Práctica 1. Introducción al simulador *CoppeliaSim*

Tecnología y Arquitectura Robótica

Grado en Ingeniería Informática

Contenidos

Introducción

CoppeliaSim es un simulador de robótica, antes conocido como V-REP, y que permite simular comportamientos de robots desde su comportamiento dinámico, su cinemática, diversos sensores y una gran multitud de elementos y efectos físicos relevantes en un sistema robotizado.

Una de las grandes ventajas del CoppeliaSim es que puede programarse en LUA (lenguaje de programación propio), C/C++, Matlab y Python. Sin embargo, utilizando la licencia gratuita de educación sólo puede programarse con Lua y C/C++.

Existe una extensa documentación oficial sobre este simulador que podéis encontrar en el siguiente enlace: https://www.coppeliarobotics.com/helpFiles/index.html. Además, hay multitud de páginas y tutoriales disponibles en Internet, los cuales pueden ser de gran ayuda.

Objetivos

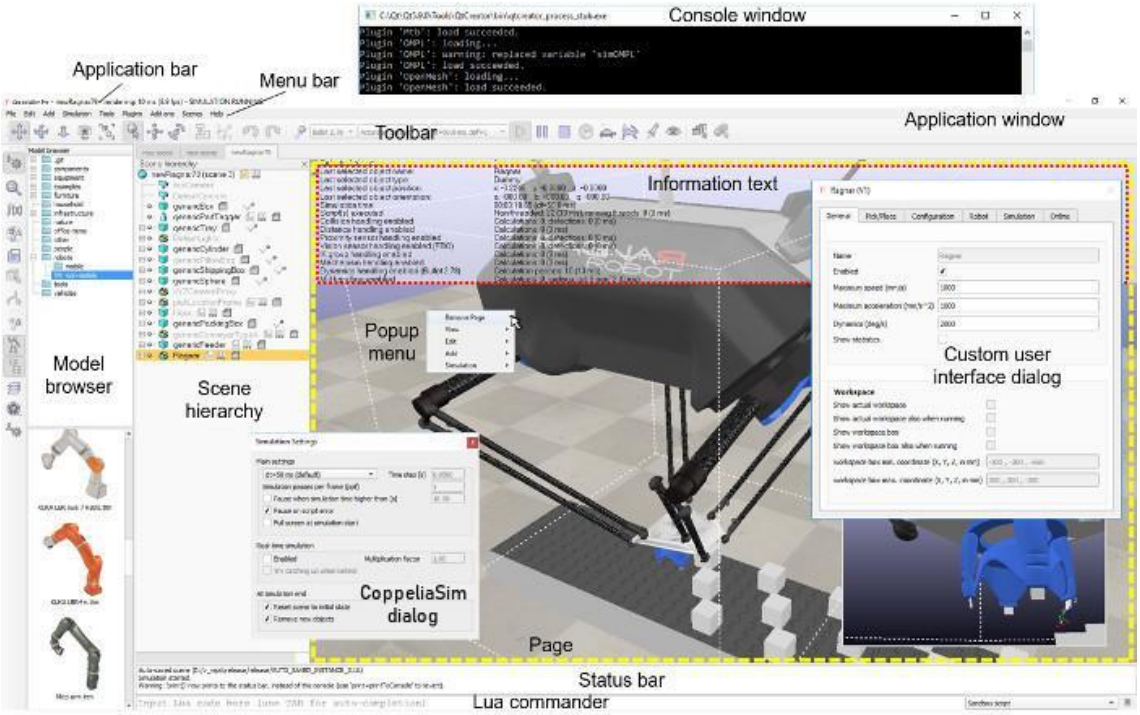
El principal objetivo de esta primera práctica es conocer los elementos básicos del simulador. Para ello exploraremos su entorno, conociendo las principales funcionalidades del mismo; luego crearemos elementos básicos y definiremos sus características para evaluar cómo se comportan en función de esas características; por último, crearemos un robot sencillo pero funcional.

Parte 1. Conociendo el entorno CoppeliaSim

En primer lugar, descargaremos el software de la página oficial en su versión ***edu*** de educación, la cual es gratuita.

Web de descarga: https://www.coppeliarobotics.com/downloads

Una vez descargado e instalado, podemos observar la interfaz que nos ofrece inicialmente. En ella podemos ver:



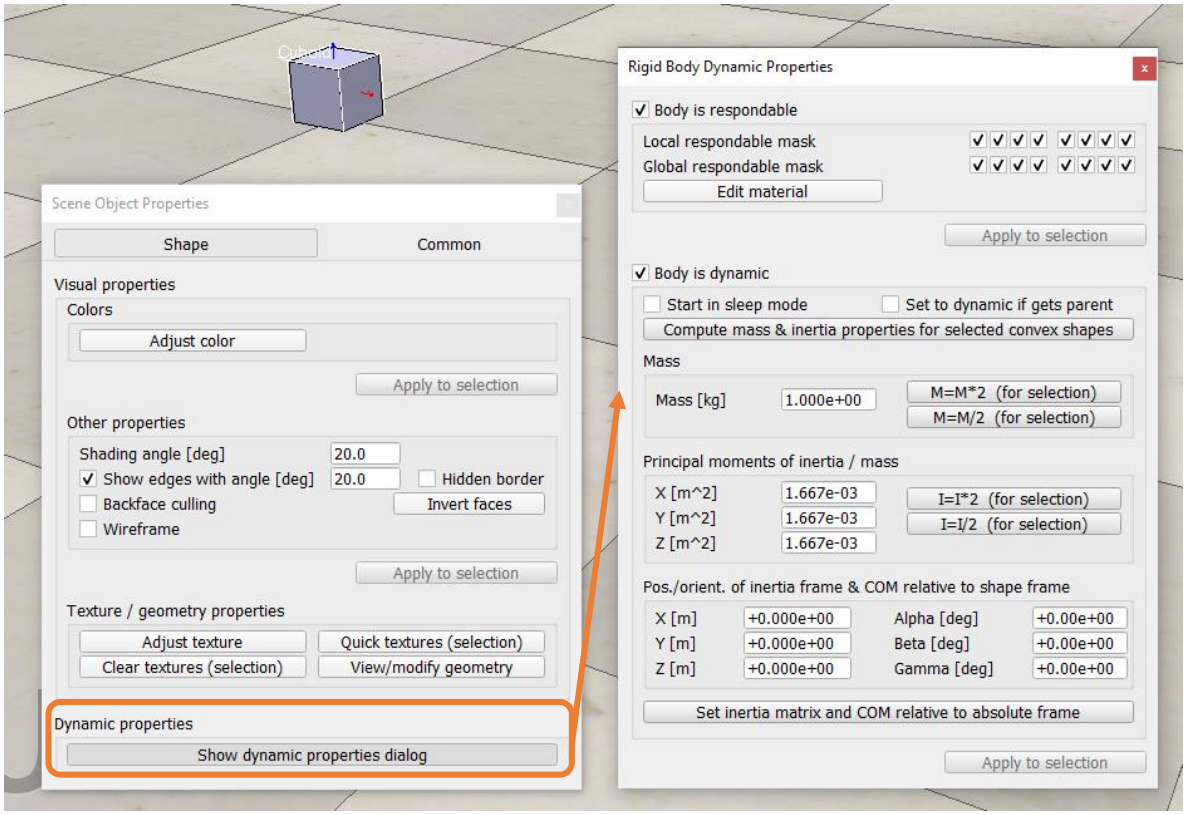
- Un menú superior con los botones File, Edit, Add… Desde este menú se puede acceder a la gran mayoría de funcionalidades del simulador. Muchas de las funcionalidades de este menú pueden ser accedidas desde otros botones de acceso rápido o pop-ups que aparecen en la pantalla.
- Una barra de herramientas horizontal. Esta tiene diversas funciones para movernos en el espacio, trasladar y rotar objetos, ejecutar la simulación y cambiar entre vistas.
- Una barra de herramientas vertical. En esta se activan diferentes opciones como la configuración de la simulación, las ventanas de modelos y jerarquía, el editor de formas o las capas visibles.
- Una ventana con los modelos predefinidos, tanto de robots, de elementos de infraestructura como paredes, como otros tantos que permiten crear nuestro entorno a simular.
- Una ventana con la jerarquía de escena. Esta muestra los elementos que hay en escena y sus dependencias parentales, cuyo efecto iremos viendo en clase, pero que se resume en que lo que afecta a un objeto padre, afecta de igual forma a los hijos.
- En la zona principal vemos la ventana en la que se muestra la escena y sus elementos.
- Eje de coordenadas del mundo. En la parte inferior derecha se ve el eje de coordenadas de la escena, con el eje *x* en rojo, el eje *y* en verde, y el eje *z* en azul. Cabe destacar que cada elemento tiene un eje de coordenadas propio que no tiene porque estar alineado con el del mundo.

Propiedades de los objetos

Los objetos primitivos o puros tienen distintas propiedades. Es importante conocer qué efecto tienen estas propiedades en la dinámica de la simulación:

Tamaño: el tamaño del objeto en *x*, *y*, y *z* definirán el espacio que ocupan.

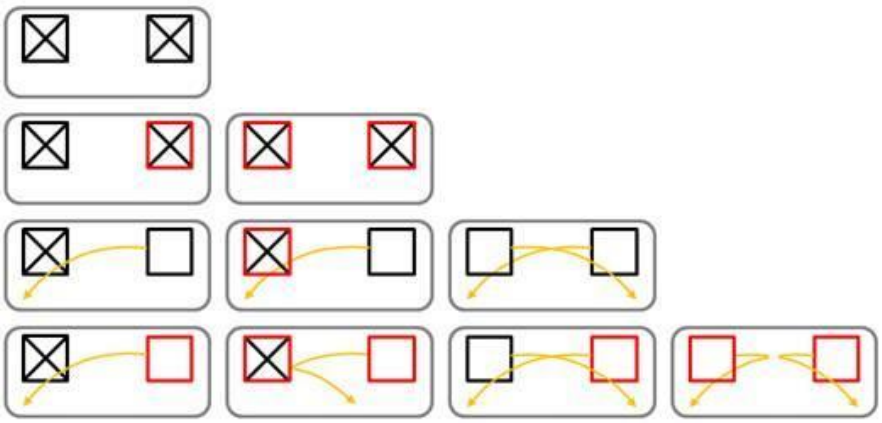
1. **Shape**: Estas propiedades definen el aspecto y el comportamiento del objeto.



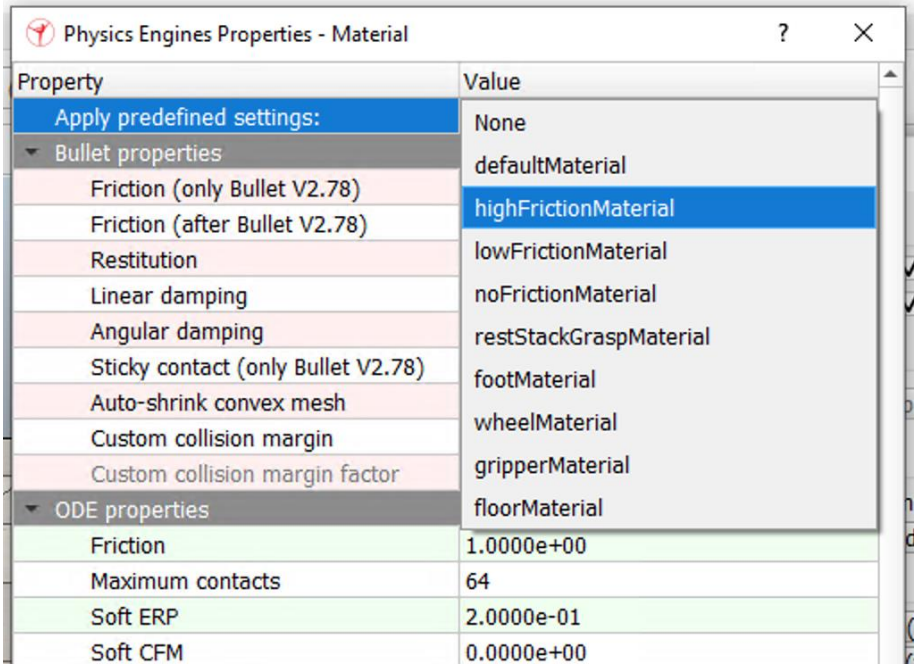
En las propiedades dinámicas del objeto vemos dos partes:

- Body is *respondable*: marcar esta propiedad supone que el objeto reacciona cuando colisiona con otro objeto. De esta forma podemos encontrarnos con las siguientes casuísticas.

	Static	Non-static
Non-respondable	☒	☐
Respondable	☒	☐



El botón Edit Material tiene abre una ventana con la configuración específica de los aspectos que le hacen reaccionar, permitiendo elegir ciertos pre-sets (*Apply predefined settings)* que ya definen multitud de parámetros, como el *highFrictionMaterial* que se puede utilizar para que las ruedas de un robot no patinen.

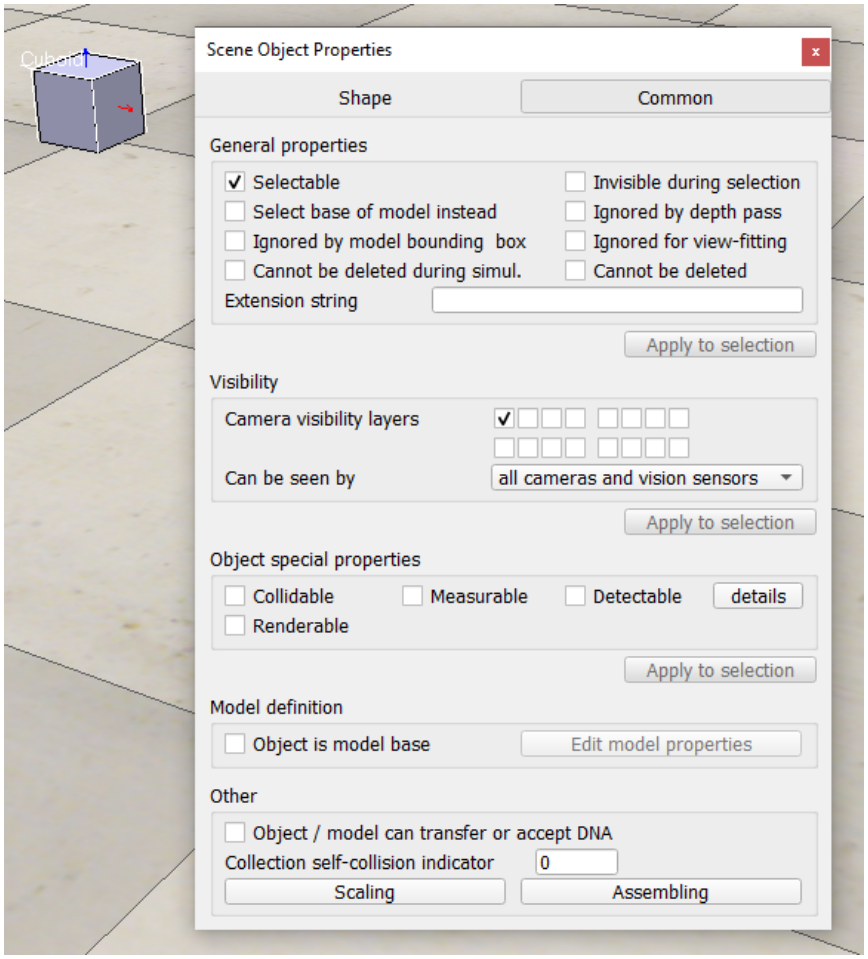


- Body is dynamic: esta propiedad indica si el objeto es dinámico o no, es decir, si se ve afectado por las fuerzas existentes como la gravedad, momentos de inercia, etc.

2. **Common:** son opciones comunes que determinan si puedes o no seleccionar el objeto, si puedes visibilizarlo y en qué capa, etc. En este menú la parte más importante son las Propiedades Especiales del Objeto:
    - Collidable: Permite activar la detección de colisiones
    - Measurable: Permite activar el cálculo de distancia mínima hacia el objeto por parte de sensores.
    - Detectable:: Permite activar la detección por sensores de proximidad. Si se accede al botón *details* se puede seleccionar concretamente qué sensores podrán detectarlo. De esta forma podemos simular que el objeto es cristal, por

lo que un sensor láser no podría detectarlo, o un objeto metálico, por lo que los sensores inductivos podrán detectarlo.

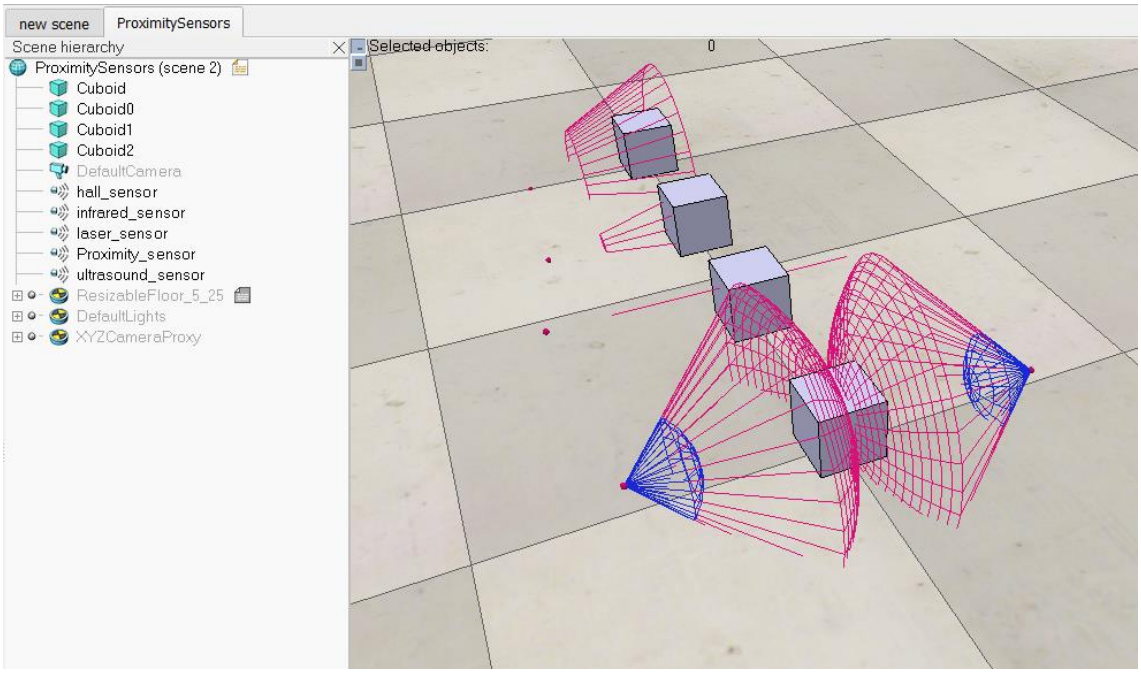
- Renderable: permite activar su detección por sensores de visión.



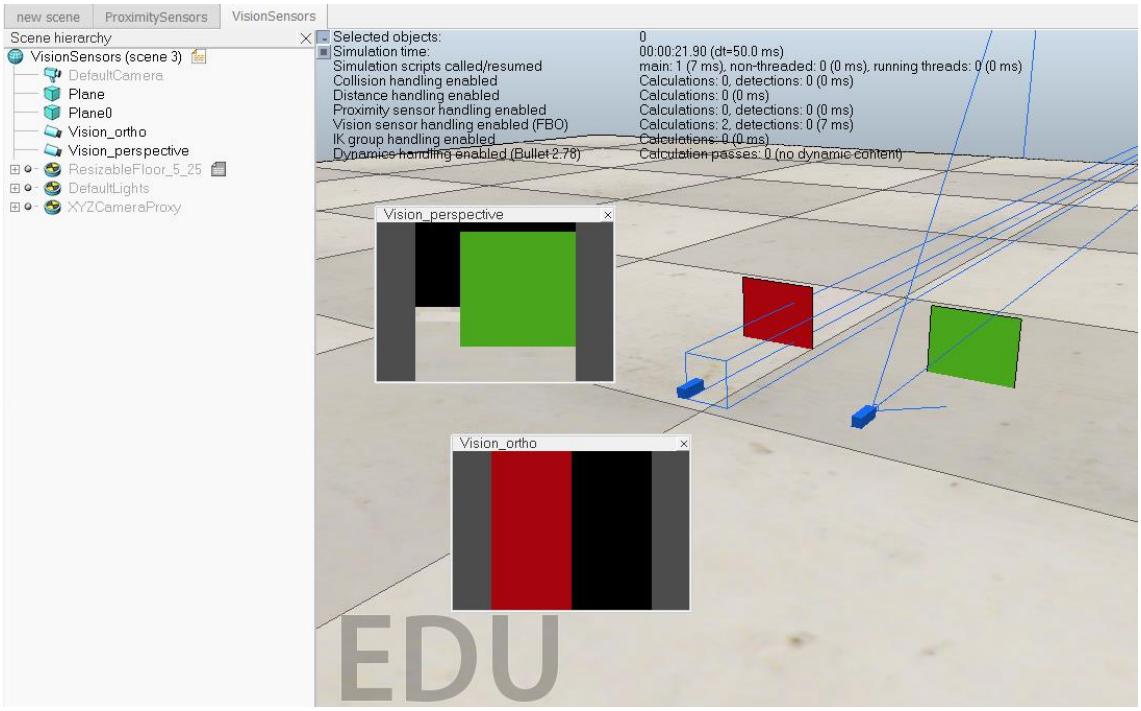
Sensores

Los sensores nos permiten percibir el entorno del robot. Existen diferentes tipos de sensores, y en CoppeliaSim se dividen en tres grandes grupos:

- Proximity sensor: estos sensores detectan los objetos más cercanos. Se diferencian en la forma del haz que utilizan. Éstos, pueden configurarse para que detecten cierto tipo de materiales, por ejemplo, pueden configurarse como inductivos y sólo detectarán elementos que sean Detectable con sensores Inductive. NOTA: si el suelo (*ResizableFloor*) tiene la propiedad de Detectable activada, puede que los sensores detecten primero el suelo por estar más próximo al sensor.

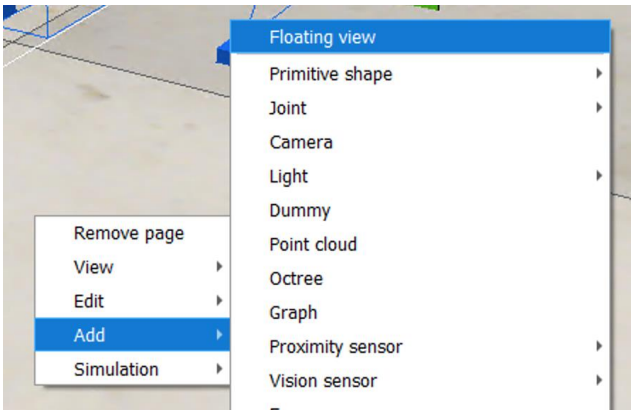


- Vision sensor: son sensores del tipo cámara, que puede ser 2D o 3D. Pueden tener dos formas principales, ortográfica y en perspectiva. Éstos pueden configurarse con distinta resolución, distancias mínimas y máximas de percepción, etc.

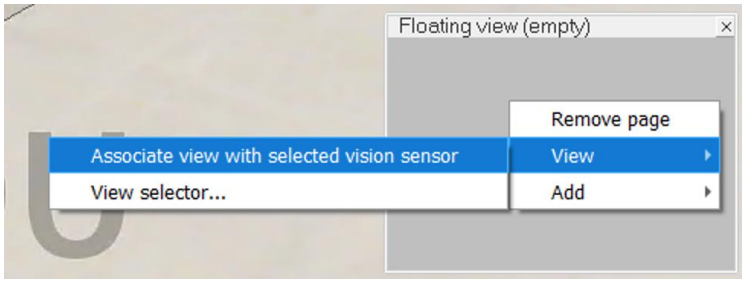


Para poder ver lo que los sensores de visión capturan, podemos crear ventanas flotantes en el CoppeliaSim. Para crear la ventana y asociarla al sensor, seguiremos los siguientes pasos:

- Botón derecho sobre la escena principal > Add > Floating view



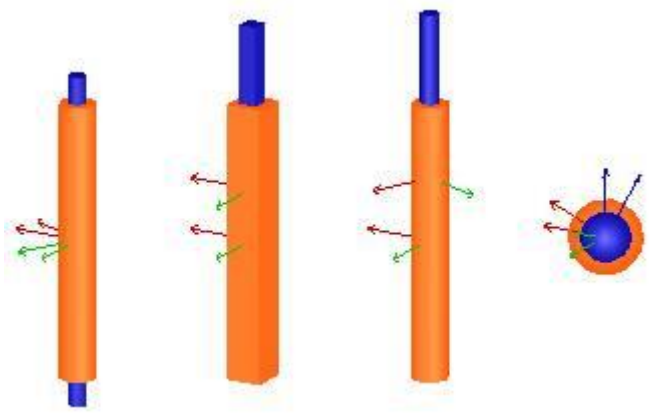
- Seleccionar el sensor en la ventana de jerarquía de objetos y hacer botón derecho sobre la ventana flotante > View > Associate view…



- Force sensor: estos sensores reaccionan ante una fuerza, como si fueran botones. Se pueden configurar con la sensibilidad sobre la que reaccionan (*Force/Torque Thresholds*).

Articulaciones (joints)

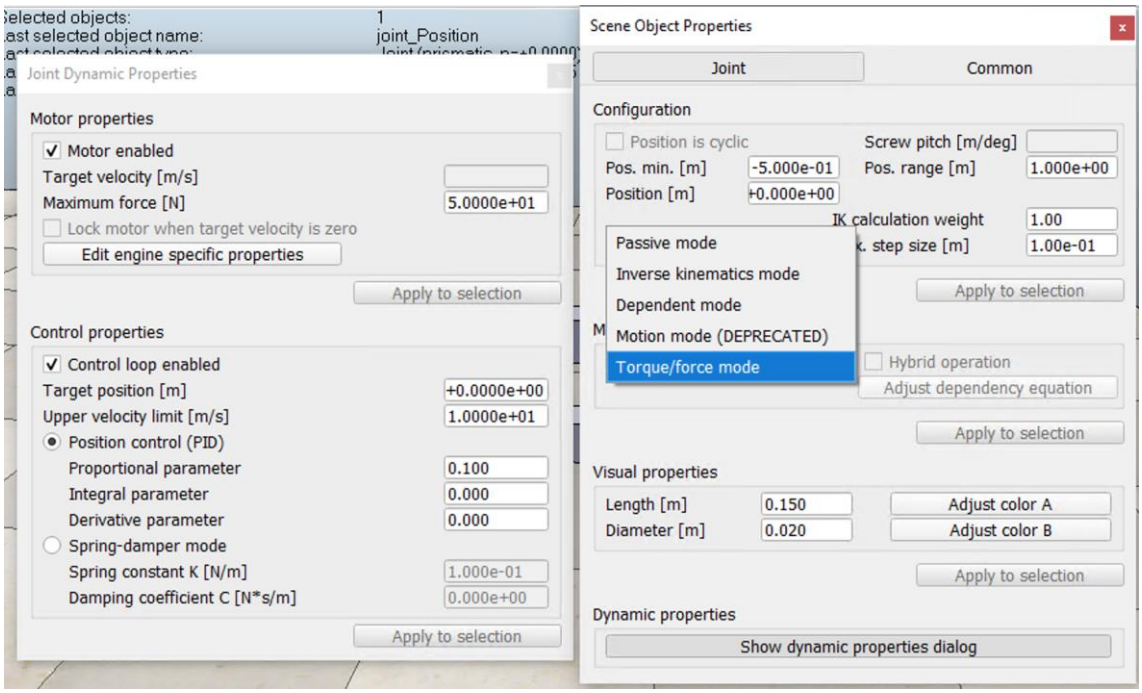
Las articulaciones permiten definir la interacción entre objetos, de forma similar a una bisagra, a un pistón o a una bola pivotante. CoppeliaSim tiene 4 tipos fundamentales de *joints*:



Joints: Revolute joint, prismatic joint, screw and spherical joint

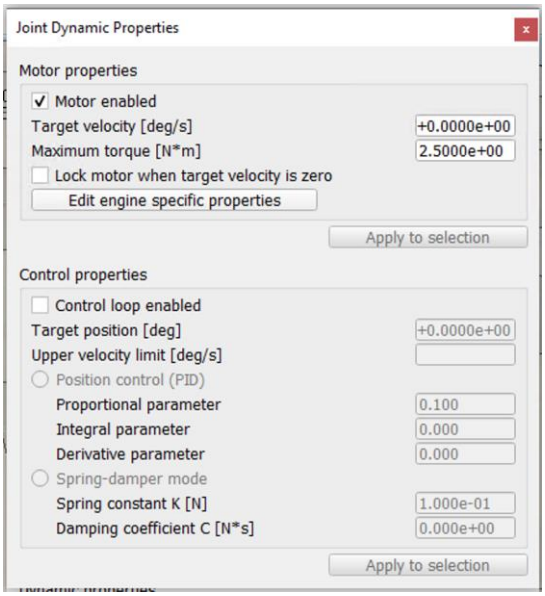
- **Revolute** joint: este tipo de articulación permite girar con un único grado de libertad.
- **Prismatic** joint: este tipo de articulación permite desplazar en una única dirección, como un pistón.
- **Screw** joint: este tipo de articulación es una fusión del revolute y el prismatic joint.

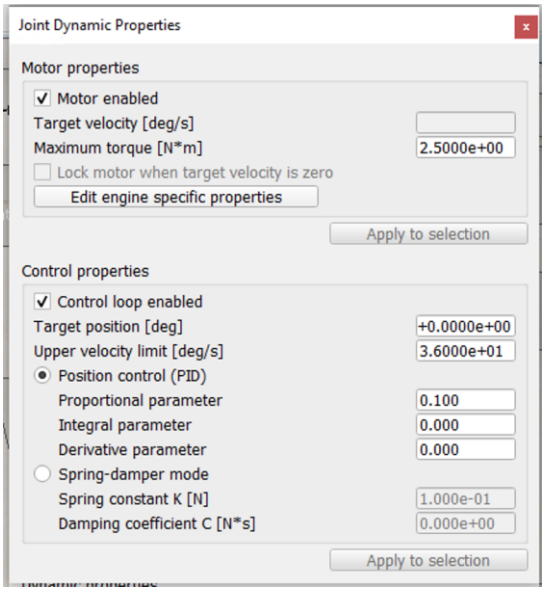
- **Spherical** joint: este tipo de articulación permite 3 grados de libertad, actuando como una bola pivotante.



Dentro de las propiedades, encontramos el modo en el que se comporta la articulación, donde *Passive mode* supone que no se puede controlar la articulación y es como una unión fija. El otro modo que más se va a utilizar es el *Torque/force mode*, el cual permite definir la fuerza que va a ejercer. En sus propiedades dinámicas, según sea la articulación *Revolute* o *Prismatic*, se configuran sus características en grados o en metros.

En el caso de los *joint Revolute* pueden controlarse por posición o por velocidad. El control por posición permite definir un ángulo final al que se emplaza la articulación, como un servomotor. Por lo contrario, en el control de velocidad al joint se le da mayor o menor velocidad para controlarlo.



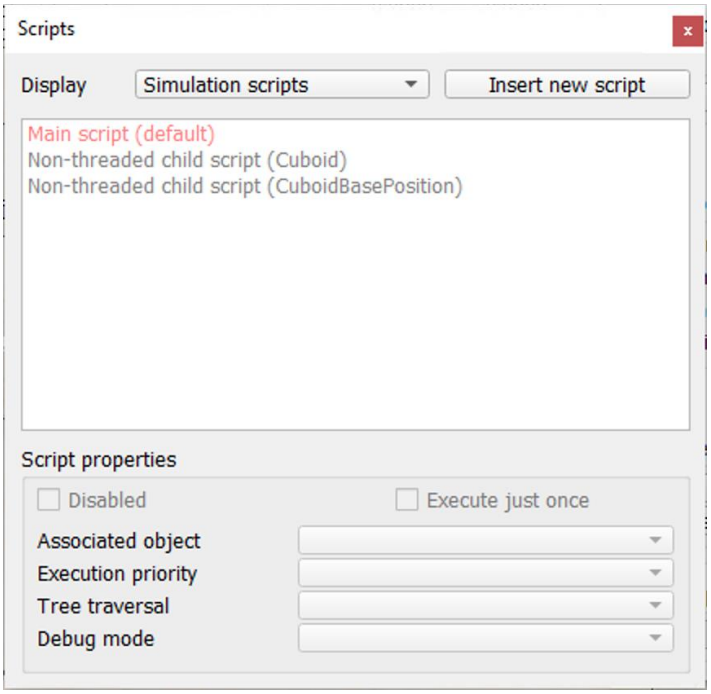


Control por velocidad Control por posición

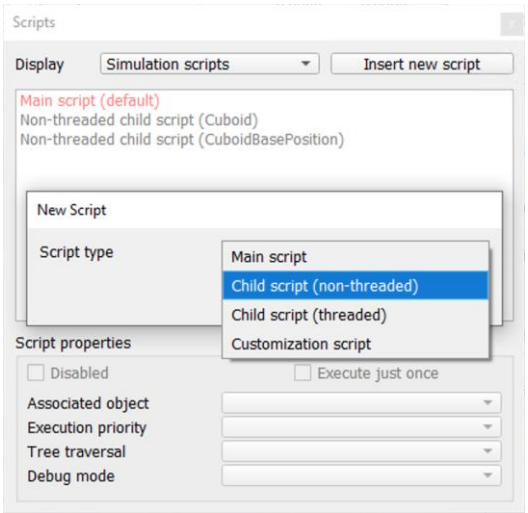
Scripts

Para poder programar ciertos comportamientos, es necesario crear un script y programarlo. En este caso lo vamos a hacer utilizando el lenguaje de programación LUA, propio de CoppeliaSim.

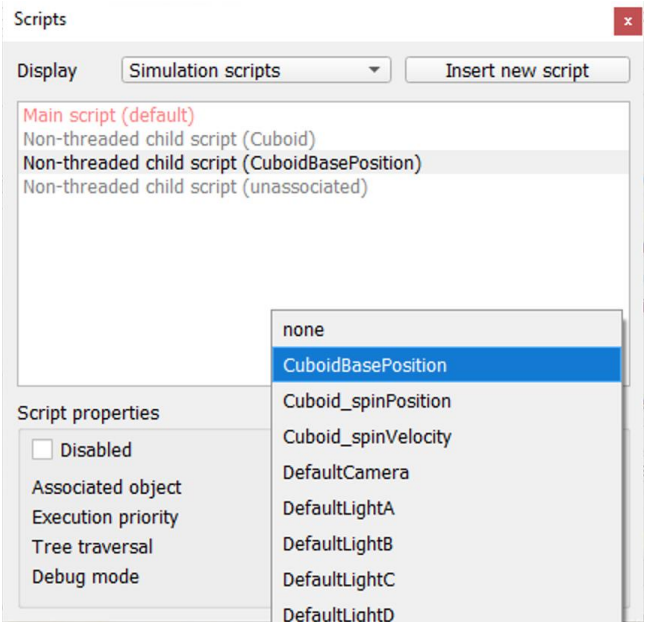
Para crear un script, en la barra de herramientas seleccionamos Tool > Scripts. Se abrirá la siguiente ventana:



Pinchamos en *Insert new script* y definimos el tipo, siendo en nuestro caso *Non-threaded child*



Una vez seleccionado, lo asociamos al objeto que deseamos. Se asigna el script al objeto padre del conjunto de elementos que vayan a verse afectados por ese script. Si tenemos dos objetos unidos por un *joint* el script se asociaría al objeto base al que se le ha unido la articulación y el resto de objetos.



Una vez creado, veremos en la ventana de jerarquía de elementos, un nuevo símbolo al lado del objeto al que se le ha asignado el script.

Por defecto se crean cuatro funciones:

- init: que se llama cuando se inicia la simulación
- actuation: función que se llama iterativamente durante la simulación
- sensing: función que se llama cuando hay una sensorización
- cleanup: función de finalización para borrar elementos tras la simulación

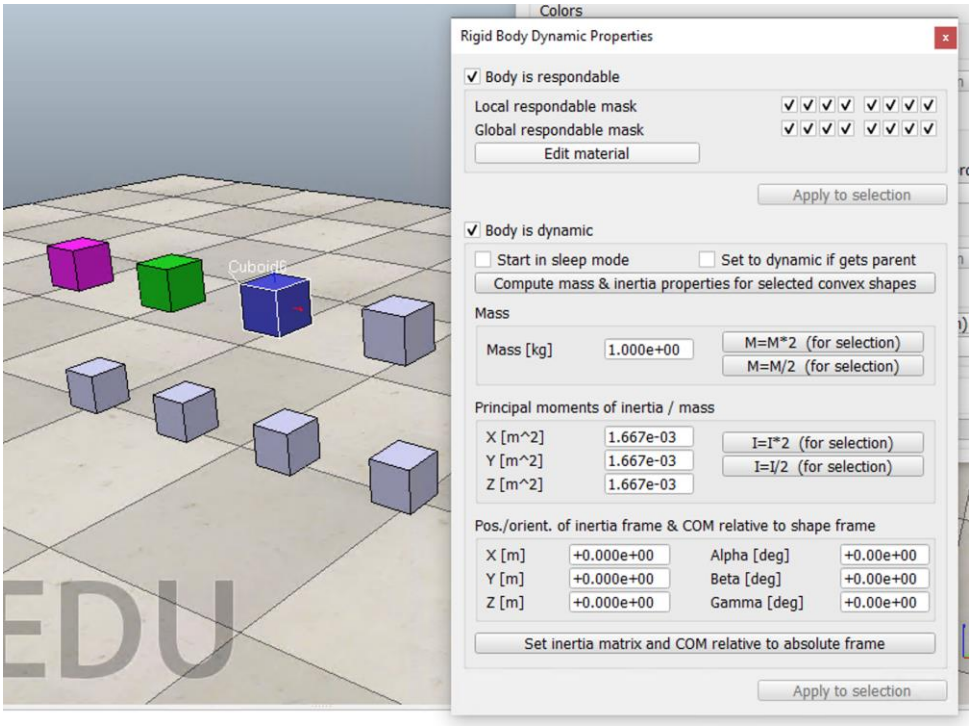
Ejercicios

Para comprender mejor las diferentes propiedades de los objetos y articulaciones, se van a realizar unos ejercicios.

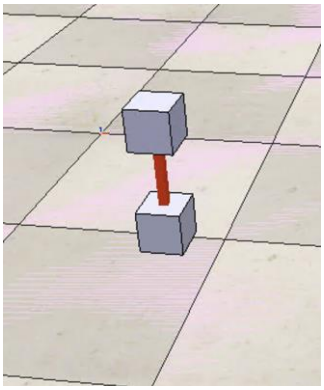
Ejercicio 1. Propiedades de los elementos. Cread un escenario con diferentes elementos básicos del tipo cuboide de tal forma que estén alineados verticalmente en parejas. Aplicad

diferentes propiedades dinámicas a los elementos superiores e inferiores y estudiad el comportamiento del sistema en cada situación. Recordad que el suelo también puede ser *Respondable*.

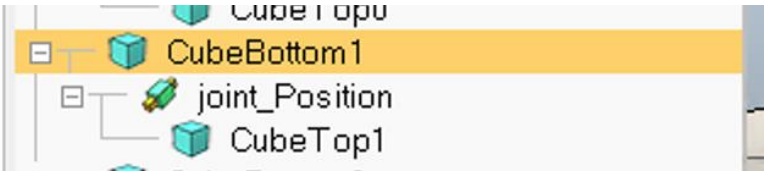
- Elementos superior/inferior
    - *Respondable/non-responsable*
    - *Dynamic/non-dynamic*



Ejercicio 2. Articulaciones prismáticas. Cread un escenario con figuras geométricas básicas de tipo cubo tal hayan parejas de cubos unidas por articulaciones *Prismatic* como se muestra en la siguiente figura:



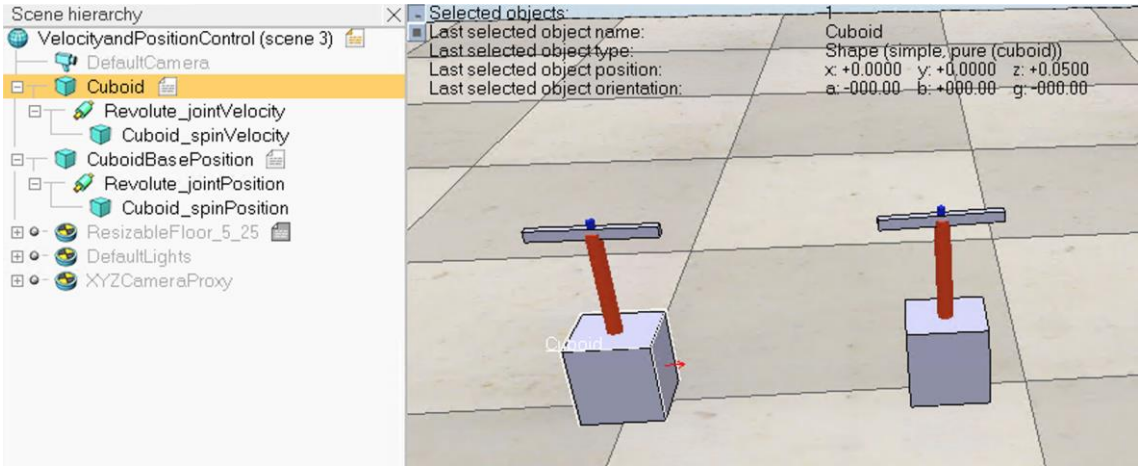
Es importante destacar que la jerarquía de elementos de ser de nivel superior a inferior: CuboInferior > Articulación > CuboSuperior:



Estudiad como se comportaría el sistema con las diferentes combinaciones de configuración. Para ello cread tantos ejemplos como casos posibles utilizando las siguientes propiedades de objetos y articulaciones:

- Objetos:
  - dinámicos/estáticos
  - cambiando sus propiedades de masa
- Articulaciones *Prismáticas*
  - Diferentes pares de fuerza
- Pasivas
  - Hybrid/No hybrid
- Par/fuerza
  - Motor enabled
  - Control loop enabled
    - PID
    - Spring-damper

Ejercicio 3. Articulaciones por revolución. Cread un escenario como el de la siguiente figura donde el *joint* es del tipo *Revolute*.



Para poder ejecutar el comportamiento de las articulaciones, utilizad el siguiente código modificando los valores para probar cómo reacciona el sistema en función de ellos.

Para el controlar el *joint por velocidad*, utilizad el siguiente código como ejemplo:

```
function sysCall_init()
    -- do some initialization here
    joint =  sim.getObjectHandle('Revolute_jointVelocity')
    sim.setJointTargetVelocity(joint, 1)
```

```
end
```

Para el *joint por posición*, utilizad el siguiente código como ejemplo:

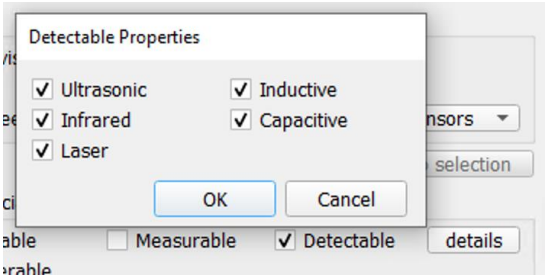
```
function sysCall_init()
    -- do some initialization here
    joint =  sim.getObjectHandle('Revolute_jointPosition')
    last_time = sim.getSimulationTime()
    ccw = false -- direcci?n de giro
end

function sysCall_actuation()
    -- put your actuation code here
    if ((sim.getSimulationTime()-last_time) >5 ) then
        time_elapsed = true
        last_time = sim.getSimulationTime()
        print('Time '..last_time)
    else
        time_elapsed = false
    end

    if (time_elapsed and ccw==true) then
        ccw = false
        sim.setJointTargetPosition(joint, 90*math.pi/180)
    elseif (time_elapsed and ccw==false) then
        ccw = true
        sim.setJointTargetPosition(joint, -90*math.pi/180)
    end
end
```

Ejercicio 4. Sensores

Cread un escenario con múltiples sensores de proximidad y sensores de visión. Comparad qué ocurre con distintos parámetros para cada uno de los sensores.

1. Proximidad:
    a. Tipos: ray, cone, pyramid

b. Sensor subtype: Ultrasonic, Infrared, Laser, Inductive, Capacitive
c. Distintos tipos *Detectable* de objetos

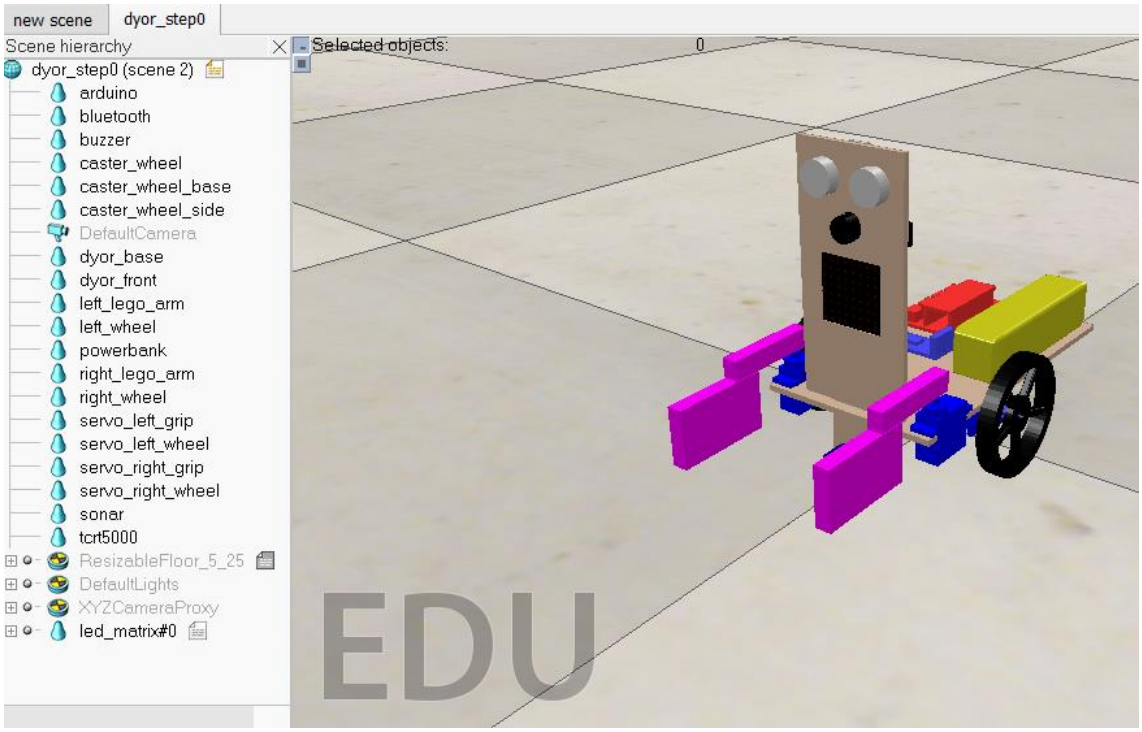


2. Visión
    a. Tipo: *orthographic* y *perspective*
    b. Cambiar el color a los objetos frente a los sensores

Parte 2. Creación de un robot sencillo

El objetivo de esta parte es la creación de un robot desde cero cuya programación se implementará en la siguiente práctica. Para ello, se deberán realizar los siguientes pasos:

- Importar los objetos (mallas) de tipo stl del robot que podréis encontrar en vuestro UACloud.
- Importar los objetos tipo modelo.
- Crear los objetos dinámicos puros relativos a las mallas: cuerpo del robot, ruedas, pinzas, rueda trasera.
- Añadir las articulaciones necesarias para el control de las palas y el giro de las ruedas.
- Establecer las relaciones de jerarquía adecuadas.
- Configurar las articulaciones que permitan controlar el giro de las ruedas por velocidad y el movimiento de las palas por posición.
- Añadir sensores de visión y de proximidad al robot.



Modo y fecha de entrega y evaluación de la práctica

Cada una de las escenas (*archivos.ttt*) de los ejercicios de la parte 1 y del robot se entregarán en un único archivo comprimido nombrado como *Apellido1Apellido2.zip.*

Esta primera práctica se desarrollará **individualmente**. La fecha de entrega será la 5ª sesión de clase de prácticas (06/03/2022) y contará un 33% de la nota de prácticas.