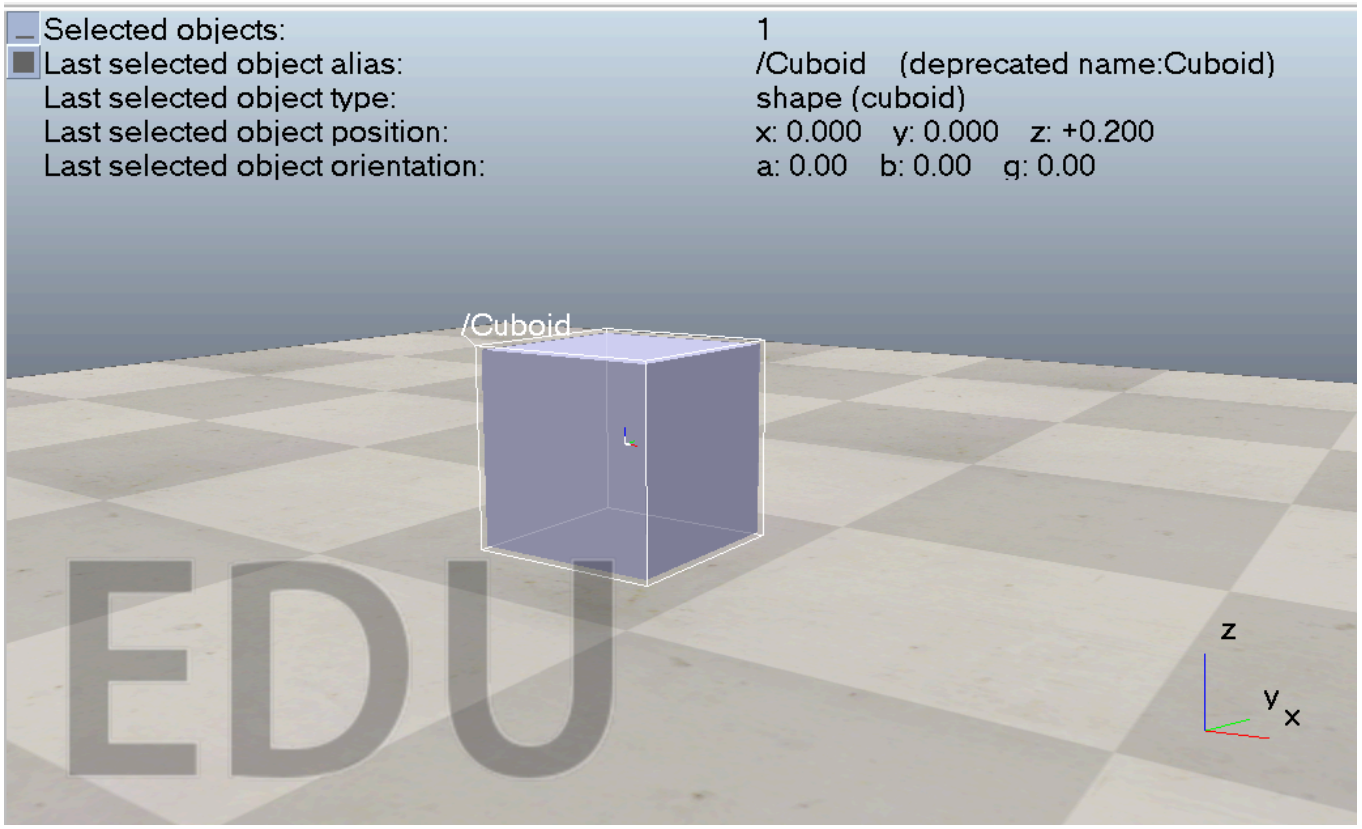
Índice

1. Enunciado

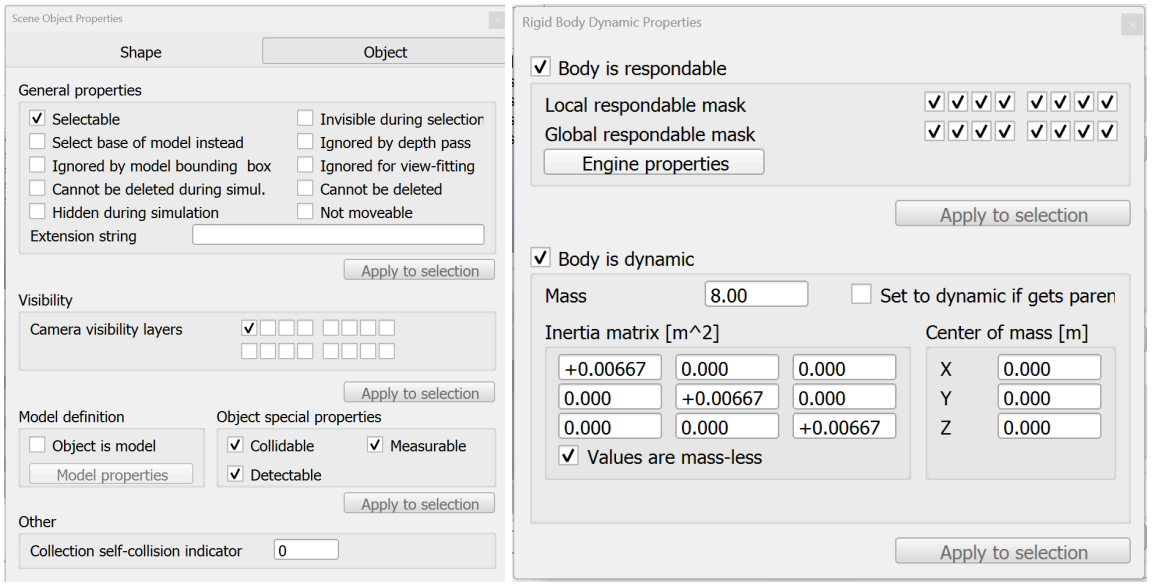
Siguiendo con las instrucciones del punto 5.7, así como las instrucciones facilitadas en clase, realizar el diseño de un robot básico. En el caso en el que no puedas seguir las instrucciones en clase a continuación tienes un enlace para poder realizar la actividad.

2. Parte I: Creación del robot

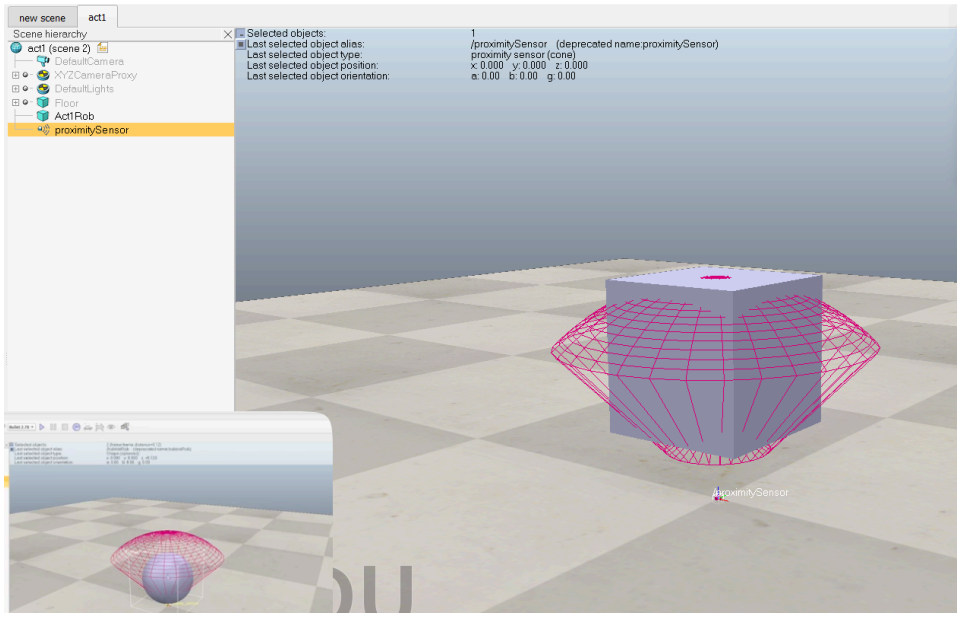
Lo primero que haremos será abrir la aplicación de CoppeliaSim y crearemos un cubo en ***Add/Primitive shape/Cuboid*** de 0.2 de alto, ancho y profundidad. Y lo elevamos 0.2 en el eje z.



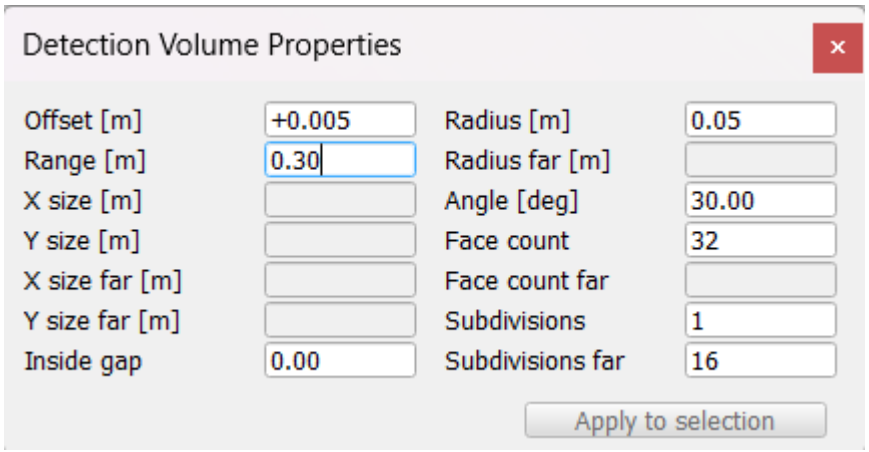
Revisamos sus propiedades y hacemos que sea dinámico, reactivo, colisionable, detectable y medible.



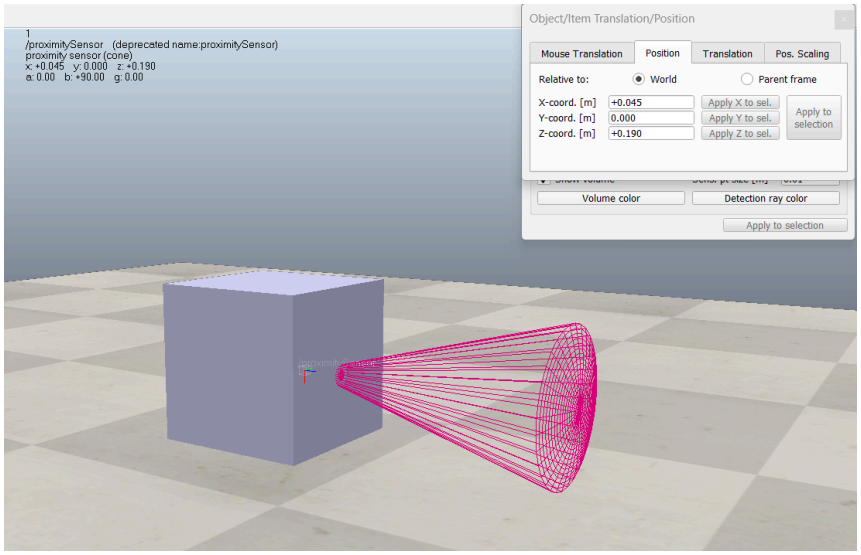
Con eso, tenemos de momento el cuerpo de nuestro robot. Lo siguiente es agregar un sensor que detectará los objetos a su alrededor en ***Add/Proximity sensor/Cone type***.



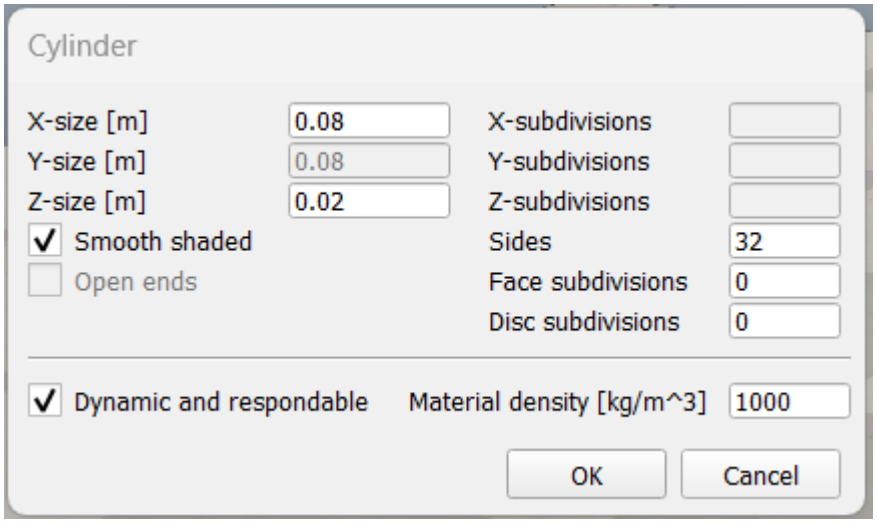
Cambiamos sus propiedades de detección a las siguientes:



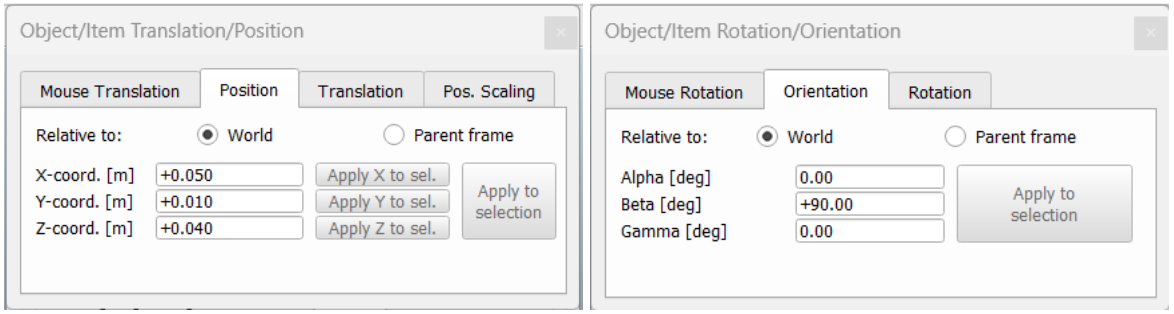
Y lo agregamos a la jerarquía de nuestro robot. Para ello lo arrastraremos hasta ponerlo encima del cubo y al soltar se quedará dentro. Lo debemos poner en esta posición.



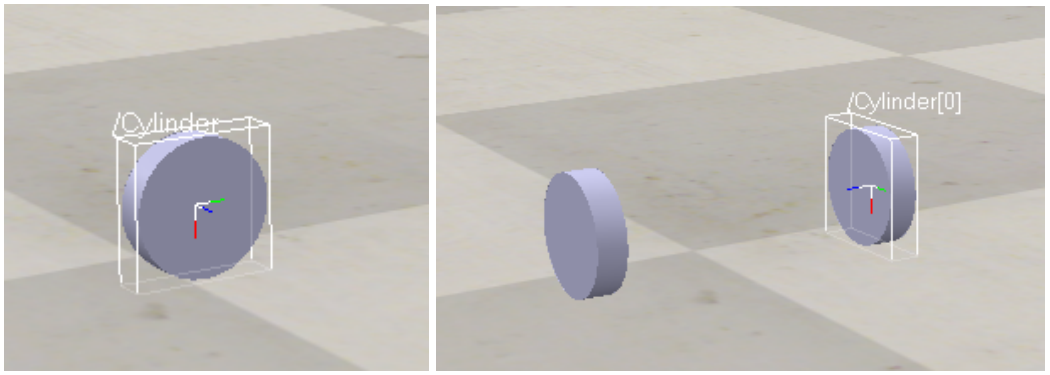
Ahora, en otra escena, crearemos las ruedas con las que nuestro robot se moverá. Para ello, crearemos un cilindro en ***Add/Primitive shapes/Cylinder*** con las siguientes medidas:



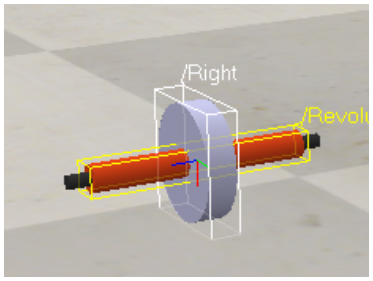
A continuación, debemos posicionar y orientar la rueda de la siguiente manera:



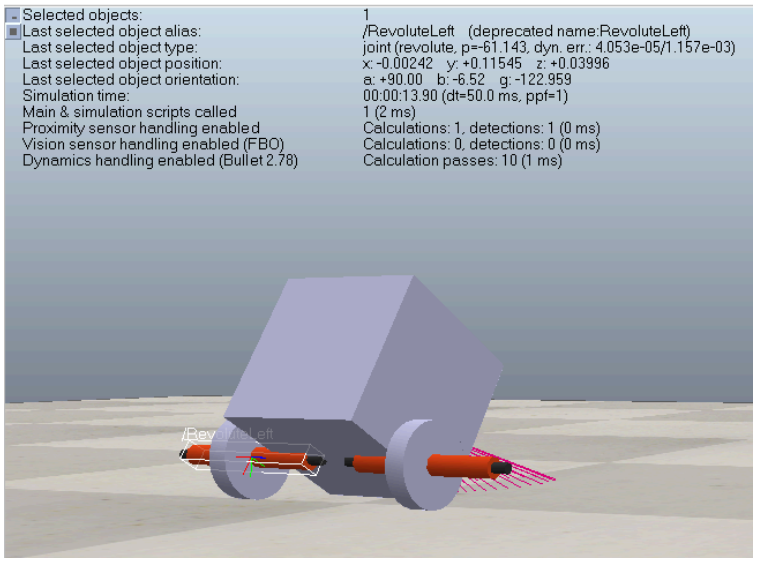
Esto hará que nuestra rueda quede perfectamente orientada. Solamente la clonamos y ya estarían nuestras ruedas.



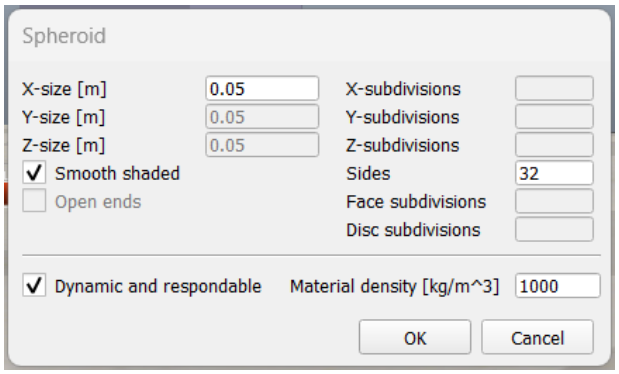
Lo siguiente será agregar una articulación en ***Add/Joint/Revolute***. Para hacer que la articulación quede encajada dentro de la rueda debemos primero seleccionar la articulación y luego la rueda, una vez seleccionadas las dos iremos tanto al apartado de posición como de rotación y le daremos al botón de "Apply to selection". Esto hará que nuestra articulación y rueda queden casi unidas. Solamente la rueda debe estar dentro de la articulación.



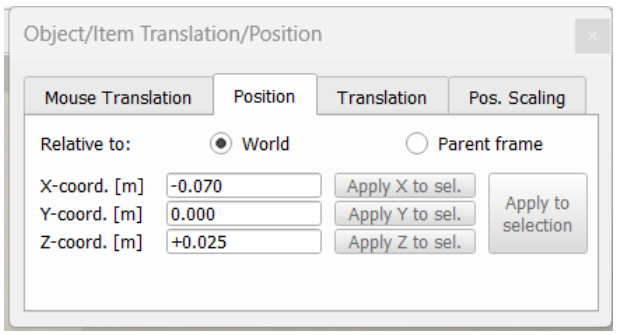
Tras terminar, pasamos las ruedas al robot. Si le damos al Play, veremos que nuestro robot se cae.

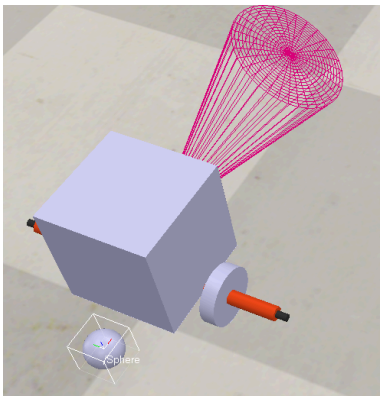


Para evitar esto, crearemos un conector que le sirva de peso para el robot. Para ello, primero agregaremos una esfera con un radio de 0.05.

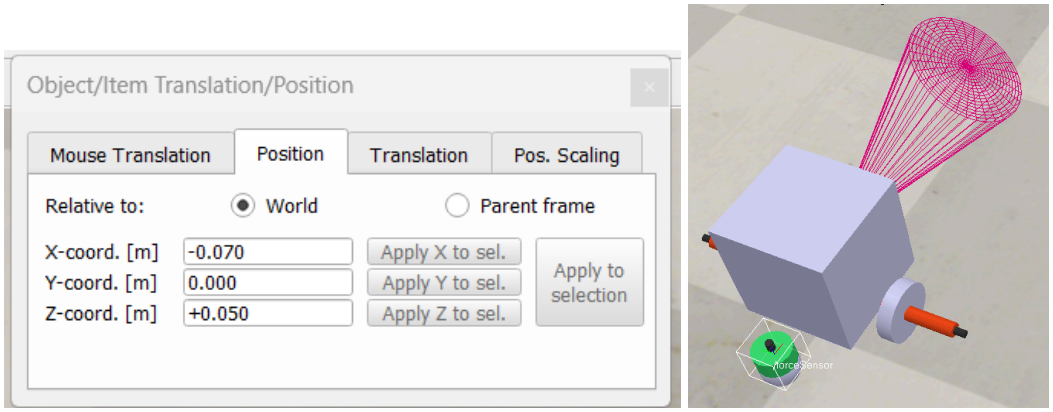


Que debe estar en esta posición.

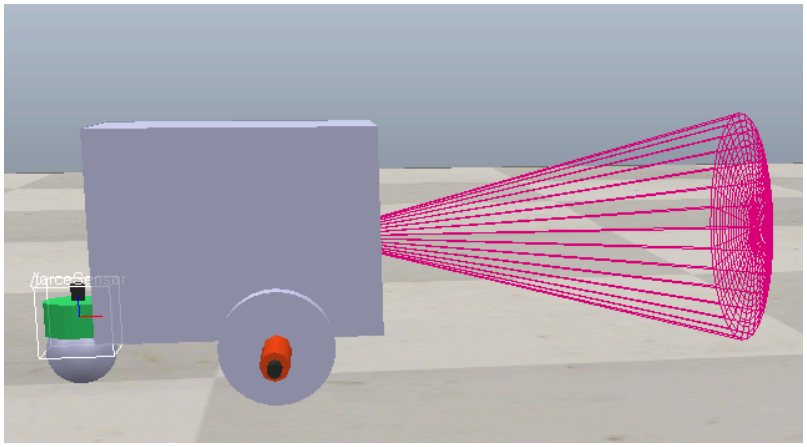




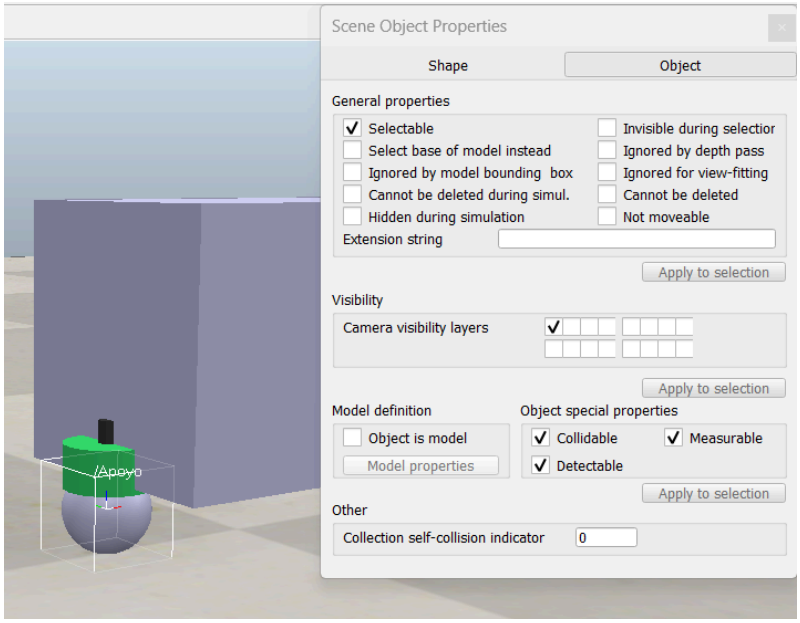
Lo siguiente será añadir un sensor de fuerza en ***Add/Force sensor***. Este debe estar encima de la esfera.



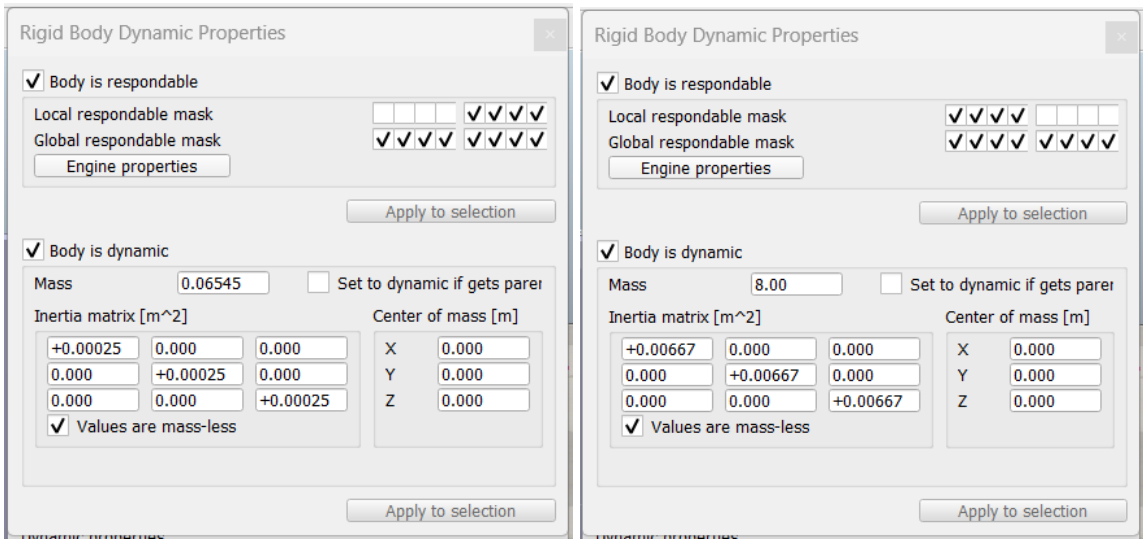
Como se queda fuera del robot, lo modificamos para que sea una caja.



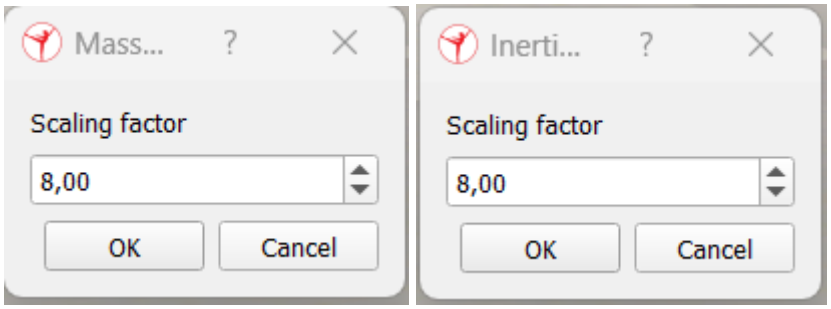
Lo siguiente será hacer a la esfera colisionable, detectable y medible.



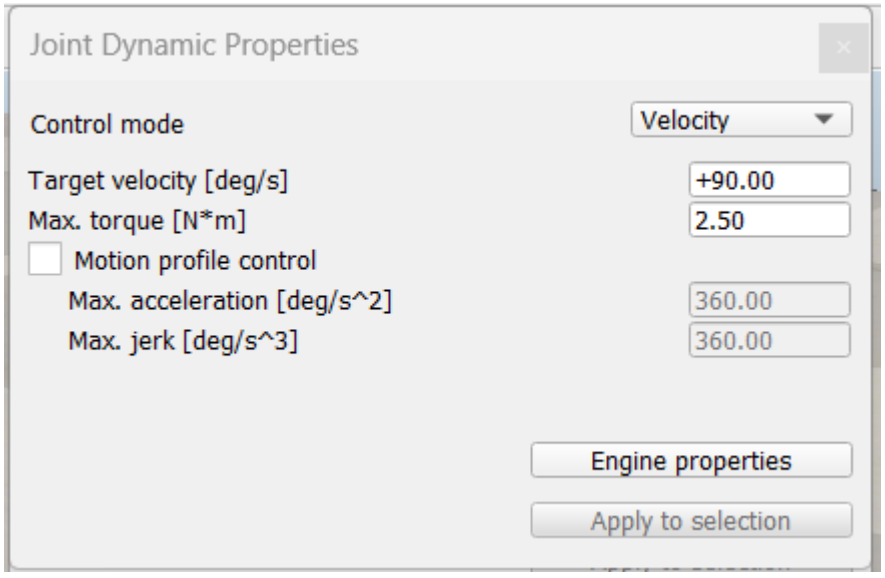
Sin embargo, cuando le damos a Play nuestro robot y la esfera entran en contacto y eso hace que haya fricción. Como eso no lo queremos debemos desmarcar las máscaras de respuesta local, para el robot será la segunda hilera mientras que para la esfera será la primera.



Esto impedirá que nuestro robot y el conector no provoquen este comportamiento. Sin embargo, si le damos a Play de nuevo veremos que nuestro robot se mueve de manera rara. Esto se debe al tipo de simulación que usa CoppeliaSim y como están los pesos y las inercias de los objetos del robot, lo que se recomienda en la documentación es seleccionar todos los objetos (el cubo, las ruedas y la esfera) y nos iremos a ***Edit/Shape mass and inertia*** y escogeremos tanto ***scale mass*** como ***scale inertia*** con un valor de 8 ambas.

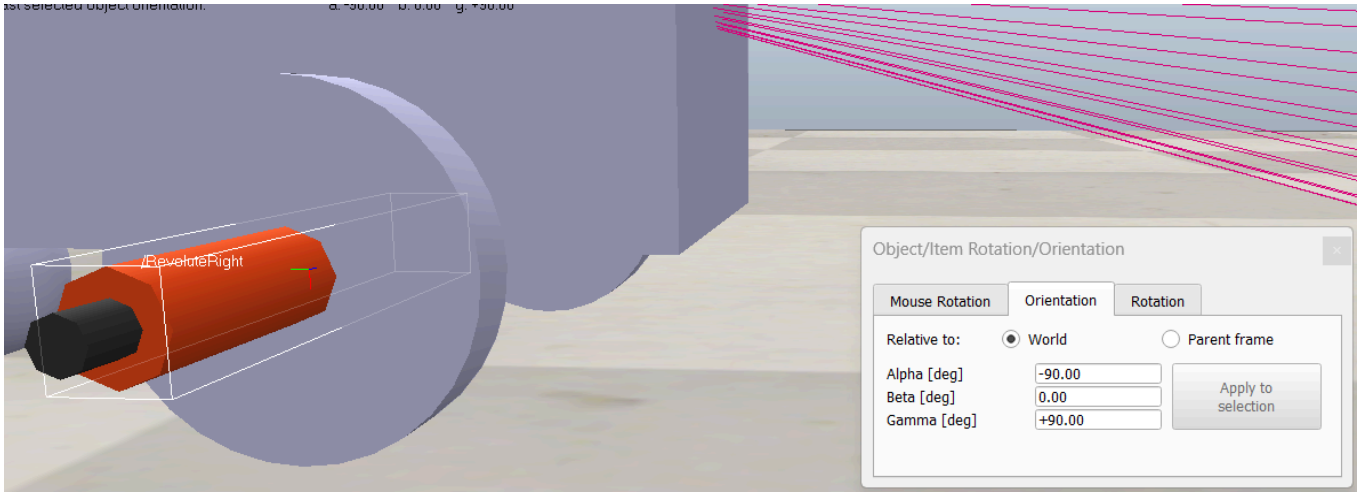


Con esto, las masas y las inercias de los objetos se balancearán y el robot no se moverá en la simulación. Por último, debemos ponerle a los motores (articulaciones) una velocidad para que se mueva el robot. Por lo que seleccionamos los motores y ponemos lo siguiente:

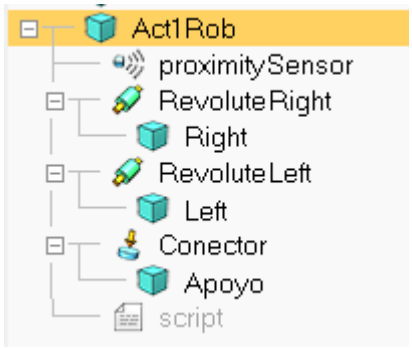


Cambiamos el control mode de Free a Velocity y ponemos una velocidad de 90º por segundo.

Con esto, nuestro robot ya se mueve. Pero va hacia atrás, la solución es girar los motores 180º sobre el eje x.

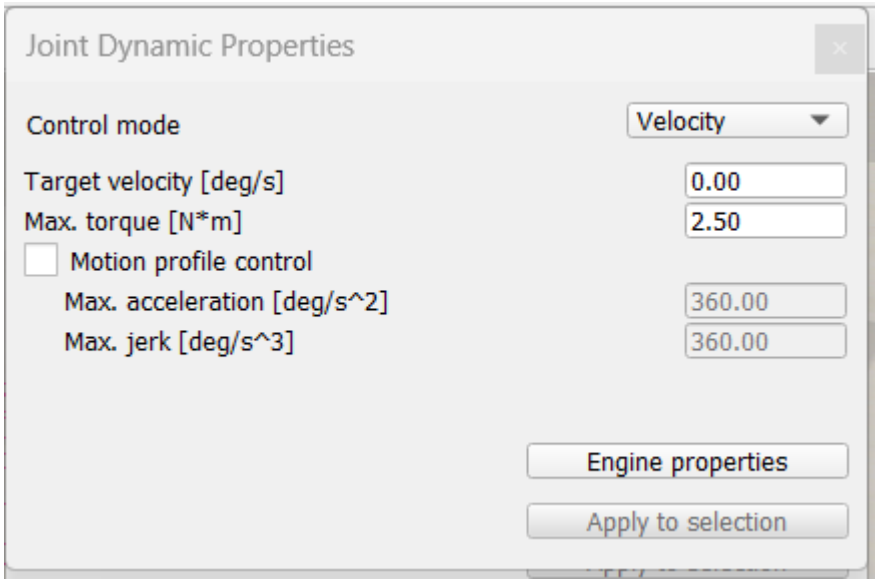


De esta manera, nuestro robot ya puede andar perfectamente. La jerarquía del robot debe quedar más o menos de la siguiente manera:



3. Parte II: Codificando el comportamiento del robot

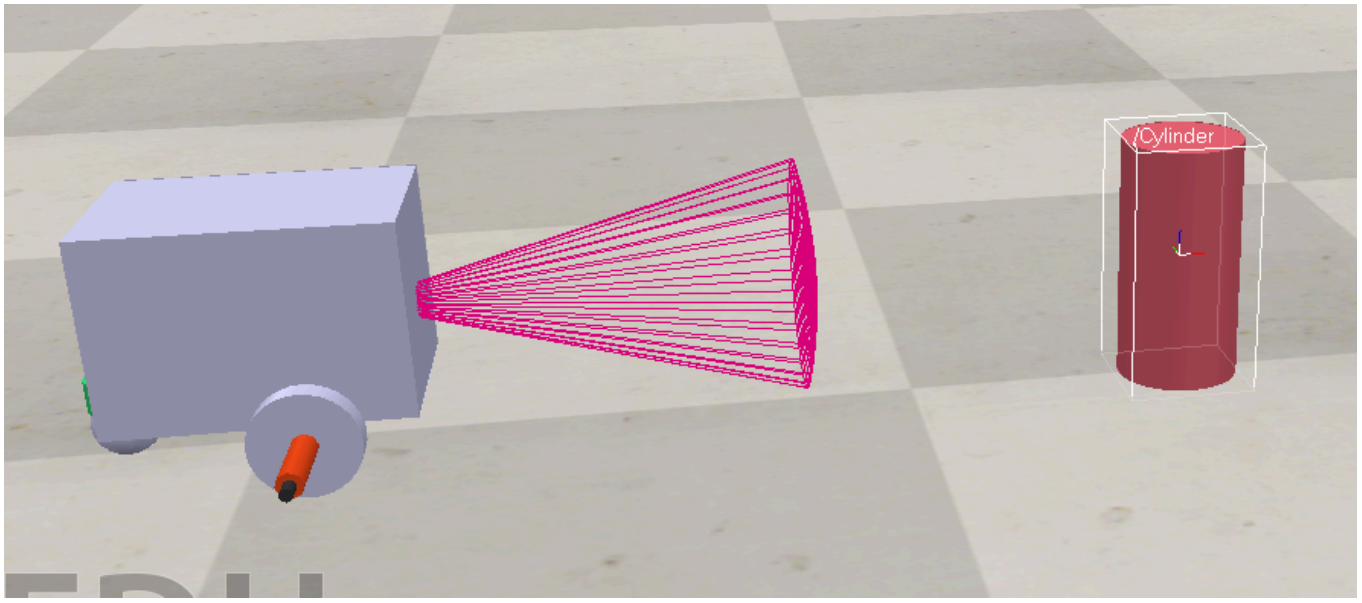
Lo siguiente será hacer que nuestro robot se mueva mediante código y graficar lo que el sensor está captando a su alrededor, con esto podremos ir viendo y pensar como el robot debe actuar cuando detecta un objeto delante de él. Primero, volvemos a la velocidad de los motores y ponemos que sea 0.



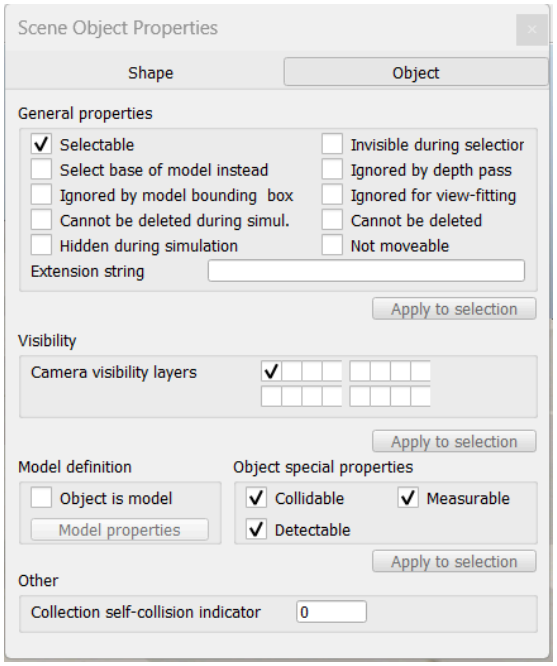
Lo siguiente será crear un script de simulación en ***Add/Script/Simulation script/Non threaded/Lua***. Lo hacemos en Lua debido a que Coppelia no encuentra el intérprete de Python y no sabemos donde se define, por lo que hacerlo en Lua será lo mejor. El script debe estar dentro del robot también. Damos click sobre el símbolo del script para abrir el editor y escribimos el siguiente código:

```
function sysCall_init()
    sim = require('sim')

    -- Asegúrate de inicializar 'self'
    self = {}

    -- Usa nombre absoluto si lo prefieres
    self.leftMotor = sim.getObject('/Act1Rob/RevoluteLeft')
    self.rightMotor = sim.getObject('/Act1Rob/RevoluteRight')

    -- Validación
    if self.leftMotor == -1 then
        sim.addLog(sim.verbosity_errors, "No se encontró /Act1Rob/RevoluteLe
    end
    if self.rightMotor == -1 then
        sim.addLog(sim.verbosity_errors, "No se encontró /Act1Rob/RevoluteRi
    end

    -- Velocidad en rad/s
    self.initSpeed = 90 * math.pi / 180

    -- Asignar velocidad
    sim.setJointTargetVelocity(self.leftMotor, self.initSpeed)
    sim.setJointTargetVelocity(self.rightMotor, self.initSpeed)

    -- do some initialization here
end
```

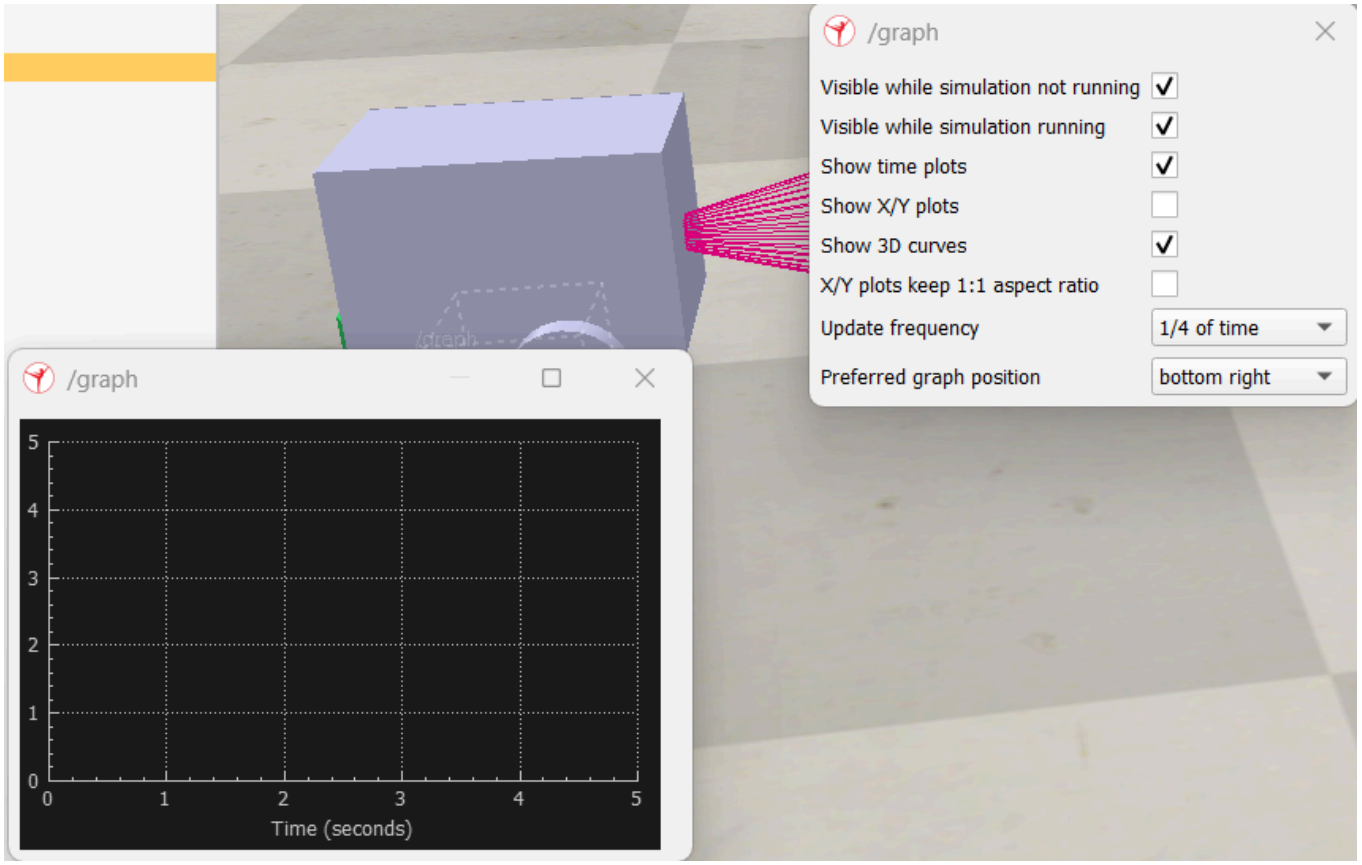
Este código buscará los motores del robot y cuando los encuentre se empezará a mover a una velocidad inicial de 90º por segundo, debemos tener en cuenta que la velocidad debe estar en radianes por segundo. Cuando le damos a Play, nuestro robot ahora se mueve gracias al código. Lo siguiente será agregar un cilindro y probar el sensor de proximidad.



Este cilindro debe ser colisionable, detectable y medible.



Otra cosa que también podemos hacer es agregar un gráfico que nos permita ver que está recogiendo el sensor. Para ello le daremos a ***Add/Graph***. De este gráfico solo quitaremos que muestre los ejes X/Y.



Ahora, en el código debemos primero definir el sensor de la misma forma que hemos hecho con los motores. También definiremos el gráfico y la línea que trazará la cual llamaremos ***distancia_libre***.

```
-- Usa nombre absoluto si lo prefieres
self.leftMotor = sim.getObject('/Act1Rob/RevoluteLeft')
self.rightMotor = sim.getObject('/Act1Rob/RevoluteRight')
self.sensor = sim.getObject('/Act1Rob/proximitySensor')
self.graph = sim.getObject('/Act1Rob/graph')
self.stream = sim.addGraphStream(self.graph, 'distancia_libre', 'm, 0,
```

Las funciones que usará nuestro sensor deben ir en la función ***sysCall_sensing()*** que es de la siguiente forma:

```
function sysCall_sensing()
    -- put your sensing code here

    res, dist, point, obj, n = sim.readProximitySensor(self.sensor)

    if res == 1 then
        -- Asignar nueva velocidad
        sim.setJointTargetVelocity(self.leftMotor, self.initSpeed*dist*5)
        sim.setJointTargetVelocity(self.rightMotor, self.initSpeed*dist*5)
    end
```

```
if res == 1 then
    -- Agregamos línea de gráfico
    sim.setGraphStreamValue(self.graph, self.stream, dist)
    -- Asignar nueva velocidad
    sim.setJointTargetVelocity(self.leftMotor, self.initSpeed*dist*5)
    sim.setJointTargetVelocity(self.rightMotor, self.initSpeed*dist*5)
end
```

Primero recogemos las lecturas del sensor y luego si todo funciona correctamente, le dirá que rebaje la velocidad multiplicando la velocidad inicial por la distancia (el 5 era de prueba). También se irá recogiendo la distancia y se imprimirá en el gráfico.