UNIVERSIDAD CARLOS III DE MADRID
ESCUELA POLITÉCNICA SUPERIOR

Departamento de Ingeniería de Sistemas y Automática.

HERRAMIENTA PARA LA CREACIÓN DE UNA LIBRERÍA DE ELEMENTOS ROBÓTICOS EN OPENRAVE

TRABAJO FIN DE GRADO

INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

Autor: Jaime Rodríguez García

Tutora: Dra. Concepción Alicia Monje Micharet

Directora: Tamara Ramos Cambero

JUNIO DE 2014

HERRAMIENTA PARA LA CREACIÓN DE UNA LIBRERÍA DE ELEMENTOS ROBÓTICOS EN OPENRAVE

Por

Jaime Rodríguez García

Tutora del Trabajo Fin de Grado

Dra. Concepción Alicia Monje Micharet

Directora del Trabajo Fin de Grado

Tamara Ramos Cambero

TRIBUNAL

Arturo de Pablo Martínez

Fernando García Fernández

Celia López Ongil

Madrid, Junio de 2014

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

AGRADECIMIENTOS

En primer lugar me gustaría mencionar a las personas con las que he realizado este proyecto y que me han ayudado a lo largo de estos meses a llevarlo a cabo, a mi tutora en el proyecto, la Dra. Concepción Alicia Monje Micharet y a la directora del proyecto, Tamara Ramos Cambero; muchas gracias a las dos por vuestra ayuda en todo momento a lo largo del proyecto, por vuestra orientación y supervisión y por haberme ayudado a llegar hasta este momento. También me gustaría nombrar a Jorge Muñoz Yañez, por su colaboración inestimable para ayudarme a entender el funcionamiento del movimiento de los robots en OpenRAVE.

En segundo lugar me gustaría agradecer a mis amigos su apoyo y el haberme ayudado a desconectar de vez en cuando, sin esos momentos seguro que no podría haber llegado hasta aquí.

Por último me gustaría mencionar a mi familia, a mis padres y a mi hermano, por su esfuerzo en intentar ayudarme en todo lo posible y por el apoyo que me han dado a lo largo de este tiempo.

RESUMEN

La finalidad de este proyecto es desarrollar una herramienta para la creación de una librería de elementos robóticos para OpenRAVE. Se creará una librería que contará con siete modelos de robots, y se creará una interfaz a través de la cual se permita al usuario escoger uno de los modelos existentes en la librería mediante la selección de unos requisitos de configuración del robot. Con esta interfaz se permitirá al usuario modificar las características del robot y realizar su simulación en OpenRAVE.

ABSTRACT

The aim of this project is to develop a tool to create a library of robotics elements for OpenRAVE. The library will have seven models of robots; also an interface will be created. The interface will allow the user to choose one of the models in the library through the selection of certain robot configuration requirements. With this interface the user is allowed to modify some of the characteristics of the robot and perform its simulation with OpenRAVE.

1. INTRODUCCIÓN.

1. 1. Introducción.

La evolución de la humanidad ha estado marcada a lo lardo de la historia por grandes revoluciones tecnológicas como lo ha sido internet. Posiblemente la próxima gran evolución venga del mundo de la robótica, un mundo que actualmente se encuentra en constante y rápida evolución. La Universidad Calos III de Madrid, concretamente en el departamento de Ingeniería de Sistemas y Automática, cuenta con un equipo que se dedica a la investigación en el mundo de la robótica.

En la investigación robótica con fundamentales los software de simulación, ya que nos permite realizar pruebas sin necesidad de utilizar robots reales, evitando riesgos innecesarios, contando con una gran semejanza al comportamiento en la vida real.

1. 2. Objetivos.

El objetivo principal de este proyecto es crear una herramienta para la creación de una librería de elementos robóticos para el simulador OpenRAVE.

Se creará una librería que contará con siete modelos de robots con distintas configuraciones. Además crearemos una interfaz para que el usuario, mediante la selección de unos requisitos, escoja uno de los modelos robóticos de la librería, pudiendo realizar modificaciones sobre el robot, y simularlo mediante OpenRAVE.

Se quiere que mediante esta interfaz se permita:

- Configurar el robot solicitando sus características básicas, y a partir de estas escoger el modelo de la librería robótico más adecuado. Estas características de configuración serán el número de extremidades y el número de grados de libertad por cada extremidad.
- Realizar algunas modificaciones sobre el modelo de robot escogido. Una vez escogido el robot el usuario podrá modificar su color, sus dimensiones o su masa.
- Simular el robot mediante OpenRAVE, permitiendo al usuario escoger la velocidad máxima y el movimiento de las articulaciones del robot.
- Cambiar la configuración del robot, permitiendo al usuario escoger entre cualquiera de los modelos existentes.

1. 3. Estado del arte.

Los simuladores de robótica se utilizan para crear aplicaciones para un robot sin necesidad de contar físicamente con el robot, ahorrando costes innecesarios y tiempo. En ocasiones estas aplicaciones se pueden transferir directamente al robot sin realizar ninguna modificación. Una de las aplicaciones más populares para los simuladores de robótica es el modelado en 3D y la representación del robot y su entorno. Con este tipo de software se puede simular el movimiento real de un robot en su entorno de trabajo. [1]

A continuación comentaremos los principales simuladores de robótica que hay disponibles.

- Webots: Es un entorno de desarrollo utilizado para modelar, programar y simular robots móviles, es ampliamente usado con fines educativos. Con Webots el usuario puede diseñar configuraciones robóticas complejas en un entorno virtual en 3D. El usuario puede elegir las propiedades de cada objeto, como la forma, el color, la masa o la fricción. Además existe una gran variedad se sensores y actuadores para equipar los robots. [2] [3]

- Gazebo: Es un simulador 3D que permite cargar diferentes tipos de robots, sensores y actuadores dentro del entorno virtual. Permite probar rápidamente los algoritmos y los robots diseñados, así como realizar pruebas de regresión utilizando escenarios realistas. Ofrece la posibilidad de simular con precisión en entornos exteriores e interiores, además dispone de un robusto motor de física, gráficos de alta calidad y varias interfaces de programación. [4]

- OpenHRP3: Open Architecture Humoniod Robotics Plataform 3. Desarrollado por NIAIST (National Institute of Advanced Industrial Science and Technology of Japan), es una plataforma para simulaciones de robots humanoides y desarrollo software que permite a los usuarios inspeccionar el modelo original del robot y programa de control a través de una simulación dinámica. Además proporciona varios componentes software de cálculo y bibliotecas que se pueden utilizar para desarrollar software relacionado con la robótica. [5]

A pesar de contar con estas opciones, para el desarrollo del proyecto utilizaremos el simulador OpenRAVE (ver 3. OPENRAVE.), se trata de un simulador multiplataforma en código abierto que es el que actualmente se está utilizando en varias investigaciones del departamento de Ingeniería de Sistemas y Automática, por lo que se ha optado por su utilización para este proyecto.

1. 4. Distribución de capítulos.

- CAPÍTULO 1: INTRODUCCIÓN: Se expone una introducción sobre el contenido del proyecto y sus objetivos.
- CAPITULO 2: REQUERIMIENTOS DE DISEÑO: Se describe la herramienta creada para el proyecto.
- CAPITULO 3: OPENRAVE: Se presenta el software de simulación utilizado en el proyecto, con sus características y su arquitectura, y se habla de las funcionalidades específicas que se utilizan.
- CAPITULO 4: BIBLIOTECA DE ELEMENTOS ROBÓTICOS: Se realiza una pequeña introducción sobre el lenguaje XML y se explica cómo se crea un robot para OpenRAVE mediante el formato propio del que dispone para XML. Además se detallan las diferentes configuraciones de robots creadas que formarán la biblioteca de elementos robóticos.
- CAPITULO 5: INTERFAZ: Se introduce el lenguaje Python, con el que está programada la interfaz, se detalla su programación y su funcionamiento.
- CAPITULO 6: DISCUSIÓN DE RESULTADOS: Ejemplo práctico del funcionamiento de la herramienta.
- CAPITULO 7: MARCO SOCIO-ECONÓMICO: Planificación y presupuesto del proyecto.
- CAPITULO 8: CONCLUSIONES Y TRABAJOS FUTUROS: Se realiza un análisis de los resultados y se comenta posibles propuestas para la mejora del proyecto.
- REFERENCIAS: Se incluyen las fuentes de las cuales se ha obtenido información para el desarrollo del proyecto.
- ANEXOS: Incluye el código con la programación de la interfaz y de los diferentes modelos de robots.

2. REQUERIMIENTOS DE DISEÑO

La herramienta a crear estará compuesta por una interfaz y una biblioteca que contendrá los elementos robóticos. La interfaz interactuará con el usuario, con el simulador OpenRAVE y con la biblioteca de elementos robóticos, como se muestra en el diagrama de la Figura 1.

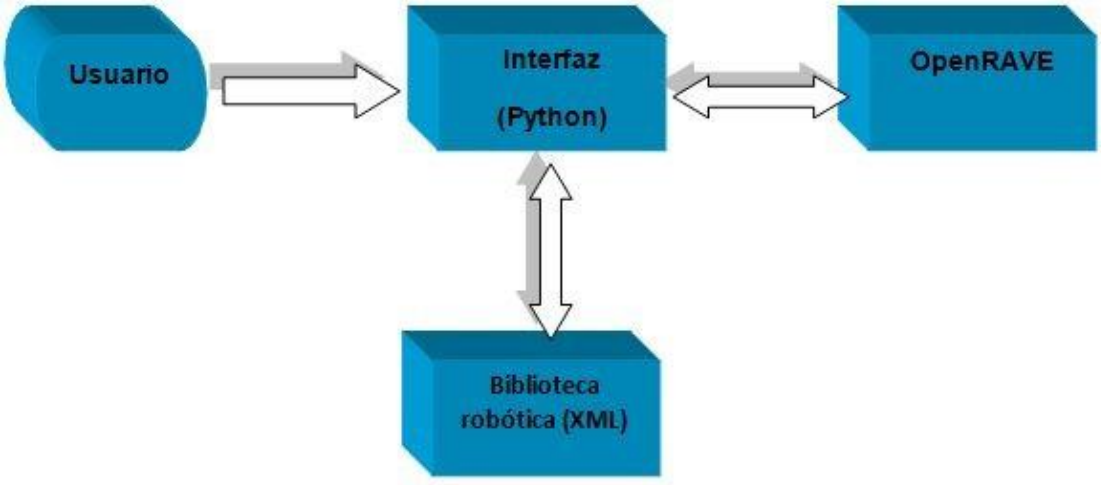

Figura 1: Diagrama de la herramienta del proyecto

El usuario, a través de la interfaz, escogerá uno de los modelos robóticos que estén disponibles en la biblioteca. La interfaz recogerá de la biblioteca el robot (modelado en XML) que seleccione el usuario. Posteriormente ofrecerá la posibilidad de realizar modificaciones sobre el robot, que una vez realizadas serán guardadas en la biblioteca. Se podrán modificar las dimensiones del robot, su masa y su color.

Una vez seleccionado el robot el usuario también tendrá la opción de realizar una simulación en OpenRAVE. La interfaz solicitará al usuario los datos necesarios para la simulación, para posteriormente llamar a OpenRAVE y abrir una nueva ventana con el visor del simulador. Luego la interfaz enviará al simulador el robot y las instrucciones necesarias para que OpenRAVE realice el movimiento.

Cuando finalice la simulación el usuario podrá realizar modificaciones sobre el robot y volver a simularlo, o podrá cambiar la configuración del robot escogiendo otro de los modelos de la biblioteca.

3. OPENRAVE.

3. 1. Introducción.

OpenRAVE es un entorno de simulación para robots que fue desarrollado por Rosen Diankov en “the Quality of Life Technology Center” de la Carnegie Mellon University Robotics Institute. Se inspira en el simulador RAVE que James Kuffner había empezado a desarrollar en 1995 y ha utilizado para sus experimentos desde entonces. El proyecto OpenRAVE se inició en 2006 y es una reescritura completa de RAVE. El mantenimiento se lleva a cabo en JSK Lab de la Universidad de Tokio.

Uno de los retos en el desarrollo de robots autónomos en el mundo real es la necesidad de integrar y probar rigurosamente *scripting* de alto nivel (prueba de secuencias de alto nivel), la planificación de movimiento, la percepción y los algoritmos de control. Con este fin, se introduce una arquitectura de software multiplataforma de código abierto llamado OpenRAVE (Open Robotics and Animation Virtual Environment).

OpenRAVE está diseñado para aplicaciones de robots autónomos del mundo real, e incluye una perfecta integración de la simulación en 3D, visualización, planificación, programación y control. Una arquitectura de *plugin* permite a los usuarios escribir con facilidad controladores personalizados o ampliar la funcionalidad. Usando *plugins* en OpenRAVE, cualquier algoritmo de planificación, de control del robot, o subsistema de detección se puede distribuir y cargar dinámicamente en tiempo de ejecución, libera a los desarrolladores de luchar con el código de bases monolíticas. Los usuarios de OpenRAVE pueden concentrarse en el desarrollo de los aspectos de planificación y de secuencias de comandos de un problema, sin tener que gestionar de forma explícita los detalles de la cinemática y la dinámica del robot, detección de colisiones, actualizaciones del entorno, y el control del robot. La arquitectura OpenRAVE proporciona una interfaz flexible que se puede utilizar junto con otros paquetes de robótica populares como Player y ROS, ya que se centra en la planificación del movimiento autónomo y secuencias de comandos de alto nivel en lugar de control de bajo nivel y protocolos de mensajes. OpenRAVE también soporta un potente entorno de programación de la red que hace que sea fácil de controlar y supervisar los robots y cambiar el flujo de ejecución en tiempo de ejecución.

Además, una de las principales ventajas de las arquitecturas de componentes abiertos es que permiten a la comunidad investigadora de la robótica para compartir y comparar fácilmente algoritmos. [6] [7]

3. 2. Arquitectura de OpenRAVE.

La arquitectura de OpenRAVE está dividida en varias capas: la interfaz gráfica del usuario (*User Side*), el guión de secuencias (*Scripting Environment*), el núcleo de OpenRAVE (*Core*) y los componentes (*Plugins*).

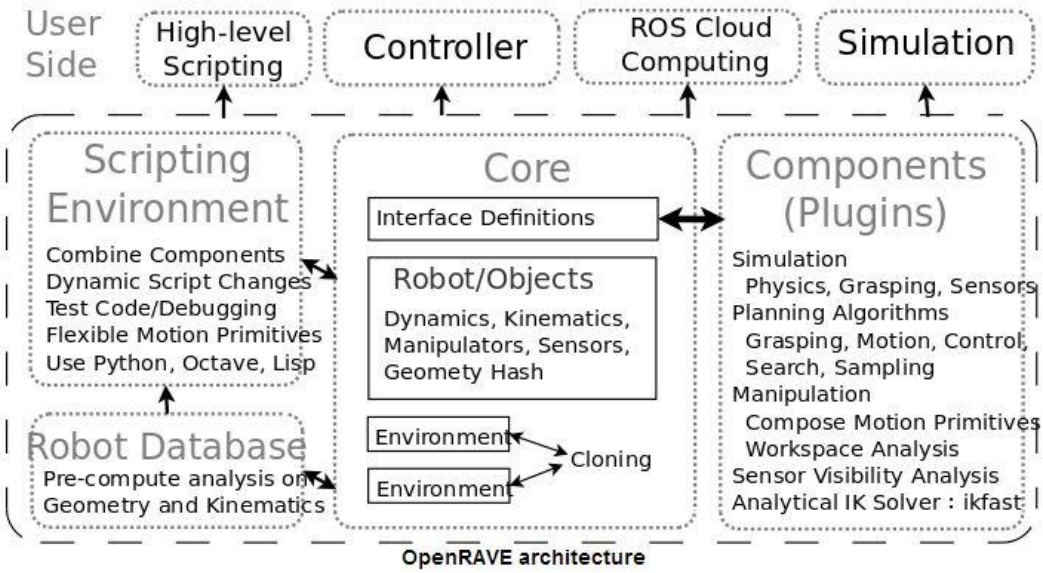

Figura 2: Arquitectura OpenRAVE

La ilustración de arriba nos muestra los cuatro componentes principales en los cuales se divide OpenRAVE:

Capa de núcleo: El núcleo está compuesto por un conjunto de Clases Base de Interfaz (*Base Interface Classes*) que definen cómo los *plugins* comparten información, y proporciona una interfaz de entorno que mantiene un estado global, el cual nos sirve como puerta para otras funciones que se ofrecen a través de OpenRAVE. El estado global de OpenRAVE maneja los *plugins* cargados, los múltiples entornos independientes y el *logging*. Por otro lado el entorno combina la comprobación de colisiones, los usuarios, los motores físicos, la cinemática del entorno y todas las interfaces en coherencia con el estado del entorno robótico.

Capa de *plugins*: OpenRAVE está diseñado como una arquitectura basada en *plugins*, que nos ofrecen la implementación de las clases base de interfaz que están cargados dinámicamente en el entorno. Los diferentes *plugins* pueden ser unidos con otras librerías robóticas que permiten a OpenRAVE ampliar su funcionalidad.

Capa de scripting: OpenRAVE nos proporciona entornos de scripts para Python y para Octave/Matlab. Python se comunica con la capa núcleo de OpenRAVE directamente con las llamadas en memoria, lo que lo hace extremadamente rápido.

Por otro lado, con Octave/Matlab, el protocolo se realiza enviando comando mediante TCP/IP con un *plugin* que ofrece un servidor de texto en OpenRAVE. El scripting permite modificaciones en tiempo real de cualquier aspecto del entorno sin necesidad de apagar, por lo que es ideal para probar nuevos algoritmos. El script de Python es muy potente y la mayoría de los ejemplos de OpenRAVE se ofrecen en este lenguaje. De hecho los usuarios deben tratar el lenguaje de scripting como una parte integral de todo el sistema, no como un reemplazo para la API C++.

Capa de base de datos de los robots: Implementa una base de conocimientos de planificación y proporciona una interfaz simple para su acceso y generación de parámetros. La propia base de datos se compone de los análisis cinemáticos, dinámicos y geométricos del robot y de su tarea. [6] [8]

3.3. Principales objetivos y características de OpenRAVE.

Los principales objetivos y características de diseño que nos ofrece el simulador OpenRAVE son:

- Contar con arquitectura basada en *plugins* que permite al usuario aumentar la funcionalidad sin necesidad de recompilar todo el código base. La mayor parte de la funcionalidad es ofrecida como *plugins*, manteniendo así el núcleo lo más simple posible.
- Ofrecer muchas implementaciones de algoritmos de planificación de movimiento que se pueden extender fácilmente a las nuevas tareas.
- Fácil depuración de los componentes durante el tiempo de ejecución sin tener que recompilar ni reiniciar todo el sistema.
- Permitir que el núcleo de OpenRAVE pueda ser usado como entorno de simulación, como entorno de programación de alto nivel, como resultado de un módulo de control cinemático y dinámico del robot o como caja negra para la manipulación de la planificación en un entorno robótico distribuido.
- Apoyar un entorno multi-hilo para permitir un fácil desarrollo en paralelo de los planificadores y otras funciones con la sincronización mínima requerida por el usuario.

Uno de los puntos fuertes de OpenRAVE en comparación con otros paquetes de planificación es el ser capaz de aplicar algoritmos a cualquier escenario, con muy pocas modificaciones cuando los robots o los objetivos de destino cambian. Los usuarios pueden concentrarse en el desarrollo de los aspectos de planificación y de

secuencias de comandos de un problema son tener que gestionar de forma explícita los detalles de cinemática de robots, dinámica, detección de colisiones actualizaciones del entorno, modelado de sensores y control del robot.

3.4. Funcionalidades específicas. Robots en OpenRAVE y su movimiento.

OpenRAVE permite el modelado de robots en dos formatos: Collada y XML. OpenRAVE soporta el formato de archivo Collada para especificar robots y añade su propio conjunto de extensiones-robot específico. El formato Collada se puede utilizar para especificar todos los robots y la información de la escena relacionada. Debido a que Collada puede ser un poco difícil de editar a mano, OpenRAVE define su propio formato mediante el empleo del formato XML para facilitar la creación de los modelos. Además es posible convertir los robots de XML a Collada mediante el siguiente comando: *-save myrobot.zae myrobot.xml.* Los siete modelos de los robots que se realizan en este proyecto son creados a través del formato propio que facilita OpenRAVE mediante el empleo de XML.

En cuanto al movimiento de los robots, OpenRAVE cuenta con una serie de planificadores ya definidos que nos dan la mejor trayectoria para nuestro robot en un determinado escenario evitando colisiones con posibles obstáculos. En este proyecto no contaremos con un escenario definido, y únicamente queremos visualizar el movimiento de las extremidades de robot, por lo que no hará falta usar un planificador. Se utilizará la función de OpenRAVE *MoveActiveJoints* para ejecutar el movimiento del robot. A esta función se le indica el punto objetivo al que se quiere llegar y ella se encargará de realizarlo. Este punto serán las posiciones finales de cada una de las articulaciones. El punto objetivo que tenemos que indicarle no es un punto geométrico en 3D relativo a la posición final en el entorno, si no que es un punto angular de las articulaciones del robot en radianes. Si disponemos de un robot con dos extremidades y un grado de libertad cada una, tendremos que pasarle un punto objetivo de 2 dimensiones, pero si el robot de dos extremidades cuenta con dos grados de libertad, tenemos que pasarle un punto de cuatro dimensiones.

4. BIBLIOTECA DE ELEMENTOS ROBÓTICOS.

4. 1. Introducción.

La biblioteca de elementos robóticos está compuesta por siete modelos de robots con distintas configuraciones creados para OpenRAVE, cuyas características podrán ser modificadas posteriormente por el usuario a través de la interfaz.

OpenRAVE utiliza XML para almacenar todas las descripciones de los robots y de los escenarios. El formato XML es lo suficientemente flexible para enlazar un archivo XML de otro o como para incluir objetos o robots ya creados en un entorno. También es posible especificar archivos en formatos VRML o IV dentro de ella para importar modelos. [9]

4. 2. XML.

XML, siglas en ingles de *eXensible Markup Language*, es un lenguaje muy simple pero estricto que juega un papel fundamental en el intercambio de una gran variedad de datos. Desarrollado por el World Wide Web Consortium (W3C), es una derivación del lenguaje SGML. No se creó únicamente para su aplicación en internet, si no que se propone como un estándar para el intercambio de información estructurada entre diferentes plataformas. Es un lenguaje de etiquetado extensible similar a HTML pero su función principal es describir datos y no mostrarlos como es el caso de HTML. Sirve para estructurar, almacenar e intercambiar información. Mediante los XML conseguimos que la información se estructure de forma bien definida. Esta estructuración se realiza mediante lo que se llama elementos, los cuales a su vez se pueden componer de otros elementos, y se las señala mediante etiquetas.

Una etiqueta consiste en una marca hecha en el documento, que señala una parte de éste como un elemento. Es una parte de información con un sentido claro y definido. Las etiquetas tienen la siguiente forma: <*Nombre*>, donde *Nombre* es el nombre del elemento que se está señalando. Y mediante <*/Nombre*> indicaremos el fin de dicho elemento. Un elemento puede consistir en varias etiquetas anidadas (por lo tanto de varios elementos) formando árboles. Además las etiquetas también pueden disponer de atributos, los cuales nos sirven para indicar características específicas de la etiqueta. Estos atributos se indicarán de la siguiente forma: <*Nombre atributo1* = “0”>. [10] [11]

4. 3. Creación de los robots.

Se han realizado siete modelos con diferentes configuraciones de robots que formarán la biblioteca de elementos robóticos. La estructura básica de todos ellos es la misma, están compuestos por una base que tiene forma cúbica y un número determinado de extremidades con forma cilíndrica. Los distintos robots se diferencian entre sí en función del número de extremidades que tiene cada uno y el número de grados de libertad que tienen sus extremidades. Tendremos dos robots con dos extremidades, dos robots con tres extremidades y tres robots con cuatro extremidades. Los robots con el mismo número de extremidades se diferencian en el número de grados de libertad por extremidad, teniendo un grado de libertad o dos. Solamente en el caso de cuatro extremidades y dos grados de libertad tenemos dos modelos, que se diferencian entre sí en la distinta colocación de las extremidades.

Por lo tanto dispondremos de siete XML diferentes en los que se guarda la descripción completa de cada modelo de robot. A continuación se van a describir las características básicas de los modelos en XML de los robots, y posteriormente se detallarán las diferentes configuraciones.

4. 3. 1. Creación de los XML.

Todos los archivos XML de los robots tienen la misma estructura. Para comenzar a crear el XML de un robot lo primero que se debe indicar es que se trata precisamente de un robot, por lo que el elemento primario y del que van a colgar los demás elementos será <Robot>. Este únicamente tendrá un atributo que será el nombre que le demos al robot.

Una vez indicado que se está creando un robot, hay que indicar que se trata de un cuerpo cinemático. Para ello creamos el elemento <KinBody>, del cual derivarán el resto de elementos que compongan el robot. Este cuerpo cinemático consiste en una colección de cuerpos rígidos (<Body>) y articulaciones (<Joint>) para conectar estos cuerpos entre sí y permitir el movimiento entre ellos. En nuestros modelos siempre dispondremos de un mínimo de tres cuerpos rígidos, uno para la base y uno por cada extremidad, y un mínimo de dos articulaciones para unir las extremidades a la base. Además si las extremidades disponen de dos grados de libertad, estas estarán compuestas por dos cuerpos rígidos cilíndricos, unidos por una articulación, de manera similar a una pierna humana.

La Figura 3 muestra como sería el XML con lo visto hasta ahora, un esquema simple de la composición básica de un robot:

```
<Robot name="robot">
  <KinBody>
    <Body name="Body0" type="dynamic">
      ...
    </Body>
    <Body name="Body1" type="dynamic">
      ...
    </Body>
    <Joint name="J0" type="hinge">
      ...
    </Joint>
  </KinBody>
</Robot>
```

Figura 3: XML esquema básico robot

Los cuerpos rígidos que derivan del cuerpo cinemático son cada una de las distintas formas geométricas que, en conjunto, van a componer el robot. Como atributos del cuerpo pondremos un nombre asociado a este y le indicaremos que es de tipo dinámico para que el cuerpo pueda moverse. Además a cada cuerpo le incluiremos como elementos tres características: la posición inicial, la masa y la geometría.

Para trasladar el cuerpo a su posición inicial en el entorno de simulación utilizamos el elemento <translation>, el cual está formado por tres números, que serán las coordenadas respecto de los ejes ‘x’, ‘y’, ‘z’ del entorno. Podemos indicar la posición de un cuerpo en función de otro cuerpo en lugar de hacerlo respecto del sistema global, para ello utilizamos <offsetfrom>, donde indicaremos el nombre del cuerpo desde el cual posicionaremos el nuevo cuerpo.

Para indicar la masa (<mass>), debemos pasarle como atributo el tipo de geometría de nuestro cuerpo, y luego indicar el peso que queremos asignarle.

En cuanto a la geometría (<Geom>) puede ser cúbica (<box>), cilíndrica (<cylinder>) o esférica (<sphere>). En nuestros robots utilizamos la forma cúbica para la base y la cilíndrica para las extremidades. Dentro del elemento de la geometría indicaremos tres características.

La primera de estas características es el color (<diffuseColor>) que queremos asignarle al cuerpo rígido. Para definir el color, indicamos tres cantidades, la primera de ellas será la cantidad de color rojo, la segunda la cantidad de color verde y por último la cantidad de azul. Modificando estas cantidades podremos conseguir cualquier color. Además, poniendo las tres cantidades a cero tendremos el color negro y poniendo las tres iguales tendremos blanco.

La segunda característica es el eje de rotación (<RotationAxis>). Con este eje de rotación, podemos girar el cuerpo en cualquiera de los ejes y a cualquier ángulo. Contendrá 4 números, los 3 primeros son los ejes (x, y, z) donde indicaremos cuáles de ellos queremos rotar (si no se quiere rotar ese eje se indica un 0) y en el último número indicaremos el ángulo en grados en el que queremos rotar el cuerpo.

Como última característica de la geometría del cuerpo para terminar la composición de un cuerpo rígido definimos sus dimensiones. Se indican de diferente forma en función de tipo de geometría. Para una geometría tipo <box> indicamos las longitudes de los tres ejes, mediante el elemento <extents>. Para las geometrías cilíndricas indicamos el radio (<radius>) y la altura <height>. Y para las esféricas únicamente se indicaría el radio.

El segundo elemento necesario para componer el cuerpo rígido del robot son las articulaciones (<Joint>). Una articulación nos permite unir los diferentes cuerpos entre sí, además de permitir el movimiento entre esos cuerpos. Para declarar una articulación le introducimos como atributos el nombre que le queramos dar y el tipo de articulación. Para nuestros robots todas las articulaciones serán bisagras (*hinge*). Dentro de las articulaciones tendremos los siguientes elementos: <Body>, <offsetfrom>, <anchor>, <axis>, <limitsdeg>, <weight> y <maxveldeg>.

El primero de los elementos, <Body>, será con el identificamos los distintos cuerpos que forman parte de la unión. Se crea la etiqueta <Body> y a continuación indicamos el nombre que le hayamos asignado al cuerpo, por lo que este cuerpo debe haber sido previamente definido como cuerpo rígido. Como las articulaciones son una unión entre dos cuerpos, utilizamos esta etiqueta dos veces por cada articulación para determinar los cuerpos que forman parte de la unión.

Mediante la etiqueta <offsetfrom> indicamos el nombre del cuerpo de la unión respecto al cual la articulación va a realizar el movimiento de giro. En nuestros modelos las articulaciones servirán para la unión de la base con las extremidades o para la unión de los dos segmentos de las extremidades en el caso que tengan dos grados de libertad. En el primer caso, la unión se hace respecto de la base, ya que queremos que la extremidad realice un movimiento respecto de la base del robot. Y en el caso de tener extremidades con dos grados de libertad, la unión se hace respecto al segmento de extremidad que está unido a la base.

Cuando indicamos con <offsetfrom> el cuerpo sobre el que se realiza el movimiento en la articulación, estamos indicando que el movimiento se realizará relativo al centro de masas del ese cuerpo, pero lo que queremos es realizar un movimiento respecto al punto de unión de los dos cuerpos que se unen en la

articulación. Para ello utilizamos <anchor>, donde indicaremos mediante tres coordenadas (x, y, z respecto del centro de gravedad del cuerpo) la posición sobre la que se realizará el giro.

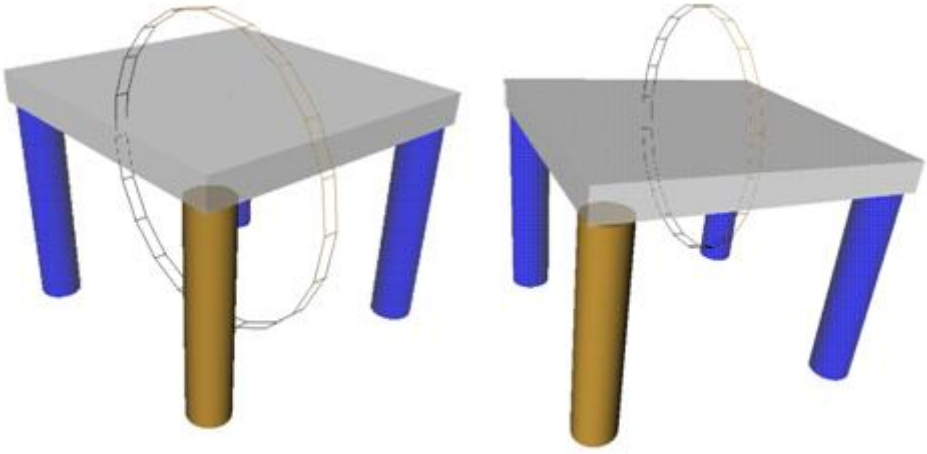

Figura 4: Ejemplo robot con y sin <anchor>

El eje sobre el que se realiza el movimiento se indica mediante el elemento <axis>. Este elemento está compuesto por 3 números, que representan los ejes ‘x’, ‘y’, ‘z’. Únicamente se coloca un 1 en el eje sobre el que se realizará el movimiento y un 0 sobre los otros.

Para limitar el rango de movimiento tenemos <limitsdeg>, donde indicamos dos valores que corresponden con las posiciones mínima y máxima en grados de la posición angular final del elemento que gira respecto al indicado en <offsetfrom>.

El elemento <weight> indica el peso de la articulación, y el elemento <maxveldeg> nos indica la máxima velocidad de movimiento de la articulación en grados por segundo.

Con estas características se han creado los diferentes ficheros de XML de configuración de los robots. A continuación se detallará la composición de los archivos XML que contienen la configuración de cada uno de los siete modelos. A lo lardo de la descripción de cada una de las configuraciones hay distintas figuras en las que a aparecen los robot, en ellas aparecen unos ejes de coordenadas, aclarar que el eje ‘x’ es el rojo, el eje ‘y’ es el verde y el ‘z’ es el azul.

4. 3. 2. Robot dos extremidades y un grado de libertad.

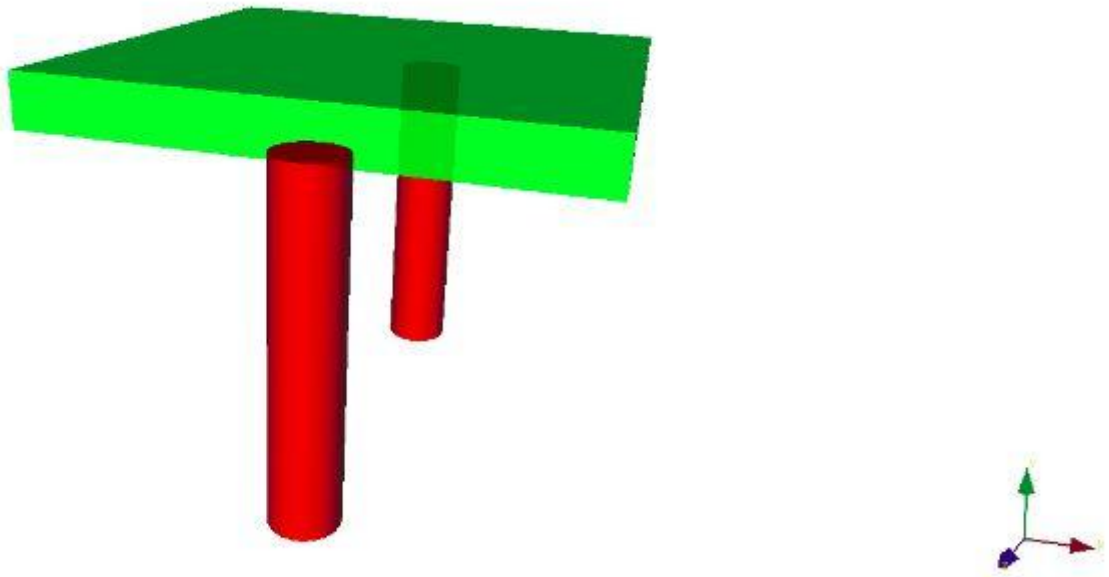

Figura 5: Robot dos extremidades un grado libertad

Este primer robot está formado por un rectángulo que será la base del robot y dos extremidades con un grado de libertad.

Como hemos descrito anteriormente, para crear el fichero XML lo primero que indicamos es que se trata de un robot, al cual le ponemos como nombre para etiquetarlo *robot_2p*. Además creamos el cuerpo cinemático e indicamos la primera geometría de la que está compuesta el robot, la base.

```
<Robot name="robot_2p">
  <KinBody>
    <Body name="Base" type="dynamic">
      <Translation>0.0 0.0 0.0</Translation>
      <Geom type="box">
    <Extents>0.5 0.05 0.5</Extents>
    <RotationAxis>0 0 0 0</RotationAxis>
        <diffuseColor>1 0 0</diffuseColor>
      </Geom>
      <Mass type="box">
        <total>10.0</total>
      </Mass>
    </Body>
```

Figura 6: XML de la base robot_2p

Con el código de la Figura 6, se crea una estructura cúbica con las dimensiones indicadas, está será la base del robot.

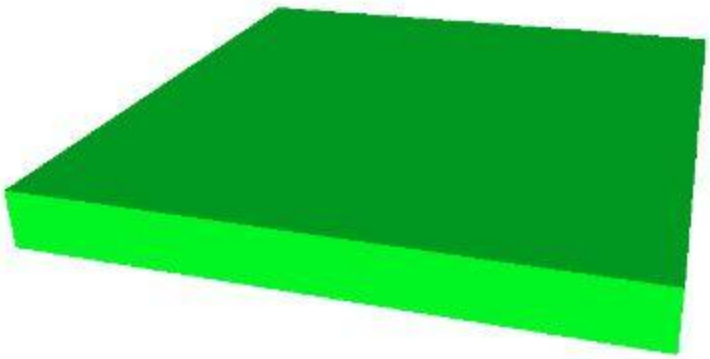

Figura 7: Base robot

La posición de la base (indicada con <Translation>) es la 0, 0, 0. Es decir, lo colocamos en el centro del entorno en la simulación, como haremos con todos los modelos. Le indicamos que la geometría es de tipo *box* y ponemos las dimensiones del largo, alto y ancho mediante <Extents>. En la rotación de los ejes están las variables a cero, ya que no queremos modificar su orientación. Además indicamos el color y la masa.

A continuación creamos la primera de las dos extremidades de las que está compuesto este robot.

```
<!-- the first movable link-->
<Body name="Arm0" type="dynamic">
  <offsetfrom>Base</offsetfrom>
  <Translation>0.0 -0.4 -0.43</Translation>
  <Geom type="cylinder">
<rotationaxis>0 0 0 0</rotationaxis>
    <radius>0.07</radius>
    <height>0.7</height>
    <diffuseColor>0 0 0</diffuseColor>
  </Geom>
  <Mass type="mimicgeom">
    <total>1.0</total>
  </Mass>
</Body>
```

Figura 8: XML de la primera extremidad robot_2p

El nombre de esta primera pata del robot es *Arm0* y la posición se la indicaremos relativa a la base, para ello utilizamos el <offsetfrom>. Colocamos la primera extremidad de tal forma que al final ambas extremidades estén a ambos lados del centro de la base del robot y justo debajo de la base del robot. Indicamos que la forma geométrica de este elemento es un cilindro, y sus dimensiones con el radio y la longitud, además del color y de la masa.

Posteriormente creamos el primer *joint*, es decir, la primera unión que se produce entre la base y la primera extremidad.

```
<Joint name="Arm0" type="hinge">
  <Body>Base</Body>
  <Body>Arm0</Body>
  <offsetfrom>Base</offsetfrom>
  <weight>4</weight>
  <limitsdeg>-60 60</limitsdeg>
  <axis>0 0 1</axis>
  <maxveldeg>10.0</maxveldeg>
  <resolution>1</resolution>
  <anchor>0.0 0.0 -0.43</anchor>
</Joint>
```

Figura 9: XML de la primera articulación robor_2p

Esta articulación tiene el nombre *Arm0* y, como se indica, es la unión entre los dos elementos ya creados anteriormente *Base* y *Arm0* (Nombre el cuerpo creado anteriormente, no de la articulación que se llama igual). Indicamos con <offsetfrom> que el movimiento entre ambos elementos se hará respecto a la base, y mediante <anchor> que el movimiento será relativo a la posición sobre la base en la que se unen base y extremidad, es decir, que el punto de unión entre los cuerpos será fijo y la extremidad realizará el movimiento desde ese punto.

Mediante la etiqueta <axis> hacemos que el movimiento se realice únicamente sobre el eje sobre el eje ‘z’. Además le indicamos el peso, la máxima velocidad, la resolución y el límite del movimiento.

En la siguiente ilustración se muestra como quedaría el robot con la base y la primera extremidad creada.

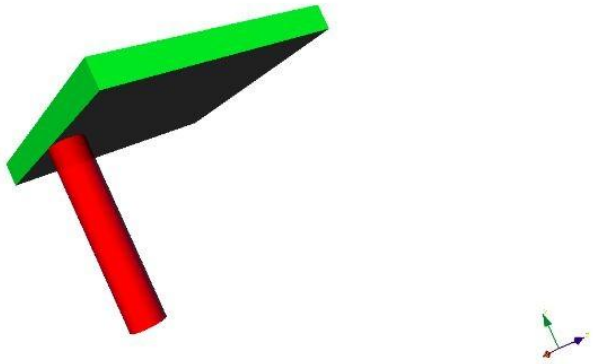

Figura 10: robot_2p con una extremidad creada

Finalmente incluiremos la segunda extremidad y la articulación para la unión entre esta y la base del robot.

```
<!-- the second movable link-->
   <Body name="Arm1" type="dynamic">
     <offsetfrom>Base</offsetfrom>
     <!-- Translation relative to Base-->
     <Translation>0.0 -4.5 7.3</Translation>
     <Geom type="cylinder">
   <rotationaxis>0 0 0 0</rotationaxis>
       <radius>0.7</radius>
       <height>8.0</height>
       <diffuseColor>1 0 0</diffuseColor>
     </Geom>
     <Mass type="mimicgeom">
       <total>10.0</total>
     </Mass>
   </Body>
   <Joint name="Arm1" type="hinge">
     <Body>Base</Body>
     <Body>Arm1</Body>
     <offsetfrom>Base</offsetfrom>
     <weight>4</weight>
     <limitsdeg>-60 60</limitsdeg>
     <axis>0 0 1</axis>
     <maxveldeg>10.0</maxveldeg>
     <resolution>1</resolution>
     <anchor>0.0 0.0 7.3</anchor>
   </Joint>

 </KinBody>
</Robot>
```

Figura 11: XML de la segunda extremidad robot_2p

La segunda extremidad la definimos igual que la primera. Únicamente varía la posición, que para esta segunda extremidad se ha colocado en el lado contrario del centro de la base del robot en la que pusimos la primera, y el nombre de la extremidad. Con la unión ocurre lo mismo, se crea igual que la primera, únicamente indicando que la articulación une la base con la segunda extremidad.

4. 3. 3. Robot dos extremidades y dos grados de libertad.

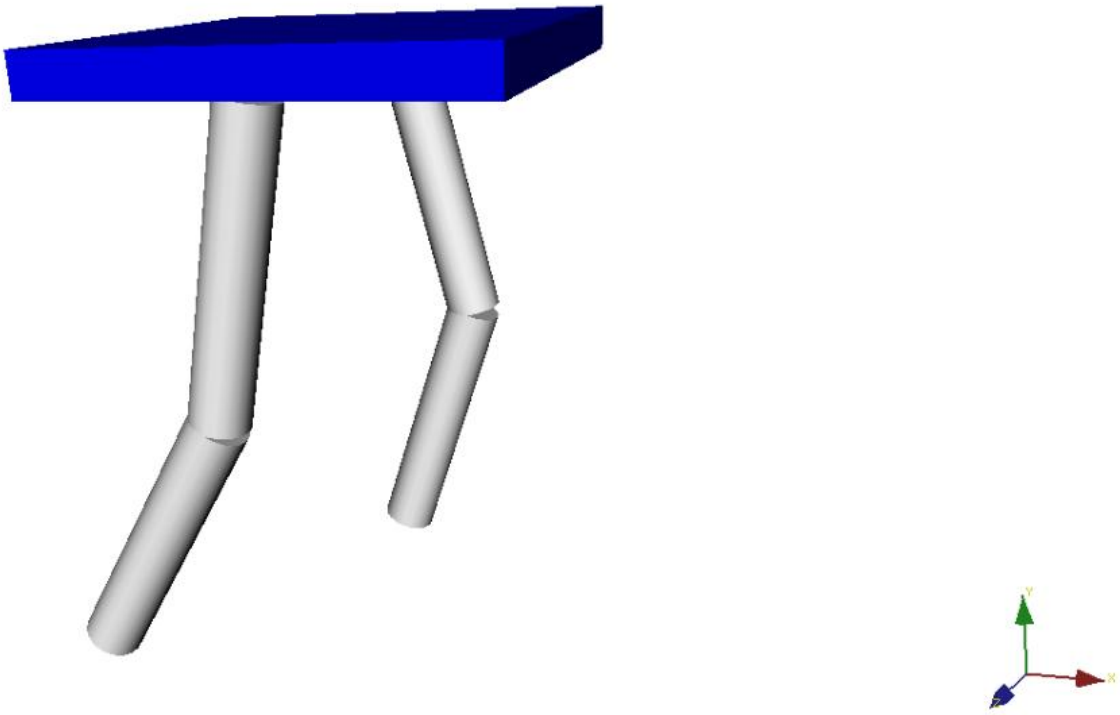

Figura 12: Robot 2 extremidades y 2 grados de libertad

Este robot es parecido al anterior, también está compuesto por una base rectangular y por dos extremidades, pero en este caso, estas extremidades dispondrán de dos grados de libertad cada una de ellas. Para su creación, a cada una de las extremidades la dividiremos en dos segmentos, es decir, crearemos dos cilindros, e incluiremos una articulación entre las dos partes para unirlas y permitir el movimiento.

Empezamos creando la base, el primer segmento de la primera extremidad y la unión entre ambos, con el nombre para el robot de *robot_2p2*.

```
<Robot name="robot_2p2">
  <KinBody>
    <Body name="Base" type="dynamic">
      <Translation>0.0 0.0 0.0</Translation>
      <Geom type="box">
    <Extents>5.0 0.5 5.0</Extents>
    <RotationAxis>0 0 0 0</RotationAxis>
        <diffuseColor>0 0 1</diffuseColor>
      </Geom>
      <Mass type="box">
        <total>10.0</total>
      </Mass>
    </Body>

    <!-- the first movable link-->
    <Body name="Arm0" type="dynamic">
      <offsetfrom>Base</offsetfrom>
      <Translation>0.0 -4.0 -4.3</Translation>
      <Geom type="cylinder">
    <rotationaxis>0 0 0 0</rotationaxis>
        <radius>0.7</radius>
        <height>7.0</height>
        <diffuseColor>1 1 1</diffuseColor>
      </Geom>
      <Mass type="mimicgeom">
        <total>10.0</total>
      </Mass>
    </Body>

    <Joint name="Arm0" type="hinge">
      <Body>Base</Body>
      <Body>Arm0</Body>
      <offsetfrom>Base</offsetfrom>
      <weight>4</weight>
      <limitsdeg>-60 60</limitsdeg>
      <axis>0 0 1</axis>
      <maxveldeg>15.0</maxveldeg>
      <resolution>1</resolution>
      <anchor>0.0 0.0 -4.3</anchor>
    </Joint>
```

Figura 13: XML primer segmento de la primera extremidad del robot_2p2

Vemos que se ha realizado de la misma manera en la que se creó la base, la primera extremidad y la articulación del primer robot. A continuación creamos el segundo segmento que forma parte de la primera extremidad.

```
<!--Segmento 2 de la pata-->
<Body name="Arm01" type="dynamic">
  <offsetfrom>Arm0</offsetfrom>
  <Translation>0.0 -7.0 0.0</Translation>
  <Geom type="cylinder">
<rotationaxis>0 0 0 0</rotationaxis>
    <radius>0.7</radius>
    <height>7.0</height>
    <diffuseColor>1 1 1</diffuseColor>
  </Geom>
  <Mass type="mimicgeom">
    <total>10.0</total>
  </Mass>
</Body>
```

Figura 14: XML segundo segmento de la primera extremidad robot_2p2

La creamos con el nombre *Arm01,* y al igual que el primer segmento será un cilindro de las mismas dimensiones, peso y color que el primer segmento. Para situarlo en su posición lo haremos indicando que esta posición es relativa al primer segmento de la extremidad. Por lo tanto únicamente debemos bajarlo para situarlo justo debajo del final del anterior segmento.

Una vez creada la primera extremidad entera, unimos las dos partes mediante una articulación que permita el movimiento entre ambas.

```
<!--union segmentos de las extremidades -->
<Joint name="Arm01" type="hinge">
  <Body>Arm0</Body>
  <Body>Arm01</Body>
  <offsetfrom>Arm0</offsetfrom>
  <weight>4</weight>
  <limitsdeg>-60 60</limitsdeg>
  <axis>0 0 1</axis>
  <maxveldeg>15.0</maxveldeg>
  <resolution>1</resolution>
  <anchor>0.0 -3.5 0.0</anchor>
</Joint>
```

Figura 15: XML unión de los segmentos de la primera extremidad del robot_2p2

Indicamos mediante los elementos <Body> que la unión es entre los dos segmentos de la extremidad y que el movimiento se realizará relativo a la primera parte de la extremidad desde su parte inferior. Con <axis> fijamos que el movimiento entre las dos partes de la extremidad se realice únicamente en el eje ‘z’. Como se

explica en el apartado anterior, con <anchor> indicamos que el movimiento será relativo a la posición en la que se unen los dos cilindros que forman la extremidad. Si no se incluye este elemento el movimiento sería relativo al centro de gravedad del primer segmento (ya que es el indicado en <offsetfrom>) por ello con <anchor> indicamos que el punto estará (según las dimensiones de la imagen anterior) a -3,5 metros en el eje 'y' del centro de gravedad (el centro de gravedad está en el centro del cilindro, su longitud es de 7 metros, por lo que lo bajamos 3,5 metros). También indicamos el peso, los límites del movimiento en grados y la máxima velocidad.

Posteriormente crearemos la segunda extremidad, que al igual que la anterior estará formada por dos elementos cilíndricos, uno debajo del otro, que formarán la extremidad, unidos entre ellos por una articulación, y con otra articulación para unir la extremidad a la base.

```
<!-- the second movable link-->
   <Body name="Arm1" type="dynamic">
     <offsetfrom>Base</offsetfrom>
     <!-- Translation relative to Base-->
     <Translation>0.0 -4.0 4.3</Translation>
     <Geom type="cylinder">
   <rotationaxis>0 0 0 0</rotationaxis>
       <radius>0.7</radius>
       <height>7.0</height>
       <diffuseColor>1 1 1</diffuseColor>
     </Geom>
     <Mass type="mimicgeom">
       <total>10.0</total>
     </Mass>
   </Body>
   <Joint name="Arm1" type="hinge">
     <Body>Base</Body>
     <Body>Arm1</Body>
     <offsetfrom>Base</offsetfrom>
     <weight>4</weight>
     <limitsdeg>-60 60</limitsdeg>
     <axis>0 0 1</axis>
     <maxveldeg>15.0</maxveldeg>
     <resolution>1</resolution>
     <anchor>0.0 0.0 4.3</anchor>
   </Joint>
```

Figura 16: XML primer segmento de la segunda extremidad del robot_2p2

```
<!--Segmento 2 de la extremidad-->
<Body name="Arm11" type="dynamic">
  <offsetfrom>Arm1</offsetfrom>
  <Translation>0.0 -7.0 0.0</Translation>
  <Geom type="cylinder">
<rotationaxis>0 0 0 0</rotationaxis>
    <radius>0.7</radius>
    <height>7.0</height>
    <diffuseColor>1 1 1</diffuseColor>
  </Geom>
  <Mass type="mimicgeom">
    <total>10.0</total>
  </Mass>
</Body>
<!--union segmentos de las patas -->
<Joint name="Arm11" type="hinge">
  <Body>Arm1</Body>
  <Body>Arm11</Body>
  <offsetfrom>Arm1</offsetfrom>
  <weight>4</weight>
  <limitsdeg>-60 60</limitsdeg>
  <axis>0 0 1</axis>
  <maxveldeg>15.0</maxveldeg>
  <resolution>1</resolution>
  <anchor>0.0 -3.5 0.0</anchor>
</Joint>

 </KinBody>
</Robot>
```

Figura 17: XML segundo segmento y articulación de la segunda extremidad del robot _2p2

Vemos que el primer segmento es igual que el de la anterior extremidad pero colocado en el lado contrario de la base del robot. El segundo segmento se define como en la otra extremidad, sólo que relativo al primer segmento de esta segunda pata. La primera articulación es entre la base y el primer segmento de la extremidad y la segunda articulación es para unir los dos miembros de la extremidad.

4. 3. 4. Robot tres extremidades y un grado de libertad.

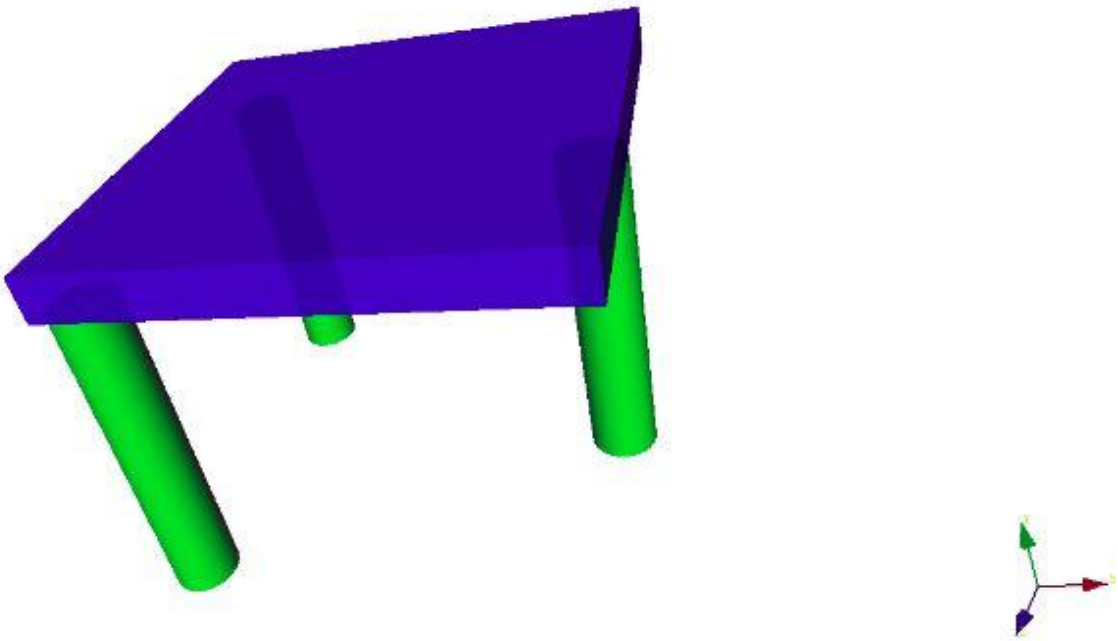

Figura 18: Robots de tres extremidades y un grado de libertad

El modelo de este robot está formado por un rectángulo como base y tres extremidades que dispondrán de un grado de libertad cada una. Dos de las extremidades estarán situadas en las esquinas de uno de los lados, y la otra irá en el centro del lado opuesto como se aprecia en la imagen.

```
<Robot name="robot_3p">
  <KinBody>
    <Body name="Base" type="dynamic">
      <Translation>0.0 0.0 0.0</Translation>
      <Geom type="box">
    <Extents>5.0 0.5 5.0</Extents>
    <RotationAxis>0 0 0 0</RotationAxis>
        <diffuseColor>0 1 0</diffuseColor>
      </Geom>
      <Mass type="box">
        <total>1.0</total>
      </Mass>
    </Body>

    <!-- the first movable link-->
    <Body name="Arm0" type="dynamic">
      <offsetfrom>Base</offsetfrom>
      <Translation>-4.0 -4.5 -4.0</Translation>
      <Geom type="cylinder">
    <rotationaxis>0 0 0 0</rotationaxis>
        <radius>1.0</radius>
        <height>8.0</height>
        <diffuseColor>0 0 1</diffuseColor>
      </Geom>
      <Mass type="mimicgeom">
        <total>10.0</total>
      </Mass>
    </Body>

    <Joint name="Arm0" type="hinge">
      <Body>Base</Body>
      <Body>Arm0</Body>
      <offsetfrom>Base</offsetfrom>
      <weight>4</weight>
      <limitsdeg>-60 60</limitsdeg>
      <axis>0 0 1</axis>
      <maxveldeg>50.0</maxveldeg>
      <resolution>1</resolution>
      <anchor>-4.0 0.0 -4.0</anchor>
    </Joint>
```

Figura 19: XML primera extremidad robot_3p

El nombre del robot es *robot_3p*, como en los casos anteriores creamos la base como un rectángulo situado en el centro del entorno y la primera extremidad como un cilindro situada, en este caso, en una de las esquinas del robot. Los unimos mediante una articulación, y a continuación creamos la segunda extremidad.

```
<!-- the second movable link-->
   <Body name="Arm1" type="dynamic">
     <offsetfrom>Base</offsetfrom>
     <!-- Translation relative to Base-->
     <Translation>-4.0 -4.5 4.0</Translation>
     <Geom type="cylinder">
   <rotationaxis>0 0 0 0</rotationaxis>
       <radius>1.0</radius>
       <height>8.0</height>
       <diffuseColor>0 0 1</diffuseColor>
     </Geom>
     <Mass type="mimicgeom">
       <total>10.0</total>
     </Mass>
   </Body>
   <Joint name="Arm1" type="hinge">
     <Body>Base</Body>
     <Body>Arm1</Body>
     <offsetfrom>Base</offsetfrom>
     <weight>4</weight>
     <limitsdeg>-60 60</limitsdeg>
     <axis>0 0 1</axis>
     <maxveldeg>50.0</maxveldeg>
     <resolution>1</resolution>
     <anchor>-4.0 0.0 4.0</anchor>
   </Joint>
```

Figura 20: XML segunda extremidad robot_3p

Esta segunda pata será la que se encuentre pegada a la otra esquina del robot en el mismo lado que la primera extremidad, por lo que la creamos igual que la anterior excepto por la posición en el eje ‘z’.

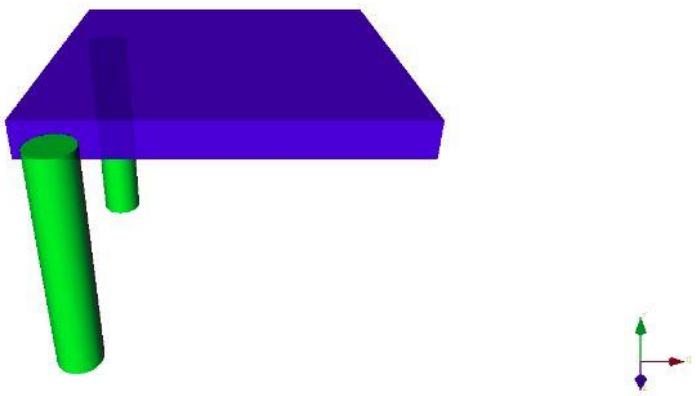

Figura 21: robot_3p con solo dos extremidades creadas

Por último creamos la última extremidad, tendrá las mismas características que las anteriores, pero colocando en el lado opuesto a las otras dos patas y en el centro, entre las dos esquinas de esas cara. Unimos la extremidad a la base y ya tenemos creado el modelo.

```
<!-- the thrid movable link-->
    <Body name="Arm2" type="dynamic">
      <offsetfrom>Base</offsetfrom>
      <!-- Translation relative to Base-->
      <Translation>4.0 -4.5 0.0</Translation>
      <Geom type="cylinder">
    <rotationaxis>0 0 0 0</rotationaxis>
        <radius>1.0</radius>
        <height>8.0</height>
        <diffuseColor>0 0 1</diffuseColor>
      </Geom>
      <Mass type="mimicgeom">
        <total>10.0</total>
      </Mass>
    </Body>
    <Joint name="Arm2" type="hinge">
      <Body>Base</Body>
      <Body>Arm2</Body>
      <offsetfrom>Base</offsetfrom>
      <weight>4</weight>
      <limitsdeg>-60 60</limitsdeg>
      <axis>0 0 1</axis>
      <maxveldeg>50.0</maxveldeg>
      <resolution>1</resolution>
      <anchor>4.0 0.0 0.0</anchor>
    </Joint>

  </KinBody>
</Robot>
```

Figura 22: XML tercera extremidad del robot_3p

4. 3. 5. Robot tres extremidades y dos grados de libertad.

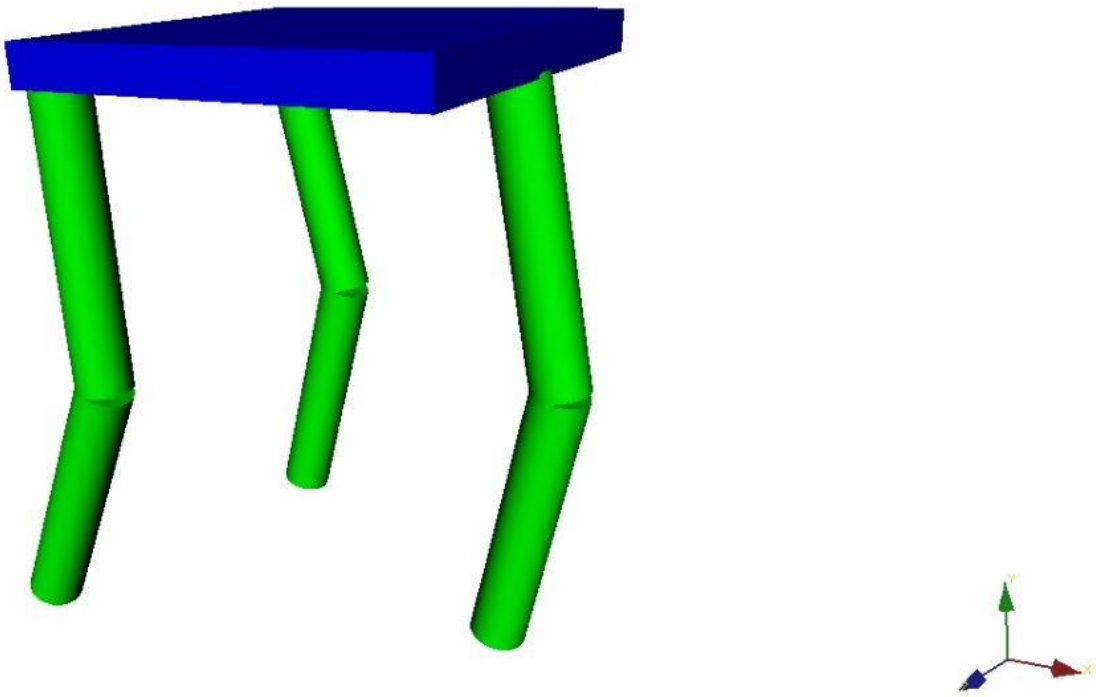

Figura 23: Robot tres extremidades y dos grados de libertad

Esta configuración es como el modelo anterior, pero al cual se le incluye un grado más de libertad a cada una de las extremidades. Por ello, el XML será igual que el anterior pero incluyendo a cada una de las tres patas un segundo segmento, colocado justo debajo del anterior y unidos entre sí por una articulación. Cada uno de estos nuevos segmentos se crean de la misma manera, únicamente se modifica el <offsetfrom> para que cada uno haga referencia a su extremidad correspondiente.

A continuación tenemos la parte del XML que correspondería al segundo segmento de una de las extremidades.

```
<!--Segmento 2 de la pata-->
<Body name="Arm01" type="dynamic">
  <offsetfrom>Arm0</offsetfrom>
  <Translation>0.0 -7.0 0.0</Translation>
  <Geom type="cylinder">
<rotationaxis>0 0 0 0</rotationaxis>
    <radius>0.7</radius>
    <height>7.0</height>
    <diffuseColor>0 1 0</diffuseColor>
  </Geom>
  <Mass type="mimicgeom">
    <total>1.0</total>
  </Mass>
</Body>
<!--union segmentos de las patas -->
<Joint name="Arm01" type="hinge">
  <Body>Arm0</Body>
  <Body>Arm01</Body>
  <offsetfrom>Arm0</offsetfrom>
  <weight>4</weight>
  <limitsdeg>-60 60</limitsdeg>
  <axis>0 0 1</axis>
  <maxveldeg>5.0</maxveldeg>
  <resolution>1</resolution>
  <anchor>0.0 -3.5 0.0</anchor>
</Joint>
```

Figura 24: XML segundo segmento de una extremidad del robot_3p2

4. 3. 6. Robot cuatro extremidades y un grado de libertad.

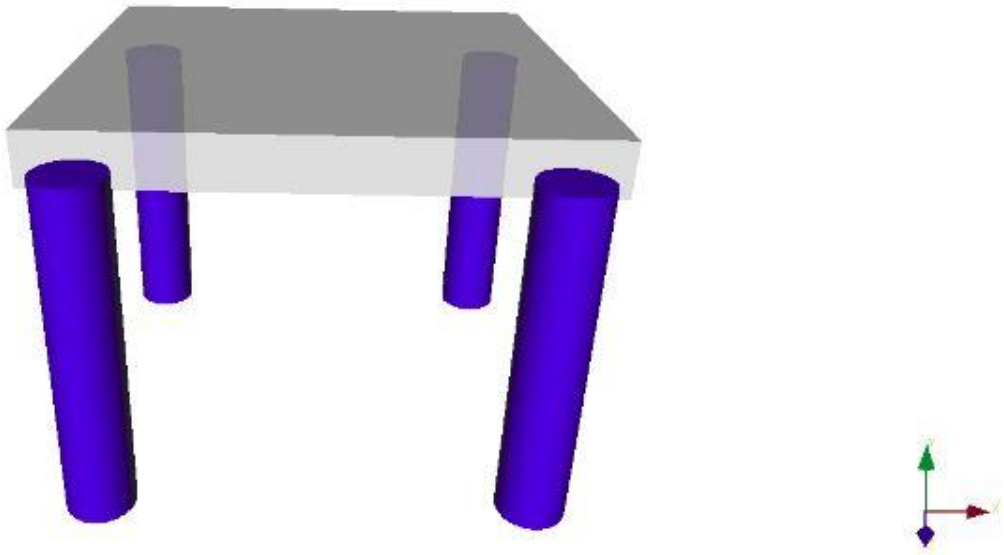

Figura 25: Robot cuatro extremidades y un grado de libertad

El quinto modelo está compuesto por una base a la cual van a estar unidas cuatro extremidades con un grado de libertad cada una. Las extremidades se crean del mismo modo a las realizadas en los demás modelos. En esta ocasión, al disponer de cuatro extremidades, las colocaremos en cada una de las esquinas del robot como se muestra en la ilustración anterior.

En la siguiente figura mostramos la creación del robot con una de sus patas, las otras cuatro tendrán las mismas características, únicamente varía <Translation>, para colocarlas en las distintas esquinas de la base y <anchor> para fijar el punto desde donde se realiza el movimiento.

```
<Robot name="robot_4p">
  <KinBody>
    <Body name="Base" type="dynamic">
      <Translation>0.0 0.0 0.0</Translation>
      <Geom type="box">
    <Extents>5.0 0.5 5.0</Extents>
    <RotationAxis>0 0 0 0</RotationAxis>
        <diffuseColor>1 1 1</diffuseColor>
      </Geom>
      <Mass type="box">
        <total>10.0</total>
      </Mass>
    </Body>

    <!-- the first movable link-->
    <Body name="Arm0" type="dynamic">
      <offsetfrom>Base</offsetfrom>
      <Translation>-4.3 -4.0 -4.3</Translation>
      <Geom type="cylinder">
    <rotationaxis>0 0 0 0</rotationaxis>
        <radius>0.7</radius>
        <height>7.0</height>
        <diffuseColor>0 0 1</diffuseColor>
      </Geom>
      <Mass type="mimicgeom">
        <total>10.0</total>
      </Mass>
    </Body>

    <Joint name="Arm0" type="hinge">
      <Body>Base</Body>
      <Body>Arm0</Body>
      <offsetfrom>Base</offsetfrom>
      <weight>4</weight>
      <limitsdeg>-60 60</limitsdeg>
      <axis>0 0 1</axis>
      <maxveldeg>1.0</maxveldeg>
      <resolution>1</resolution>
      <anchor>-4.3 0.0 -4.3</anchor>
    </Joint>
```

Figura 26: XML primera extremidad del robot_4p

4. 3. 7. Robot cuatro extremidades y dos grados de libertad, tipo mesa.

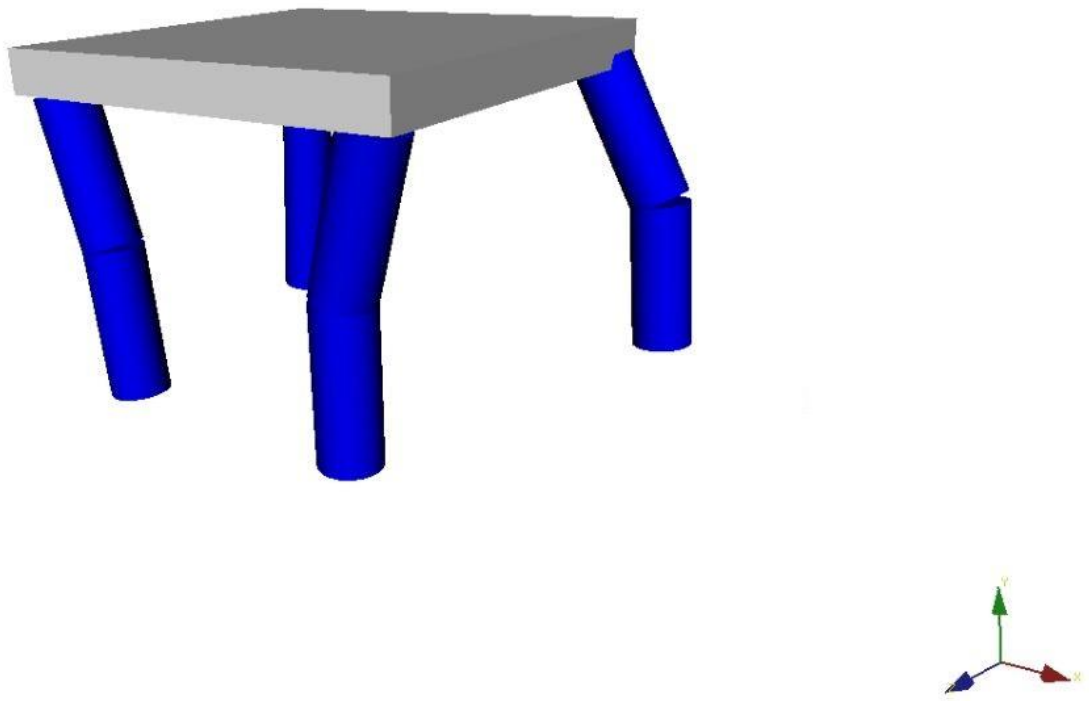

Figura 27: Robot cuatro extremidades y dos grados de libertad, tipo mesa

Los dos últimos modelos son robots con cuatro extremidades y dos grados de libertad por extremidad, pero se diferenciarán en la colocación de las extremidades. Este modelo se basa en la anterior configuración de cuatro extremidades, únicamente se ha añadido un grado de libertad más a cada unas de las patas, tal y como se ha hecho anteriormente en otros modelos con dos grados de libertad. Es decir, le incluimos un cilindro más debajo de cada extremidad para la obtención de ese grado más de libertad extra y los unimos mediante una articulación. En la siguiente imagen se muestra esta segunda parte de una de las cuatro extremidades y su unión a la primera parte de la extremidad.

```
<!--Segmento 2 de la pata-->
<Body name="Arm01" type="dynamic">
  <offsetfrom>Arm0</offsetfrom>
  <Translation>0.0 -3.5 0.0</Translation>
  <Geom type="cylinder">
<rotationaxis>0 0 0 0</rotationaxis>
    <radius>0.7</radius>
    <height>3.5</height>
    <diffuseColor>0 0 1</diffuseColor>
  </Geom>
  <Mass type="mimicgeom">
    <total>2.0</total>
  </Mass>
</Body>
<!--union segmentos de las patas -->
<Joint name="Arm01" type="hinge">
  <Body>Arm0</Body>
  <Body>Arm01</Body>
  <offsetfrom>Arm0</offsetfrom>
  <weight>4</weight>
  <limitsdeg>-60 60</limitsdeg>
  <axis>0 0 1</axis>
  <maxveldeg>5.0</maxveldeg>
  <resolution>1</resolution>
  <anchor>0.0 -1.75 0.0</anchor>
</Joint>
```

Figura 28: XML segundo segmento robot_4p2

4. 3. 8. Robot cuatro extremidades y dos grados de libertad, tipo araña.

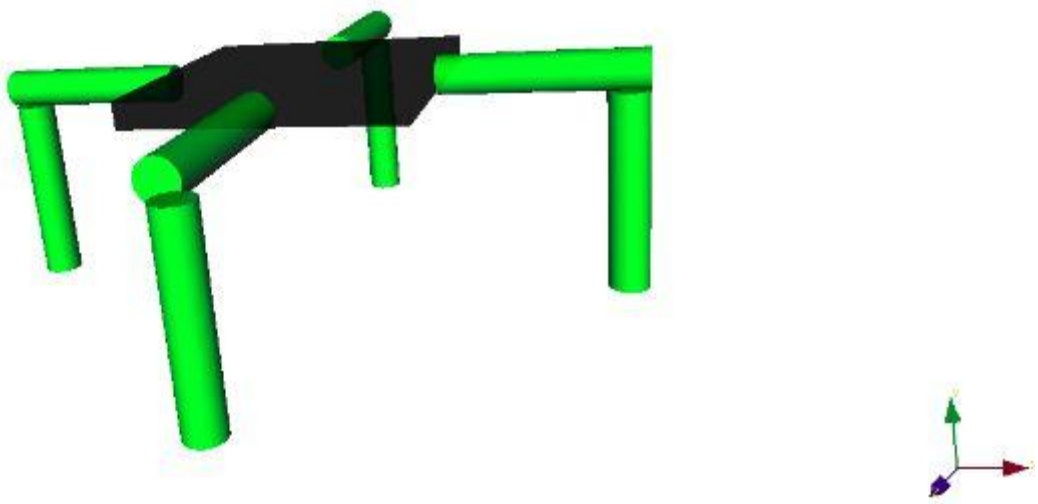

Figura 29: Robot cuatro extremidades y dos grados de libertad, tipo araña

Como último modelo tenemos la segunda variación del robot con cuatro extremidades y dos grados de libertad por extremidad. Este robot también está compuesto por una base y cuatro patas, pero estas no parten desde debajo de la base del robot, sino que el primero de los segmentos de cada pata se une a la base por una sus caras verticales, como se muestra en la ilustración anterior, y se coloca en posición horizontal. Al final de este primer segmento le unimos el segundo segmento, en esta ocasión colocado en posición vertical.

En el fichero del robot, indicamos como nombre del robot *robot_spider*. La base la creamos como hemos hecho hasta ahora, pero el primer segmento de la primera extremidad lo rotamos en torno al eje ‘x’ 90 grados mediante el elemento del XML <rotationaxis> y así conseguimos colocar este primer segmento de la en posición horizontal. Trasladamos su posición para colocarlo en uno de los lados del robot y creamos una articulación para unirlo a la base. Mediante el elemento <axis> de la articulación indicamos que el movimiento se hará sobre el eje ‘x’.

```
<Robot name="robot_spider">
  <KinBody>
    <Body name="Base" type="dynamic">
      <Translation>0.0 0.0 0.0</Translation>
      <Geom type="box">
    <Extents>5.0 0.5 5.0</Extents>
    <RotationAxis>0 0 0 0</RotationAxis>
        <diffuseColor>0 0 1</diffuseColor>
      </Geom>
      <Mass type="box">
        <total>1.0</total>
      </Mass>
    </Body>

    <!-- FIRST LEG-->
    <Body name="Arm0" type="dynamic">
      <offsetfrom>Base</offsetfrom>
      <Translation>0.0 0.0 7.25</Translation>
      <Geom type="cylinder">
    <rotationaxis>1 0 0 90</rotationaxis>
        <radius>0.7</radius>
        <height>4.5</height>
        <diffuseColor>0 1 0</diffuseColor>
      </Geom>
      <Mass type="mimicgeom">
        <total>1.0</total>
      </Mass>
    </Body>
    <!--union pata con base -->
    <Joint name="Arm0" type="hinge">
      <Body>Base</Body>
      <Body>Arm0</Body>
      <offsetfrom>Base</offsetfrom>
      <weight>4</weight>
      <limitsdeg>-60 60</limitsdeg>
      <axis>1 0 0</axis>
      <maxveldeg>10.0</maxveldeg>
      <resolution>1</resolution>
      <anchor>0.0 0.0 3.625</anchor>
    </Joint>
```

Figura 30: XML base y primer segmento de la primera extremidad del robot_spider

En la Figura 31 se muestra como queda este primer segmento de una de las extremidades:

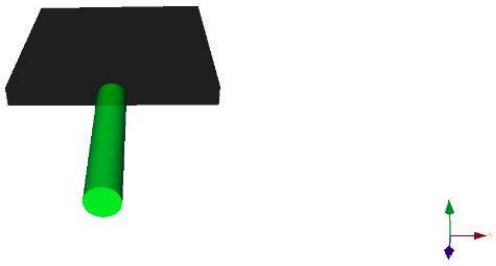

Figura 31: robot_spider con el primer segmento de una extremidad

A continuación se crea el segundo segmento de la primera extremidad. Este segundo segmento (*Arm01)* se coloca debajo del extremo del segmento anterior en posición vertical, es decir, en este caso no hace falta rotar la pieza. Además creamos una articulación entre ambos para permitir el movimiento entre las dos partes de la extremidad.

```
<!--Segmento 2 de la pata-->
<Body name="Arm01" type="dynamic">
  <offsetfrom>Arm0</offsetfrom>
  <Translation>0.0 -2.95 1.55</Translation>
  <Geom type="cylinder">
<rotationaxis>1 0 0 0</rotationaxis>
    <radius>0.7</radius>
    <height>4.5</height>
    <diffuseColor>0 1 0</diffuseColor>
  </Geom>
  <Mass type="mimicgeom">
    <total>1.0</total>
  </Mass>
</Body>
<!--union segmentos de las patas -->
<Joint name="Arm01" type="hinge">
  <Body>Arm0</Body>
  <Body>Arm01</Body>
  <offsetfrom>Arm0</offsetfrom>
  <weight>4</weight>
  <limitsdeg>-60 60</limitsdeg>
  <axis>1 0 0</axis>
  <maxveldeg>10.0</maxveldeg>
  <resolution>1</resolution>
  <anchor>0.0 0.0 1.55</anchor>
</Joint>
```

Figura 32: XML segundo segmento de la primera extremidad del robot_spider

Se observa como se ha creado la segunda parte de la extremidad desde la parte anterior, permitiendo el giro sobre el eje ‘x’.

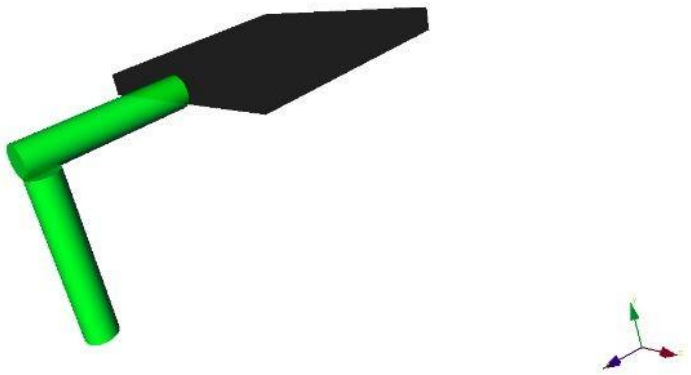

Figura 33: robot_spider con una extremidad creada

Creamos una segunda extremidad, con sus dos segmentos, pero esta vez en una de las caras perpendiculares a la cara donde creamos la anterior extremidad. Lo creamos de la misma forma que anteriormente, giramos el primer trozo de la extremidad 90 grados pero en esta ocasión lo hacemos en torno al eje ‘z’ para colocarlo en otra de las caras verticales de la base del robot, y lo trasladamos a su posición. Creamos el segundo segmento debajo del primero y la unión de los miembros de la extremidad. En las articulaciones indicaremos que este caso el giro se realizará sobre eje ‘z’.

```
<!-- SECOND LEG-->
<Body name="Arm1" type="dynamic">
  <offsetfrom>Base</offsetfrom>
  <Translation>7.25 0.0 0.0</Translation>
  <Geom type="cylinder">
<rotationaxis>0 0 1 90</rotationaxis>
    <radius>0.7</radius>
    <height>4.5</height>
    <diffuseColor>0 1 0</diffuseColor>
  </Geom>
  <Mass type="mimicgeom">
    <total>1.0</total>
  </Mass>
</Body>
<Joint name="Arm1" type="hinge">
  <Body>Base</Body>
  <Body>Arm1</Body>
  <offsetfrom>Base</offsetfrom>
  <weight>4</weight>
  <limitsdeg>-60 60</limitsdeg>
  <axis>0 0 1</axis>
  <maxveldeg>10.0</maxveldeg>
  <resolution>1</resolution>
  <anchor>3.625 0.0 0.0</anchor>
</Joint>
```

Figura 34: XML primer segmento de la segunda extremidad del robot_spider

```
<!--Segmento 2 de la pata-->
<Body name="Arm11" type="dynamic">
  <offsetfrom>Arm1</offsetfrom>
  <Translation>1.55 -2.95 0.0</Translation>
  <Geom type="cylinder">
<rotationaxis>1 0 0 0</rotationaxis>
    <radius>0.7</radius>
    <height>4.5</height>
    <diffuseColor>0 1 0</diffuseColor>
  </Geom>
  <Mass type="mimicgeom">
    <total>1.0</total>
  </Mass>
</Body>
<!--union segmentos de las patas -->
<Joint name="Arm11" type="hinge">
  <Body>Arm1</Body>
  <Body>Arm11</Body>
  <offsetfrom>Arm1</offsetfrom>
  <weight>4</weight>
  <limitsdeg>-60 60</limitsdeg>
  <axis>0 0 1</axis>
  <maxveldeg>10.0</maxveldeg>
  <resolution>1</resolution>
  <anchor>1.55 0.0 0.0</anchor>
</Joint>
```

Figura 35: XML segundo segmento de la segunda extremidad del robot_spider

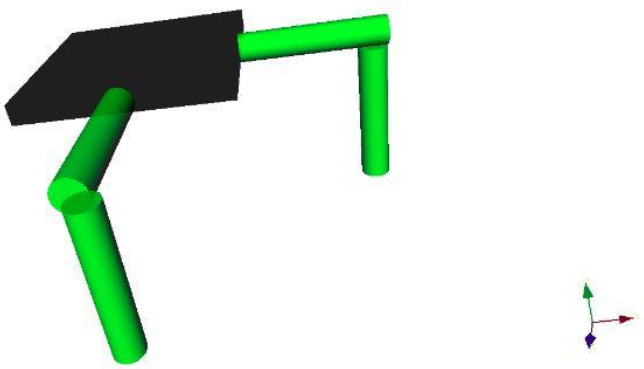

Figura 36: robot_spider con dos extremidades creadas

Las dos siguientes extremidades las creamos igual que las ya creadas pero colocándolas en los lados opuestos de la base. El primer segmento de la tercera pata está girada 90 grados respecto al eje 'x', pero la trasladamos al lado contrario de lo que hicimos con la primera pata. Y la cuarta pata la situamos en el lado apuesto a segunda de las extremidades.

```
<!-- THIRD LEG-->
<Body name="Arm2" type="dynamic">
  <offsetfrom>Base</offsetfrom>
  <Translation>0.0 0.0 -7.25</Translation>
  <Geom type="cylinder">
<rotationaxis>1 0 0 90</rotationaxis>
    <radius>0.7</radius>
    <height>4.5</height>
    <diffuseColor>0 1 0</diffuseColor>
  </Geom>
  <Mass type="mimicgeom">
    <total>1.0</total>
  </Mass>
</Body>
<Joint name="Arm2" type="hinge">
  <Body>Base</Body>
  <Body>Arm2</Body>
  <offsetfrom>Base</offsetfrom>
  <weight>4</weight>
  <limitsdeg>-60 60</limitsdeg>
  <axis>1 0 0</axis>
  <maxveldeg>10.0</maxveldeg>
  <resolution>1</resolution>
  <anchor>0.0 0.0 -3.625</anchor>
</Joint>
```

Figura 37: XML primer segmento de la tercera extremidad del robot_spider

```
<!--Segmento 2 de la pata-->
<Body name="Arm21" type="dynamic">
  <offsetfrom>Arm2</offsetfrom>
  <Translation>0.0 -2.95 -1.55</Translation>
  <Geom type="cylinder">
<rotationaxis>1 0 0 0</rotationaxis>
    <radius>0.7</radius>
    <height>4.5</height>
    <diffuseColor>0 1 0</diffuseColor>
  </Geom>
  <Mass type="mimicgeom">
    <total>1.0</total>
  </Mass>
</Body>
<!--union segmentos de las patas -->
<Joint name="Arm21" type="hinge">
  <Body>Arm2</Body>
  <Body>Arm21</Body>
  <offsetfrom>Arm2</offsetfrom>
  <weight>4</weight>
  <limitsdeg>-60 60</limitsdeg>
  <axis>1 0 0</axis>
  <maxveldeg>10.0</maxveldeg>
  <resolution>1</resolution>
  <anchor>0.0 0.0 -1.55</anchor>
</Joint>
```

Figura 38: XML segundo segmento de la tercera extremidad del robot_spider

```
<!-- FOURTH LEG-->
<Body name="Arm3" type="dynamic">
  <offsetfrom>Base</offsetfrom>
  <Translation>-7.25 0.0 0.0</Translation>
  <Geom type="cylinder">
<rotationaxis>0 0 1 90</rotationaxis>
    <radius>0.7</radius>
    <height>4.5</height>
    <diffuseColor>0 1 0</diffuseColor>
  </Geom>
  <Mass type="mimicgeom">
    <total>1.0</total>
  </Mass>
</Body>
<Joint name="Arm3" type="hinge">
  <Body>Base</Body>
  <Body>Arm3</Body>
  <offsetfrom>Base</offsetfrom>
  <weight>4</weight>
  <limitsdeg>-60 60</limitsdeg>
  <axis>0 0 1</axis>
  <maxveldeg>10.0</maxveldeg>
  <resolution>1</resolution>
  <anchor>-3.625 0.0 0.0</anchor>
</Joint>
```

Figura 39: XML primer segmento de la cuarta extremidad del robot_spider

```
<!--Segmento 2 de la pata-->
<Body name="Arm31" type="dynamic">
  <offsetfrom>Arm3</offsetfrom>
  <Translation>-1.55 -2.95 0.0</Translation>
  <Geom type="cylinder">
<rotationaxis>1 0 0 0</rotationaxis>
    <radius>0.7</radius>
    <height>4.5</height>
    <diffuseColor>0 1 0</diffuseColor>
  </Geom>
  <Mass type="mimicgeom">
    <total>1.0</total>
  </Mass>
</Body>
<!--union segmentos de las patas -->
<Joint name="Arm31" type="hinge">
  <Body>Arm3</Body>
  <Body>Arm31</Body>
  <offsetfrom>Arm3</offsetfrom>
  <weight>4</weight>
  <limitsdeg>-60 60</limitsdeg>
  <axis>0 0 1</axis>
  <maxveldeg>10.0</maxveldeg>
  <resolution>1</resolution>
  <anchor>-1.55 0.0 0.0</anchor>
</Joint>

 </KinBody>
</Robot>
```

Figura 40 XML segundo segmento de la cuarta extremidad del robot_spider

5. INTERFAZ.

5. 1. Descripción.

Para la visualización en el simulador OpenRAVE de los diferentes robots que se han creado en XML y que forman la biblioteca de elementos robóticos, tenemos una interfaz programada mediante el lenguaje Python.

El objetivo de la interfaz es cargar en OpenRAVE los modelos en función de los requerimientos del usuario. Mediante esta interfaz el usuario especificará una serie de requisitos que se le solicitarán y en función de estos requisitos, se cargará el modelo más adecuado de los disponibles y se podrá lanzar el simulador OpenRAVE. Además se podrá modificar algunas de las características del robot, como sus dimensiones o su masa, y se permitirá al usuario indicar la velocidad máxima y la posición final de las articulaciones para describir el movimiento a realizar en la simulación.

5. 2. Python.

Python es un lenguaje de programación interpretado orientado a objetos e interactivo, que usa tipado dinámico y es multiplataforma. Es administrado por la Python Software Foundation y posee una licencia en código abierto denominada Python Software Foundation License.

Fue creado a finales de los años 80 por Guido van Rossum en el Centro para las Matemáticas y la Informática (CWI, Centrum Wiskunde & Informatica), en los Países Bajos, como un sucesor del lenguaje de programación ABC.

Python es un lenguaje de programación multiparadigma. Esto significa que más que forzar a los programadores a adoptar un estilo particular de programación permite diferentes estilos: programación orientada a objetos, programación imperativa y programación funcional. Otros paradigmas están soportados mediante el uso de extensiones.

Otro objetivo del diseño del lenguaje es la facilidad de extensión. Tiene una sintaxis clara y sencilla, además es posible escribir nuevos módulos fácilmente en C o C++ y puede comunicarse por medio de intercambio de datos mediante XML. [12] [13]

5. 3. Funcionamiento y programación de la interfaz.

En este apartado se explicará el funcionamiento que nos ofrece la interfaz para la simulación de los robots en OpenRAVE y se detallarán las acciones principales del programa.

Al inicio del programa se solicitan al usuario las características básicas del robot para su configuración, que serán el número de extremidades y el número de grados de libertad de las extremidades. Con esos datos el programa cargará el fichero XML con el robot que se adecúe a esas características solicitadas.

Posteriormente mostraremos un menú en el cual indicaremos las opciones disponibles y donde el usuario indicará que opción desea ejecutar. Además, mostraremos un mensaje de aviso al usuario indicando que las especificaciones de masa, color y dimensiones actuales del robot son las que se quedaron tras la última ejecución con esa configuración. Después de realizarse la opción escogida por el usuario el programa volverá a este menú para solicitar una nueva acción.

La primera de las opciones es cambiar la configuración del robot, de esta manera si al inicio del programa el usuario se confunde introduciendo las características puede escoger otra configuración para el robot, o si ha realizado las acciones con un robot y desea probar con otro puede hacerlo.

Las opciones 2, 3 y 4 son las opciones de modificación. La opción 2 permite cambiar las dimensiones del robot. En esta opción se solicitarán las nuevas dimensiones de la base del robot: el alto, el ancho y el largo; y de las extremidades: su longitud y el radio. La opción 3 permite al usuario modificar la masa de la base del robot y de las extremidades del robot. Y la opción 4 nos permite modificar el color del robot, pudiendo escoger entre una lista de colores para la base y las extremidades.

La opción 5 nos permite la simulación del robot escogido mediante OpenRAVE. Antes de comenzar con la simulación se le solicitará al usuario que introduzca la velocidad máxima del movimiento y la posición angular final de las articulaciones para realizar el movimiento que desee el usuario. La primera vez que entremos en esta opción se lanzará una nueva ventana con el visor de OpenRAVE, y se enviará el robot. Aparecerá un mensaje en la interfaz del programa anunciando que el usuario podrá finalizar la simulación de ese robot pulsando la tecla enter. Una vez pulsado enter, se eliminará el robot del visor del entorno de OpenRAVE, pero se mantendrá la ventana de OpenRAVE por si el usuario quiere realizar alguna modificación sobre el robot y volver a probarlo, o por si quiere cambiar de robot y realizar otra simulación. Únicamente cerraremos la ventana de OpenRAVE al finalizar el programa.

La última de las opciones, la 6, es la opción para salir del programa y finalizar la ejecución.

5. 3.1. Solicitud de las características.

Esta acción es la primera acción que se realiza al iniciar el programa, además también se ejecuta cuando el usuario selecciona la opción 1 del menú, cambiar configuración del robot.

La solicitud de las características consiste en solicitar al usuario las características básicas del robot, que son el número de extremidades y el número de grados de libertad de las extremidades. El número de extremidades que puede escoger el usuario está entre 2, 3 y 4, ya que disponemos de robots modelados en XML con 2, 3 y 4 patas. Y el número de grados de libertad será 1 ó 2. Si el usuario escoge 4 extremidades y dos grados de libertad aparecerá otro mensaje solicitando el tipo de robot que se quiere, tipo mesa o tipo araña, ya que como se explicó en los apartados 4. 3. 7. Robot cuatro extremidades y dos grados de libertad, tipo mesa. y 4. 3. 8. Robot cuatro extremidades y dos grados de libertad, tipo araña., se dispone de estos dos modelos de robot con esas características.

En la Figura 41 y Figura 42 tenemos la función *escoger_robot()* que es la que se encarga de realizar esta acción.

```
#Escoger robot
def escoger_robot(opcion, extremidades, glibertad):
    opcion = "0"
    while True:
        try:
            extremidades = raw_input(" - Introduzca el número de extremidades del robot (mínimo 2 y máximo 4): ")
            if extremidades == "2" or extremidades == "3" or extremidades == "4":
                break
            else:
                print("\nEl número de extremidades debe estar entre 2 y 4.\n")
        except:
            print("\nEl número de extremidades debe estar entre 2 y 4.\n")
```

Figura 41: Solicitar número extremidades

```
    while True:
        try:
            glibertad = raw_input("\n - Introduzca el número de grados de libertad (entre 1 y 2) para las"
"extremidades del robot: ")
            if glibertad == "1" or glibertad == "2":
                break
            else:
                print("\nEl número de grados de libertad debe ser 1 ó 2.")
        except:
            print("\nEl número de grados de libertad debe ser 1 ó 2.")
    if extremidades == "4" and glibertad == "2":
        print("\nHay dos tipos de robots con esas características: \n")
        print(" 1 - Robot tipo mesa.")
        print(" 2 - Robot tipo araña.")
        while True:
            try:
                opcion = raw_input("\nEscoja una de la dos opciones: ")
                if opcion == "1" or opcion == "2":
                    break
                else:
                    print("\nDebes escoger la opición 1 ó 2.")
            except:
                print("\nDebes escoger la opición 1 ó la opción 2.")
    return opcion, extremidades, glibertad
```

Figura 42: Solicitar número de grados de libertad

Como se puede apreciar en el código, se pide al usuario que introduzca los datos solicitados por pantalla, y se valida que los datos introducidos son correctos. El número de extremidades se guarda sobre la variable *extremidades*, el número de grados de libertad en *glibertad*, y si se tiene que escoger entre los dos tipos de robot con 4 extremidades y 2 grados de libertad, se almacena la selección en la variable *opcion*. De esta manera a lo largo del programa vamos a saber cuál ha sido la opción seleccionada por el usuario y con qué configuración estamos trabajando.

5. 3.2. Carga del fichero XML.

Después de solicitar al usuario las características básicas del robot, se procede a cargar el fichero XML correspondiente con esos requisitos seleccionados por el usuario. Es decir, mediante el número de extremidades seleccionadas y el número de grados de libertad obtenemos el archivo XML con el robot que cumple esos requisitos y lo cargamos en el programa.

Para cargar este fichero en el script lo que se hace es utilizar la librería *xml.dom.minidom*. Esta librería contiene un método para parsear todo el archivo XML, es decir, lee el fichero y analiza su estructura, la cual cargaremos sobre una variable de programa para trabajar sobre ella y así no tener que hacer modificaciones directamente sobre el fichero.

```
#Cargamos el fichero xml corresponiente con los requisitos
    import xml.dom.minidom
    if   extremidades == "2" and glibertad == "1":
        xmldoc = xml.dom.minidom.parse("robot_2p.xml")
    elif extremidades == "2" and glibertad == "2":
        xmldoc = xml.dom.minidom.parse("robot_2p2.xml")
    elif extremidades == "3" and glibertad == "1":
        xmldoc = xml.dom.minidom.parse("robot_3p.xml")
    elif extremidades == "3" and glibertad == "2":
        xmldoc = xml.dom.minidom.parse("robot_3p2.xml")
    elif extremidades == "4":
        if   glibertad == "1":
            xmldoc = xml.dom.minidom.parse("robot_4p.xml")
        elif opcion == "1":
            xmldoc = xml.dom.minidom.parse("robot_4p2.xml")
        elif opcion == "2":
            xmldoc = xml.dom.minidom.parse("robot_spider.xml")
```

Figura 43: Carga del fichero correspondiente

En la Figura 43 se muestra la parte del código en la cual realizamos esta acción. Primero importamos la librería que nos permite hacer este tratamiento mediante la sentencia *import.* A continuación, en función de los requisitos que se han solicitado al usuario, cargamos el fichero XML correspondiente con la función *xml.dom.minidom.parse("nombre_fichero.xml")* y se lo asignamos a la variable del programa *xmldoc*, que será la variable con la que trabajaremos y que contiene toda la información del XML con la configuración del robot seleccionado.

5. 3.3. Menú de opciones.

El menú de opciones es un bucle que nos muestra las opciones que tiene el usuario y que ejecuta la acción seleccionada por este. Estas seis opciones del menú se muestran después de que el usuario haya escogido el robot, y también se mostrará cada vez que se termine de realizar una de las opciones.

Antes de mostrar el menú con las opciones que dispone el usuario, se muestra la configuración que tiene en ese momento el robot escogido. Llamaremos a la función o*btener_caracteristicas_xml()* para obtener las características que buscamos. Esta función recorre la variable *xmldoc* que contiene toda la estructura del robot seleccionado y obtiene los valores de las características que posteriormente mostraremos al usuario. Recorremos la variable con la función *xmldoc.getElementsByTagName("Nombre")* con la cual nos posicionamos sobre el elementos del XML que sea *Nombre*, y con la función *fristChild.data* obtenemos su

valor. De este modo recuperamos las dimensiones de la base y de las extremidades, la masa y el color. Una vez obtenidas las características se le muestran al usuario.

```
#Esoger la acción a realizar
    while accion!= "1" and accion!= "6":
        try:
            masa_ext, masa_base, color_ext, color_base, longitud, radio, base = obtener_caracteristicas_xml()
            print("\nConfiguración actual:")
            if opcion == "0":
                if glibertad == "1":
                    print " - Robot de", extremidades, "extremidades y", glibertad, "grado de libertad por extremidad"
                elif glibertad == "2":
                    print " - Robot de", extremidades, "extremidades y", glibertad, "grados de libertad por extremidad"
            elif opcion == "1":
                print " - Robot de", extremidades, "extremidades y", glibertad, "grados de libertad por extremidad, tipo mesa"
            elif opcion == "2":
                print " - Robot de", extremidades, "extremidades y", glibertad, "grados de libertad por extremidad, tipo araña"
            print " - Dimensiones de la base: ", base
            print " - Longitud de las extremidades: ", longitud*2
            print " - Radio de las extremidades: ", radio
            print " - Masa de las extremidades: ", masa_ext
            print " - Masa de la base: ", masa_base
            print " - Color de las extremidades: ", color_ext
            print " - Color de la base: ", color_base

            print("\n\nAcciones disponibles: \n")
            print(" 1 - Configurar un nuevo robot")
            print(" 2 - Modificar las dimensiones")
            print(" 3 - Modificar la masa")
            print(" 4 - Modificar el color")
            print(" 5 - Simular con OpenRAVE")
            print(" 6 - Salir")
            accion = raw_input("\nEscoge una de las opciones: ")
```

Figura 44: Configuración actual y menú de opciones

Después de la configuración del robot se muestran las seis opciones del menú, a través de las cuales el usuario podrá ir realizando diferentes acciones hasta la finalización del programa que se realizará a través de la opción 6.

5. 3.4. Opción 1: Configurar un nuevo robot.

En esta primera opción, lo que se hace es salir del bucle del menú principal y volver al inicio del programa para solicitar de nuevo las características del nuevo robot que se desea seleccionar (ver 5. 3.1. Solicitud de las características.). Una vez configurado el nuevo robot se vuelve al menú de opciones.

5. 3.5. Opción 2: Modificar las dimensiones.

Mediante esta opción el usuario podrá modificar las dimensiones del robot, tanto de la base como de las extremidades. Lo primero será solicitar al usuario las nuevas dimensiones, posteriormente se calcularán las nuevas posiciones de las extremidades, ya que al variar las dimensiones de la base varía su posición. Una vez calculadas se modifica la variable *xmldoc* para que contenga las nuevas posiciones y dimensiones, y se sobrescribe el fichero XML del robot modificado por la variable *xmldoc*.

```
elif   accion == "2":
    longitud, radio, base = dimensiones()
    modificar_longitudes()
    modificar_xml(extremidades, glibertad, opcion)
```

Figura 45: Opción 2

En primer lugar se llama a la función *dimensiones()*, la cual se encarga de solicitar las nuevas medidas del robot. Se pide al usuario que introduzca las dimensiones tanto de la base como de las extremidades. Para la base se solicita el alto, ancho y largo; y para las extremidades su longitud y su radio.

Como se explica en el apartado 4. 3. Creación de los robots., cuando disponemos de un robot con dos grados de libertad por extremidad, lo que hacemos es crear la extremidad mediante dos cilindros unidos por una articulación. Por lo que si el usuario ha seleccionado un robot con dos grados de libertad y está modificando las dimensiones, la variable en la que guardamos la longitud de la extremidad que introduce el usuario se divide entre dos, para que la longitud total de la extremidad sumando sus dos partes sea la longitud requerida por el usuario.

Una vez tenemos las dimensiones se llama a la función *modificar_longitudes().* La primera acción que se realiza dentro de esta función es llamar a la función *posiciones()* para determinar las posiciones de las extremidades en función de las nuevas dimensiones.

Con la función *posiciones()* se calcula la posición respecto de la base en la que van a estar colocadas las extremidades. Además también calcularemos los puntos donde se unen las diferentes partes del robot (base y cada una de las extremidades; y si tiene dos grados de libertad, la unión de los dos segmentos de cada extremidad) que serán los puntos fijos (definidos en el XML en la etiqueta <anchor>) desde los cuales se producirá el movimiento. Calculamos las posiciones según la configuración del robot seleccionado y en función de las dimensiones del robot. Las dimensiones de la base están guardadas en la variables *base[x, y , z]* y las dimensiones de las

extremidades en *radio* y *longitud*. Iremos guardado las posiciones de cada extremidad en el array *pos[]* y el punto de ancla en el array *ancla[]*.

A continuación se muestra la Figura 46, es una parte de la función *posiciones()* en la cual se calculan las posiciones y los puntos de ancla de las extremidades para el robot con dos extremidades y dos grados de libertad. En la posición 0 de la variable *pos* se guarda la posición (x, y, z) del primer segmento de la primera extremidad, en la posición 1 se guarda la posición del segundo segmento, y en las posiciones 2 y 3 se guardan los datos correspondientes a la segunda extremidad. Lo mismo ocurre con la variable *ancla*, donde guardamos los datos de los punto fijos para el movimiento.

```
elif extremidades == "2" and glibertad == "2":
    pos[0] = [0.00, -(longitud/2 + base[1]), -(base[2] - radio)]
    pos[1] = [0.00,               -longitud,               0.00]
    pos[2] = [0.00, -(longitud/2 + base[1]),  (base[2] - radio)]
    pos[3] = [0.00,               -longitud,               0.00]

    ancla[0] = [0.00,        0.00, -(base[2] - radio)]
    ancla[1] = [0.00, -longitud/2,              0.00]
    ancla[2] = [0.00,        0.00,  (base[2] - radio)]
    ancla[3] = [0.00, -longitud/2,              0.00]
```

Figura 46: Calculo posiciones de las extremidades para el robot_2p2

Las posiciones y los puntos de ancla están almacenadas en arrays de tipo float (variable numérica real), y para insertarlo en el XML debemos cambiarlo a una variable cadena de texto. Las posiciones las pasamos a la variable *translation* y los puntos de ancla a la variable *anchor*.

```
for n in range (0, 8):  #Convertimos de numerico a cadena
    aux0 = pos[n,0]
    aux0 = str(aux0)
    aux1 = pos[n,1]
    aux1 = str(aux1)
    aux2 = pos[n,2]
    aux2 = str(aux2)

    aux  = aux0 + " " + aux1 + " " + aux2
    translation.append(aux)

for n in range (0, 8): #Convertimos de numerico a cadena
    aux0 = ancla[n,0]
    aux0 = str(aux0)
    aux1 = ancla[n,1]
    aux1 = str(aux1)
    aux2 = ancla[n,2]
    aux2 = str(aux2)

    aux  = aux0 + " " + aux1 + " " + aux2
    anchor.append(aux)
```

Figura 47: Conversión a cadena de texto

Una vez obtenidas las posiciones modificamos la variable *xmldoc* que contiene la estructura del XML para incluir las especificaciones del usuario. Con la función *getElementsByTagName("Nombre")* nos posicionamos sobre el elemento del XML que corresponda con "*Nombre",* y con la función *fristChild.data* obtenemos el valor de ese elemento, que modificaremos con los nuevos datos especificados por el usuario. De este modo, modificamos en *xmldoc* la posición de los cuerpos ("Translation"), el punto de ancla ("anchor"), las dimensiones de la base ("Extents"), el radio de las extremidades ("radius") y la longitud de las extremidades ("height").

```
#Modificar xml longitudes
def modificar_longitudes():
##Calculamos las posiciones
    posiciones()
##Convertimos las dimensiones de la base del robot de float a string
    base[0] = str(base[0])
    base[1] = str(base[1])
    base[2] = str(base[2])
##Unimos las coordenadas x,y,z de las dimensiones de la base en una unica cadena
    extents_base = base[0] + " " + base[1] + " " + base[2]
    i = 0
##Modificamos posiciones
    for n in xmldoc.getElementsByTagName("Translation"):
        n.firstChild.data = translation[i]
        i+=1
##Modificamos posicion fija de las extremidades
    i = 0
    for n in xmldoc.getElementsByTagName("anchor"):
        n.firstChild.data = anchor[i]
        i+=1
##Modificamos dimensiones de la base
    for n in xmldoc.getElementsByTagName("Extents"):
        n.firstChild.data = extents_base
##Modificamos el radio de las extremidades
    for n in xmldoc.getElementsByTagName("radius"):
        n.firstChild.data = radio
##Modificamos la longitud de las extremidades
    for n in xmldoc.getElementsByTagName("height"):
        n.firstChild.data = longitud
```

Figura 48: Modificar longitudes

Por último, una vez finalizada la función *modificar_longitudes()*, se llama a la función *modificar_xml().* Esta función se encarga de abrir el fichero XML con la configuración del robot que se está trabajando, y reescribirlo con la variable *xmldoc* que contiene la estructura del XML con los cambios realizados. Después de reescribirlo se cierra el fichero.

```
#Modificar fichero xml
def modificar_xml(extremidades, glibertad, opcion):
##Abrimos el fichero xml
    if   extremidades == "2" and glibertad == "1":
        nf = open('robot_2p.xml','w')
    elif extremidades == "2" and glibertad == "2":
        nf = open('robot_2p2.xml','w')
    elif extremidades == "3" and glibertad == "1":
        nf = open('robot_3p.xml','w')
    elif extremidades == "3" and glibertad == "2":
        nf = open('robot_3p2.xml','w')
    elif extremidades == "4":
        if   glibertad == "1":
            nf = open('robot_4p.xml','w')
        elif opcion == "1":
            nf = open('robot_4p2.xml','w')
        elif opcion == "2":
            nf = open('robot_spider.xml','w')

    nf.write(xmldoc.toxml()) #Escribimos el fichero xml
    nf.close() #Cerramos el fichero xml
```

Figura 49: Modificar fichero XML

5. 3.6. Opción 3: Modificar la masa.

Con la opción 3 el usuario tiene la opción de modificar tanto la masa de la base del robot como la masa de las extremidades. Las acciones que se realizarán son las mismas que para modificar longitudes, en primer lugar se solicitarán los datos al usuario, posteriormente se guardarán los datos en variable *xmldoc* y por último se sobrescribe el fichero del XML correspondiente.

```
elif accion == "3":
    masa_ext, masa_base = masa()
    modificar_masa(masa_ext, masa_base)
    modificar_xml(extremidades, glibertad, opcion)
```

Figura 50: Opción 3

Mediante la función *masa()* solicitamos al usuario las nuevas masas para la base del robot y las extremidades.

```
#Escoger la masa
def masa():
    os.system("clear")
    print("\nMasa de las extremidades del robot.")
    while True:
        try:
            masa_ext = float(input("\n - Introduzca la masa en kg: "))
            if  masa_ext <= 0:
                print("\nIntroduzca una masa mayor de 0.")
            else:
                break
        except:
            print("\nIntroduzca una masa válida.")
    os.system("clear")
    print("\nMasa de la base del robot.")
    while True:
        try:
            masa_base = float(input("\n - Introduzca la masa en kg: "))
            if  masa_base <= 0:
                print("\nIntroduzca una masa mayor de 0.")
            else:
                break
        except:
            print("\nIntroduzca una masa válida.")
    os.system("clear")
    return masa_ext, masa_base
```

Figura 51: Escoger masa

Posteriormente con la función *modificar_masa()* recorremos la estructura del XML que tenemos guardado en el programa y sustituimos las masas por sus nuevos valores del mismo modo que se hace para modificar las longitudes.

```
#Modificar xml masa
def modificar_masa(masa_ext, masa_base):
#Modificamos el xml correspondiente para que tenga la masa seleccionada
    primera_vez = True
    for n in xmldoc.getElementsByTagName("total"):
        if primera_vez == True:
            n.firstChild.data = masa_base
        else:
            n.firstChild.data = masa_ext
        primera_vez = False
```

Figura 52: Modificar masa

Por último modificamos el fichero XML llamando a la función *modificar_xml()* descrita en el apartado anterior.

5. 3.7. Opción 4: Modificar el color.

En esta opción el usuario será capaz de modificar el color de la base y de las extremidades del robot. En primer lugar se solicitan al usuario los colores de la base y las extremidades, posteriormente se modifica la variable *xmldoc* y por último modificamos el XML.

```
elif accion == "4":
    color_ext, color_base = color()
    modificar_color(color_ext, color_base)
    modificar_xml(extremidades, glibertad, opcion)
```

Figura 53: Opción 4

El funcionamiento es similar a la opción anterior. En primer lugar, mediante la función *color(),* se le solicitan al usuario los colores que desea, mostrando una lista con los colores disponibles. Si el color introducido no está en la lista mostrada se volverá a solicitar.

Los colores disponibles son el blanco, negro, azul, verde y rojo. Son variables definidas en el programa como en la Figura 54, siguiendo lo descrito en el apartado 4. 3. 1. Creación de los XML., donde se explica cómo a través de la variación de los tres números de cada variable se obtiene un color distinto.

```
#Colores disponibles
blanco = "1 1 1"
negro = "0 0 0"
azul = "0 0 1"
verde = "0 1 0"
rojo = "1 0 0"
```

Figura 54: Colores disponibles

Luego con la función *modificar_color()* cambiamos nuestra variable que contiene la estructura del XML y por último mediante la función *modificar_xml()* modificamos el archivo correspondiente.

```
#Modificar xml color
def modificar_color(color_ext, color_base):
    primera_vez = True
#Modificamos el xml correspondiente para que tenga el color seleccionado
    for n in xmldoc.getElementsByTagName("diffuseColor"):
        if primera_vez == True:
            n.firstChild.data = color_base
        else:
            n.firstChild.data = color_ext
        primera_vez = False
```

Figura 55: Modificar color

5. 3.8. Opción 5: Simular con OpenRAVE.

Con la opción 5 el usuario podrá simular el robot mediante OpenRAVE. La primera vez que se realiza una simulación durante una ejecución del programa se abre una nueva ventana con el visor de OpenRAVE. Esta solo se cerrará una vez finalice la ejecución del programa.

En esta opción lo que se hace es, antes de lanzar el robot al simulador, solicitar al usuario la velocidad máxima de movimiento de las articulaciones para la simulación, y la posición angular final de cada articulación. Una vez obtenidos los datos, se modifica la velocidad en el archivo XML del robot correspondiente y por último se procede a la simulación del robot.

Hay que tener en cuenta que el movimiento que se va a realizar es un movimiento coordinado, en el cual cada articulación llega a su punto final a la vez que las demás articulaciones, por lo que la velocidad no será para todas la misma, y será máxima para aquella articulación que tenga que realizar un mayor desplazamiento.

Para los robots con dos grados de libertad se solicitará por cada extremidad la posición final de cada una de las dos articulaciones, siendo siempre la primera la articulación que une base y extremidad, y la segunda la unión entre las dos partes de la extremidad.

Se solicitará la posición final de cada articulación teniendo en cuenta que en la configuración inicial de los robots las articulaciónes de encuentran a 0 grados. A continuación se muestra la Figura 56 con las configuraciones iniciales de las articulaciones de cada uno de los modelos.

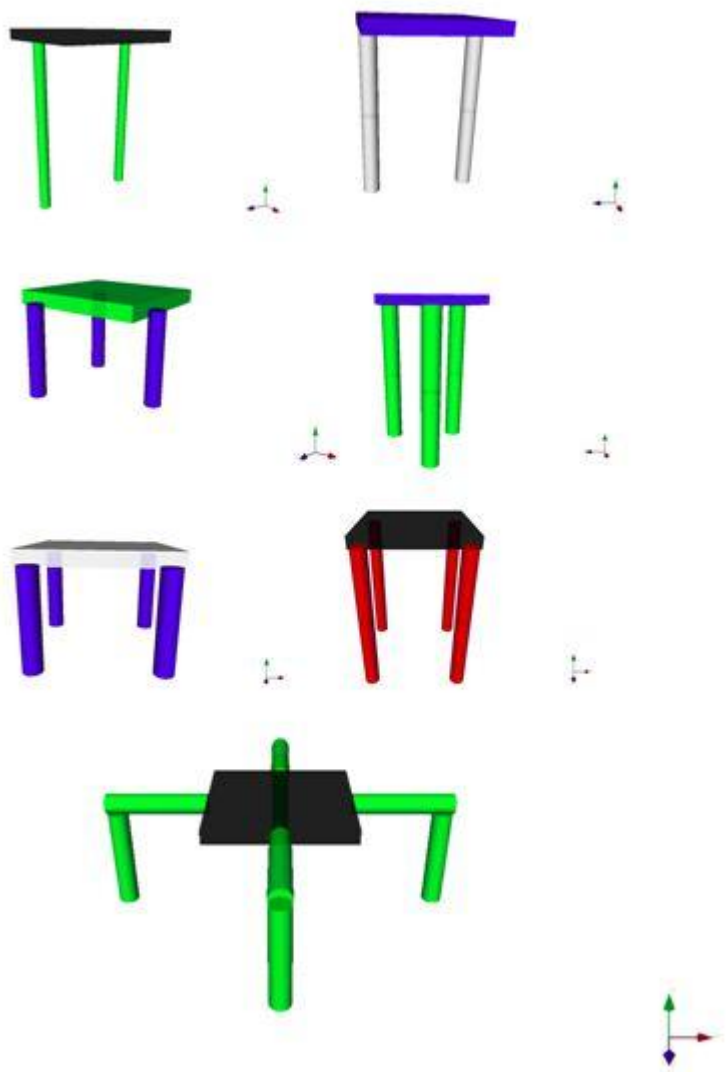

Figura 56: Configuraciones iniciales de las articulaciones de los robots

```
elif accion == "5":
    ang, vel = solicitar_datos_simulacion()
    modificar_velocidad()
    modificar_xml(extremidades, glibertad, opcion)
```

Figura 57: Opción 5

Para la solicitud de los datos de simulación se llama a la función *solicitar_datos_simulacion()*. Con esta función solicitamos al usuario la velocidad máxima de movimiento para las articulaciones en grados por segundo, y la posición angular final en grados de cada una de las articulaciones. Pasamos el ángulo

solicitado en grados a radianes, ya que será necesario pasárselo a OpenRAVE en estas unidades para realizar el movimiento.

```
while True:
    try:
        angulo = float(input("\n\t - Introduzca la posición angular final del movimiento en grados: "))
        if  angulo < -60 or angulo > 60:
            print("\nEl máximo ángulo de giro son 60 grados")
        else:
            break
    except:
        print("\nIntroduzca un ángulo válido.")
angulo = angulo * math.pi / 180 #Convertimos de grados a radianes
return angulo
```

Figura 58: Escoger posición angular final

Una vez obtenidos los datos de simulación modificamos, mediante la función *modificar_velocidad(),* la variable *xmldoc* para que contenga el nuevo dato de velocidad, y luego modificamos el fichero correspondiente con la función *modificar_xml().*

Si es la primera vez que se simula en esa ejecución del programa lo primero que hace es que crear es un entorno en OpenRAVE mediante la función *Environment()*. Una vez creado el entorno lanzamos el visor con la función *SetViewer()*.

```
if cargar_viewer == True:
    env = Environment() # create openrave environment
    env.SetViewer('qtcoin') # attach viewer
    cargar_viewer = False
```

Figura 59: Carga visor de OpenRAVE

Una vez cargado OpenRAVE, cargamos el fichero XML que contiene las características del robot con el que se esté trabajando mediante la función *env.Load("robot.xml")*. Con esta función cargamos el robot en el entorno del simulador por lo que aparecerá en la ventana de OpenRAVE. Con el robot ya en el simulador, y mediante la función *GetRobot(),* asignamos el robot una variable para poder realizar acciones sobre él. Con esta variable, se llama a la función *MoveActiveJoints que* se encargará de mover las articulaciones a la posición indicada. Esta posición estará compuesta por los ángulos finales en radianes de las articulaciones que se han solicitado al usuario.

```
#cargamos el robot
if   extremidades == "2" and glibertad == "1":
    env.Load('robot_2p.xml')
elif extremidades == "2" and glibertad == "2":
    env.Load('robot_2p2.xml')
elif extremidades == "3" and glibertad == "1":
    env.Load('robot_3p.xml')
elif extremidades == "3" and glibertad == "2":
    env.Load('robot_3p2.xml')
elif extremidades == "4":
    if   glibertad == "1":
        env.Load('robot_4p.xml')
    elif opcion == "1":
        env.Load('robot_4p2.xml')
    elif opcion == "2":
        env.Load('robot_spider.xml')
```

Figura 60: Carga del robot en OpenRAVE

```
robot = env.GetRobots()[0] #obtenemos el robot
robot.basemanip = interfaces.BaseManipulation(robot)
#Realizamos movimiento
if   extremidades == "2" and glibertad == "1":
    goal1 = [ang[0], ang[1]]
    robot.basemanip.MoveActiveJoints(goal=goal1,maxiter=3000,steplength=0.1)
elif extremidades == "2" and glibertad == "2":
    goal2 = [ang[0], ang[1], ang[2], ang[3]]
    robot.basemanip.MoveActiveJoints(goal=goal2,maxiter=3000,steplength=0.1)
elif extremidades == "3" and glibertad == "1":
    goal3 = [ang[0], ang[1], ang[2]]
    robot.basemanip.MoveActiveJoints(goal=goal3,maxiter=3000,steplength=0.1)
elif extremidades == "3" and glibertad == "2":
    goal4 = [ang[0], ang[1], ang[2], ang[3], ang[4], ang[5]]
    robot.basemanip.MoveActiveJoints(goal=goal4,maxiter=3000,steplength=0.1)
elif extremidades == "4":
    if   glibertad == "1":
        goal5 = [ang[0], ang[1], ang[2], ang[3]]
        robot.basemanip.MoveActiveJoints(goal=goal5,maxiter=3000,steplength=0.1)
    elif opcion == "1":
        goal6 = [ang[0], ang[1], ang[2], ang[3], ang[4], ang[5], ang[6], ang[7]]
        robot.basemanip.MoveActiveJoints(goal=goal6,maxiter=3000,steplength=0.1)
    elif opcion == "2":
        goal7 = [ang[0], ang[1], ang[2], ang[3], ang[4], ang[5], ang[6], ang[7]]
        robot.basemanip.MoveActiveJoints(goal=goal7,maxiter=3000,steplength=0.1)
```

Figura 61: Movimiento del robot en OpenRAVE

Una vez finalice el movimiento, en la ventana del programa, aparecerá un mensaje indicándole al usuario que presione la tecla enter cuando desee finalizar la

simulación. Una vez se pulse la tecla, se eliminará al robot del entorno de simulación de OpenRAVE mediante la función *env.Remove("robot.xml").* Una vez se elimine el robot el programa volverá al menú principal y solicitará una nueva acción al usuario.

```
#Esperemos a que termine el movimiento
while not robot.GetController().IsDone():
        time.sleep(0.01)
#
time.sleep(1)
os.system("clear")
raw_input ("\nPresione enter para finalizar la simulación.")
os.system("clear")

with env:
    env.Remove(robot) #Eliminamos robot del entorno de OpenRAVE
```

Figura 62: Eliminar robot de OpenRAVE

5. 3.9. Opción 6: Salir.

La opción 6 de salir es para finalizar la ejecución del programa. Si se ha realizado una simulación mediante la opción 5 del menú, se cerrará la ventana de OpenRAVE.

6. DISCUSIÓN DE RESULTADOS.

En este apartado se explicará por medio de un ejemplo práctico el resultado final del proyecto, mostrando el funcionamiento de la herramienta y las diferentes acciones que el usuario puede realizar. Haremos un recorrido por la herramienta accediendo a todas las opciones del menú y realizando simulaciones con alguno de los modelos robóticos de la biblioteca.

Lo primero de todo se debe indicar que para que el usuario pueda realizar las simulaciones es imprescindible tener instalado el entorno de simulación OpenRAVE en el ordenador.

Al comienzo de la ejecución del programa lo primero que la interfaz nos solicita son las características para la configuración del robot. Seleccionaremos el modelo de robot con dos extremidades y un grado de libertad.

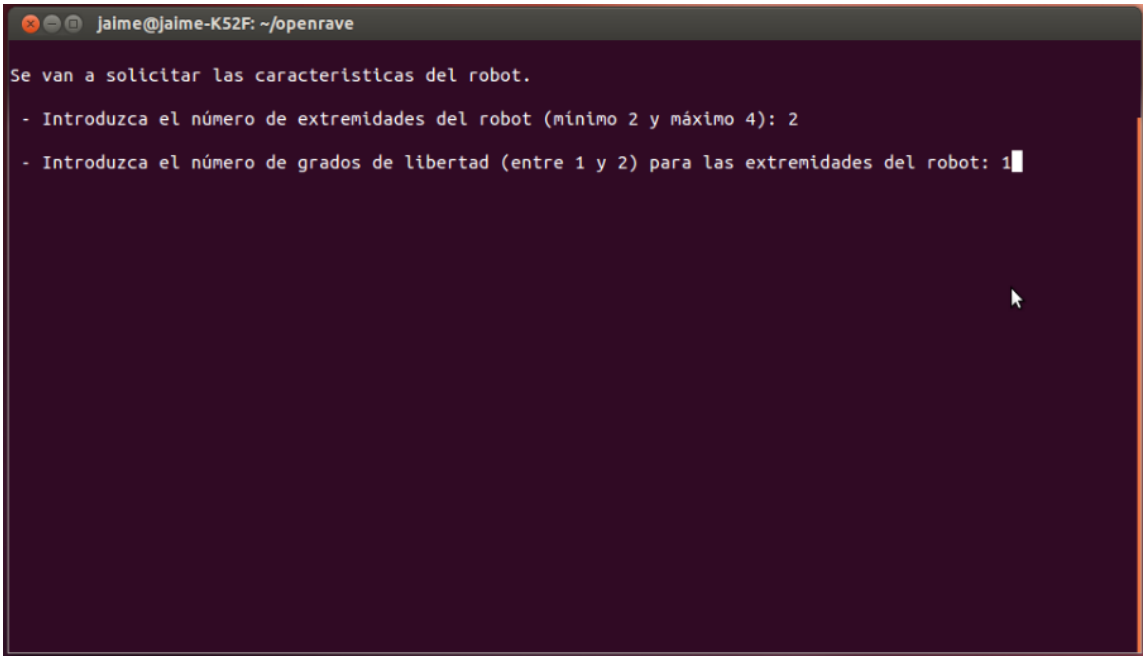

Figura 63: Inicio del programa

Posteriormente nos aparece un mensaje de aviso diciendo que las especificaciones actuales del robot son las que se quedaron tras la última ejecución del programa con ese modelo.

Se nos muestra la configuración actual del robot escogido y el menú de acciones que tenemos disponibles. Como se muestra en la Figura 64, las dimensiones de la base del robot son 10 x 0.5 x 8.5 metros, la longitud de las extremidades es de 7

metros y su radio de 0.7 metros, la masa es de10 kg tanto para la base como para cada extremidad y los colores de robot son rojo para dos extremidades y negro para la base.

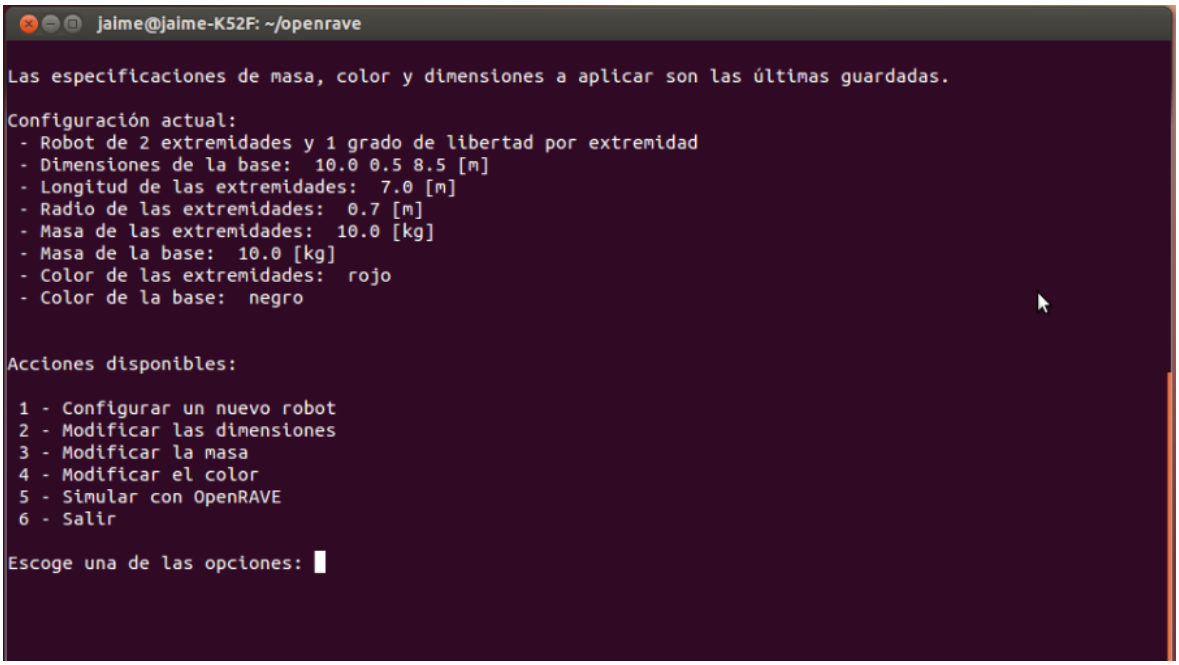

Figura 64: Menú principal

Seleccionamos la opción 5 para realizar una simulación en OpenRAVE. Se nos solicita que seleccionemos la velocidad máxima de movimiento en grados por segundo y la posición angular final para cada extremidad. Introducimos una velocidad de 10, y una posición final de 45 grados para la primera extremidad y de -30 para la segunda.

Figura 65: Opción 5, simular con OpenRAVE

Se nos abre una nueva ventana con el visor de OpenRAVE, donde nos aparece el robot y realiza el movimiento indicado, partiendo sus extremidades desde la posición inicial de cero grados y finalizando en las posiciones indicadas. Una vez terminado el movimiento nos aparece un mensaje en la interfaz diciéndonos que para finalizar la simulación presionemos la tecla enter.

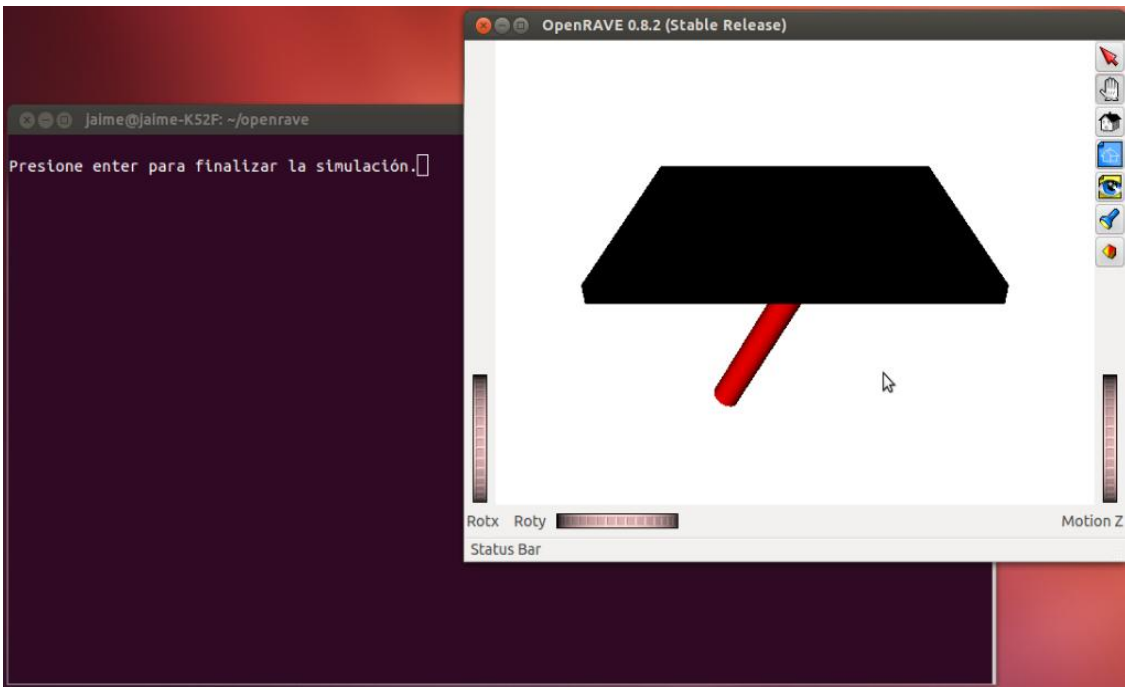

Figura 66: Simulación en OpenRAVE

Nos posicionamos de nuevo en la interfaz, pulsamos enter y volvemos al menú principal. Del visor de OpenRAVE nos desaparece el robot, pero la ventana del simulador se mantiene abierta.

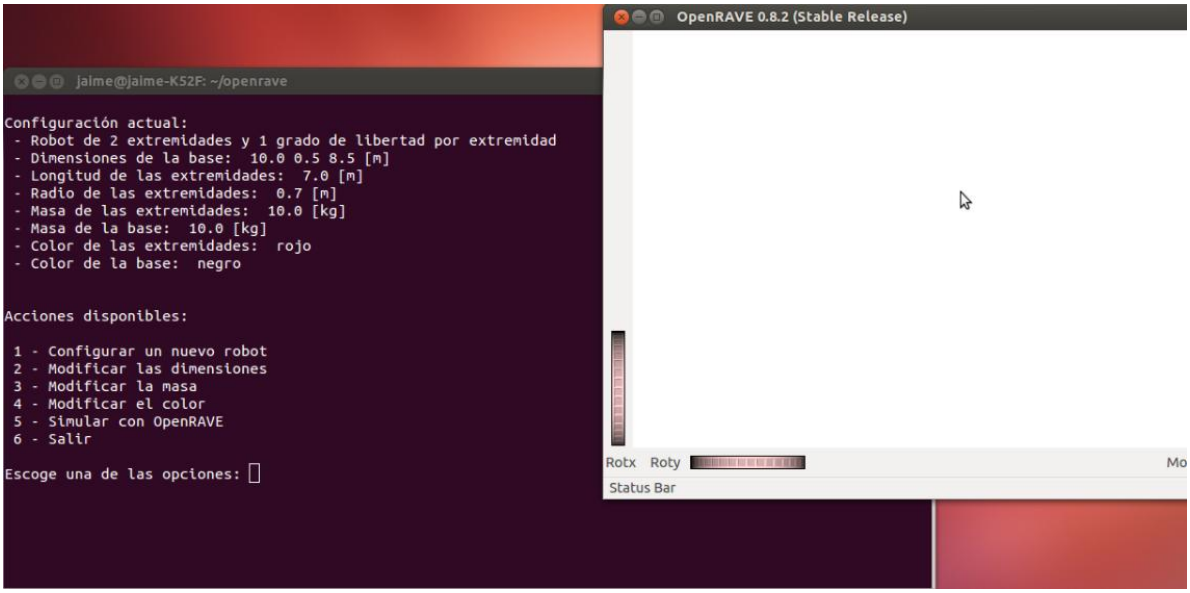

Figura 67: Después de la simulación

Ahora seleccionamos la opción 2 del menú principal, modificar dimensiones. Se nos solicitan las nuevas dimensiones tanto de la base como de las extremidades, e introducimos las que aparecen en la Figura 68.

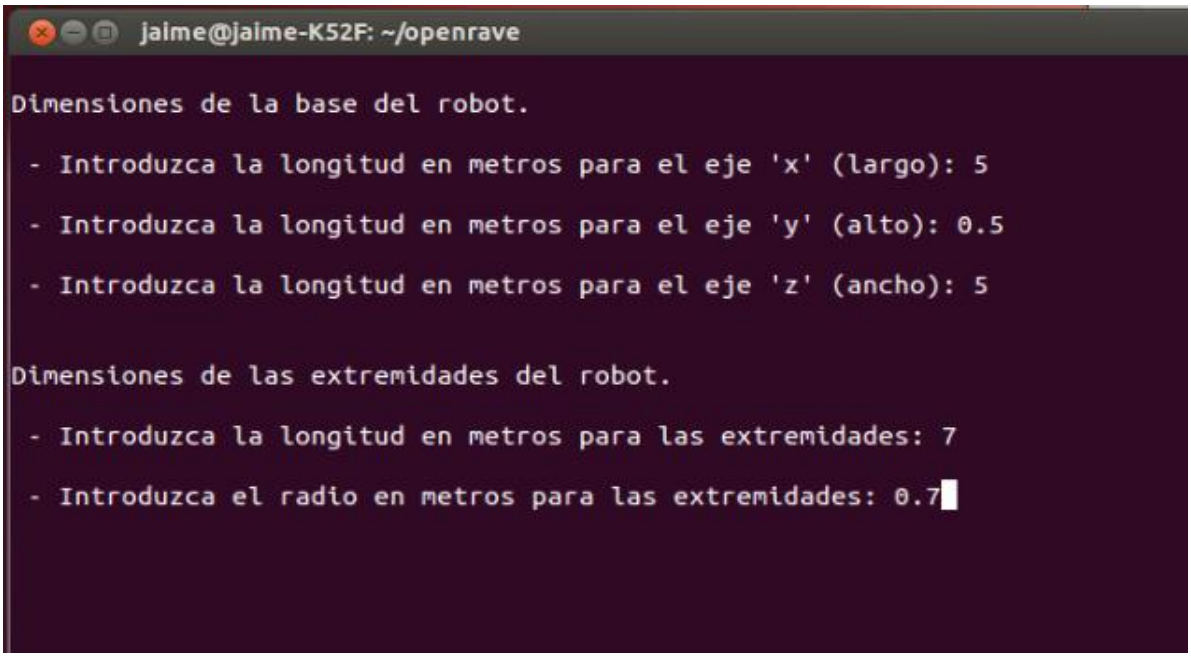

Figura 68: Opción 2, modificar longitudes

Al regresar al menú principal observamos que en los datos de la configuración actual se nos han modificado las dimensiones que teníamos anteriormente y nos aparecen las nuevas que hemos introducido.

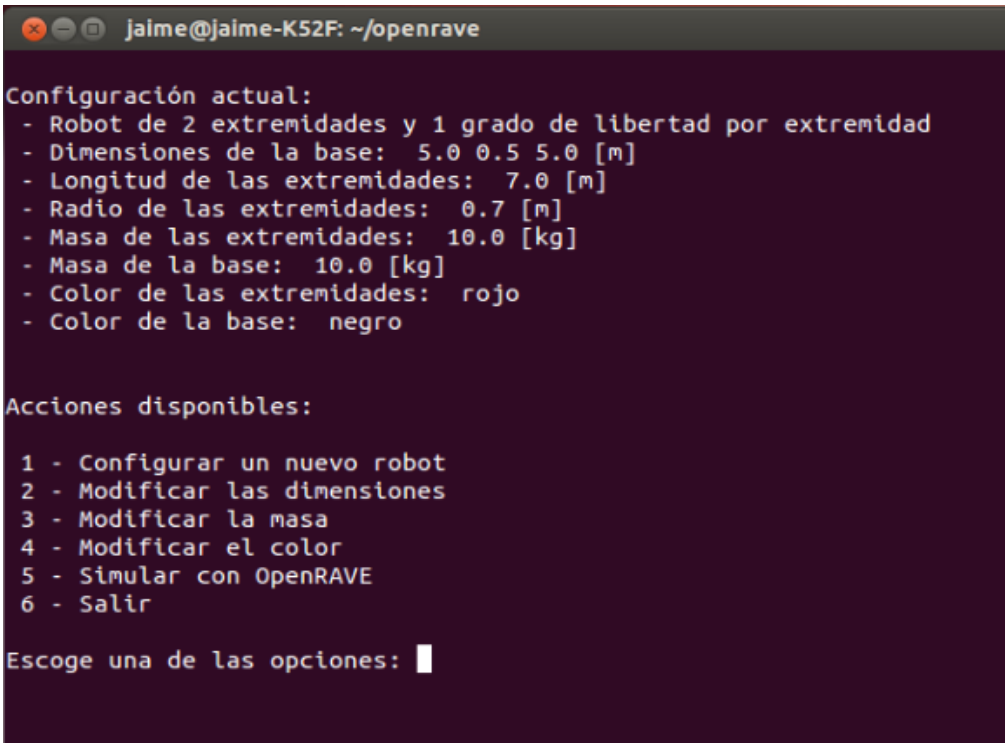

Figura 69: Longitudes modificadas

Volvemos al menú principal y seleccionamos la opción 3 para modificar la masa. Nos aparece un mensaje solicitando la masa de la base del robot y de las extremidades. Introduciremos 5 kg para cada uno.

Figura 70: Opción 3, modificar masa

Al volver al menú principal, seleccionamos la opción 4 para modificar el color del robot. Se solicita el nuevo color para la base y para las extremidades, mostrándonos una lista de colores para escoger uno de ellos.

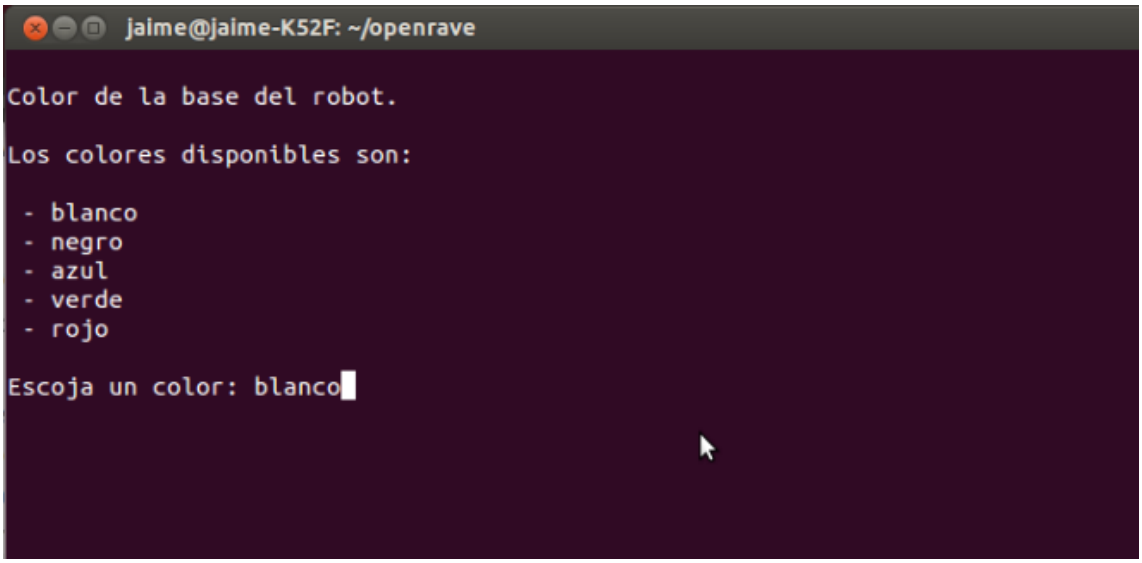

Figura 71: Opción 4, modificar color

Seleccionamos de nuevo la opción 5 y volvemos a realizar una simulación, donde nos vuelven a solicitar los datos para la simulación. Se puede observar en la ventana del simulador de la siguiente imagen cómo se ha modificado nuestro robot respecto al que teníamos en la primera simulación.

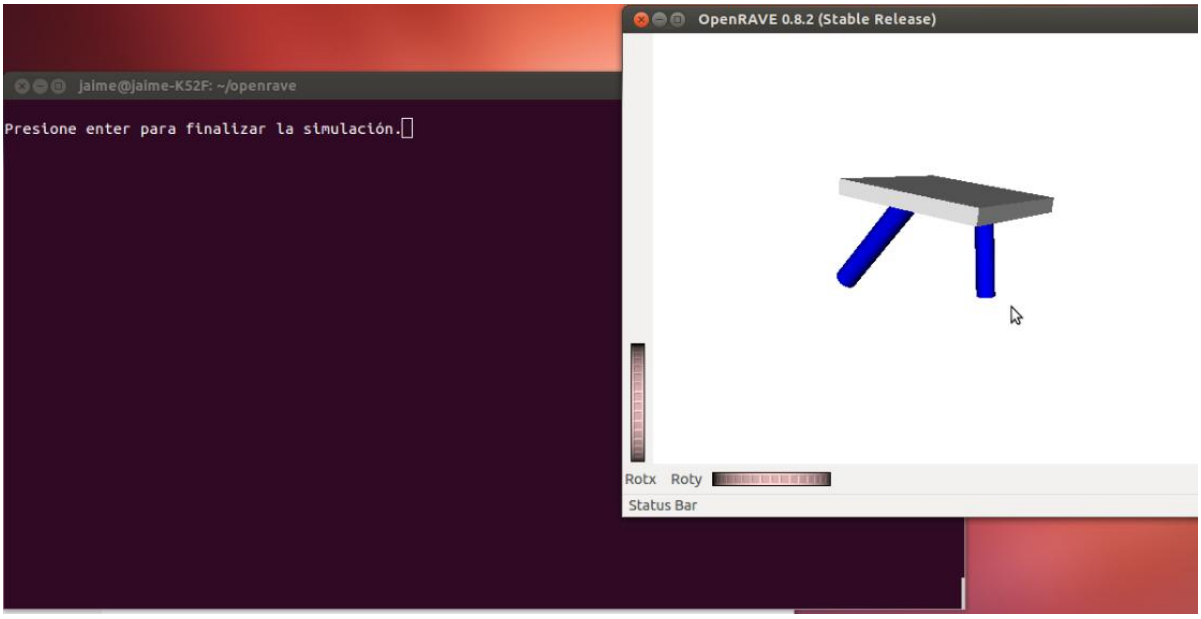

Figura 72: Segunda simulación en OpenRAVE

A continuación volvemos al menú principal y seleccionamos la opción 1, configurar un nuevo robot, que nos devuelve al inicio del programa y nos vuelve a solicitar las nuevas características para el robot. Seleccionamos 4 extremidades y 2 grados de libertad, al disponer de dos opciones, se pide que seleccionemos una de ellas, seleccionaremos la segunda opción, el robot tipo araña.

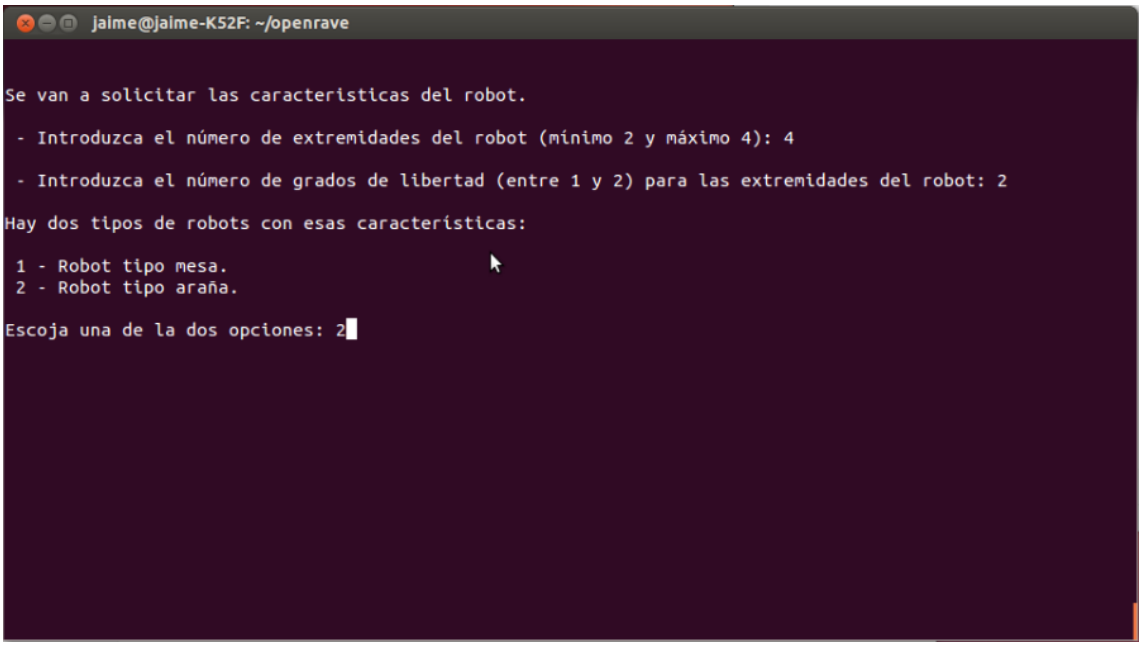

Figura 73: Opción 1, escoger nueva configuración

Volvemos al menú principal y seleccionamos la opción de simular con OpenRAVE. En este caso los datos de las posiciones angulares nos las irá pidiendo la interfaz articulación por articulación como se muestra en la Figura 74.

```
A continuación se van a solicitar los datos para la simulación.

- Introduzca la velocidad máxima de movimiento en grados por segundo: 10
- Datos de la extremidad 1 , primera articulación:
        - Introduzca la posición angular final del movimiento en grados: -30
- Datos de la extremidad 1 , segunda articulación:
        - Introduzca la posición angular final del movimiento en grados: -30
- Datos de la extremidad 2 , primera articulación:
        - Introduzca la posición angular final del movimiento en grados: 30
- Datos de la extremidad 2 , segunda articulación:
        - Introduzca la posición angular final del movimiento en grados: 30
- Datos de la extremidad 3 , primera articulación:
        - Introduzca la posición angular final del movimiento en grados: 30
- Datos de la extremidad 3 , segunda articulación:
        - Introduzca la posición angular final del movimiento en grados: 30
- Datos de la extremidad 4 , primera articulación:
        - Introduzca la posición angular final del movimiento en grados: -30
- Datos de la extremidad 4 , segunda articulación:
        - Introduzca la posición angular final del movimiento en grados: -30
```

Figura 74: Opción 5, con dos grados de libertad

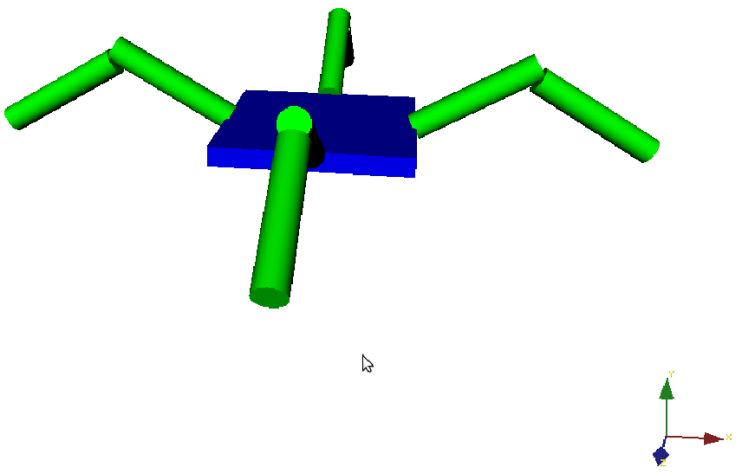

Figura 75: Tercera simulación en OpenRAVE

Una vez hayamos terminado la simulación, regresamos al menú principal y seleccionamos la opción 6 para salir del programa. Cuando salgamos del programa también se nos cerrará la ventana del simulador OpenRAVE y se dará por finalizada la ejecución de programa.

A lo largo de este apartado hemos visto como es una simulación con la herramienta creada y como hemos podido realizar diferentes simulaciones, escogiendo varios modelos y realizando varias modificaciones sobre ellos.

7. MARCO SOCIO-ECONÓMICO.

7.1 Planificación.

El proyectó se inició con la primera reunión el 27 de noviembre de 2013 donde se asentaron las bases del proyecto y se hizo una primera planificación para el desarrollo del proyecto.

El desarrollo del proyecto no ha seguido el ciclo de vida habitual de un proyecto en cascada donde primero se realiza una planificación de todo el proyecto, luego se implementa y por último se realizan las pruebas. El proyecto ha seguido un ciclo iterativo e incremental, donde el software se ha ido realizando poco a poco añadiéndole valor de modo incremental. A medida que se realizaba, también se realizaban las pruebas, y se han ido sucediendo diversas reuniones donde se han ido fijando las ampliaciones del proyecto.

El primer paso para la realización del proyecto fue realizar una primera planificación fijando unos objetivos y fechas para la finalización del proyecto.

Un segundo paso fue una etapa de documentación en la que se realizó un estudio de las herramientas a utilizar y se valoraron alternativas para el desarrollo final del proyecto.

Posteriormente se comenzó con la creación los modelos de los robots en XML. Han ido sufriendo varias modificaciones a lo largo del proyecto, por lo que no se finalizó con su creación hasta antes de las pruebas finales del proyecto.

Una vez se tuvieron los primeros modelos de los robots en XML se comenzó con el desarrollo de la interfaz. La cual como hemos comentado se fue desarrollando poco a poco de modo incremental. Se comenzó únicamente solicitando al usuario un modelo y mostrando con OpenRAVE, y ha ido evolucionando hasta acabar en el modelo final del proyecto.

A medida que se iba realizando la interfaz y los modelos de los XML iban evolucionando, se han ido realizando pruebas, por lo que estas han durado desde prácticamente el inicio de la implementación del proyecto hasta su finalización con las pruebas finales.

A continuación se mostrará una tabla con las tareas y sus fechas de inicio y fin y un diagrama de Gantt con la planificación del proyecto.

Tarea

Nombre	Fecha de inicio	Fecha de fin
Planificación	27/11/13	2/12/13
Documentación	3/12/13	13/12/13
Creación modelos XML	16/12/13	9/06/14
Creación de la interfaz	16/12/13	9/06/14
Pruebas	22/01/14	16/06/14
Elaboración de la memoria	16/12/13	22/06/14

Tabla 1: Planificación de tareas

En esta tabla se muestran las diferentes tareas a lo largo del proyecto, con sus fechas de inicio y sus fechas de fin. Vemos como las tareas creación de los XML, la interfaz y las pruebas se han ido desarrollando a la vez a lo largo del proyecto. Y como la creación de la memoria comenzó después de la documentación y se ha ido desarrollando poco a poco a lo largo de todo el proyecto, aunque la mayor parte de ella se ha realizado en las últimas fechas del proyecto.

Diagrama de Gantt

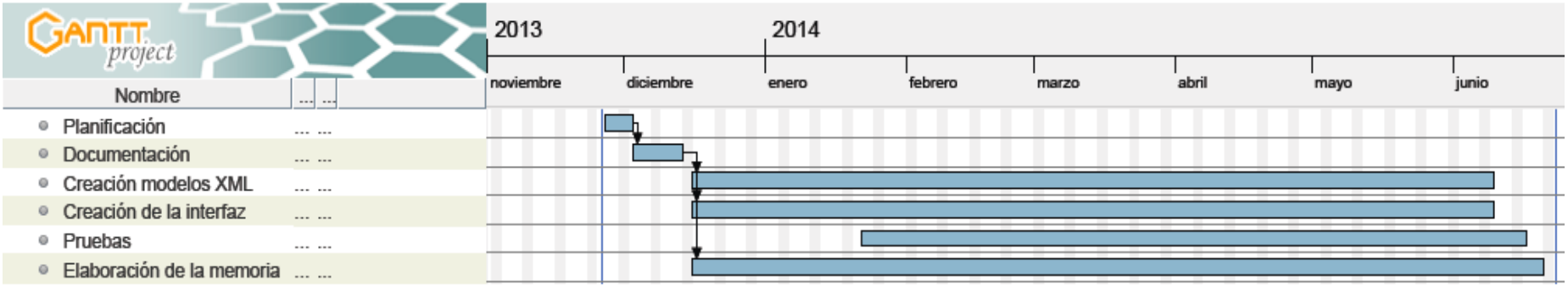

Tabla 2: Diagrama de Gantt

7.2 Presupuesto.

A continuación se detallará el presupuesto para la realización de este proyecto. Se hará un desglose en del presupuesto en función de los costes de personal, de software y de hardware, y posteriormente se mostrará un resumen con el coste total del proyecto.

Coste Personal			
Categoría	**Nº Horas**	**Coste hora**	**Coste Total**
Jefe proyecto	50 horas	35 €/hora	1750 €
Ingeniero	400 horas	10 €/hora	4000 €
		TOTAL	5750€

Tabla 3: Coste personal

Para calcular los costes de software y hardware se hace uso de la siguiente fórmula:

$$Costes = (A / B) \cdot C \cdot D$$

A = Número de meses desde la fecha de facturación en que el equipo es utilizado.

B = Periodo de depreciación.

C = Coste del equipo sin IVA

D = Porcentaje de uso que de dedica al proyecto. Se fija al 100%.

Coste Software				
Descripción	**Coste**	**Dedicación (meses)**	**Periodo de depreciación**	**Coste imputable**
Ubuntu 12.04	0 €	6 meses	60	0 €
OpenRAVE	0 €	6 meses	60	0 €
Open Office	0 €	6 meses	60	0 €
			TOTAL	0 €

Tabla 4: Coste Software

Coste Hardware				
Descripción	**Coste**	**Dedicación (meses)**	**Periodo de depreciación**	**Coste imputable**
Portátil Asus K52F	499 €	6 meses	60	49,90 €
Ratón Microsoft wireless 1000	15,99 €	6 meses	60	1,59 €
			TOTAL	51,49 €

Tabla 5: Coste Hardware

Costes Totales	
Coste Personal	5750€
Coste Software	0 €
Coste Hardware	51,49 €
Subtotal 1	**5801,49 €**
Beneficios Empresariales (25% subtotal 1)	1450,37 €
Subtotal 2	**7251,80 €**
IVA (21%)	1522,88 €
TOTAL	**8774,68 €**

Tabla 6: Costes totales

8. CONCLUSIONES Y TRABAJOS FUTUROS.

8.1. Conclusiones.

El objetivo principal del proyecto era la construcción de una biblioteca de elementos robóticos y de una interfaz a través de la cual pudiéramos modificar esa biblioteca y realizar simulaciones a través de OpenRAVE.

Una vez se ha estudiado el entorno de simulación de OpenRAVE se han desarrollado varios modelos de robots para la biblioteca en el formato propio que dispone OpenRAVE mediante XML.

Posteriormente se ha creado una interfaz para que el usuario pudiera acceder a la biblioteca y poder realizar modificaciones sobre las características de los robots. Además a través de la interfaz el usuario puede realizar simulaciones en OpenRAVE de los modelos de la biblioteca

Como ya hemos comprobado en el Capítulo 6 discusión de resultados, se han cumplido los objetivos presentados al inicio, y se ha conseguido crear una herramienta que cumple todos los requisitos y especificaciones.

8.2. Trabajos futuros.

En este aparatado trataremos las posibles mejoras que se pueden realizar sobre el proyecto y algunas ideas sobre trabajos futuros que se pueden llevar a cabo. A pesar de haber realizado un proyecto cumpliendo los objetivos planteados como se ha visto en las conclusiones del apartado anterior, siempre se pueden realizar mejoras.

En cuanto a las mejoras que se podrían hacer sobre la interfaz, una de ellas puede ser la optimización de los algoritmos creados en Python para crear una aplicación más eficaz y rápida. Otra de las mejoras que se pueden llevar a cabo es la creación de una interfaz gráfica para que sea más visual y accesible para el usuario. Otra mejora podría ser no dejar al usuario modificar las longitudes del robot si las dimensiones introducidas son ilógicas, por ejemplo si el usuario crea un robot con una base con unas dimensiones muy superiores a las de las extremidades. También se podría dar la posibilidad al usuario de realizar únicamente modificaciones sobre la base del robot o sobre las extremidades, ya que actualmente si se modifica una característica se solicitan los datos de modificación para ambos. Por último se podría ofrecer la posibilidad de, una vez realizadas las modificaciones, guardarlas o no guardarlas, puesto que ahora mismo siempre que se realiza un cambio se guarda automáticamente.

En cuanto a los robots, se podrían añadir más características a sus configuraciones en los ficheros XML, como por ejemplo introducir el máximo par a aplicar sobre sus articulaciones, la inercia de cada cuerpo rígido o la máxima aceleración; además se permitiría al usuario modificar estas características desde la interfaz.

Sobre los trabajos futuros, tengo algunas ideas que permitirían mejorar y ampliar este proyecto. La primera de ellas sería crear una biblioteca de robots más amplia, creando más configuraciones de robots, creando modelos diferentes a los actuales. También se podrían crear diferentes partes de robots (brazos, extremidades, cuerpos…) para que luego el usuario por medio de la interfaz uniese los que más le interesen.

Además de ampliar la biblioteca se podría dejar al usuario que seleccionara el tipo de geometría que viese oportuno para cada parte del robot, por ejemplo modificando la base de un robot y hacerla cilíndrica o creando las extremidades con forma cúbica.

Otra idea sería incluir entornos en la simulación de los robots, es decir, un escenario donde se pueda ver en la simulación al robot actuando en una posible situación real. Los entornos en OpenRAVE también se crean como ficheros XML por lo que se podría crear una biblioteca con distintos escenarios y dar la posibilidad al usuario para escoger entre ellos y poder simular los robots en los distintos escenarios.

Una vez se disponga de estos escenarios, se podrían utilizar los planificadores de movimiento disponibles en OpenRAVE para realizar movimientos del robot y trasladar de un punto a otro de un escenario por el camino más eficaz. También se podría permitir al usuario introducir trayectorias y ver como es el comportamiento del robot en los distintos entornos.

REFERENCIAS

[1] «http://en.wikipedia.org/wiki/Robotics_simulator,» [En línea]. [Último acceso: 12 2013].

[2] «http://es.wikipedia.org/wiki/Webots,» [En línea]. [Último acceso: 12 2013].

[3] «http://www.cyberbotics.com/overview,» [En línea]. [Último acceso: 12 2013].

[4] «http://www.gazebosim.org/,» [En línea]. [Último acceso: 12 2013].

[5] «http://www.openrtp.jp/openhrp3/en/index.html,» [En línea]. [Último acceso: 12 2013].

[6] R. Diankov y J. Kuffner, «A plannig architecture for autonomous robotics.,» 2008.

[7] «http://openrave.org/docs/latest_stable/overview/,» [En línea]. [Último acceso: 06 2014].

[8] «http://openrave.org,» [En línea]. [Último acceso: 06 2014].

[9] «http://openrave.programmingvision.com/wiki/index.php/Format:XML,» [En línea]. [Último acceso: 06 2014].

[10] «http://www.w3c.es/Divulgacion/GuiasBreves/TecnologiasXML,» [En línea]. [Último acceso: 12 2013].

[11] «http://www.learn-xml-tutorial.com/xml-basics.cfm,» [En línea]. [Último acceso: 12 2013].

[12] «http://es.wikipedia.org/wiki/Python,» [En línea]. [Último acceso: 12 2013].

[13] «https://www.python.org,» [En línea]. [Último acceso: 06 2014].

ANEXOS

Programa: programa.py

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
import os, sys, time, math
from numpy import *
from openravepy import *

os.system("clear") #Limpiar la pantalla

#Inicialización de variables de trabajo
accion       = "0" #Acción seleccionada en el menú
opcion       = "0" #Opción mesa/araña para robot de 4 ext y 2gl.
glibertad    = "0" #Número de grados de libertad por extremidad
extremidades = "0" #Número de extremidades
seleccion    = True #Control para finalizar el programa
cargar_viewer = True #Controlar si ya se ha abierto OpenRAVE

base=[] #Dimensiones de la base
base.append(0.00)
base.append(0.00)
base.append(0.00)

pos=zeros([8,3],float) #Posiciones
translation=[] #Posiciones (cadena)

ancla=zeros([8,3],float) #Ancla
anchor=[] #Ancla (cadena)

ang=zeros([8],float) #Angulos de movimiento
vel=0.0 #Velocidad max de movimiento


#Colores disponibles
blanco = "1 1 1"
negro = "0 0 0"
azul = "0 0 1"
verde = "0 1 0"
rojo = "1 0 0"


#FUNCIONES

#Calculo de las posiciones de las extremidades
def posiciones():
      #Iniciañizamos variables
      pos=zeros([8,3],float)   #posiciones de las extremidades
      ancla=zeros([8,3],float) #posición fija de la extremidad

      #Borramos listas
      del translation[0:]
      del anchor[0:]

      if   extremidades == "2" and glibertad == "1":
```

```
            pos[0] = [0.00, -(longitud/2 + base[1]), -(base[2] -
radio)]
            pos[1] = [0.00, -(longitud/2 + base[1]),  (base[2] -
radio)]

            ancla[0] = [0.00, 0.00, -(base[2] - radio)]
            ancla[1] = [0.00, 0.00,  (base[2] - radio)]

      elif extremidades == "2" and glibertad == "2":
            pos[0] = [0.00, -(longitud/2 + base[1]), -(base[2] -
radio)]
            pos[1] = [0.00,               -longitud,
0.00]
            pos[2] = [0.00, -(longitud/2 + base[1]),  (base[2] -
radio)]
            pos[3] = [0.00,               -longitud,
0.00]

            ancla[0] = [0.00,        0.00, -(base[2] - radio)]
            ancla[1] = [0.00, -longitud/2,               0.00]
            ancla[2] = [0.00,        0.00,  (base[2] - radio)]
            ancla[3] = [0.00, -longitud/2,               0.00]

      elif extremidades == "3" and glibertad == "1":
            pos[0] = [-(base[0] - radio), -(longitud/2 + base[1]), -
(base[2] - radio)]
            pos[1] = [-(base[0] - radio), -(longitud/2 + base[1]),
(base[2] - radio)]
            pos[2] = [ (base[0] - radio), -(longitud/2 + base[1]),
0.00]

            ancla[0] = [-(base[0] - radio), 0.00, -(base[2] -
radio)]
            ancla[1] = [-(base[0] - radio), 0.00,  (base[2] -
radio)]
            ancla[2] = [ (base[0] - radio), 0.00,
0.00]

      elif extremidades == "3" and glibertad == "2":
            pos[0] = [-(base[0] - radio), -(longitud/2 + base[1]), -
(base[2] - radio)]
            pos[1] = [             0.00,               -longitud,
0.00]
            pos[2] = [-(base[0] - radio), -(longitud/2 + base[1]),
(base[2] - radio)]
            pos[3] = [             0.00,               -longitud,
0.00]
            pos[4] = [ (base[0] - radio), -(longitud/2 + base[1]),
0.00]
            pos[5] = [             0.00,               -longitud,
0.00]

            ancla[0] = [-(base[0] - radio),                   0.00,
-(base[2] - radio)]
            ancla[1] = [             0.00,            -longitud/2,
0.00]
            ancla[2] = [-(base[0] - radio),                   0.00,
(base[2] - radio)]
```

```
            ancla[3] = [              0.00,             -longitud/2,
0.00]
            ancla[4] = [ (base[0] - radio),                    0.00,
0.00]
            ancla[5] = [              0.00,             -longitud/2,
0.00]

      elif extremidades == "4" and glibertad == "1" and opcion !=
"1" and opcion != "2":
            pos[0] = [-(base[0] - radio), -(longitud/2 + base[1]), -
(base[2] - radio)]
            pos[1] = [-(base[0] - radio), -(longitud/2 + base[1]),
(base[2] - radio)]
            pos[2] = [ (base[0] - radio), -(longitud/2 + base[1]), -
(base[2] - radio)]
            pos[3] = [ (base[0] - radio), -(longitud/2 + base[1]),
(base[2] - radio)]

            ancla[0] = [-(base[0] - radio), 0.00, -(base[2] -
radio)]
            ancla[1] = [-(base[0] - radio), 0.00,  (base[2] -
radio)]
            ancla[2] = [ (base[0] - radio), 0.00, -(base[2] -
radio)]
            ancla[3] = [ (base[0] - radio), 0.00,  (base[2] -
radio)]

      elif extremidades == "4" and glibertad == "2" and opcion ==
"1":
            pos[0] = [-(base[0] - radio), -(longitud/2 + base[1]), -
(base[2] - radio)]
            pos[1] = [              0.00,               -longitud,
0.00]
            pos[2] = [-(base[0] - radio), -(longitud/2 + base[1]),
(base[2] - radio)]
            pos[3] = [              0.00,               -longitud,
0.00]
            pos[4] = [ (base[0] - radio), -(longitud/2 + base[1]), -
(base[2] - radio)]
            pos[5] = [              0.00,               -longitud,
0.00]
            pos[6] = [ (base[0] - radio), -(longitud/2 + base[1]),
(base[2] - radio)]
            pos[7] = [              0.00,               -longitud,
0.00]

            ancla[0] = [-(base[0] - radio),           0.00, -
(base[2] - radio)]
            ancla[1] = [              0.00,             -longitud/2,
0.00]
            ancla[2] = [-(base[0] - radio),                0.00,
(base[2] - radio)]
            ancla[3] = [              0.00,             -longitud/2,
0.00]
            ancla[4] = [ (base[0] - radio),           0.00, -
(base[2] - radio)]
            ancla[5] = [              0.00,             -longitud/2,
0.00]
```

```
            ancla[6] = [ (base[0] - radio),           0.00,
(base[2] - radio)]
            ancla[7] = [              0.00,             -longitud/2,
0.00]


      elif extremidades == "4" and glibertad == "2" and opcion ==
"2":
            pos[0] = [                    0.00,
0.00,  (base[2] + longitud/2)]
            pos[1] = [                    0.00, - (longitud/2 +
radio),      longitud/2 - radio]
            pos[2] = [ (base[0] + longitud/2),
0.00,                    0.00]
            pos[3] = [     longitud/2 - radio, - (longitud/2 +
radio),                     0.00]
            pos[4] = [                    0.00,
0.00, -(base[2] + longitud/2)]
            pos[5] = [                    0.00, - (longitud/2 +
radio),  - (longitud/2 - radio)]
            pos[6] = [-(base[0] + longitud/2),
0.00,                    0.00]
            pos[7] = [  -(longitud/2 - radio), - (longitud/2 +
radio),                     0.00]

            ancla[0] = [                    0.00,
0.00,  (base[2] + longitud/2)/2]
            ancla[1] = [                    0.00,
0.00,      longitud/2 - radio]
            ancla[2] = [ (base[0] + longitud/2)/2,
0.00,                    0.00]
            ancla[3] = [     longitud/2 - radio,
0.00,                    0.00]
            ancla[4] = [                    0.00,
0.00, -(base[2] + longitud/2)/2]
            ancla[5] = [                    0.00,
0.00,  - (longitud/2 - radio)]
            ancla[6] = [-(base[0] + longitud/2)/2,
0.00,                    0.00]
            ancla[7] = [  -(longitud/2 - radio),
0.00,                    0.00]

      aux = "0.0 0.0 0.0"    #Posicion de la base del robot
      translation.append(aux)

      for n in range (0, 8): #Convertimos de numerico a cadena
            aux0 = pos[n,0]
            aux0 = str(aux0)
            aux1 = pos[n,1]
            aux1 = str(aux1)
            aux2 = pos[n,2]
            aux2 = str(aux2)

            aux  = aux0 + " " + aux1 + " " + aux2
            translation.append(aux)

      for n in range (0, 8): #Convertimos de numerico a cadena
            aux0 = ancla[n,0]
```

```
            aux0 = str(aux0)
            aux1 = ancla[n,1]
            aux1 = str(aux1)
            aux2 = ancla[n,2]
            aux2 = str(aux2)

            aux  = aux0 + " " + aux1 + " " + aux2
            anchor.append(aux)

      return pos, translation, ancla, anchor


#Escoger robot
def escoger_robot(opcion, extremidades, glibertad):
      opcion = "0"
      while True:
            try:
                  extremidades = raw_input(" - Introduzca el número
de extremidades del robot (mínimo 2 y máximo 4): ")
                  if extremidades == "2" or extremidades == "3" or
extremidades == "4":
                        break
                  else:
                        print("\nEl número de extremidades debe estar
entre 2 y 4.\n")
            except:
                  print("\nEl número de extremidades debe estar
entre 2 y 4.\n")

      while True:
            try:
                  glibertad = raw_input("\n - Introduzca el número
de grados de libertad (entre 1 y 2) para las extremidades del
robot: ")
                  if glibertad == "1" or glibertad == "2":
                        break
                  else:
                        print("\nEl número de grados de libertad debe
ser 1 ó 2.")
            except:
                  print("\nEl número de grados de libertad debe ser
1 ó 2.")

      if extremidades == "4" and glibertad == "2":
            print("\nHay dos tipos de robots con esas
características: \n")
            print(" 1 - Robot tipo mesa.")
            print(" 2 - Robot tipo araña.")
            while True:
                  try:
                        opcion = raw_input("\nEscoja una de la dos
opciones: ")
                        if opcion == "1" or opcion == "2":
                              break
                        else:
                              print("\nDebes escoger la opición 1 ó
2.")
                  except:
```

```
                        print("\nDebes escoger la opición 1 ó la
opción 2.")

      return opcion, extremidades, glibertad

#Escoger dimensiones
def dimensiones():
      os.system("clear")
      base=[0, 0, 0]
      radio = 0
      longitud = 0
##Base
      print("\nDimensiones de la base del robot.")
      while True:
            try:
                  base[0] = float(input("\n - Introduzca la longitud
en metros para el eje 'x' (largo): "))
                  if base[0] <= 0:
                        print("\nIntroduzca una longitud superior a
0.")
                  else:
                        break
            except:
                  print("\nIntroduzca una longitud válida.")
      while True:
            try:
                  base[1] = float(input("\n - Introduzca la longitud
en metros para el eje 'y' (alto): "))
                  if base[1] <= 0:
                        print("\nIntroduzca una longitud superior a
0.")
                  else:
                        break
            except:
                  print("\nIntroduzca una longitud válida.")
      while True:
            try:
                  base[2] = float(input("\n - Introduzca la longitud
en metros para el eje 'z' (ancho): "))
                  if base[2] <= 0:
                        print("\nIntroduzca una longitud superior a
0.")
                  else:
                        break
            except:
                  print("\nIntroduzca una longitud válida.")

##Extremidades
      print("\n\nDimensiones de las extremidades del robot.")
      while True:
            try:
                  longitud = float(input("\n - Introduzca la
longitud en metros para las extremidades: "))
                  if longitud <= 0:
                        print("\nIntroduzca una longitud superior a
0.")
                  else:
                        break
```

```
            except:
                  print("\nIntroduzca una longitud válida.")
      while True:
            try:
                  radio = float(input("\n - Introduzca el radio en
metros para las extremidades: "))
                  if radio <= 0:
                        print("\nIntroduzca un radio superior a 0.")
                  else:
                        break
            except:
                  print("\nIntroduzca un radio válida.")
      os.system("clear")
      if glibertad == "2":
            longitud = longitud/2

      return longitud, radio, base

#Escoger el color
def color():
      os.system("clear")
#Color extremidades
      print("\nColor de las extremidades del robot.")
      print("\nLos colores disponibles son: ")
      print("\n - blanco\n - negro\n - azul\n - verde\n - rojo")
      while True:
            try:
                  color_ext = raw_input("\nEscoja un color: ")
                  if  color_ext != "blanco" and color_ext != "negro"
and color_ext != "azul" and color_ext != "verde" and color_ext !=
"rojo":
                        print("\nColor no disponible.")
                  else:
                        break
            except:
                  print("\nColor no disponible.")

      if   color_ext == "blanco":
            color_ext = blanco
      elif color_ext == "negro":
            color_ext = negro
      elif color_ext == "azul":
            color_ext = azul
      elif color_ext == "verde":
            color_ext = verde
      elif color_ext == "rojo":
            color_ext = rojo

#Color base
      os.system("clear")
      print("\nColor de la base del robot.")
      print("\nLos colores disponibles son: ")
      print("\n - blanco\n - negro\n - azul\n - verde\n - rojo")
      while True:
            try:
                  color_base = raw_input("\nEscoja un color: ")
```

```
                if  color_base != "blanco" and color_base !=
"negro" and color_base != "azul" and color_base != "verde" and
color_base != "rojo":
                        print("\nColor no disponible.")
                else:
                        break
            except:
                print("\nColor no disponible.")

      if   color_base == "blanco":
            color_base = blanco
      elif color_base == "negro":
            color_base = negro
      elif color_base == "azul":
            color_base = azul
      elif color_base == "verde":
            color_base = verde
      elif color_base == "rojo":
            color_base = rojo

      os.system("clear")

      return color_ext, color_base

#Escoger la masa
def masa():
      os.system("clear")
      print("\nMasa de las extremidades del robot.")
      while True:
            try:
                masa_ext = float(input("\n - Introduzca la masa en
kg: "))
                if  masa_ext <= 0:
                      print("\nIntroduzca una masa mayor de 0.")
                else:
                      break
            except:
                print("\nIntroduzca una masa válida.")
      os.system("clear")
      print("\nMasa de la base del robot.")
      while True:
            try:
                masa_base = float(input("\n - Introduzca la masa
en kg: "))
                if  masa_base <= 0:
                      print("\nIntroduzca una masa mayor de 0.")
                else:
                      break
            except:
                print("\nIntroduzca una masa válida.")
      os.system("clear")
      return masa_ext, masa_base

#Modificar xmldoc color
def modificar_color(color_ext, color_base):

      primera_vez = True
      for n in xmldoc.getElementsByTagName("diffuseColor"):
```

```
            if primera_vez == True:
                  n.firstChild.data = color_base
            else:
                  n.firstChild.data = color_ext
            primera_vez = False

#Modificar xmldoc masa
def modificar_masa(masa_ext, masa_base):

      primera_vez = True
      for n in xmldoc.getElementsByTagName("total"):
            if primera_vez == True:
                  n.firstChild.data = masa_base
            else:
                  n.firstChild.data = masa_ext
            primera_vez = False

#Modificar xmldoc longitudes
def modificar_longitudes():
##Calculamos las posiciones
      posiciones()

##Convertimos las dimensiones de la base del robot de float a
string
      base[0] = str(base[0])
      base[1] = str(base[1])
      base[2] = str(base[2])

##Unimos las coordenadas x,y,z de las dimensiones de la base en una
unica cadena
      extents_base = base[0] + " " + base[1] + " " + base[2]

      i = 0
##Modificamos posiciones
      for n in xmldoc.getElementsByTagName("Translation"):
            n.firstChild.data = translation[i]
            i+=1

##Modificamos posicion fija de las extremidades
      i = 0
      for n in xmldoc.getElementsByTagName("anchor"):
            n.firstChild.data = anchor[i]
            i+=1

##Modificamos dimensiones de la base
      for n in xmldoc.getElementsByTagName("Extents"):
            n.firstChild.data = extents_base

##Modificamos el radio de las extremidades
      for n in xmldoc.getElementsByTagName("radius"):
            n.firstChild.data = radio

##Modificamos la longitud de las extremidades
      for n in xmldoc.getElementsByTagName("height"):
            n.firstChild.data = longitud


#Modificar fichero xml
```

```
def modificar_xml(extremidades, glibertad, opcion):
##Abrimos el fichero xml
      if   extremidades == "2" and glibertad == "1":
            nf = open('robot_2p.xml','w')
      elif extremidades == "2" and glibertad == "2":
            nf = open('robot_2p2.xml','w')
      elif extremidades == "3" and glibertad == "1":
            nf = open('robot_3p.xml','w')
      elif extremidades == "3" and glibertad == "2":
            nf = open('robot_3p2.xml','w')
      elif extremidades == "4":
            if   glibertad == "1":
                  nf = open('robot_4p.xml','w')
            elif opcion == "1":
                  nf = open('robot_4p2.xml','w')
            elif opcion == "2":
                  nf = open('robot_spider.xml','w')

      nf.write(xmldoc.toxml()) #Escribimos el fichero xml
      nf.close() #Cerramos el fichero xml

#Obtenemos las caracteristicas del robot que hay actualmente en el
xml
def obtener_caracteristicas_xml():
      for n in xmldoc.getElementsByTagName("Extents"):
            base=n.firstChild.data
            break
      for n in xmldoc.getElementsByTagName("radius"):
            radio = n.firstChild.data
            break
      for n in xmldoc.getElementsByTagName("height"):
            longitud = n.firstChild.data
            break
      longitud = float(longitud)
      j=1
      for n in xmldoc.getElementsByTagName("total"):
            if j == 1:
                  masa_base = n.firstChild.data
            elif j==2:
                  masa_ext = n.firstChild.data
                  break
            j+=1
      j=1
      for n in xmldoc.getElementsByTagName("diffuseColor"):
            if j == 1:
                  color_base = n.firstChild.data
            elif j==2:
                  color_ext = n.firstChild.data
                  break
            j+=1

      if   color_ext == blanco:
            color_ext = "blanco"
      elif color_ext == negro:
            color_ext = "negro"
      elif color_ext == azul:
            color_ext = "azul"
      elif color_ext == verde:
```

```
            color_ext = "verde"
      elif color_ext == rojo:
            color_ext = "rojo"

      if   color_base == blanco:
            color_base = "blanco"
      elif color_base == negro:
            color_base = "negro"
      elif color_base == azul:
            color_base = "azul"
      elif color_base == verde:
            color_base = "verde"
      elif color_base == rojo:
            color_base = "rojo"

      return masa_ext, masa_base, color_ext, color_base, longitud,
radio, base

#Solicitamos los datos para la simulacion
def solicitar_datos_simulacion():
      time.sleep(0.5)
      os.system("clear")
      print("\n\nA continuación se van a solicitar los datos para
la simulación.\n")
      while True:
            try:
                  vel = float(input("\n - Introduzca la velocidad
máxima de movimiento en grados por segundo: "))
                  if  vel <= 0:
                        print("\nIntroduzca una velocidad superior a
0.")
                  else:
                        break
            except:
                  print("\nIntroduzca una velocidad válida.")
      if   extremidades == "2" and glibertad == "1":
            for i in range (2):
                  print "\n - Datos de la extremidad", i+1, ": "
                  ang[i] = recoger_datos_sim()

      elif extremidades == "2" and glibertad == "2":
            j=0
            for i in range (2):
                  print "\n - Datos de la extremidad", i+1, ",
primera articulación: "
                  ang[j] = recoger_datos_sim()
                  j+=1
                  print "\n - Datos de la extremidad", i+1, ",
segunda articulación: "
                  ang[j] = recoger_datos_sim()
                  j+=1


      elif extremidades == "3" and glibertad == "1":
            for i in range (3):
                  print "\n - Datos de la extremidad", i+1, ": "
                  ang[i] = recoger_datos_sim()
```

```
      elif extremidades == "3" and glibertad == "2":
            j=0
            for i in range (3):
                  print "\n - Datos de la extremidad", i+1, ",
primera articulación: "
                  ang[j] = recoger_datos_sim()
                  j+=1
                  print "\n - Datos de la extremidad", i+1, ",
segunda articulación: "
                  ang[j] = recoger_datos_sim()
                  j+=1

      elif extremidades == "4":
            if   glibertad == "1":
                  for i in range (4):
                        print "\n - Datos de la extremidad", i+1, ":
"
                        ang[i] = recoger_datos_sim()
            else:
                  j=0
                  for i in range (4):
                        print "\n - Datos de la extremidad", i+1, ",
primera articulación: "
                        ang[j] = recoger_datos_sim()
                        j+=1
                        print "\n - Datos de la extremidad", i+1, ",
segunda articulación: "
                        ang[j] = recoger_datos_sim()
                        j+=1
      return ang, vel

def recoger_datos_sim():
      while True:
            try:
                  angulo = float(input("\n\t - Introduzca la
posición angular final del movimiento en grados: "))
                  if  angulo < -60 or angulo > 60:
                        print("\nEl máximo ángulo de giro son 60
grados")
                  else:
                        break
            except:
                  print("\nIntroduzca un ángulo válido.")
      angulo = angulo * math.pi / 180 #Convertimos de grados a
radianes
      return angulo

#Modificamos xml velocidad
def modificar_velocidad():
      for n in xmldoc.getElementsByTagName("maxveldeg"):
            n.firstChild.data = vel



##### PROGRAMA PRINCIPAL #####

while seleccion:
```

```
      print("\n\nSe van a solicitar las caracteristicas del
robot.\n")
# Escogemos el robot
      opcion, extremidades, glibertad = escoger_robot(opcion,
extremidades, glibertad)

#Cargamos el fichero xml corresponiente con los requisitos
      import xml.dom.minidom
      if   extremidades == "2" and glibertad == "1":
            xmldoc = xml.dom.minidom.parse("robot_2p.xml")
      elif extremidades == "2" and glibertad == "2":
            xmldoc = xml.dom.minidom.parse("robot_2p2.xml")
      elif extremidades == "3" and glibertad == "1":
            xmldoc = xml.dom.minidom.parse("robot_3p.xml")
      elif extremidades == "3" and glibertad == "2":
            xmldoc = xml.dom.minidom.parse("robot_3p2.xml")
      elif extremidades == "4":
            if   glibertad == "1":
                  xmldoc = xml.dom.minidom.parse("robot_4p.xml")
            elif opcion == "1":
                  xmldoc = xml.dom.minidom.parse("robot_4p2.xml")
            elif opcion == "2":
                  xmldoc = xml.dom.minidom.parse("robot_spider.xml")

      seleccion = False
      accion    = "0"

      os.system("clear")
      print("\nLas especificaciones de masa, color y dimensiones a
aplicar son las últimas guardadas.")

#Esoger la acción a realizar
      while accion!= "1" and accion!= "6":
            try:

#Configuración actual
                  masa_ext, masa_base, color_ext, color_base,
longitud, radio, base = obtener_caracteristicas_xml()
                  print("\nConfiguración actual:")
                  if opcion == "0":
                        if glibertad == "1":
                              print " - Robot de", extremidades,
"extremidades y", glibertad, "grado de libertad por extremidad"
                        elif glibertad == "2":
                              print " - Robot de", extremidades,
"extremidades y", glibertad, "grados de libertad por extremidad"
                  elif opcion == "1":
                        print " - Robot de", extremidades,
"extremidades y", glibertad, "grados de libertad por extremidad,
tipo mesa"
                  elif opcion == "2":
                        print " - Robot de", extremidades,
"extremidades y", glibertad, "grados de libertad por extremidad,
tipo araña"
                  print " - Dimensiones de la base: ", base, "[m]"
                  if glibertad == "1":
                        print " - Longitud de las extremidades: ",
longitud, "[m]"
```

```
                elif glibertad == "2":
                      print " - Longitud de las extremidades: ",
longitud*2, "[m]"
                print " - Radio de las extremidades: ", radio,
"[m]"
                print " - Masa de las extremidades: ", masa_ext,
"[kg]"
                print " - Masa de la base: ", masa_base, "[kg]"
                print " - Color de las extremidades: ", color_ext
                print " - Color de la base: ", color_base

#Menú de opciones
                print("\n\nAcciones disponibles: \n")
                print(" 1 - Configurar un nuevo robot")
                print(" 2 - Modificar las dimensiones")
                print(" 3 - Modificar la masa")
                print(" 4 - Modificar el color")
                print(" 5 - Simular con OpenRAVE")
                print(" 6 - Salir")
                accion = raw_input("\nEscoge una de las opciones:
")

                if   accion == "1":
                      os.system("clear")
                      seleccion = True
                elif   accion == "2":
                      longitud, radio, base = dimensiones()
                      modificar_longitudes()
                      modificar_xml(extremidades, glibertad,
opcion)
                elif accion == "3":
                      masa_ext, masa_base = masa()
                      modificar_masa(masa_ext, masa_base)
                      modificar_xml(extremidades, glibertad,
opcion)
                elif accion == "4":
                      color_ext, color_base = color()
                      modificar_color(color_ext, color_base)
                      modificar_xml(extremidades, glibertad,
opcion)
                elif accion == "5":
                      ang, vel = solicitar_datos_simulacion()
                      modificar_velocidad()
                      modificar_xml(extremidades, glibertad,
opcion)

                      if cargar_viewer == True:
                            env = Environment() # create openrave
environment
                            env.SetViewer('qtcoin') # attach viewer
                            cargar_viewer = False
                      #cargamos el robot
                      if   extremidades == "2" and glibertad ==
"1":
                            env.Load('robot_2p.xml')
                      elif extremidades == "2" and glibertad ==
"2":
                            env.Load('robot_2p2.xml')
```

```
                        elif extremidades == "3" and glibertad ==
"1":
                              env.Load('robot_3p.xml')
                        elif extremidades == "3" and glibertad ==
"2":
                              env.Load('robot_3p2.xml')
                        elif extremidades == "4":
                              if   glibertad == "1":
                                    env.Load('robot_4p.xml')
                              elif opcion == "1":
                                    env.Load('robot_4p2.xml')
                              elif opcion == "2":
                                    env.Load('robot_spider.xml')

                        robot = env.GetRobots()[0] #obtenemos el
robot
                        robot.basemanip =
interfaces.BaseManipulation(robot)
                        #Realizamos movimiento
                        if   extremidades == "2" and glibertad ==
"1":
                              goal1 = [ang[0], ang[1]]

      robot.basemanip.MoveActiveJoints(goal=goal1,maxiter=3000,step
length=0.1)
                        elif extremidades == "2" and glibertad ==
"2":
                              goal2 = [ang[0], ang[1], ang[2],
ang[3]]

      robot.basemanip.MoveActiveJoints(goal=goal2,maxiter=3000,step
length=0.1)
                        elif extremidades == "3" and glibertad ==
"1":
                              goal3 = [ang[0], ang[1], ang[2]]

      robot.basemanip.MoveActiveJoints(goal=goal3,maxiter=3000,step
length=0.1)
                        elif extremidades == "3" and glibertad ==
"2":
                              goal4 = [ang[0], ang[1], ang[2],
ang[3], ang[4], ang[5]]

      robot.basemanip.MoveActiveJoints(goal=goal4,maxiter=3000,step
length=0.1)
                        elif extremidades == "4":
                              if   glibertad == "1":
                                    goal5 = [ang[0], ang[1], ang[2],
ang[3]]

      robot.basemanip.MoveActiveJoints(goal=goal5,maxiter=3000,step
length=0.1)
                              elif opcion == "1":
                                    goal6 = [ang[0], ang[1], ang[2],
ang[3], ang[4], ang[5], ang[6], ang[7]]

      robot.basemanip.MoveActiveJoints(goal=goal6,maxiter=3000,step
length=0.1)
```

```
                              elif opcion == "2":
                                    goal7 = [ang[0], ang[1], ang[2],
ang[3], ang[4], ang[5], ang[6], ang[7]]

      robot.basemanip.MoveActiveJoints(goal=goal7,maxiter=3000,step
length=0.1)

                        #Esperemos a que termine el movimiento
(Copiado de los ejemplos de OpenRAVE)
                        while not robot.GetController().IsDone():
                              time.sleep(0.01)
                        #

                        time.sleep(1)
                        os.system("clear")
                        raw_input ("\nPresione enter para finalizar
la simulación.")
                        os.system("clear")

                        with env:
                              env.Remove(robot) #Eliminamos robot del
entorno de OpenRAVE

                  elif accion == "6":
                        seleccion = False
                  else:
                        os.system("clear")
                        print("\nOpción escogida incorrecta.")
                        time.sleep(0.5)

            except:
                  os.system("clear")
                  print("\nOpción escogida incorrecta.")
                  time.sleep(0.5)
os.system("clear")
```

Robot dos extremidades y un grado de libertad: robot_2p.xml

```
<?xml version="1.0" ?><Robot name="robot_2p">

 <KinBody>

  <Body name="Base" type="dynamic">

   <Translation>0.0 0.0 0.0</Translation>

   <Geom type="box">

       <Extents>5.0 0.5 8.0</Extents>

       <RotationAxis>0 0 0 0</RotationAxis>

    <diffuseColor>0 0 0</diffuseColor>

   </Geom>
```

```
  <Mass type="box">
    <total>10.0</total>
  </Mass>
</Body>

<!-- the first movable link-->
<Body name="Arm0" type="dynamic">
  <offsetfrom>Base</offsetfrom>
  <Translation>0.0 -4.5 -7.3</Translation>
  <Geom type="cylinder">
        <rotationaxis>0 0 0 0</rotationaxis>
    <radius>0.7</radius>
    <height>8.0</height>
    <diffuseColor>1 0 0</diffuseColor>
  </Geom>
  <Mass type="mimicgeom">
    <total>10.0</total>
  </Mass>
</Body>

<!--<Joint circular="true" name="Arm0" type="hinge"> -->
<Joint name="Arm0" type="hinge">
  <Body>Base</Body>
  <Body>Arm0</Body>
  <offsetfrom>Base</offsetfrom>
  <weight>4</weight>
  <limitsdeg>-60 60</limitsdeg>
```

```
    <axis>0 0 1</axis>
    <maxveldeg>10.0</maxveldeg>
    <resolution>1</resolution>
    <anchor>0.0 0.0 -7.3</anchor>
   </Joint>

<!-- the second movable link-->
   <Body name="Arm1" type="dynamic">
    <offsetfrom>Base</offsetfrom>
    <!-- Translation relative to Base-->
    <Translation>0.0 -4.5 7.3</Translation>
    <Geom type="cylinder">
          <rotationaxis>0 0 0 0</rotationaxis>
     <radius>0.7</radius>
     <height>8.0</height>
     <diffuseColor>1 0 0</diffuseColor>
    </Geom>
    <Mass type="mimicgeom">
     <total>10.0</total>
    </Mass>
   </Body>
   <Joint name="Arm1" type="hinge">
    <Body>Base</Body>
    <Body>Arm1</Body>
    <offsetfrom>Base</offsetfrom>
    <weight>4</weight>
    <limitsdeg>-60 60</limitsdeg>
```

```
    <axis>0 0 1</axis>
    <maxveldeg>10.0</maxveldeg>
    <resolution>1</resolution>
    <anchor>0.0 0.0 7.3</anchor>
   </Joint>

 </KinBody>
</Robot>
```

Robot dos extremidades y dos grados de libertad: robot_2p2.xml

```
<?xml version="1.0" ?><Robot name="robot_2p2">
 <KinBody>
   <Body name="Base" type="dynamic">
    <Translation>0.0 0.0 0.0</Translation>
    <Geom type="box">
          <Extents>5.0 0.5 5.0</Extents>
          <RotationAxis>0 0 0 0</RotationAxis>
     <diffuseColor>0 0 1</diffuseColor>
    </Geom>
    <Mass type="box">
     <total>10.0</total>
    </Mass>
   </Body>

   <!-- the first movable link-->
   <Body name="Arm0" type="dynamic">
    <offsetfrom>Base</offsetfrom>
```

```
  <Translation>0.0 -4.0 -4.3</Translation>
  <Geom type="cylinder">
        <rotationaxis>0 0 0 0</rotationaxis>
   <radius>0.7</radius>
   <height>7.0</height>
   <diffuseColor>1 1 1</diffuseColor>
  </Geom>
  <Mass type="mimicgeom">
   <total>10.0</total>
  </Mass>
</Body>

<Joint name="Arm0" type="hinge">
  <Body>Base</Body>
  <Body>Arm0</Body>
  <offsetfrom>Base</offsetfrom>
  <weight>4</weight>
  <limitsdeg>-60 60</limitsdeg>
  <axis>0 0 1</axis>
  <maxveldeg>15.0</maxveldeg>
  <resolution>1</resolution>
  <anchor>0.0 0.0 -4.3</anchor>
</Joint>
<!--Segmento 2 de la pata-->
<Body name="Arm01" type="dynamic">
  <offsetfrom>Arm0</offsetfrom>
  <Translation>0.0 -7.0 0.0</Translation>
```

```
    <Geom type="cylinder">
          <rotationaxis>0 0 0 0</rotationaxis>
      <radius>0.7</radius>
      <height>7.0</height>
      <diffuseColor>1 1 1</diffuseColor>
    </Geom>
    <Mass type="mimicgeom">
      <total>10.0</total>
    </Mass>
  </Body>
  <!--union segmentos de las patas -->
  <Joint name="Arm01" type="hinge">
    <Body>Arm0</Body>
    <Body>Arm01</Body>
    <offsetfrom>Arm0</offsetfrom>
    <weight>4</weight>
    <limitsdeg>-60 60</limitsdeg>
    <axis>0 0 1</axis>
    <maxveldeg>15.0</maxveldeg>
    <resolution>1</resolution>
    <anchor>0.0 -3.5 0.0</anchor>
  </Joint>

<!-- the second movable link-->
  <Body name="Arm1" type="dynamic">
    <offsetfrom>Base</offsetfrom>
    <!-- Translation relative to Base-->
```

```
  <Translation>0.0 -4.0 4.3</Translation>
  <Geom type="cylinder">
        <rotationaxis>0 0 0 0</rotationaxis>
    <radius>0.7</radius>
    <height>7.0</height>
    <diffuseColor>1 1 1</diffuseColor>
  </Geom>
  <Mass type="mimicgeom">
    <total>10.0</total>
  </Mass>
</Body>
<Joint name="Arm1" type="hinge">
  <Body>Base</Body>
  <Body>Arm1</Body>
  <offsetfrom>Base</offsetfrom>
  <weight>4</weight>
  <limitsdeg>-60 60</limitsdeg>
  <axis>0 0 1</axis>
  <maxveldeg>15.0</maxveldeg>
  <resolution>1</resolution>
  <anchor>0.0 0.0 4.3</anchor>
</Joint>
<!--Segmento 2 de la pata-->
<Body name="Arm11" type="dynamic">
  <offsetfrom>Arm1</offsetfrom>
  <Translation>0.0 -7.0 0.0</Translation>
  <Geom type="cylinder">
```

```
        <rotationaxis>0 0 0 0</rotationaxis>
      <radius>0.7</radius>
      <height>7.0</height>
      <diffuseColor>1 1 1</diffuseColor>
    </Geom>
    <Mass type="mimicgeom">
      <total>10.0</total>
    </Mass>
  </Body>
  <!--union segmentos de las patas -->
  <Joint name="Arm11" type="hinge">
    <Body>Arm1</Body>
    <Body>Arm11</Body>
    <offsetfrom>Arm1</offsetfrom>
    <weight>4</weight>
    <limitsdeg>-60 60</limitsdeg>
    <axis>0 0 1</axis>
    <maxveldeg>15.0</maxveldeg>
    <resolution>1</resolution>
    <anchor>0.0 -3.5 0.0</anchor>
  </Joint>

 </KinBody>
</Robot>
```

Robot tres extremidades y un grado de libertad: robot_3p.xml

```
<?xml version="1.0" ?><Robot name="robot_3p">
```

```
<KinBody>
 <Body name="Base" type="dynamic">
  <Translation>0.0 0.0 0.0</Translation>
  <Geom type="box">
        <Extents>5.0 0.5 5.0</Extents>
        <RotationAxis>0 0 0 0</RotationAxis>
   <diffuseColor>0 1 0</diffuseColor>
  </Geom>
  <Mass type="box">
   <total>1.0</total>
  </Mass>
 </Body>

 <!-- the first movable link-->
 <Body name="Arm0" type="dynamic">
  <offsetfrom>Base</offsetfrom>
  <Translation>-4.0 -4.5 -4.0</Translation>
  <Geom type="cylinder">
        <rotationaxis>0 0 0 0</rotationaxis>
   <radius>1.0</radius>
   <height>8.0</height>
   <diffuseColor>0 0 1</diffuseColor>
  </Geom>
  <Mass type="mimicgeom">
   <total>10.0</total>
  </Mass>
 </Body>
```

```
  <Joint name="Arm0" type="hinge">
    <Body>Base</Body>
    <Body>Arm0</Body>
    <offsetfrom>Base</offsetfrom>
    <weight>4</weight>
    <limitsdeg>-60 60</limitsdeg>
    <axis>0 0 1</axis>
    <maxveldeg>50.0</maxveldeg>
    <resolution>1</resolution>
    <anchor>-4.0 0.0 -4.0</anchor>
  </Joint>

<!-- the second movable link-->
  <Body name="Arm1" type="dynamic">
    <offsetfrom>Base</offsetfrom>
    <!-- Translation relative to Base-->
    <Translation>-4.0 -4.5 4.0</Translation>
    <Geom type="cylinder">
          <rotationaxis>0 0 0 0</rotationaxis>
      <radius>1.0</radius>
      <height>8.0</height>
      <diffuseColor>0 0 1</diffuseColor>
    </Geom>
    <Mass type="mimicgeom">
      <total>10.0</total>
    </Mass>
```

```
    </Body>
    <Joint name="Arm1" type="hinge">
      <Body>Base</Body>
      <Body>Arm1</Body>
      <offsetfrom>Base</offsetfrom>
      <weight>4</weight>
      <limitsdeg>-60 60</limitsdeg>
      <axis>0 0 1</axis>
      <maxveldeg>50.0</maxveldeg>
      <resolution>1</resolution>
      <anchor>-4.0 0.0 4.0</anchor>
    </Joint>

<!-- the thrid movable link-->
    <Body name="Arm2" type="dynamic">
      <offsetfrom>Base</offsetfrom>
      <!-- Translation relative to Base-->
      <Translation>4.0 -4.5 0.0</Translation>
      <Geom type="cylinder">
            <rotationaxis>0 0 0 0</rotationaxis>
        <radius>1.0</radius>
        <height>8.0</height>
        <diffuseColor>0 0 1</diffuseColor>
      </Geom>
      <Mass type="mimicgeom">
        <total>10.0</total>
      </Mass>
```

```
  </Body>
  <Joint name="Arm2" type="hinge">
   <Body>Base</Body>
   <Body>Arm2</Body>
   <offsetfrom>Base</offsetfrom>
   <weight>4</weight>
   <limitsdeg>-60 60</limitsdeg>
   <axis>0 0 1</axis>
   <maxveldeg>50.0</maxveldeg>
   <resolution>1</resolution>
   <anchor>4.0 0.0 0.0</anchor>
  </Joint>

 </KinBody>
</Robot>
```

Robot tres extremidades y dos grados de libertad: robot_3p2.xml

```
<?xml version="1.0" ?><Robot name="robot_3p2">
 <KinBody>
  <Body name="Base" type="dynamic">
   <Translation>0.0 0.0 0.0</Translation>
   <Geom type="box">
        <Extents>5.0 0.5 5.0</Extents>
        <RotationAxis>0 0 0 0</RotationAxis>
    <diffuseColor>0 0 1</diffuseColor>
   </Geom>
   <Mass type="box">
```

```
   <total>1.0</total>
  </Mass>
</Body>

<!-- the first movable link-->
<Body name="Arm0" type="dynamic">
  <offsetfrom>Base</offsetfrom>
  <Translation>-4.3 -4.0 -4.3</Translation>
  <Geom type="cylinder">
        <rotationaxis>0 0 0 0</rotationaxis>
    <radius>0.7</radius>
    <height>7.0</height>
    <diffuseColor>0 1 0</diffuseColor>
  </Geom>
  <Mass type="mimicgeom">
    <total>1.0</total>
  </Mass>
</Body>

<Joint name="Arm0" type="hinge">
  <Body>Base</Body>
  <Body>Arm0</Body>
  <offsetfrom>Base</offsetfrom>
  <weight>4</weight>
  <limitsdeg>-60 60</limitsdeg>
  <axis>0 0 1</axis>
  <maxveldeg>5.0</maxveldeg>
```

```
  <resolution>1</resolution>
  <anchor>-4.3 0.0 -4.3</anchor>
</Joint>
<!--Segmento 2 de la pata-->
<Body name="Arm01" type="dynamic">
  <offsetfrom>Arm0</offsetfrom>
  <Translation>0.0 -7.0 0.0</Translation>
  <Geom type="cylinder">
        <rotationaxis>0 0 0 0</rotationaxis>
    <radius>0.7</radius>
    <height>7.0</height>
    <diffuseColor>0 1 0</diffuseColor>
  </Geom>
  <Mass type="mimicgeom">
    <total>1.0</total>
  </Mass>
</Body>
<!--union segmentos de las patas -->
<Joint name="Arm01" type="hinge">
  <Body>Arm0</Body>
  <Body>Arm01</Body>
  <offsetfrom>Arm0</offsetfrom>
  <weight>4</weight>
  <limitsdeg>-60 60</limitsdeg>
  <axis>0 0 1</axis>
  <maxveldeg>5.0</maxveldeg>
  <resolution>1</resolution>
```

```
    <anchor>0.0 -3.5 0.0</anchor>
   </Joint>

<!-- the second movable link-->
   <Body name="Arm1" type="dynamic">
    <offsetfrom>Base</offsetfrom>
    <!-- Translation relative to Base-->
    <Translation>-4.3 -4.0 4.3</Translation>
    <Geom type="cylinder">
          <rotationaxis>0 0 0 0</rotationaxis>
     <radius>0.7</radius>
     <height>7.0</height>
     <diffuseColor>0 1 0</diffuseColor>
    </Geom>
    <Mass type="mimicgeom">
     <total>1.0</total>
    </Mass>
   </Body>
   <Joint name="Arm1" type="hinge">
    <Body>Base</Body>
    <Body>Arm1</Body>
    <offsetfrom>Base</offsetfrom>
    <weight>4</weight>
    <limitsdeg>-60 60</limitsdeg>
    <axis>0 0 1</axis>
    <maxveldeg>5.0</maxveldeg>
    <resolution>1</resolution>
```

```
  <anchor>-4.3 0.0 4.3</anchor>
</Joint>
<!--Segmento 2 de la pata-->
<Body name="Arm11" type="dynamic">
  <offsetfrom>Arm1</offsetfrom>
  <Translation>0.0 -7.0 0.0</Translation>
  <Geom type="cylinder">
        <rotationaxis>0 0 0 0</rotationaxis>
    <radius>0.7</radius>
    <height>7.0</height>
    <diffuseColor>0 1 0</diffuseColor>
  </Geom>
  <Mass type="mimicgeom">
    <total>1.0</total>
  </Mass>
</Body>
<!--union segmentos de las patas -->
<Joint name="Arm11" type="hinge">
  <Body>Arm1</Body>
  <Body>Arm11</Body>
  <offsetfrom>Arm1</offsetfrom>
  <weight>4</weight>
  <limitsdeg>-60 60</limitsdeg>
  <axis>0 0 1</axis>
  <maxveldeg>5.0</maxveldeg>
  <resolution>1</resolution>
  <anchor>0.0 -3.5 0.0</anchor>
```

```
  </Joint>



<!-- the thrid movable link-->
  <Body name="Arm2" type="dynamic">
    <offsetfrom>Base</offsetfrom>
    <!-- Translation relative to Base-->
    <Translation>4.3 -4.0 0.0</Translation>
    <Geom type="cylinder">
          <rotationaxis>0 0 0 0</rotationaxis>
      <radius>0.7</radius>
      <height>7.0</height>
      <diffuseColor>0 1 0</diffuseColor>
    </Geom>
    <Mass type="mimicgeom">
      <total>1.0</total>
    </Mass>
  </Body>
  <Joint name="Arm2" type="hinge">
    <Body>Base</Body>
    <Body>Arm2</Body>
    <offsetfrom>Base</offsetfrom>
    <weight>4</weight>
    <limitsdeg>-60 60</limitsdeg>
    <axis>0 0 1</axis>
    <maxveldeg>5.0</maxveldeg>
    <resolution>1</resolution>
```

```
  <anchor>4.3 0.0 0.0</anchor>
</Joint>
<!--Segmento 2 de la pata-->
<Body name="Arm21" type="dynamic">
  <offsetfrom>Arm2</offsetfrom>
  <Translation>0.0 -7.0 0.0</Translation>
  <Geom type="cylinder">
        <rotationaxis>0 0 0 0</rotationaxis>
    <radius>0.7</radius>
    <height>7.0</height>
    <diffuseColor>0 1 0</diffuseColor>
  </Geom>
  <Mass type="mimicgeom">
    <total>1.0</total>
  </Mass>
</Body>
<!--union segmentos de las patas -->
<Joint name="Arm21" type="hinge">
  <Body>Arm2</Body>
  <Body>Arm21</Body>
  <offsetfrom>Arm2</offsetfrom>
  <weight>4</weight>
  <limitsdeg>-60 60</limitsdeg>
  <axis>0 0 1</axis>
  <maxveldeg>5.0</maxveldeg>
  <resolution>1</resolution>
  <anchor>0.0 -3.5 0.0</anchor>
```

```
  </Joint>



 </KinBody>
</Robot>
```

Robot cuatro extremidades y un grado de libertad: robot_4p.xml

```
<?xml version="1.0" ?><Robot name="robot_4p">
 <KinBody>
  <Body name="Base" type="dynamic">
   <Translation>0.0 0.0 0.0</Translation>
   <Geom type="box">
        <Extents>5.0 0.5 5.0</Extents>
        <RotationAxis>0 0 0 0</RotationAxis>
    <diffuseColor>1 1 1</diffuseColor>
   </Geom>
   <Mass type="box">
    <total>10.0</total>
   </Mass>
  </Body>

  <!-- the first movable link-->
  <Body name="Arm0" type="dynamic">
   <offsetfrom>Base</offsetfrom>
   <Translation>-4.3 -4.0 -4.3</Translation>
   <Geom type="cylinder">
        <rotationaxis>0 0 0 0</rotationaxis>
```

```
    <radius>0.7</radius>
    <height>7.0</height>
    <diffuseColor>0 0 1</diffuseColor>
   </Geom>
   <Mass type="mimicgeom">
    <total>10.0</total>
   </Mass>
  </Body>

  <Joint name="Arm0" type="hinge">
   <Body>Base</Body>
   <Body>Arm0</Body>
   <offsetfrom>Base</offsetfrom>
   <weight>4</weight>
   <limitsdeg>-60 60</limitsdeg>
   <axis>0 0 1</axis>
   <maxveldeg>1.0</maxveldeg>
   <resolution>1</resolution>
   <anchor>-4.3 0.0 -4.3</anchor>
  </Joint>

<!-- the second movable link-->
  <Body name="Arm1" type="dynamic">
   <offsetfrom>Base</offsetfrom>
   <Translation>-4.3 -4.0 4.3</Translation>
   <Geom type="cylinder">
        <rotationaxis>0 0 0 0</rotationaxis>
```

```
      <radius>0.7</radius>
      <height>7.0</height>
      <diffuseColor>0 0 1</diffuseColor>
    </Geom>
    <Mass type="mimicgeom">
      <total>10.0</total>
    </Mass>
  </Body>
  <Joint name="Arm1" type="hinge">
    <Body>Base</Body>
    <Body>Arm1</Body>
    <offsetfrom>Base</offsetfrom>
    <weight>4</weight>
    <limitsdeg>-60 60</limitsdeg>
    <axis>0 0 1</axis>
    <maxveldeg>1.0</maxveldeg>
    <resolution>1</resolution>
    <anchor>-4.3 0.0 4.3</anchor>
  </Joint>

<!-- the third movable link-->
  <Body name="Arm2" type="dynamic">
    <offsetfrom>Base</offsetfrom>
    <!-- Translation relative to Base-->
    <Translation>4.3 -4.0 -4.3</Translation>
    <Geom type="cylinder">
          <rotationaxis>0 0 0 0</rotationaxis>
```

```
      <radius>0.7</radius>
      <height>7.0</height>
      <diffuseColor>0 0 1</diffuseColor>
    </Geom>
    <Mass type="mimicgeom">
      <total>10.0</total>
    </Mass>
  </Body>
  <Joint name="Arm2" type="hinge">
    <Body>Base</Body>
    <Body>Arm2</Body>
    <offsetfrom>Base</offsetfrom>
    <weight>4</weight>
    <limitsdeg>-60 60</limitsdeg>
    <axis>0 0 1</axis>
    <maxveldeg>1.0</maxveldeg>
    <resolution>1</resolution>
    <anchor>4.3 0.0 -4.3</anchor>
  </Joint>

<!-- the fourth movable link-->
  <Body name="Arm3" type="dynamic">
    <offsetfrom>Base</offsetfrom>
    <!-- Translation relative to Base-->
    <Translation>4.3 -4.0 4.3</Translation>
    <Geom type="cylinder">
          <rotationaxis>0 0 0 0</rotationaxis>
```

```
      <radius>0.7</radius>
      <height>7.0</height>
      <diffuseColor>0 0 1</diffuseColor>
    </Geom>
    <Mass type="mimicgeom">
      <total>10.0</total>
    </Mass>
  </Body>
  <Joint name="Arm3" type="hinge">
    <Body>Base</Body>
    <Body>Arm3</Body>
    <offsetfrom>Base</offsetfrom>
    <weight>4</weight>
    <limitsdeg>-60 60</limitsdeg>
    <axis>0 0 1</axis>
    <maxveldeg>1.0</maxveldeg>
    <resolution>1</resolution>
    <anchor>4.3 0.0 4.3</anchor>
  </Joint>

 </KinBody>
</Robot>
```

Robot cuatro extremidades y dos grados de libertad, tipo mesa: robot_4p2.xml

```
<?xml version="1.0" ?><Robot name="robot_4p2">
 <KinBody>
```

```
<Body name="Base" type="dynamic">
 <Translation>0.0 0.0 0.0</Translation>
 <Geom type="box">
       <Extents>5.0 0.5 5.0</Extents>
       <RotationAxis>0 0 0 0</RotationAxis>
  <diffuseColor>1 1 1</diffuseColor>
 </Geom>
 <Mass type="box">
  <total>1.0</total>
 </Mass>
</Body>

<!-- FIRST LEG-->
<Body name="Arm0" type="dynamic">
 <offsetfrom>Base</offsetfrom>
 <Translation>-4.3 -2.25 -4.3</Translation>
 <Geom type="cylinder">
       <rotationaxis>1 0 0 0</rotationaxis>
  <radius>0.7</radius>
  <height>3.5</height>
  <diffuseColor>0 0 1</diffuseColor>
 </Geom>
 <Mass type="mimicgeom">
  <total>2.0</total>
 </Mass>
</Body>
<!--union pata con base -->
```

```
<Joint name="Arm0" type="hinge">
  <Body>Base</Body>
  <Body>Arm0</Body>
  <offsetfrom>Base</offsetfrom>
  <weight>4</weight>
  <limitsdeg>-60 60</limitsdeg>
  <axis>0 0 1</axis>
  <maxveldeg>5.0</maxveldeg>
  <resolution>1</resolution>
  <anchor>-4.3 0.0 -4.3</anchor>
</Joint>
<!--Segmento 2 de la pata-->
<Body name="Arm01" type="dynamic">
  <offsetfrom>Arm0</offsetfrom>
  <Translation>0.0 -3.5 0.0</Translation>
  <Geom type="cylinder">
        <rotationaxis>0 0 0 0</rotationaxis>
    <radius>0.7</radius>
    <height>3.5</height>
    <diffuseColor>0 0 1</diffuseColor>
  </Geom>
  <Mass type="mimicgeom">
    <total>2.0</total>
  </Mass>
</Body>
<!--union segmentos de las patas -->
<Joint name="Arm01" type="hinge">
```

```
  <Body>Arm0</Body>
  <Body>Arm01</Body>
  <offsetfrom>Arm0</offsetfrom>
  <weight>4</weight>
  <limitsdeg>-60 60</limitsdeg>
  <axis>0 0 1</axis>
  <maxveldeg>5.0</maxveldeg>
  <resolution>1</resolution>
  <anchor>0.0 -1.75 0.0</anchor>
</Joint>

<!-- SECOND LEG-->
<Body name="Arm1" type="dynamic">
  <offsetfrom>Base</offsetfrom>
  <Translation>-4.3 -2.25 4.3</Translation>
  <Geom type="cylinder">
        <rotationaxis>0 0 0 0</rotationaxis>
    <radius>0.7</radius>
    <height>3.5</height>
    <diffuseColor>0 0 1</diffuseColor>
  </Geom>
  <Mass type="mimicgeom">
    <total>2.0</total>
  </Mass>
</Body>
<Joint name="Arm1" type="hinge">
  <Body>Base</Body>
```

```
  <Body>Arm1</Body>
  <offsetfrom>Base</offsetfrom>
  <weight>4</weight>
  <limitsdeg>-60 60</limitsdeg>
  <axis>0 0 1</axis>
  <maxveldeg>5.0</maxveldeg>
  <resolution>1</resolution>
  <anchor>-4.3 0.0 4.3</anchor>
</Joint>
<!--Segmento 2 de la pata-->
<Body name="Arm11" type="dynamic">
  <offsetfrom>Arm1</offsetfrom>
  <Translation>0.0 -3.5 0.0</Translation>
  <Geom type="cylinder">
        <rotationaxis>0 0 0 0</rotationaxis>
    <radius>0.7</radius>
    <height>3.5</height>
    <diffuseColor>0 0 1</diffuseColor>
  </Geom>
  <Mass type="mimicgeom">
    <total>2.0</total>
  </Mass>
</Body>
<!--union segmentos de las patas -->
<Joint name="Arm11" type="hinge">
  <Body>Arm1</Body>
  <Body>Arm11</Body>
```

```
  <offsetfrom>Arm1</offsetfrom>
  <weight>4</weight>
  <limitsdeg>-60 60</limitsdeg>
  <axis>0 0 1</axis>
  <maxveldeg>5.0</maxveldeg>
  <resolution>1</resolution>
  <anchor>0.0 -1.75 0.0</anchor>
</Joint>


<!-- THIRD LEG-->
<Body name="Arm2" type="dynamic">
  <offsetfrom>Base</offsetfrom>
  <Translation>4.3 -2.25 -4.3</Translation>
  <Geom type="cylinder">
        <rotationaxis>0 0 0 0</rotationaxis>
    <radius>0.7</radius>
    <height>3.5</height>
    <diffuseColor>0 0 1</diffuseColor>
  </Geom>
  <Mass type="mimicgeom">
    <total>2.0</total>
  </Mass>
</Body>
<Joint name="Arm2" type="hinge">
  <Body>Base</Body>
  <Body>Arm2</Body>
```

```
  <offsetfrom>Base</offsetfrom>
  <weight>4</weight>
  <limitsdeg>-60 60</limitsdeg>
  <axis>0 0 1</axis>
  <maxveldeg>5.0</maxveldeg>
  <resolution>1</resolution>
  <anchor>4.3 0.0 -4.3</anchor>
</Joint>
<!--Segmento 2 de la pata-->
<Body name="Arm21" type="dynamic">
  <offsetfrom>Arm2</offsetfrom>
  <Translation>0.0 -3.5 0.0</Translation>
  <Geom type="cylinder">
        <rotationaxis>0 0 0 0</rotationaxis>
    <radius>0.7</radius>
    <height>3.5</height>
    <diffuseColor>0 0 1</diffuseColor>
  </Geom>
  <Mass type="mimicgeom">
    <total>2.0</total>
  </Mass>
</Body>
<!--union segmentos de las patas -->
<Joint name="Arm21" type="hinge">
  <Body>Arm2</Body>
  <Body>Arm21</Body>
  <offsetfrom>Arm2</offsetfrom>
```

```
  <weight>4</weight>
  <limitsdeg>-60 60</limitsdeg>
  <axis>0 0 1</axis>
  <maxveldeg>5.0</maxveldeg>
  <resolution>1</resolution>
  <anchor>0.0 -1.75 0.0</anchor>
</Joint>


<!-- FOURTH LEG-->
<Body name="Arm3" type="dynamic">
  <offsetfrom>Base</offsetfrom>
  <Translation>4.3 -2.25 4.3</Translation>
  <Geom type="cylinder">
        <rotationaxis>0 0 0 0</rotationaxis>
    <radius>0.7</radius>
    <height>3.5</height>
    <diffuseColor>0 0 1</diffuseColor>
  </Geom>
  <Mass type="mimicgeom">
    <total>2.0</total>
  </Mass>
</Body>
<Joint name="Arm3" type="hinge">
  <Body>Base</Body>
  <Body>Arm3</Body>
  <offsetfrom>Base</offsetfrom>
```

```
  <weight>4</weight>
  <limitsdeg>-60 60</limitsdeg>
  <axis>0 0 1</axis>
  <maxveldeg>5.0</maxveldeg>
  <resolution>1</resolution>
  <anchor>4.3 0.0 4.3</anchor>
</Joint>
<!--Segmento 2 de la pata-->
<Body name="Arm31" type="dynamic">
  <offsetfrom>Arm3</offsetfrom>
  <Translation>0.0 -3.5 0.0</Translation>
  <Geom type="cylinder">
        <rotationaxis>0 0 0 0</rotationaxis>
    <radius>0.7</radius>
    <height>3.5</height>
    <diffuseColor>0 0 1</diffuseColor>
  </Geom>
  <Mass type="mimicgeom">
    <total>2.0</total>
  </Mass>
</Body>
<!--union segmentos de las patas -->
<Joint name="Arm31" type="hinge">
  <Body>Arm3</Body>
  <Body>Arm31</Body>
  <offsetfrom>Arm3</offsetfrom>
  <weight>4</weight>
```

```
    <limitsdeg>-60 60</limitsdeg>
    <axis>0 0 1</axis>
    <maxveldeg>5.0</maxveldeg>
    <resolution>1</resolution>
    <anchor>0.0 -1.75 0.0</anchor>
   </Joint>


 </KinBody>
</Robot>
```

Robot cuatro extremidades y dos grado de libertad, tipo araña: robot_spider.xml

```
<?xml version="1.0" ?><Robot name="robot_spider">
  <KinBody>
    <Body name="Base" type="dynamic">
      <Translation>0.0 0.0 0.0</Translation>
      <Geom type="box">
            <Extents>5.0 0.5 5.0</Extents>
            <RotationAxis>0 0 0 0</RotationAxis>
       <diffuseColor>0 0 1</diffuseColor>
      </Geom>
      <Mass type="box">
       <total>1.0</total>
      </Mass>
    </Body>
```

```
<!-- FIRST LEG-->
<Body name="Arm0" type="dynamic">
  <offsetfrom>Base</offsetfrom>
  <Translation>0.0 0.0 7.25</Translation>
  <Geom type="cylinder">
        <rotationaxis>1 0 0 90</rotationaxis>
    <radius>0.7</radius>
    <height>4.5</height>
    <diffuseColor>0 1 0</diffuseColor>
  </Geom>
  <Mass type="mimicgeom">
    <total>1.0</total>
  </Mass>
</Body>
<!--union pata con base -->
<Joint name="Arm0" type="hinge">
  <Body>Base</Body>
  <Body>Arm0</Body>
  <offsetfrom>Base</offsetfrom>
  <weight>4</weight>
  <limitsdeg>-60 60</limitsdeg>
  <axis>1 0 0</axis>
  <maxveldeg>10.0</maxveldeg>
  <resolution>1</resolution>
  <anchor>0.0 0.0 3.625</anchor>
</Joint>
<!--Segmento 2 de la pata-->
```

```
<Body name="Arm01" type="dynamic">
  <offsetfrom>Arm0</offsetfrom>
  <Translation>0.0 -2.95 1.55</Translation>
  <Geom type="cylinder">
        <rotationaxis>1 0 0 0</rotationaxis>
   <radius>0.7</radius>
   <height>4.5</height>
   <diffuseColor>0 1 0</diffuseColor>
  </Geom>
  <Mass type="mimicgeom">
   <total>1.0</total>
  </Mass>
</Body>
<!--union segmentos de las patas -->
<Joint name="Arm01" type="hinge">
  <Body>Arm0</Body>
  <Body>Arm01</Body>
  <offsetfrom>Arm0</offsetfrom>
  <weight>4</weight>
  <limitsdeg>-60 60</limitsdeg>
  <axis>1 0 0</axis>
  <maxveldeg>10.0</maxveldeg>
  <resolution>1</resolution>
  <anchor>0.0 0.0 1.55</anchor>
</Joint>

<!-- SECOND LEG-->
```

```
<Body name="Arm1" type="dynamic">
  <offsetfrom>Base</offsetfrom>
  <Translation>7.25 0.0 0.0</Translation>
  <Geom type="cylinder">
        <rotationaxis>0 0 1 90</rotationaxis>
    <radius>0.7</radius>
    <height>4.5</height>
    <diffuseColor>0 1 0</diffuseColor>
  </Geom>
  <Mass type="mimicgeom">
    <total>1.0</total>
  </Mass>
</Body>
<Joint name="Arm1" type="hinge">
  <Body>Base</Body>
  <Body>Arm1</Body>
  <offsetfrom>Base</offsetfrom>
  <weight>4</weight>
  <limitsdeg>-60 60</limitsdeg>
  <axis>0 0 1</axis>
  <maxveldeg>10.0</maxveldeg>
  <resolution>1</resolution>
  <anchor>3.625 0.0 0.0</anchor>
</Joint>
<!--Segmento 2 de la pata-->
<Body name="Arm11" type="dynamic">
  <offsetfrom>Arm1</offsetfrom>
```

```
  <Translation>1.55 -2.95 0.0</Translation>
  <Geom type="cylinder">
        <rotationaxis>1 0 0 0</rotationaxis>
   <radius>0.7</radius>
   <height>4.5</height>
   <diffuseColor>0 1 0</diffuseColor>
  </Geom>
  <Mass type="mimicgeom">
   <total>1.0</total>
  </Mass>
</Body>
<!--union segmentos de las patas -->
<Joint name="Arm11" type="hinge">
  <Body>Arm1</Body>
  <Body>Arm11</Body>
  <offsetfrom>Arm1</offsetfrom>
  <weight>4</weight>
  <limitsdeg>-60 60</limitsdeg>
  <axis>0 0 1</axis>
  <maxveldeg>10.0</maxveldeg>
  <resolution>1</resolution>
  <anchor>1.55 0.0 0.0</anchor>
</Joint>



<!-- THIRD LEG-->
<Body name="Arm2" type="dynamic">
```

```
  <offsetfrom>Base</offsetfrom>
  <Translation>0.0 0.0 -7.25</Translation>
  <Geom type="cylinder">
        <rotationaxis>1 0 0 90</rotationaxis>
   <radius>0.7</radius>
   <height>4.5</height>
   <diffuseColor>0 1 0</diffuseColor>
  </Geom>
  <Mass type="mimicgeom">
   <total>1.0</total>
  </Mass>
</Body>
<Joint name="Arm2" type="hinge">
  <Body>Base</Body>
  <Body>Arm2</Body>
  <offsetfrom>Base</offsetfrom>
  <weight>4</weight>
  <limitsdeg>-60 60</limitsdeg>
  <axis>1 0 0</axis>
  <maxveldeg>10.0</maxveldeg>
  <resolution>1</resolution>
  <anchor>0.0 0.0 -3.625</anchor>
</Joint>
<!--Segmento 2 de la pata-->
<Body name="Arm21" type="dynamic">
  <offsetfrom>Arm2</offsetfrom>
  <Translation>0.0 -2.95 -1.55</Translation>
```

```
  <Geom type="cylinder">
        <rotationaxis>1 0 0 0</rotationaxis>
   <radius>0.7</radius>
   <height>4.5</height>
   <diffuseColor>0 1 0</diffuseColor>
  </Geom>
  <Mass type="mimicgeom">
   <total>1.0</total>
  </Mass>
</Body>
<!--union segmentos de las patas -->
<Joint name="Arm21" type="hinge">
  <Body>Arm2</Body>
  <Body>Arm21</Body>
  <offsetfrom>Arm2</offsetfrom>
  <weight>4</weight>
  <limitsdeg>-60 60</limitsdeg>
  <axis>1 0 0</axis>
  <maxveldeg>10.0</maxveldeg>
  <resolution>1</resolution>
  <anchor>0.0 0.0 -1.55</anchor>
</Joint>


<!-- FOURTH LEG-->
<Body name="Arm3" type="dynamic">
  <offsetfrom>Base</offsetfrom>
```

```
  <Translation>-7.25 0.0 0.0</Translation>
  <Geom type="cylinder">
        <rotationaxis>0 0 1 90</rotationaxis>
   <radius>0.7</radius>
   <height>4.5</height>
   <diffuseColor>0 1 0</diffuseColor>
  </Geom>
  <Mass type="mimicgeom">
   <total>1.0</total>
  </Mass>
 </Body>
 <Joint name="Arm3" type="hinge">
  <Body>Base</Body>
  <Body>Arm3</Body>
  <offsetfrom>Base</offsetfrom>
  <weight>4</weight>
  <limitsdeg>-60 60</limitsdeg>
  <axis>0 0 1</axis>
  <maxveldeg>10.0</maxveldeg>
  <resolution>1</resolution>
  <anchor>-3.625 0.0 0.0</anchor>
 </Joint>
 <!--Segmento 2 de la pata-->
 <Body name="Arm31" type="dynamic">
  <offsetfrom>Arm3</offsetfrom>
  <Translation>-1.55 -2.95 0.0</Translation>
  <Geom type="cylinder">
```

```
        <rotationaxis>1 0 0 0</rotationaxis>
      <radius>0.7</radius>
      <height>4.5</height>
      <diffuseColor>0 1 0</diffuseColor>
    </Geom>
    <Mass type="mimicgeom">
      <total>1.0</total>
    </Mass>
  </Body>
  <!--union segmentos de las patas -->
  <Joint name="Arm31" type="hinge">
    <Body>Arm3</Body>
    <Body>Arm31</Body>
    <offsetfrom>Arm3</offsetfrom>
    <weight>4</weight>
    <limitsdeg>-60 60</limitsdeg>
    <axis>0 0 1</axis>
    <maxveldeg>10.0</maxveldeg>
    <resolution>1</resolution>
    <anchor>-1.55 0.0 0.0</anchor>
  </Joint>

 </KinBody>
</Robot>
```