

Modelado y Simulación de Robots

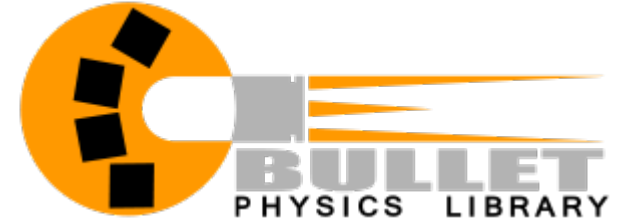
Controlando un Robot en PyBullet

Grado en Ingeniería de Robótica Software

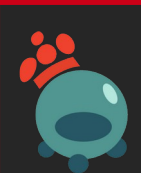
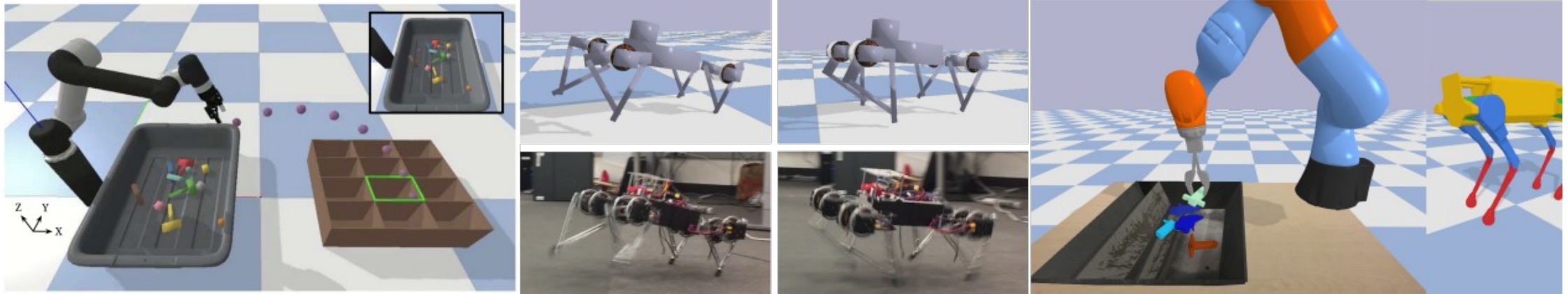
Teoría de la Señal y las Comunicaciones y
Sistemas Telemáticos y Computación

Roberto Calvo Palomino
roberto.calvo@urjc.es

Pybullet



- Motor de simulación de físicas
- Última release, Abril 2022
- <https://pybullet.org/>
- <https://github.com/bulletphysics/bullet3/>



Instalación

- Accede a los ordenadores del laboratorio (físicamente o mediante VNC) <https://labs.etsit.urjc.es/vnc/>
- Receta válida para laboratorio y vuestros ordenadores

```
$ pip3 install pybullet gym==0.19.0
```

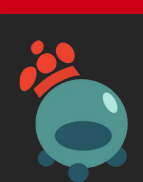
```
$ pip3 install numpy --upgrade
```

Si obtienes error al ejecutar utiliza:

```
$ export LD_PRELOAD=/usr/lib/x86_64-linux-gnu/libstdc++.so.6
```

- Instalará pybullet en vuestro \$HOME, sin necesidad de ser root ni tener permisos de administrador.
- Comprueba que se ha instalado correctamente, visualizando el directorio de ejemplos

```
$ ls $HOME/.local/lib/python3.10/site-packages/pybullet_envs/examples/
```

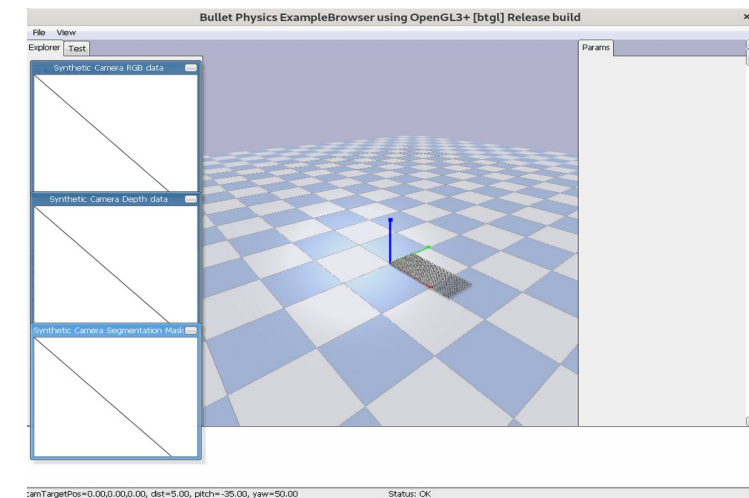


Probando la instalación

- Ejecuta el siguiente comando para iniciar el ejemplo de fichas de domino en el motor de físicas

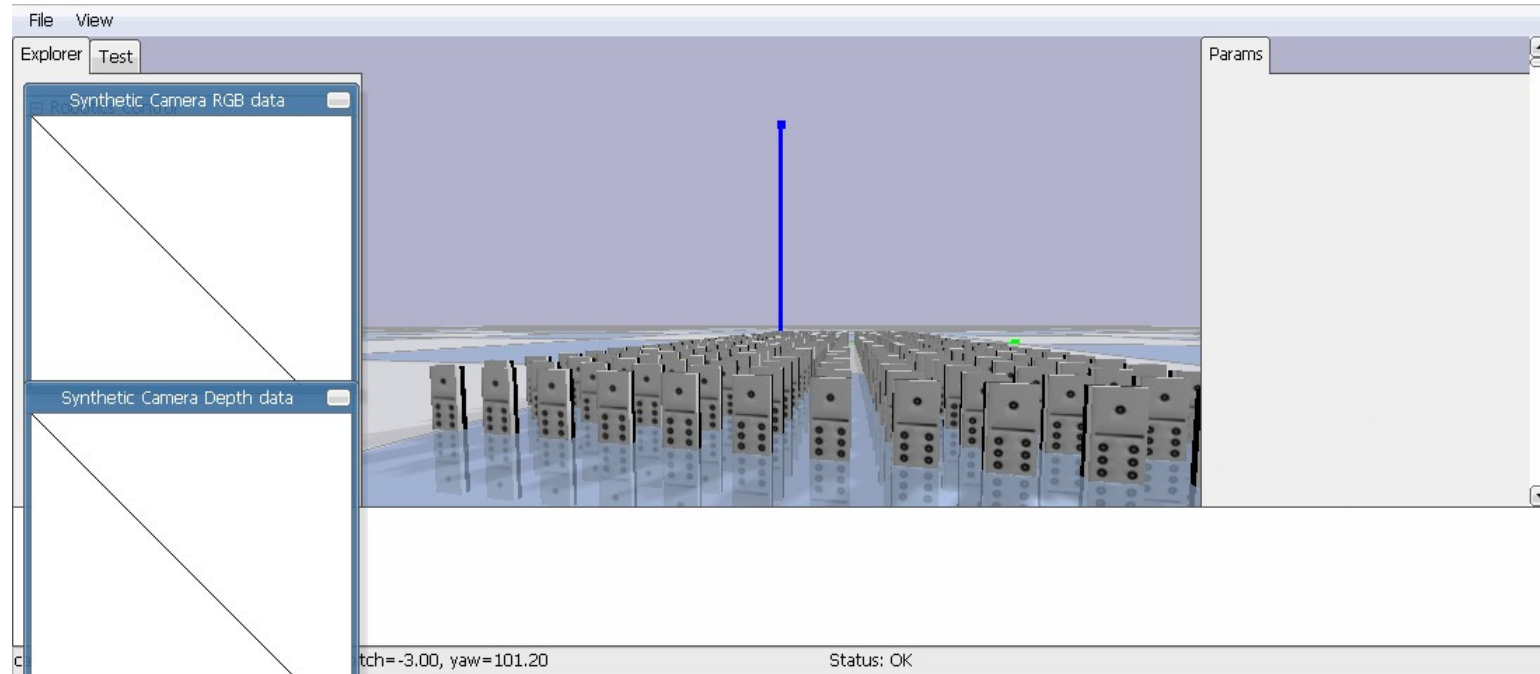
```
$ python3 -m pybullet_envs.examples.dominoes
```

- Deberías ver la interfaz



Probando la instalación

- Alt + Botón Izqdo: Cambiar Perspectiva
- Rueda ratón: Cambiar Zoom.
- Prueba a generar movimientos en las fichas de domino.



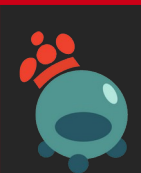
Documentación

- PyBullet Quickstart Guide (Importante)
 - <https://docs.google.com/document/d/10sXEhzFRSnvFcl3XxNGhnD4N2SedqwdAvK3dsihxVUA/edit#>

- Carpeta de ejemplos de pybullet

```
$HOME/.local/lib/python3.10/site-packages/pybullet_envs/examples/
```

(la versión de python podría cambiar dependiendo de la instalación)



Hola Mundo en Pybullet

```
import pybullet as p
import pybullet_data
import time
```

Imports de las librerías

```
physicsClient = p.connect(p.GUI)
p.setAdditionalSearchPath(pybullet_data.getDataPath())
p.setGravity(0,0,-9.8)
```

Conectamos motor con GUI

Establecemos gravedad (X,Y,Z)

```
planeId = p.loadURDF("plane.urdf")
```

Cargamos un modelo (plano)

```
euler_angles = [0,0,0]
startOrientation = p.getQuaternionFromEuler(euler_angles)
startPosition = [0,0,1]
```

```
robotId = p.loadURDF("r2d2.urdf",startPosition, startOrientation)
```

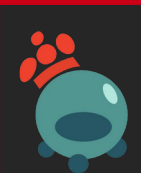
Cargamos un nuevo objeto, con una posición (x,y,z) y una orientación dada en cuaternion (C,X,Y,Z)

```
for i in range(10000):
    p.stepSimulation()
    time.sleep(1./240.)
```

Bucle principal que ejecuta los pasos de la simulación.

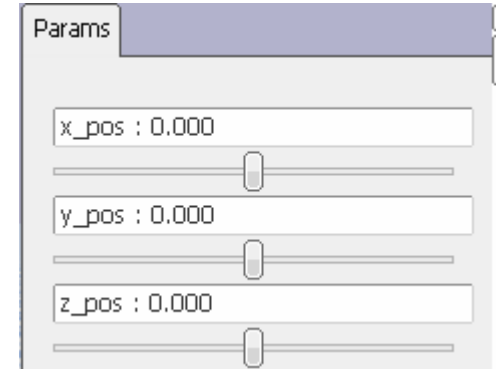
Por defecto utilizaremos siempre time step de 1/240 segundos.

```
p.disconnect()
```



Parámetros de depuración

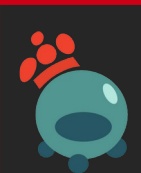
- Añadir parámetros en la pestaña “params”. Útiles para poder cambiar dinámicas en tiempo real.



```
x_pos_id = p.addUserDebugParameter("x_pos", MIN_VALUE, MAX_VALUE, INIT_VALUE)
```

- Desde el bucle de control, puedes obtener los cambios del parámetro de depuración de la siguiente manera

```
x_euler = p.readUserDebugParameter(x_euler_id)
```

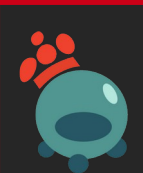


Simulación

- **stepSimulation**

- La simulación avanza solo un “step” (paso) de acuerdo con los pasos establecidos en setTimeStep
- Util cuando se quiere controlar la simulación donde se necesita procesar cada paso antes de avanzar al siguiente

```
1  import pybullet as p
2  import time
3
4  p.connect(p.GUI)
5  p.setGravity(0, 0, -9.8)
6  p.setTimeStep(1./240.)
7
8  for _ in range(1000):
9      p.stepSimulation()
10     time.sleep(1./240.)
```

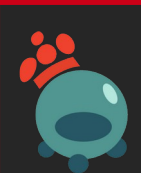


Simulación

- **SetRealTimeSimulation**

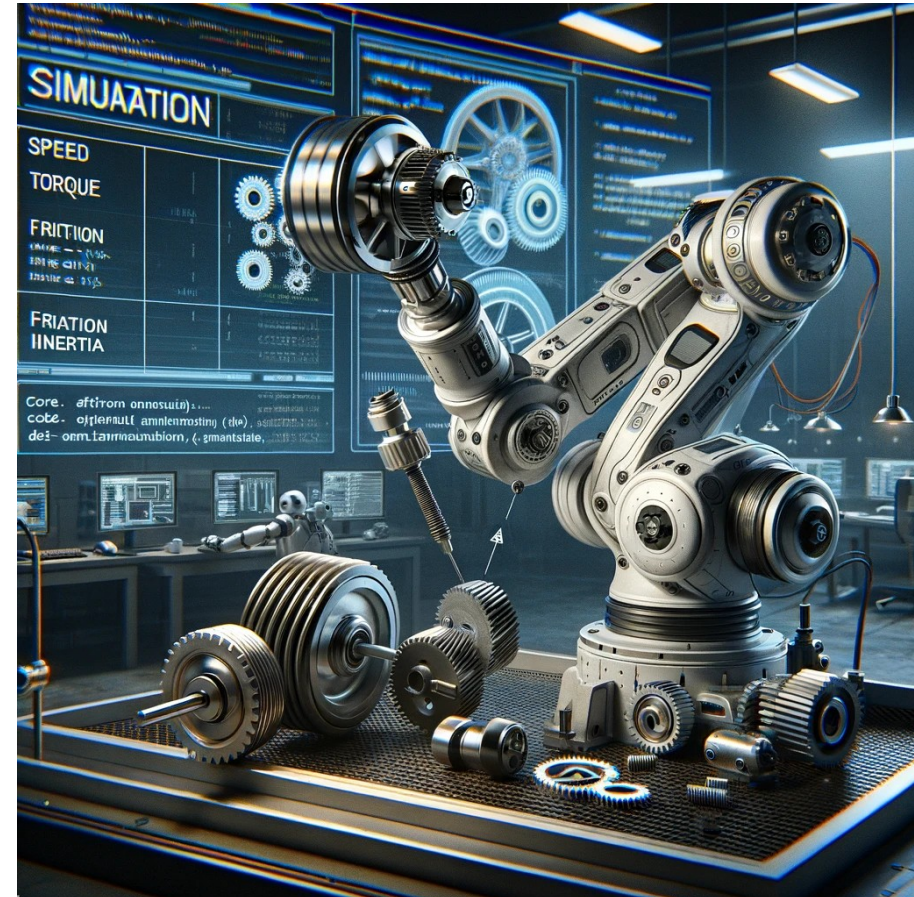
- El motor de físicas no realiza pausas en la simulación
- Ejecuta en tiempo real acorde al RTC del sistema.
- Depende del rendimiento del sistema.

```
1  import pybullet as p
2  import time
3
4  p.connect(p.GUI)
5  p.setGravity(0, 0, -9.8)
6  p.setTimeStep(1./240.)
7  p.setRealTimeSimulation(True)
8
9  for _ in range(1000):
10 |     # your code here
```



Controlando un Robot en Pybullet

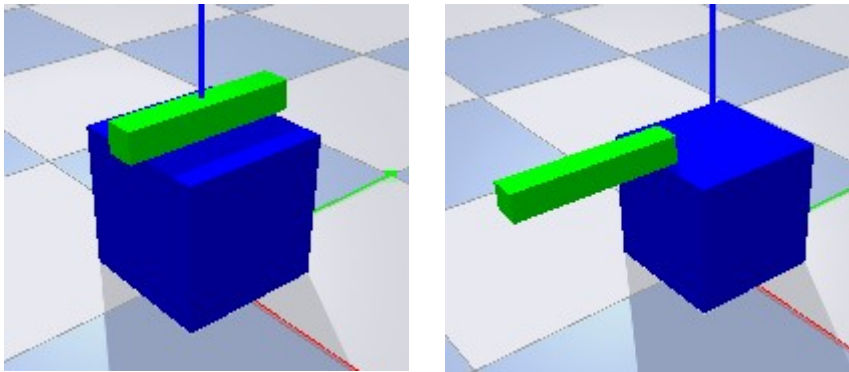
- Velocidad
- Momento de Fuerza o Torque
- Fricción
- Inercia



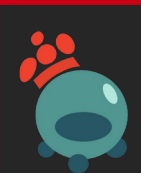
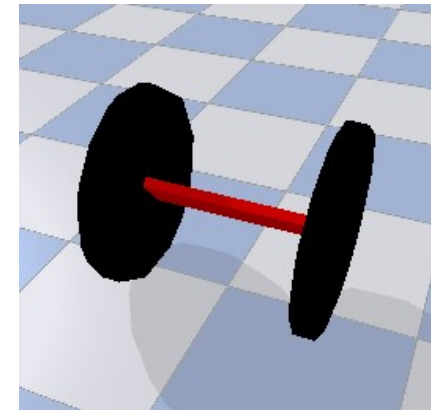
Introducción

- El control y movimiento del robot se realizará a través de los joints y fuerzas externas.
- Cada joint de tipo 'revolute' y 'prismatic' está implementado con motor en pyBullet.
- Por tanto, podemos comandar velocidades y fuerzas a ese motor.

Prismatic



Revolute



URDF - pyBullet

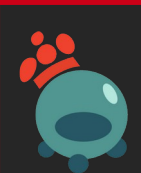
- Una vez cargado un modelo URDF en bullet, podemos saber cuantos *joints* tiene
- Cada *joint* nos permite mover partes del robot

```
robotId = p.loadURDF("myrobot.urdf",[0,0,0])

numJoints = p.getNumJoints(robotId)
print("NumJoints: " + str(numJoints))

for j in range (numJoints):
    print("%d - %s" % (p.getJointInfo(robotId,j)[0], p.getJointInfo(robotId,j)[1].decode("utf-8")))
```

```
NumJoints: 1
0 - base_to_body
```



URDF - pyBullet

- La función *setJointMotorControl2* nos permite configurar diferentes velocidades y fuerzas a las articulaciones.

```
frictionId = p.addUserDebugParameter("jointFriction", 0, 10, 5)
torqueId = p.addUserDebugParameter("joint torque", -10, 10, 5)

while (1):

    frictionForce = p.readUserDebugParameter(frictionId)
    jointTorque = p.readUserDebugParameter(torqueId)

    p.setJointMotorControl2(robotId, 0, p.TORQUE_CONTROL, force=jointTorque)
    p.setJointMotorControl2(robotId, 0, p.VELOCITY_CONTROL, targetVelocity=0, force=frictionForce)

    p.stepSimulation()
    time.sleep(1./240.)
```

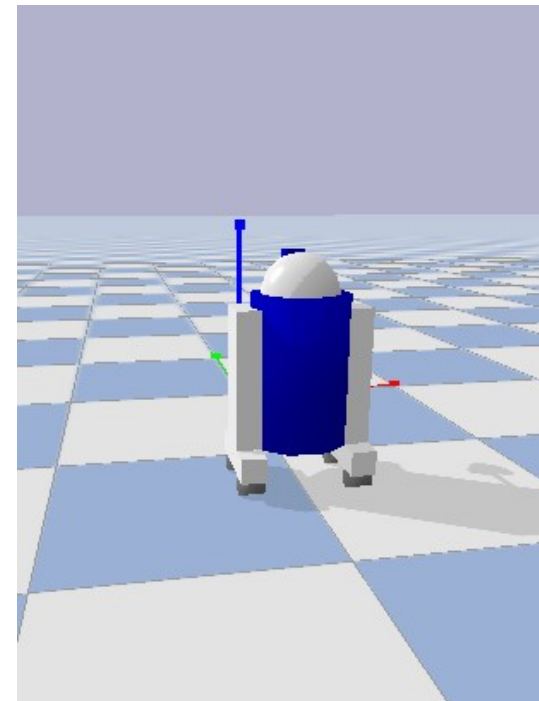


URDF - R2D2

- Usa `setJointMotorControl2` o `setJointMotorControlArray` para comandar velocidades y fuerzas a cada joint

```
p.setJointMotorControl2(robotId, JOINT, p.VELOCITY_CONTROL, targetVelocity=10)
```

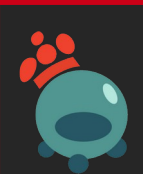
```
NumJoints: 15  
0 - base_to_right_leg  
1 - right_base_joint  
2 - right_front_wheel_joint  
3 - right_back_wheel_joint  
4 - base_to_left_leg  
5 - left_base_joint  
6 - left_front_wheel_joint  
7 - left_back_wheel_joint  
8 - gripper_extension  
9 - left_gripper_joint  
10 - left_tip_joint  
11 - right_gripper_joint  
12 - right_tip_joint  
13 - head_swivel  
14 - tobox
```



Introducción

- La velocidad asignada a un joint no implica que el robot completo adquiera esa velocidad.
- Las velocidades o fuerzas siempre estarán limitadas a las restricciones especificadas en los URDF
- La mayoría de las acciones comandadas a un joint se realizan a través de las funciones:

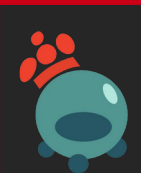
`setJointMotorControl2`
`setJointMotorControlArray`



SetJointMotorControl2

- **POSITION_CONTROL:** Mueve el joint a una posición determinada con una velocidad fijada.
 - Es necesario pasar *targetPosition* y *targetVelocity*
- **VELOCITY_CONTROL:** Establece una velocidad fija al joint
 - Es necesario pasar *targetVelocity* y opcionalmente *force*
 - *Force* define la fuerza motor (par motor, o momento de fuerza).
- **TORQUE_CONTROL:** Establecer el torque o momento de fuerza instantáneamente sobre un joint
 - Es necesario pasar *Force* que establece la fuerza

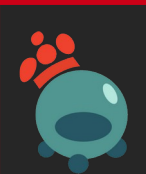
Más info en [Pybullet QuickStart Guide](#)



Velocidad

- Para generar movimiento en un robot móvil en un escenario realista con fuerzas externas, es necesario aplicar velocidades sobre sus joints que permitan movimiento sobre una superficie.
- Comandaré cierta velocidad (m/s) al motor que simula el movimiento del joint

```
p.setJointMotorControl2(robotId,  
                          2,  
                          p.VELOCITY_CONTROL,  
                          targetVelocity=10)
```



Velocidad

- Cuando el movimiento esté relacionado con la activación de varios joints simultáneamente podemos usar

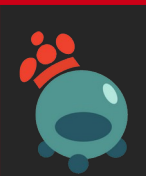
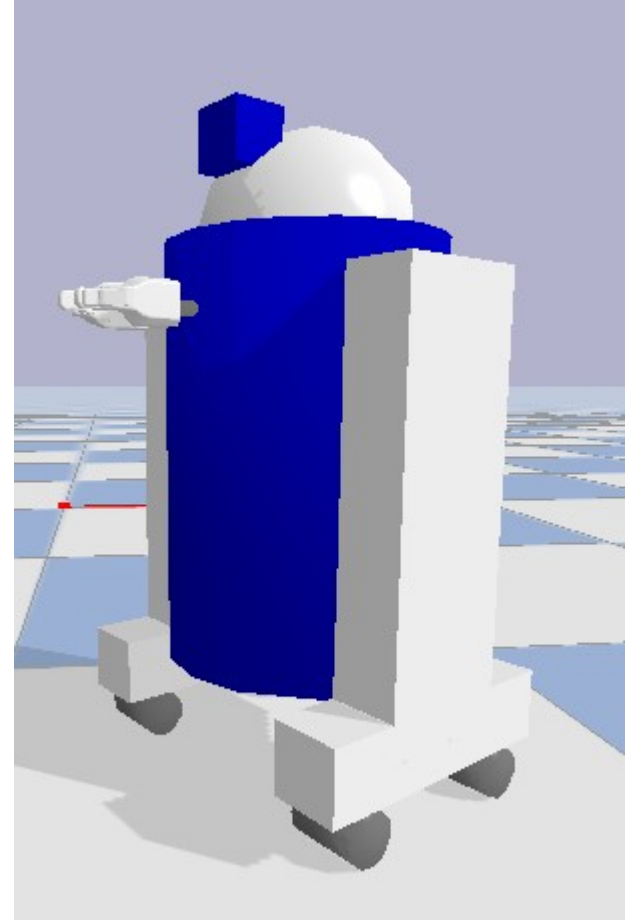
`setJointMotorControlArray`

```
p.setJointMotorControlArray(robotId,  
                             joints,  
                             p.VELOCITY_CONTROL,  
                             targetVelocities=[10,10,10,10] )
```



EJERCICIO

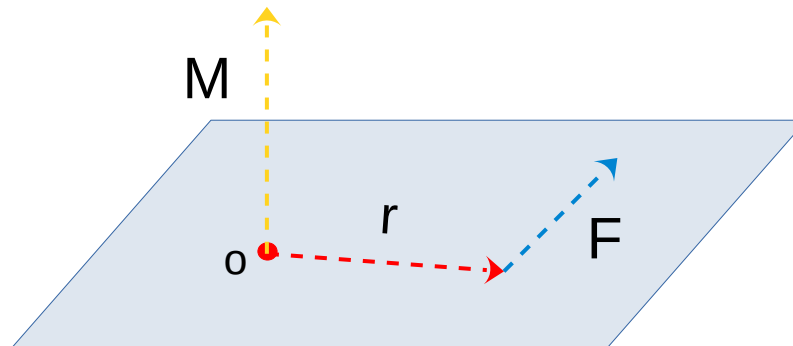
- Construye un escenario con R2D2
- Analiza sus joints y links
- Genera un movimiento en el robot para que se desplace por el mundo.
- A la vez genera movimiento en su gripper
- Utiliza control en posición y velocidad según lo consideres oportuno.



Momento de una Fuerza

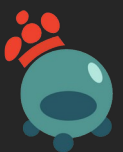
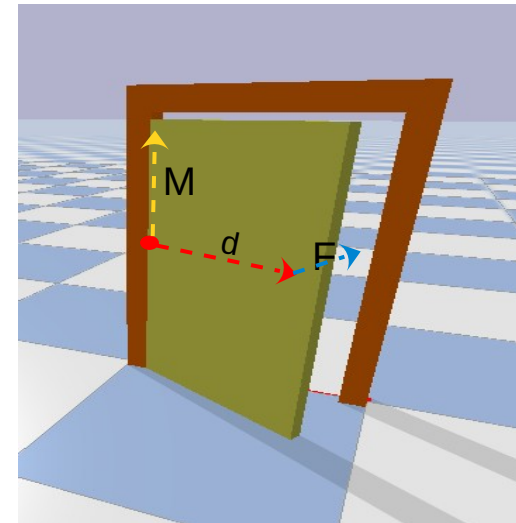
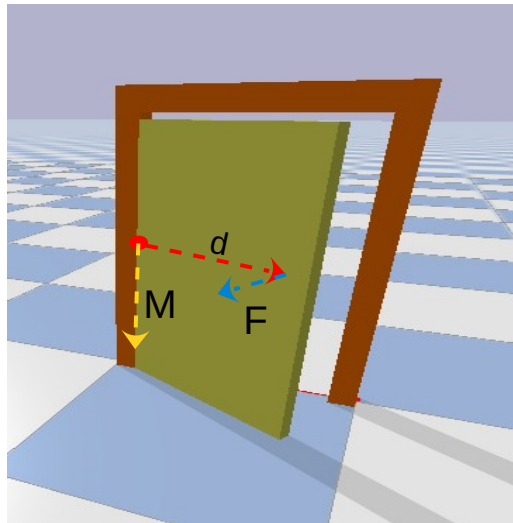
- El **momento** (M) de una fuerza respecto a un punto O , es la pseudo-fuerza resultante al multiplicar el vector posición (d) por el vector fuerza (F)
- El momento de una fuerza con respecto a un punto da a conocer en qué medida existe capacidad en una fuerza o sistema de fuerzas para cambiar el estado de la **rotación** del cuerpo alrededor de un eje que pase por dicho punto.

$$M = r \cdot F$$



Momento de una Fuerza

- El momento de fuerza cuando empujamos/tiramos de una puerta se genera en su eje de giro
 - Movimiento sentido agujas del reloj -> M negativa
 - Movimiento contrario agujas del reloj -> M positiva

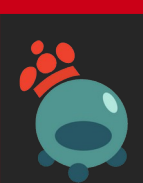


Momento de una Fuerza

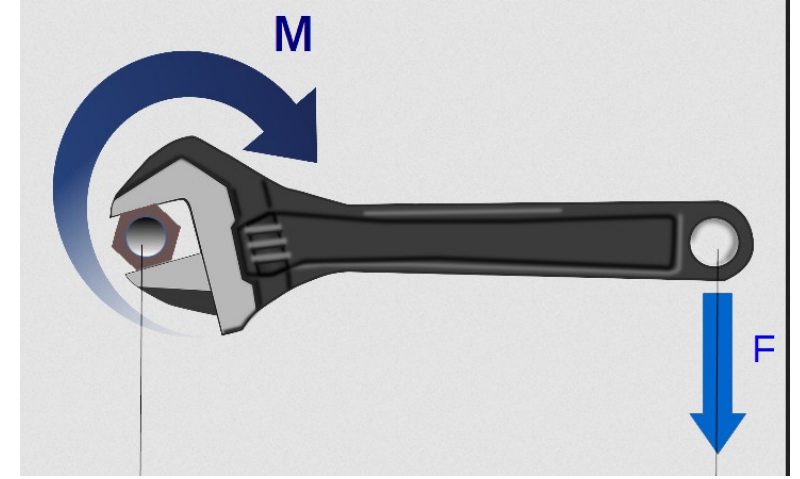
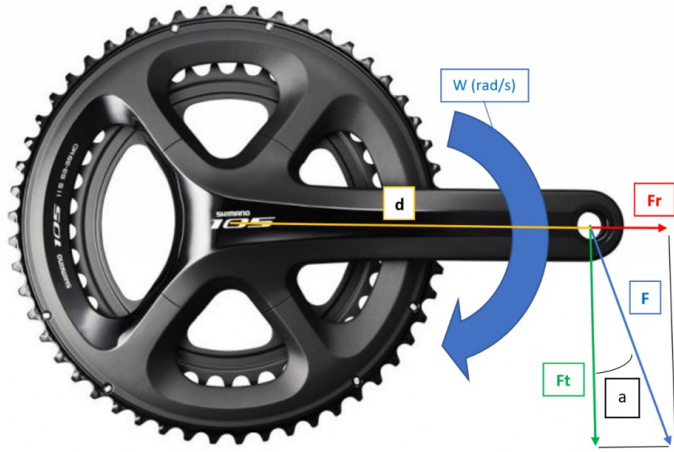
- En la vida real F genera M .
- Normalmente en los simuladores o motores de físicas, se suele definir M sobre los joints/articulaciones para simular F
- En PyBullet, en el ejemplo de la puerta, lo simulamos de la siguiente manera.

```
p.setJointMotorControl2(door, 1, p.TORQUE_CONTROL, force=value_torque)
```

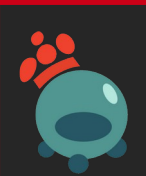
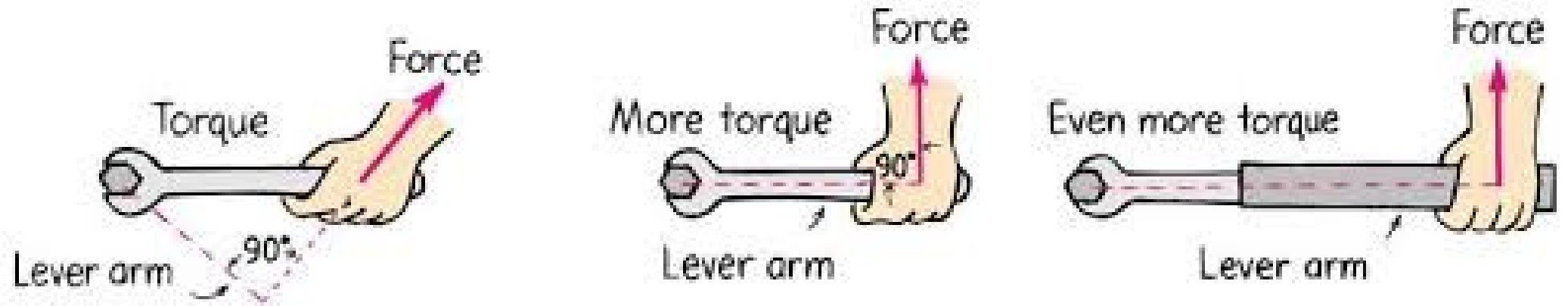
- Ejemplo: [jointFrictionAndMotor.py](#)



Momento de una Fuerza

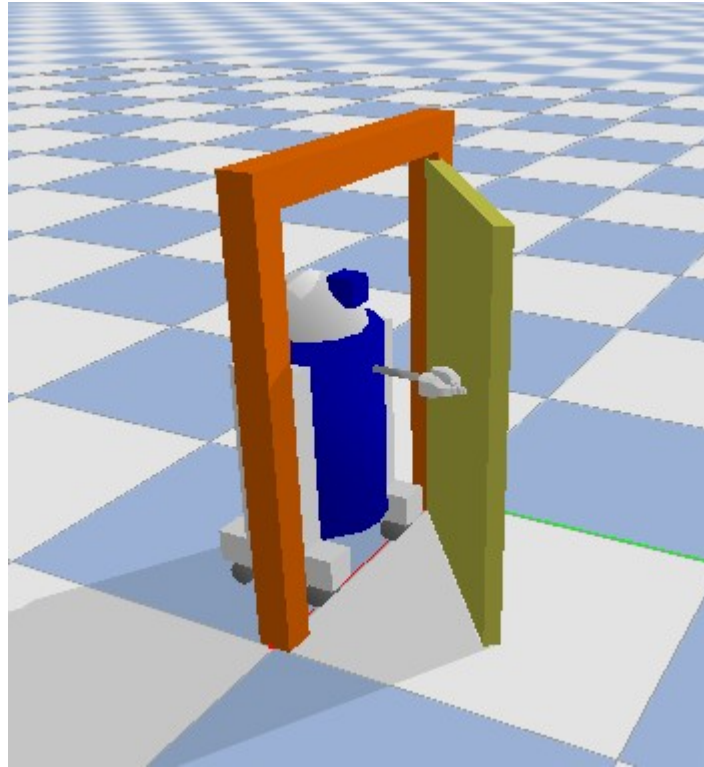


Momento de una Fuerza



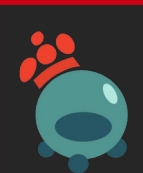
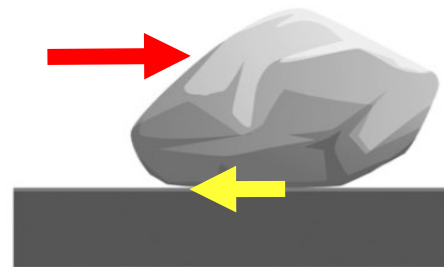
R2D2 vs Door

- Código en [aulavirtual/git](https://github.com/raulavirtual/git)



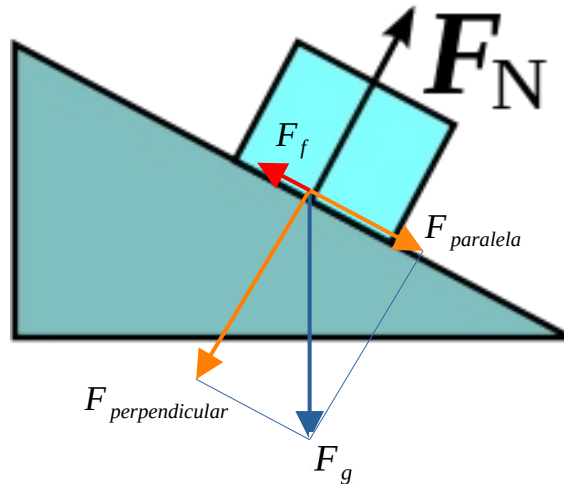
Fricción

- La fuerza de fricción es la fuerza que existe entre dos superficies en contacto, que se opone al deslizamiento
- La fuerza de rozamiento tiene dirección paralela a la superficie de apoyo.
- El coeficiente de rozamiento depende exclusivamente de la naturaleza de los cuerpos en contacto



Fricción

- La Fuerza Normal es la fuerza perpendicular a la superficie del contacto del objeto.
- Es una reacción a la fuerza que el objeto ejerce sobre la superficie, de acuerdo con la tercera ley de Newton



$$F_g = m \times g \quad \left\{ \begin{array}{l} F_{paralela} = F_g \times \sin(\theta) \\ F_{perpendicular} = F_g \times \cos(\theta) \end{array} \right.$$

$$F_n = F_{perpendicular}$$

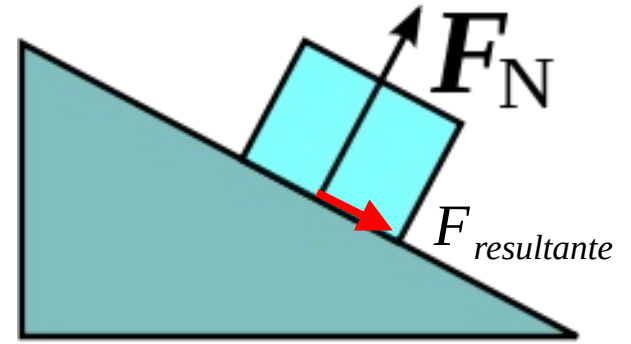
$$F_f = \mu \times F_n$$

$$F_{resultante} = F_{paralela} - F_f$$

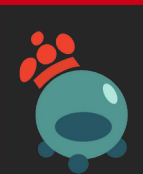
Fricción

- **Linear Friction**

- Fricción en la superficie de contacto
- Fricción de desplazamiento
- En simulaciones se modela cuando un elemento se mueve o arrastra a través de una superficie, ya sea plana o inclinada.
- La fuerza normal es la reacción a la fuerza aplicada sobre un objeto en dirección perpendicular a la superficie de soporte.



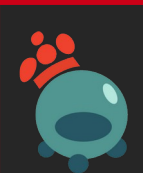
$$F_{resultante} = F_{paralela} - F_f$$



Fricción

- **Spinning Friction**

- Fricción alrededor de la normal (rueda-eje)
- La fricción de giro se refiere a la resistencia contra el movimiento rotacional alrededor del punto de contacto entre dos superficies.
- Muy utilizado en simulación de ruedas, poleas o incluso robots con partes rotativas, que interactúan con superficies de contacto
- Debe tener en cuenta, el torque, superficie de contacto, velocidades angulares, aceleraciones, etc ...



Fricción

- **Rolling Friction**

- Se produce cuando un objeto rueda sobre la superficie de otro (rueda-suelo)
- Fundamental para simular el movimiento de vehículos, ruedas, esferas o cualquier objeto que se mueva rodando en lugar de deslizándose.

$$f = u_r \cdot N$$



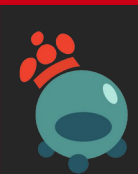
Fricción

- Coeficiente Fricción Lineal Rueda: 0.5-0.9
- Coeficiente Fricción Lineal Rueda con superficie mojada: 0.2-0.35
- Coeficiente Fricción Lineal Rueda con superficie helada: 0.05-0.1
- Coeficiente Fricción Rodadura Rueda: 0.01-0.015



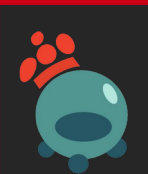
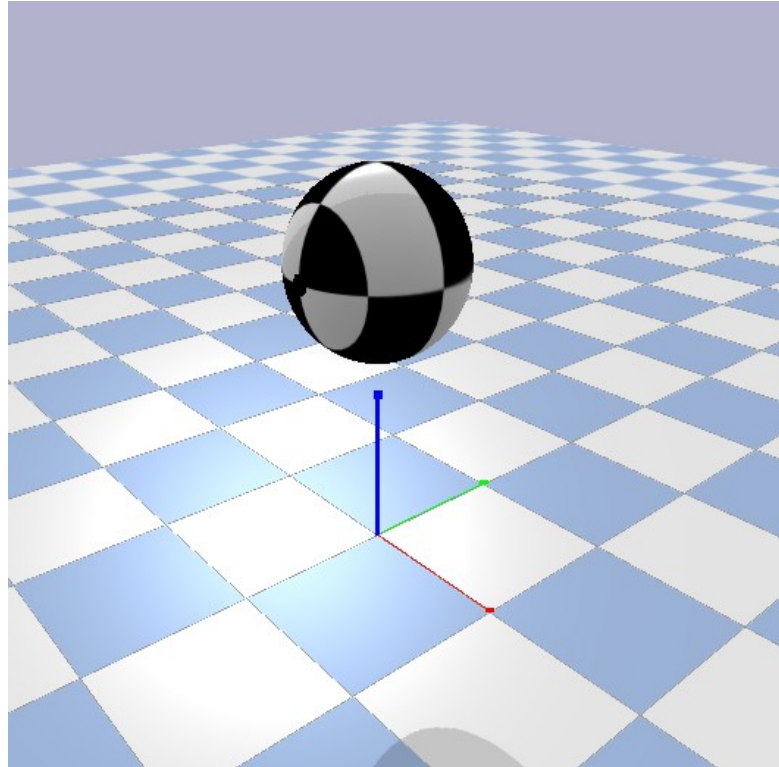
Fricción

- Restitution:
 - Define de qué manera los objetos se recuperan después de una colisión.
 - La dinámica de colisión hace uso de este coeficiente.
 - $[0..1]$, Aconsejable siempre cercano a 0.
 - 0: Los objetos se quedan pegados después de la colisión y la velocidad final es 0.
 - 1: Los objetos rebotan sin pérdida de energía cinética y la velocidad final es igual a la inicial.



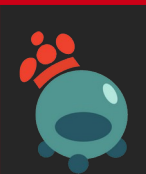
Fricción

- Restitution:
 - [Restitution.py](#)



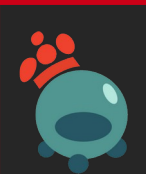
Inercia

- Es la *propiedad* que tienen los cuerpos de permanecer en su estado de reposo o movimiento (relativos)
- **Resistencia** que opone el objeto a modificar su estado de movimiento.
- Un objeto tiene *más inercia* cuando es más difícil/costoso cambiar el estado físico del mismo.
 - Inercia
 - Momento de Inercia
 - Momento Angular
 - Tensor o Matriz de Inercia



Inercia

- ¿Cómo calculamos la inercia de un coche que va a 50km/h y pesa 2000 kg cuando queremos aplicar una fuerza de frenado para que detenga su movimiento en 150 metros?



Inercia

- ¿Cómo calculamos la inercia de un coche que va a 50km/h y pesa 2000 kg cuando queremos aplicar una fuerza de frenado para que detenga su movimiento en 100 metros?

$$v_f^2 = v_o^2 + 2 \times a \times d \qquad a = \frac{v_f^2 - v_o^2}{2 \times d} \qquad a = -0.96 \, m/s^2$$

$$F = m \times a$$

$$F = -1920 \, N$$

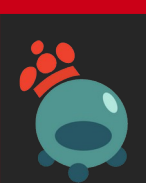
- ¿Alguna otra fuerza que deberíamos tener en cuenta?



Momento de Inercia

- El momento de inercia (I) es una medida que representa la resistencia de un objeto a cambios en su velocidad angular.
- El momento de inercia depende de:
 - La geometría del cuerpo rígido
 - La posición del eje de giro
- El momento de inercia se puede calcular como el sumatorio de los productos de la masa de partículas por el cuadrado de la distancia r (de cada partícula al eje).

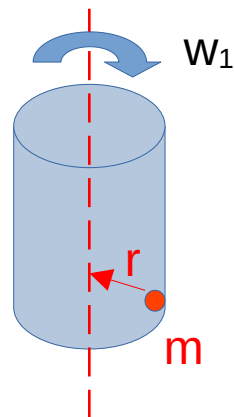
$$I = \sum m_i r_i^2$$



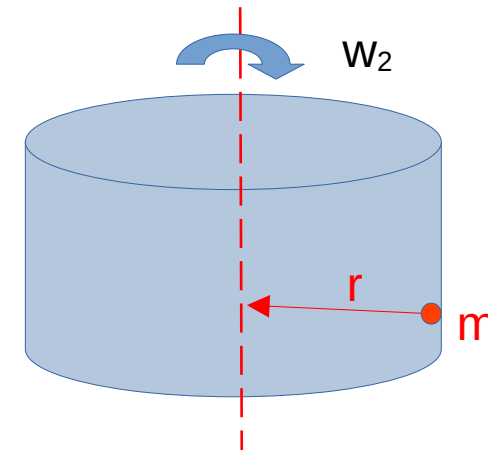
Momento de Inercia

- Análisis de sistemas rotacionales en física y en ingeniería.
- Afecta la cantidad de torque necesaria para un cambio deseado en la velocidad angular.
- Turbinas, ruedas, y satélites, optimizando su eficiencia y estabilidad.

Objeto1

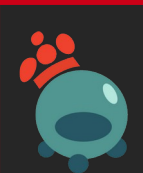


Objeto2



$$I = \sum m_i r_i^2$$

¿ $I(O1) < I(O2)$ ó $I(O1) > I(O2)$?

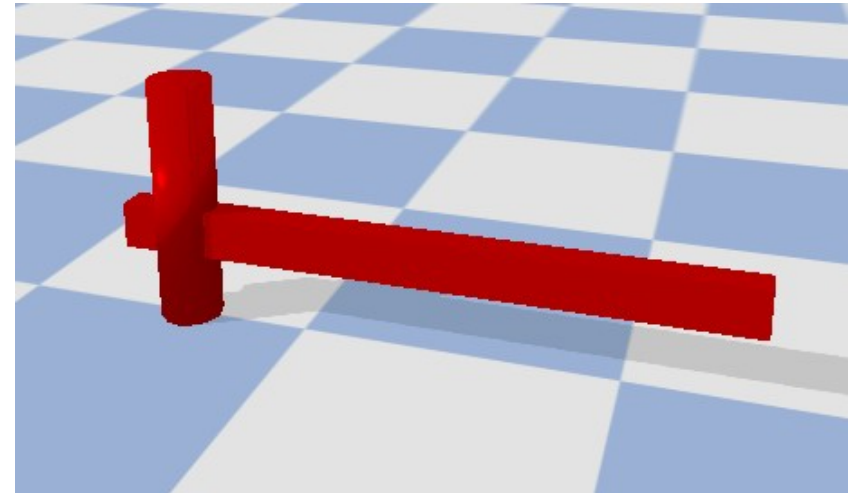


Momento angular

- El *momento angular* o *momento cinético* es una magnitud física que representa la cantidad de rotación de un objeto.

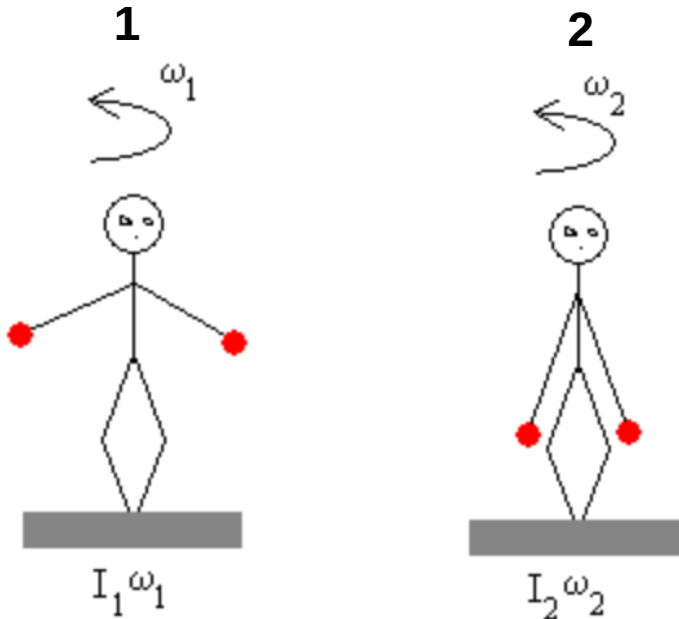
$$\mathbf{L} = \mathbf{r} \times \mathbf{p} = \mathbf{r} \times m\mathbf{v}$$

- Representación 1
- Representacion 2



Momento angular

- El **principio de conservación** del *momento angular* afirma que si el momento de las fuerzas exteriores es cero, el momento angular total se conserva, es decir, permanece constante.
- Para un cuerpo rígido que gira alrededor de su eje

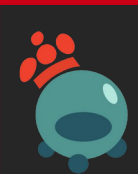


$$\mathbf{L} = I \cdot \boldsymbol{\omega}.$$

$$I = \sum m_i r_i^2$$

Momento angular

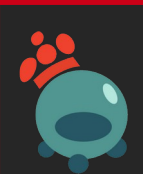
Momento Angular	Fuerza Torque
Es una medida de la cantidad de rotación de un objeto y depende tanto de la masa y velocidad del objeto como de su distancia al eje de rotación	Describe cuánto puede cambiar el estado de movimiento rotacional de un objeto debido a una fuerza aplicada.
Se relaciona con el estado de rotación de un objeto (cuánto y cómo está girando),	Se refiere a la causa de ese cambio en el estado de rotación
Un cambio en el momento angular de un objeto es el efecto de aplicar un torque sobre él.	Puede considerarse como la causa que puede alterar el momento angular de un objeto
En un sistema cerrado sin influencias externas, el momento angular se conserva	El torque describe cómo las fuerzas externas o internas están tratando de cambiar ese estado de movimiento rotacional.
$\text{kg}\cdot\text{m}^2/\text{s}$ en el SI	$\text{kg}\cdot\text{m}^2/\text{s}$ en el SI



Tensor o Matriz de Inercia

- El tensor de segundo orden o Matriz de inercia es un tensor simétrico que describe la inercia rotacional de un cuerpo solido rígido, con todos los ejes de rotación posibles.
- I_{xx}, I_{yy}, I_{zz} , son los momentos de inercia alrededor de los ejes xx , yy , y zz , respectivamente, y representan la resistencia del cuerpo a la rotación alrededor de estos ejes.

$$E_{rot} = \frac{1}{2} \begin{pmatrix} \Omega_x & \Omega_y & \Omega_z \end{pmatrix} \begin{pmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{yx} & I_{yy} & I_{yz} \\ I_{zx} & I_{zy} & I_{zz} \end{pmatrix} \begin{pmatrix} \Omega_x \\ \Omega_y \\ \Omega_z \end{pmatrix}$$



Matriz de Inercia

- La matriz de Inercia es una matriz simétrica

$$\begin{pmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{yx} & I_{yy} & I_{yz} \\ I_{zx} & I_{zy} & I_{zz} \end{pmatrix}$$

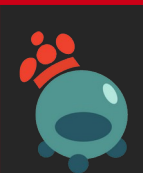
```
<inertial>
  <origin rpy="0 0 0" xyz="0 0 0.35"/>
  <mass value="1.0"/>
  <inertia ixx="0.1" ixy="0"    ixz="0"
            iyy="0.1" iyz="0"
            izz="0.1"/>
</inertial>
```



Diagonal Inercial

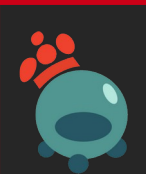
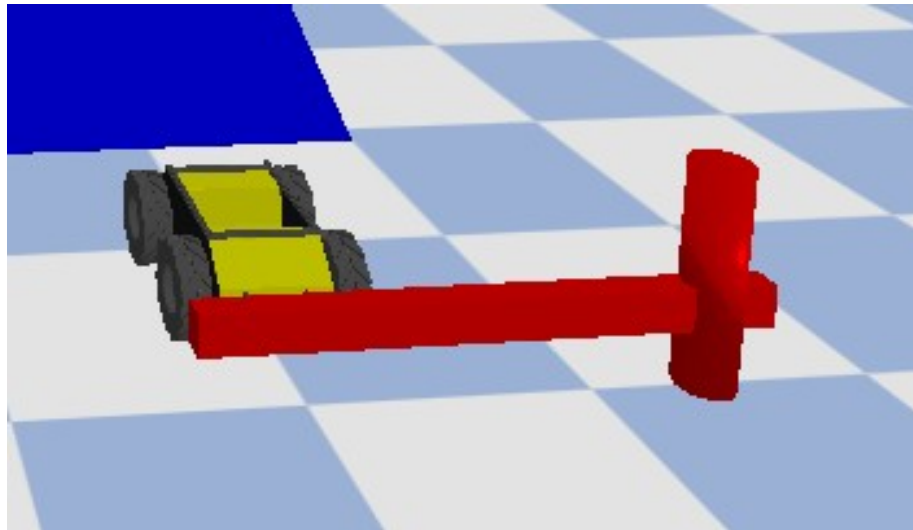
- Para cambiar la diagonal inercial de la matriz programáticamente desde pybullet se utiliza la función `changeDynamics`

```
p.changeDynamics(barrierId,0, localInertiaDiagonal=[10.1,0.0,10.1])
```



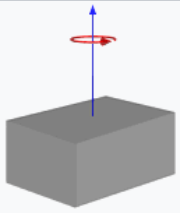
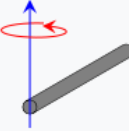
Inercia

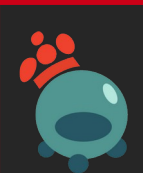
- 2 casos de estudio
 - Barrera con alta inercia
 - Barrera con baja inercia
- ¿Cómo afectará al comportamiento del robot?



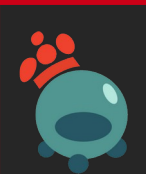
Matriz de Inercia

- Simular correctamente
- https://en.wikipedia.org/wiki/List_of_moments_of_inertia#List_of_3D_inertia_tensors

Solid cuboid of width w , height h , depth d , and mass m		$I = \begin{bmatrix} \frac{1}{12}m(h^2 + d^2) & 0 & 0 \\ 0 & \frac{1}{12}m(w^2 + d^2) & 0 \\ 0 & 0 & \frac{1}{12}m(w^2 + h^2) \end{bmatrix}$
Slender rod along y -axis of length l and mass m about end		$I = \begin{bmatrix} \frac{1}{3}ml^2 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & \frac{1}{3}ml^2 \end{bmatrix}$



Ejemplos de robots o componentes
donde tendrías que simular
correctamente: Inercia





Escuela de Ingeniería
de Fuenlabrada

