Práctica 1 Tecnología y Arquitectura Robótica

Daniel Asensi Roch dni: **48776120C**

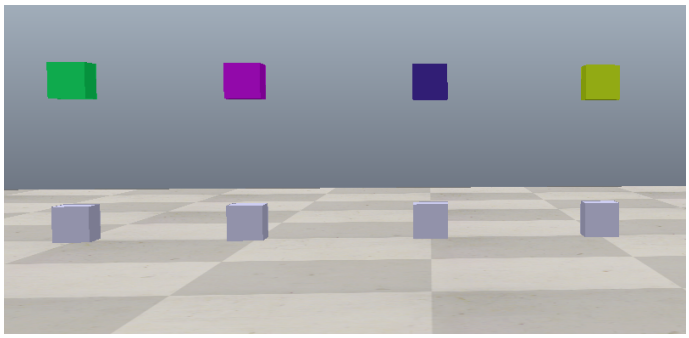
February 22, 2022

Contents

1 Parte 1: Ejercicios

1.1 Ejercicio 1

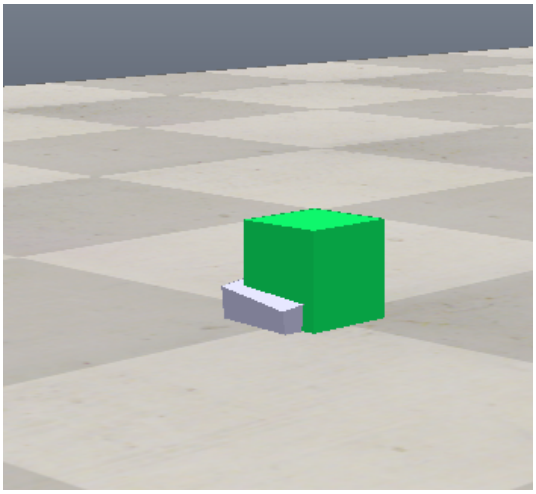
En este ejercicio se nos pedía a los alumnos que generáramos un escenario con diferentes elementos básicos de tipo cuboide con diferentes propiedades dinámicas para los elementos superiores e inferiores, además podremos modificar las propiedades pertinentes del suelo y estudiar su comportamiento:



Estado inicial

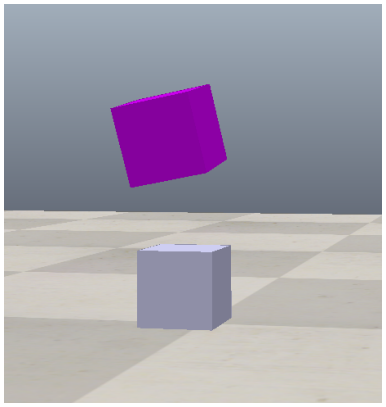
Ahora relataré los resultados obtenidos durante las pruebas realizadas con las diferentes bariaciones que he realizado en las propiedades dinámicas de los cuboides superiores y inferiores La propiedad modificada en

el cubo verde ha sido su masa *Mass[kg]* (hasta un punto absurdo, del orden de más de 8 toneladas) esto ha producido como podemos observar en la siguiente imagen que el cuboide sobre que el que cae sea aplastado, de tal manera que se produce un bug con las colisiones del suelo.

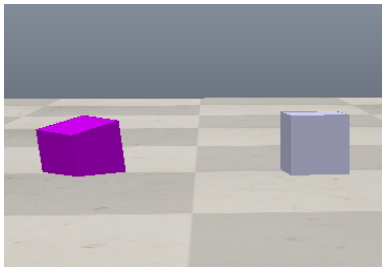


Cuboide verde siendo aplastado

Las siguientes propiedades probadas han sido las de inercia y masa *Inertia / mass* y fricción alterada en el cuboide inferior, esto nos produce resultados muy dispares, de tal manera que el cuboide morado comienza a rebotar a lo largo de la escena y pegado al suelo, a su vez el cubo gris sobre el que cae se ve desplazado hacia un lado y moviéndose de manera casi infinita hacia el lado que es empujado divido a que su fricción con el suelo es nula.

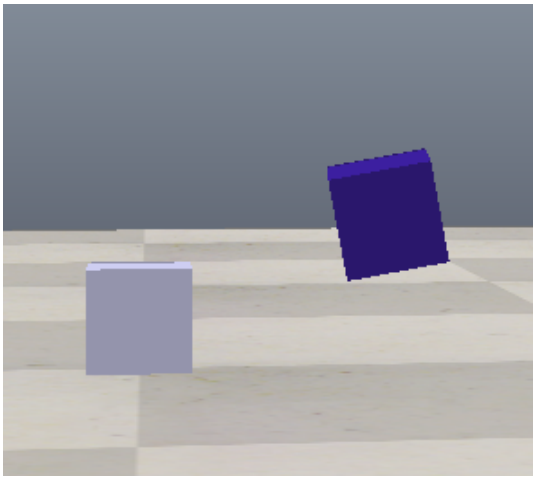


Cuboide morado rebotando contra cuboide gris



Cuboide gris desplazandose por fricción nula

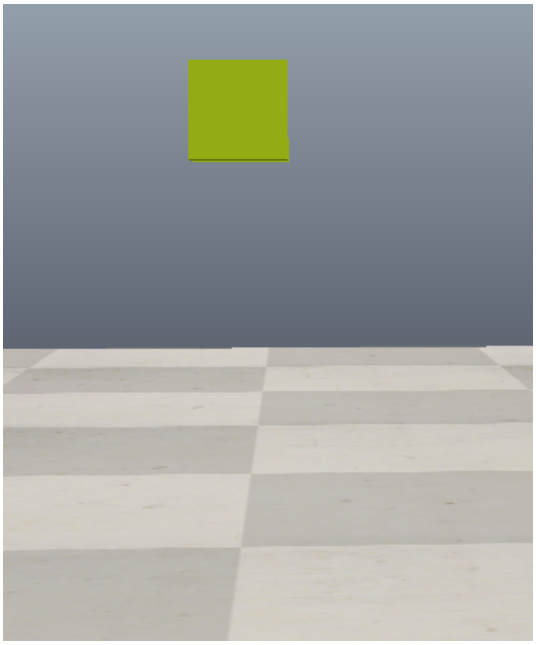
La siguiente propiedad modificada ha sido la de la inercia y masa *Inertia / mass* pero esta vez sus valores *Alpha, Beta y Gamma* esto produce que el cubo azul también rebote de por la escena, pero no de manera errática como sucedía con el morado, este se desliza por la escena con una posición fija y sin producir rebotes.



Cuboide azul rebotando y deslizandose por la escena.

La siguiente propiedad ha sido la de *Body is respondable* y *Body is dynamic*, en los cubos amarillo y gris respectivamente, desactivándolos en cada uno de ellos, al desactivar la primera propiedad podemos observar

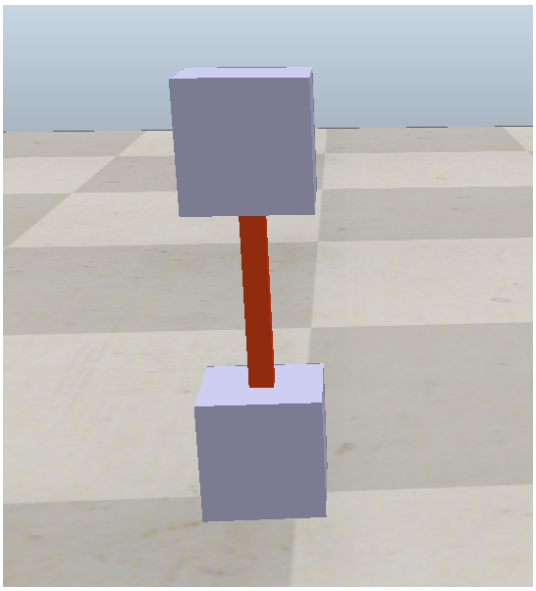
como el cubo amarillo se queda inmóvil en el aire, y podemos observar como el cuboide gris ha desaparecido ya que al si tener sus propiedad Body is respondable activa pero no is dynamic ha caído de la zona visible ya que no posee colisiones con los demás objetos.



Cuboide amarillo y azul.

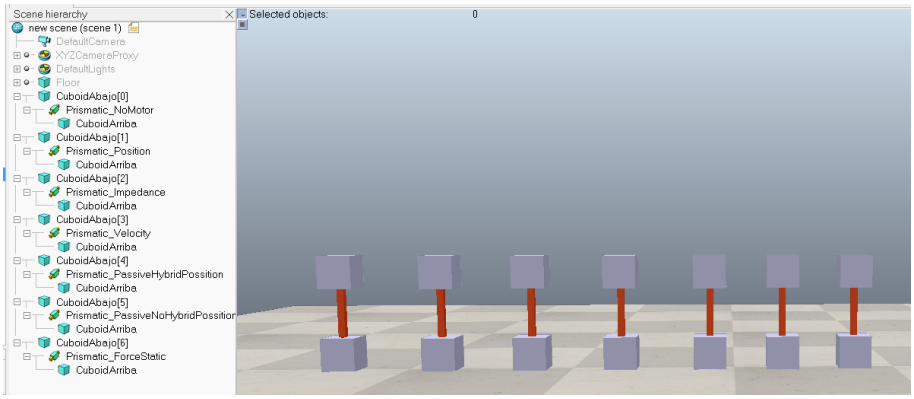
1.2 Ejercicio 2

En este ejercicio se nos pide implementar una pareja de cubos unida por una articulación y probar diferentes casos, de tal forma que el inicio del ejercicio queda de la siguiente manera.



Inicio de Ejercicio 2.

De esta manera he dispuesto en mi escena todos los casos posibles a probar con las diferentes configuraciones aplicadas tanto a la articulación como a los objetos pertinentes que la articulación ha de sostener o mover, quedando en nuestra escena de la siguiente manera.



Inicio de Ejercicio 2.

La primera configuración aplicada a la articulación, comenzando de izquierda a derecha, ha sido la de modo fuerza con el motor desactivado y sin control de propiedades. Por lo que solo dependerá de la posición del cuboide superior.

La segunda configuración aplicada ha sido con el modo fuerza, pero en este caso con el motor habilitado, y un control en bucle cerrado, más en concreto un PID.

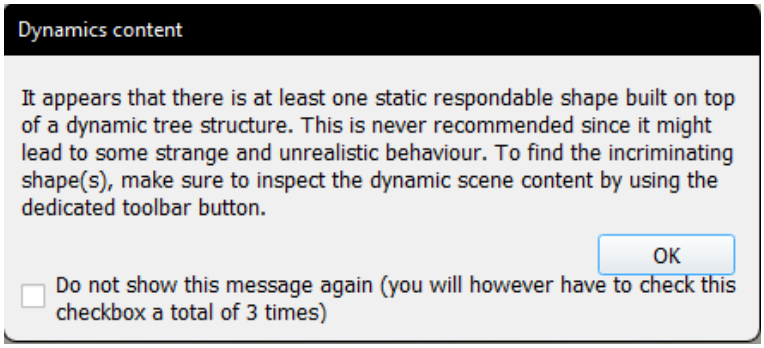
La tercera configuración aplicada al igual que la segunda ha sido con el modo fuerza, pero en este caso con el motor habilitado, y un control en bucle cerrado, esta vez activando el modo impedancia es decir el de muelle amortiguador.

La cuarta configuración aplicada ha sido con el modo fuerza activado, con un motor y sin bucle cerrado, esta configuración sería ideal para un control de velocidad

La quinta articulación configurada ha sido mediante el modo pasivo, pero activando el modo híbrido lo que nos permitirá internamente mantener un control de la articulación, mediante un control PID.

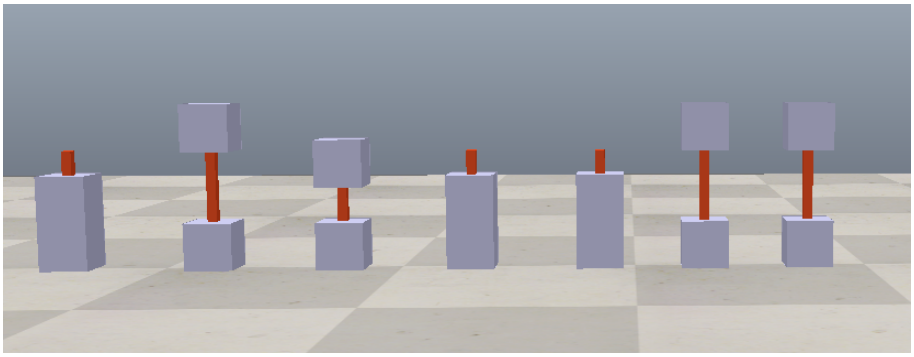
La sexta configuración aplicada es la misma que la anterior pero sin el modo híbrido, esto sumado a que el objeto de abajo no es estático sino dinámico nos generará problemas.

La séptima configuración realizada utilizamos la configuración anterior, pero la articulación intentará arrastrar un objeto estático y generará el siguiente error:



Error Generado

Por último muestro el resultado de todas las configuraciones anteriormente definidas:

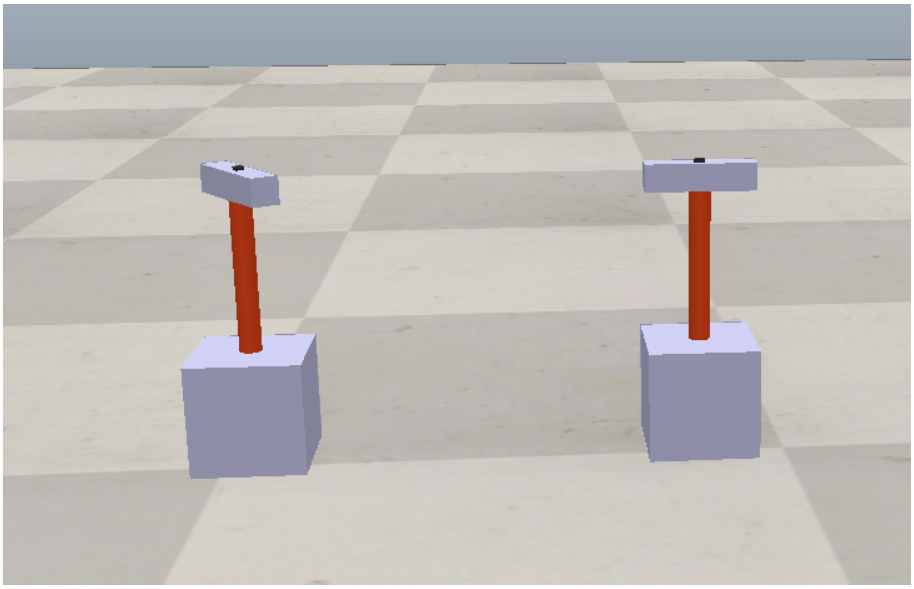


Resultado

Como conclusión de este ejercicio sacamos que debemos tener cuidado con las propiedades que utilizamos para nuestros objetos y articulaciones, ya que pueden derivar en comportamientos no deseados.

1.3 Ejercicio 3

En este ejercicio hemos tenido nuestro primer acercamiento a los scripts hechos en LUA, para ello hemos utilizado el siguiente escenario:



Escenario

En el primer objeto inyectaremos el siguiente script:

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

Con este script controlaremos la velocidad de rotación del cuboide de arriba, mediante la articulación de rotación, dando como resultado que este giré en sentido horario, al ser esto un pdf no puedo mostrar el resultado de la rotación.

El siguiente script a probar hará que el cuboide superior cambie de lado cada segundos:

```
function sysCall_init()
    -- do some initialization here
    joint = sim.getObjectHandle('Revolute_jointPosition')
    last_time = sim.getSimulationTime()
    ccw = false -- direccin 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
```

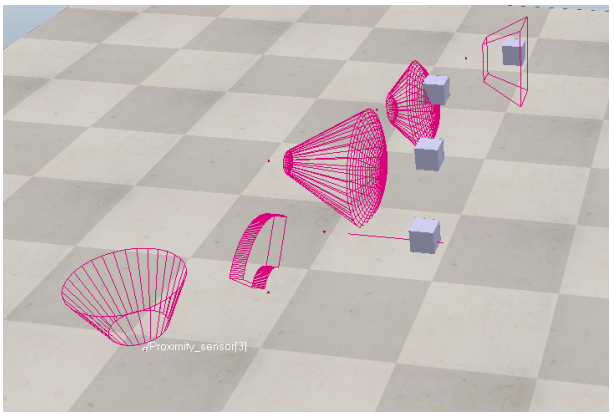
```
        ccw = false
        sim.setJointTargetPosition(joint, 90*math.pi/180)
    elseif (time_elapsed and ccw==false) then
        ccw = true
        sim.setJointTargetPosition(joint, -90*math.pi/180)
    end
end

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

function sysCall_cleanup()
    -- do some clean-up here
end
```

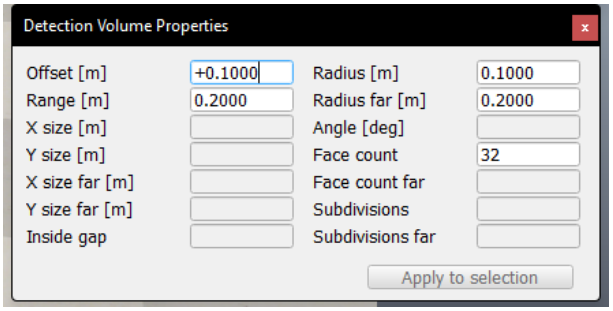
1.4 Ejercicio 4

En este ejercicio vislumbraremos las capacidades que tienen cada uno de los sensores que nos ofrece este simulador, realizando pruebas y explicando los resultados obtenidos. Desde el menú add podremos añadir diferentes formas en los sensores de proximidad que serán los primeros estudiados en este apartado, de manera que al añadirlos a la escena obtendremos lo siguiente:



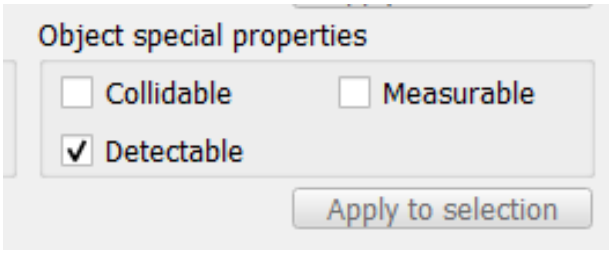
Tipos de sensores de proximidad

De todos estos sensores podremos modificar su volumen de detección para que esta sea más o menos precisa, y su campo de detección mayor o menor dependiendo de como ajustemos los volúmenes de detección



Ajuste volumen detección

Pero para que todos estos sensores puedan detectar objetos tendremos que configurar dichos objetos con la propiedad Detectable que vemos en el siguiente cuadro:



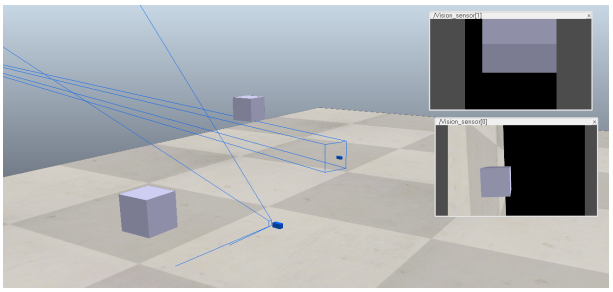
Propiedad Detectable

Al contrario que en otras versiones de Coppelia en esta no podremos configurar el tipo de detección para los objetos y serán detectables por todos los sensores.

Una vez un objeto es detectado durante la simulación se marca el haz que lo detecta de la siguiente manera:

Simulación de detección

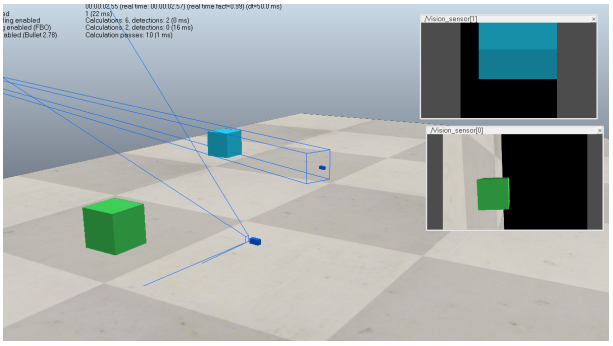
En cuanto a los sensores de visión haremos al igual que con los sensores de proximidad los añadiremos a la escena:



Sensores de visión con rederers

Una vez añadidos y configurados a nuestro gusto podremos ajustar la resolución de su visión por ejemplo el render de arriba a la derecha es de 32x32 lo que hace que sea una imagen pixelada del cubo, mientras que el de abajo del mismo es de 720x480, lo que nos ofrece una mayor calidad de imagen.

Ahora cambiaré los cuboides de delante de color, de estos es importante especificar que para que no sean invisibles a nuestros sensores de visión deberemos indicar que tienen la propiedad rendereable:

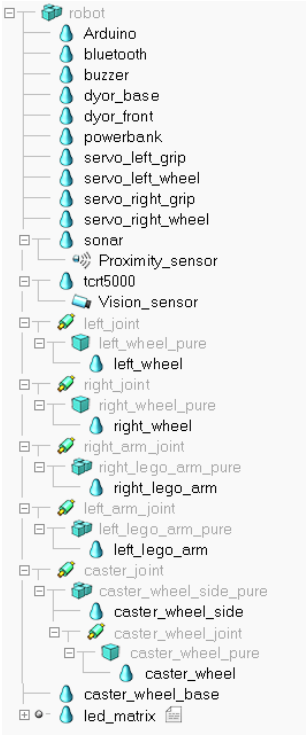


Cambiados de color los cubos

2 Parte 2: Creación de un robot sencillo

En esta parte de la práctica se nos presenta la problemática de realizar el esqueleto de un sencillo robot que más tarde programaremos para ser capaces de moverlo a voluntad.

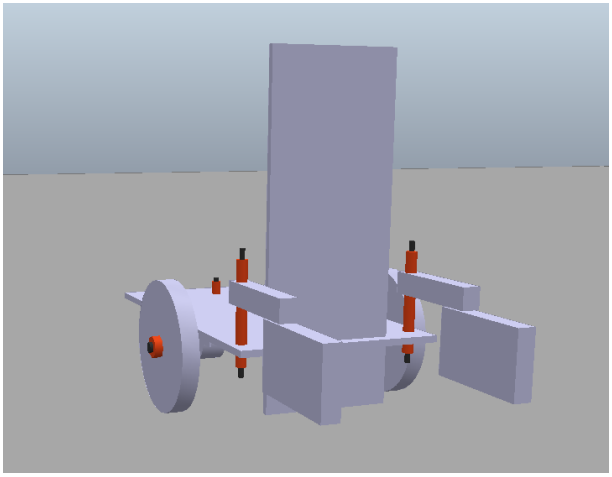
El primer paso que debemos realizar es importar las mallas que nos han proporcionado en Uacloud y para trabajar de manera más cómoda renombraremos lo archivos, para poder trabajar más adelante las jerarquías con mayor facilidad.



Nombres de los archivos

Una vez importadas las formas de cada una de ellas que vayan enganchadas a una articulación y deseemos obtener sus colisiones, obtendremos sus formas puras, es decir cuboides y cilindros.

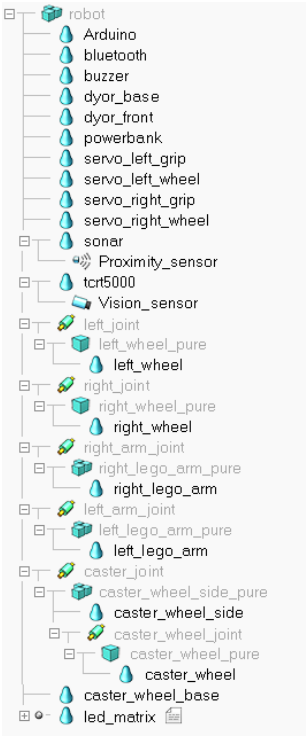
Una vez hecho esto las agruparemos formando las piezas completas de las cuales hemos sacado sus formas puras.



Formas Puras

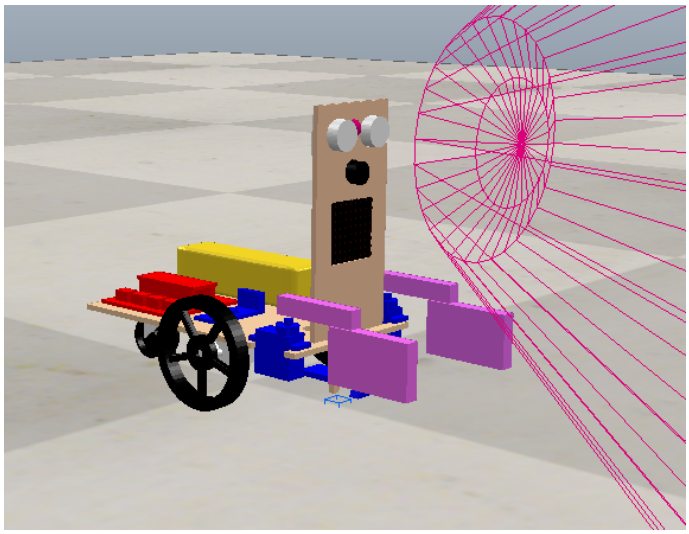
En las cuales como se puede visualizar en la fotografía hemos insertado las joint correspondientes a las articulaciones y motores. Los dos motores de las ruedas si que ejercerán fuerza, los motores de los brazos serán servos que se moverán dependiendo de la dirección del objeto, y la rueda de la parte de atrás simplemente girará de manera errática dependiendo del movimiento del robot.

Una vez hecho esto podremos proceder a realizar la jerarquía de los objetos, la cual quedará de la siguiente manera:



Nombres de los archivos

Quedando el robot completo, y añadiendo los sensores sin configurar y las piezas pintadas de la siguiente manera:



Robot Completo