Manual Técnico

Entorno de Simulación Robótico

Anthony Figueroa

pgsimrob@fing.edu.uy

Tutor

Gonzalo Tejera

Cotutores

Gustavo Armagno, Facundo Benavides, Serrana Casella

28 de julio de 2008

Instituto de Computación

Facultad de Ingeniería - Universidad de la República

Montevideo - Uruguay

Índice

1. Aspectos generales 5
   1.1. Descarga 5
   1.2. Instalación 5
   1.3. Ejecución 5
      1.3.1. Núcleo de simulación 5
      1.3.2. Interfaz para el envío de Comandos . . 5
      1.3.3. Demostración 5
   1.4. Agentes 5
   1.5. Comunicación 6
   1.6. Mensajes 6
   1.7. Sincronización 7

2. Comandos de control 9
   2.1. Registro de agente 9
   2.2. Información de agente 10
   2.3. Registro de materiales e interacciones . . . 10
   2.4. Creación de entidad 13
      2.4.1. Robot 17
      2.4.2. Campo de juego 20
      2.4.3. Pelota 24
   2.5. Inicializar simulación 25
   2.6. Especificar modo de paso de tiempo 25
   2.7. Especificar bandera de paso de tiempo 25
   2.8. Especificar modo de avance con ajuste automático 26
   2.9. Especificar modo de comunicación 26
   2.10. Sincronización con iteración 27
   2.11. Especificar tiempo de espera en comunicación sincrónica 27
   2.12. Especificar tamaño del paso de simulación . . 28
   2.13. Especificar gravedad 28
   2.14. Especificar restricciones de dureza global . 29
   2.15. Especificar restricciones de esponjosidad global 29
   2.16. Guardar simulación a archivo 29
   2.17. Cargar simulación desde archivo 30
   2.18. Especificación de modo rápido 30
   2.19. Especificación de pasos de tiempo por iteración 31

3. Comandos de entidad 33
   3.1. Especificar posición de entidad 33
   3.2. Controlar motores de entidad 33

4. Respuestas 35

5. Broadcast 37

6. Configuración 41

7. Extensibilidad 45
7.1. Modelo físico 45
7.2. Entidades 46
7.3. Comandos 47
7.4. Tipos de agentes y filtros 48

8. Interfaz para el envío de comandos 51

Índice de figuras

1. Mensajes 6
2. Empalme *WheelJoint* (extraído del sitio www.ode.org). 17
3. Extensibilidad modelo físico 46
4. Extensibilidad de entidades 47
5. Extensibilidad de comandos 48
6. Extensibilidad de tipos de agentes y filtros 49
7. SimRob Commander 51

1. Aspectos generales

1.1. Descarga

La descarga del código fuente o los binarios se puede realizar a través del sitio web http://www.fing.edu.uy/~pgsimrob en la sección "Descargas".

1.2. Instalación

La versión descargable que contiene la aplicación ya compilada no requiere instalación. La versión con el código fuente debe ser compilada utilizando el comando "make".

1.3. Ejecución

1.3.1. Núcleo de simulación

El ejecutable correspondiente al núcleo de simulación se llama "SimRob". Es posible ejecutar la aplicación con o sin interfaz gráfica, optando por el modo con interfaz gráfica a través del parámetro opcional "-gui". Ejemplo: "./SimRob -gui".

El sistema presenta diversos parámetros de configuración, que se describen en el capítulo 6.

1.3.2. Interfaz para el envío de Comandos

Para ejecutar la interfaz de envío de comandos, es necesario ejecutar el archivo FrontEnd.class (bajo la carpeta "bin"). Ejemplo: "java FrontEnd.class". Es importante destacar que las variables de entorno JAVAHOME y CLASSPATH deben estar correctamente configuradas.

1.3.3. Demostración

Esta disponible para su descarga un ejemplo de agente externo, ya compilado, pero que incluye el código fuente. Este archivo una vez ejecutado registra un agente externo, crea una simulación en la cual participan un robot y un campo de juego y luego capta eventos del teclado. Estos eventos permiten enviarle velocidades a las ruedas del robot creado. En la carpeta "src" de este ejemplo hay una serie de archivos en formato xml que permiten el correcto envío de mensajes, incluyendo la creación del modelo físico (SumBot.xml).

1.4. Agentes

La manera de interactuar con el sistema es a través de la ejecución de agentes que se conectan al mismo. En agente se registra al sistema especificando su ubicación y su tipo y a partir de este momento procede a enviarle todo tipo de mensajes con el objetivo de efectuar acciones.

1.5. Comunicación

La comunicación entre los agentes y el sistema o núcleo de simulación se realiza a través del protocolo UDP, a través del intercambio de mensajes. La dirección de red y el puerto en el cual se ejecuta el sistema se especifica en un archivo de configuración leído al momento de iniciar la ejecución. Los agentes le indican su dirección de red y puerto en el mensaje de registro.

1.6. Mensajes

En un mensaje UDP recibido por el núcleo de simulación pueden estar codificados diversos mensajes de pedidos. Estos mensajes de pedidos o "Request Messages" pueden ser de tipo "Agent Message" o "Simulation Message". El primero de estos tipos corresponde a mensajes que contienen un pedido para realizar alguna operación relacionada con agentes, como el registro de los mismos o el pedido de información de otro agente. El segundo encapsula mensajes que tienen como objetivo realizar alguna acción sobre la simulación. Estos mensajes son los denominados "Request" o pedidos y pueden ser de dos tipos: "Entity Request" y "Control Request".

Cada pedido especifica el comando que debe ser ejecutado y encapsula los datos necesarios para llevar a cabo dicho comando. Los pedidos de tipo entidad o "Entity Request" se traducen a comandos asociados a una entidad especifica en la simulación, de forma tal de realizar operaciones específicas sobre esa entidad, como por ejemplo controlar motores, mover entidades, etc. Los pedidos de control o "Control Request" no son tan específicos y se traducen a comandos que realizan acciones generales sobre la simulación, no asociados a ninguna entidad que ya exista. Ejemplos de estos pedidos son cambios en las propiedades físicas generales de la simulación, control del paso de tiempo, guardar la simulación, etc. Cabe destacar que la creación de una entidad también es un pedido de este tipo ya que la misma no existe antes de la ejecución de este comando. A continuación se muestra un diagrama que ilustra esta realidad.

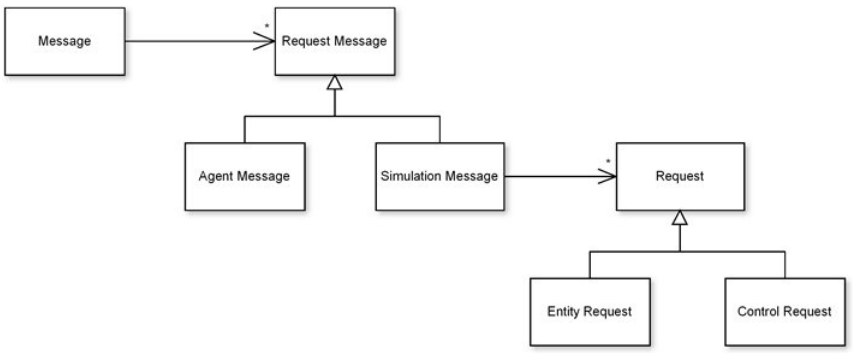

Figura 1: Mensajes

1.7. Sincronización

Los agentes conectados al sistema pueden registrarse en dos modos: sincrónico y asincrónico. Por otra parte, el sistema también puede estar ejecutándose en alguno de estos dos modos. A cada ciclo de iteración, el sistema realiza un broadcast con información acerca del estado de la simulación, enviando esta información a los agentes que corresponda. A continuación, su manera de actuar varía según cual sea su modo de ejecución.

El modo sincrónico implica que a cada ciclo de iteración, el simulador espere una notificación de cada agente conectado a el en modo sincrónico para avanzar la simulación. En caso que uno o más de estas notificaciones no lleguen, se espera un lapso de tiempo configurable y se avanza. Las notificaciones se realizan a través del envío de un mensaje que lleva a la ejecución de un comando responsable de sincronizar a un agente con una iteración. En caso que en determinada iteración, todos los agentes conectados en modo sincrónico estén sincronizados con la misma, se procede a avanzar hacia la siguiente iteración. Si no existen agentes conectados en modo sincrónico, el sistema actúa de manera análoga a la ejecución asincrónica.

Si el simulador está funcionando en modo asincrónico, no se realiza ninguna sincronización con los agentes conectados al mismo, sin importar si algunos agentes se registraron en modo sincrónico. Se avanza la simulación tan pronto sea posible, o sea, luego que se hayan terminado todas las tareas correspondientes a esa iteración, o sea, luego de la ejecución de todos los comandos relacionados a los mensajes recibidos durante la última iteración.

2. Comandos de control

2.1. Registro de agente

Todo intercambio de mensajes entre el sistema y un agente externo, comienza con el registro de dicho agente. Para esto el agente externo debe enviar un mensaje en el cual se especifican los siguientes datos:

- Clave del comando a ejecutar, en este caso es "Register_Agent"
- Tipo de agente: Con el objetivo de proveer un marco semántico y flexible para la interacción con los agentes, la arquitectura del sistema soporte la presencia de diversos tipos de agentes, cada uno de los cuales podría implementar ciertas particularidades. En la actualidad existen dos tipos de agentes, cuyo manejo en el núcleo de simulación es análogo, solamente están diferenciados con el objetivo de facilitar la implementación de filtros de información asociados a cada tipo de agente.
  - Control de robots: Agente con el objetivo de controlar robots. Su clave es "RobotController"
  - Interfaz gráfica: Agente con el objetivo de implementar una interfaz gráfica que refleje el estado de la simulación. Su clave es GUI.
- IP: Dirección de red del agente externo
- Puerto: Puerto del agente externo
- Modo de sincronización: Especifica si el agente desea conectarse en modo sincrónico al sistema. Sus posibles valores son "true" o "false".
- Tiempo de broadcast: En caso que el sistema se encuentre en modo asincrónico o el agente esté conectado en este modo, el tiempo de broadcast indica un intervalo de tiempo para el cual el agente está dispuesto a recibir información de broadcast. En caso que el valor sea -1 indica que el agente no quiere recibir información de broadcast.
- Clave: Identificador único mediante el cual se va a identificar el agente en el sistema

Un ejemplo de mensaje de registro de agente es el que se muestra a continuación.

```
<Message>
 <Agent_Message Command="Register_Agent" Agent_Type="RobotController">
    <Port>33153</Port>
    <IP>127.0.0.1</Port>
    <Sync>false</Sync>
    <Broadcast>0.1</Broadcast>
    <Key>Test</Key>
 </Agent_Message>
```

</Message>

En este caso se registra un agente con clave "Test", conectado de modo asincrónico en el puerto 33153 e IP 127.0.0.1, y va a recibir información acerca del estado del mundo a cada 0.1 segundos, en caso que exista información disponible. En caso que el registro se haga de forma correcta, el agente recibirá una respuesta del sistema con un mensaje que confirma la acción efectuada. El mensaje cuando se ha efectuado el registro con éxito es el siguiente.

<Response>Sucessfully added agent with key Test, port 33153 and ip 127.0.0.1</Response>

2.2. Información de agente

Eventualmente un agente podría realizar un pedido de información acerca de la ubicación de otro agente conectado al sistema. Para esto se debe especificar los siguientes parámetros:

- Clave del comando a ejecutar, en este caso es "Info_Request"
- Clave del agente que desea recibir información
- Clave del agente del cual se desea consultar información

En caso que la información recibida por el sistema sea coherente, se responderá con un mensaje conteniendo la información correspondiente.

Un ejemplo de un mensaje conteniendo este pedido se muestra a continuación.

```
<Message>
  <Agent_Message Command="Info_Request">
    <Requester>Test</Requester>
    <Requested>Test2</Requested>
  </Agent_Message>
</Message>
```

Una típica respuesta a este mensaje se puede ver a continuación.

```
<Response>
  <Agent_Information>
    <Key>Test2</Key>
    <Port>33154</Port>
    <IP>127.0.0.1</Port>
  </Agent_Information>
</Response>
```

2.3. Registro de materiales e interacciones

Al momento de crear entidades se debe especificar que cuerpos la componen. A cada cuerpo se le puede especificar un material. En caso que no se le especifique un material a un cuerpo, se le especificará uno por defecto. Además existe un comando que permite crear materiales y especificar interacciones. Este

debería ser ejecutado anteriormente a la creación de las entidades que utilizan los materiales que este comando registra. Cuando se detecta que dos cuerpos están colisionando, se busca los materiales que los componen y se busca en el sistema una interacción que regule el comportamiento de la colisión entre dichos materiales. En caso que se encuentre la interacción, se especifican los parámetros correspondientes para resolver la colisión. En caso contrario se asigna una interacción por defecto a la colisión y se procede de manera análoga.

Los parámetros que componen las interacciones se detallan a continuación.

- Bounciness: Cuando dos cuerpos están conectados por empalmes, el empalme implica que dichos cuerpos tengan cierta posición relativa entre sí. Durante la simulación puede ocurrir que dichos cuerpos se muevan de manera tal que esta posición relativa no sea la adecuada. Cuando esto ocurre, a cada paso de simulación se aplica una fuerza correctiva con el objetivo de llevar a los cuerpos a una posición correcta. Esta fuerza es controlada por este parametro, que puede tener un valor entre 0 y 1. Especifica que proporción del error se corregirá en la siguiente iteración. Su valor recomendado es entre 0.1 y 0.8. Cuando dos objetos colisionan, regula la fuerza que se ejerce a los cuerpos para que estos no se atraviesen.

- Hardness: Cuando dos cuerpos colisionan, esto implica que se deben cumplir algunas restricciones en cuanto al movimiento de ambos. Este parámetro denota que tan rígidas son estas restricciones. Su valor varía de 0 a 1. Utilizar valores menores implica restricciones más rígidas, contribuyendo para la estabilidad numérica y la velocidad de la simulación.

- Friction: Coeficiente de fricción (mu) aplicado al punto de contacto entre dos cuerpos, afectando a los mismos de la manera determinada por las leyes físicas. Su rango de posibles valores va de 0 hasta el infinito.

- Slip 1 y Slip 2: Estos parámetros implican que dos superficies en contacto se deslicen a una velocidad relativa proporcional a la fuerza aplicada tangentemente a la superficie. Son particularmente utiles para modelar interacciones entre ruedas y la superficie. Si se especifica la fuerza de rozamiento entre dichos cuerpos como infinita, la rueda girara perfectamente sobre si misma avanzando sobre el piso, sin deslizar hacia ninguna dirección. Sin embargo si se le aplica una fuerza perpendicular a su dirección podría desearse que la rueda se deslizara sobre la superficie en esa dirección. Sin embargo, como se especificó una fuerza de rozamiento infinita, esto no debería pasar. Para contemplar este tipo de casos se utilizan estos parámetros, uno en cada dirección del movimiento relativo entre los cuerpos.

En un mensaje de registro de materiales e interacciones se debe indicar los siguientes datos:

- Clave del comando a ejecutar, en este caso es "Register_Materials_Interactions"

- Cualquier cantidad de materiales, a los cuales se les debe indicar un identificador numérico y un nombre.
- Interacciones entre dichos materiales, y para cada interacción los parámetros ya explicados

Un ejemplo de mensaje utilizado para el registro de materiales e interacciones es el siguiente.

```
<Message>
  <Simulation_Message>
    <Control_Request Command="Register_Materials_Interactions">
      <Material>
        <Material_Id>0</Material_Id>
        <Material_Name>Wood</Material_Name>
      </Material>
      <Material>
        <Material_Id>1</Material_Id>
        <Material_Name>Metal</Material_Name>
      </Material>
      <Interaction>
        <Material_1>0</Material_1>
        <Material_2>0</Material_2>
        <Hardness>0.15</Hardness>
        <Bounciness>0.6</Bounciness>
        <Slip1>0.1</Slip1>
        <Slip2>0.1</Slip2>
        <Friction>1</Friction>
      </Interaction>
      <Interaction>
        <Material_1>0</Material_1>
        <Material_2>1</Material_2>
        <Hardness>0.1</Hardness>
        <Bounciness>0.7</Bounciness>
        <Slip1>0.1</Slip1>
        <Slip2>0.1</Slip2>
        <Friction>1</Friction>
      </Interaction>
      <Interaction>
        <Material_1>1</Material_1>
        <Material_2>1</Material_2>
        <Hardness>0.05</Hardness>
        <Bounciness>0.8</Bounciness>
        <Slip1>0.17</Slip1>
        <Slip2>0.17</Slip2>
        <Friction>1</Friction>
      </Interaction>
```

```
    </Control_Request>
  </Simulation_Message>
</Message>
```

2.4. Creación de entidad

Se debe definir que objetos van a a interactuar en la simulación. Los objetos son cuerpos y empalmes con determinada disposición en el mundo simulado que interactuarán entre sí. Para agrupar conjuntos de objetos relacionados, se presenta el concepto de entidades.

Concretamente una entidad está compuesta por cuerpos y empalmes que los conectan. En términos más abstractos se puede decir que una entidad agrupa comandos y restricciones. Agrupa comandos porque cada tipo de entidad tiene una serie de comandos de entidad relacionados a el. Estos efectúan operaciones sobre la entidad, como moverla o controlar motores en caso de que la entidad lo amerite. Agrupa restricciones porque ciertos tipos de entidades pueden estar conectados a ciertos tipos o cantidades de objetos.

Existen actualmente tres tipos de entidades: robots, campos de juego y pelota. Una entidad de tipo robot encapsula comandos para mover la entidad en el espacio, controlar los motores, entre otros. Además, no presenta restricciones en cuanto a los objetos que la componen, o sea, que puede contener la cantidad y tipos de cuerpos y empalmes que se desee. Una entidad de tipo campo de juego no presenta comandos asociados a ella, ya que no se consideró necesario. Por otra parte, presenta restricciones en cuanto a los objetos que la componen; esta entidad está asociada a cualquier cantidad de cuerpos, pero no admite empalmes en su composición. Una entidad de tipo pelota encapsula el comando que la mueve en la simulación. También presenta restricciones en cuanto a su composición, ya que solamente puede estar compuesta por un cuerpo, y ese cuerpo debe ser de tipo esfera.

Para entender de que manera se modelan las entidades, hay que comprender las propiedades de dos conceptos básicos: cuerpos y empalmes. Estos se crean en el contexto de la creación de una entiudad, no poseen un comando específico para cada cuerpo o empalme.

Cuerpos

Son componentes fundamentales de las entidades. Representan cuerpos geométricos como cubos, prismas, y esferas. Poseen diversas propiedades importantes comunes a todos los cuerpos:

- Identificador: Cada cuerpo creado debe tener un identificador único en la entidad, o sea, el identificador es local a la entidad que lo encapsula.
- Masa: Es un valor mayor que 0, representa la masa en kg del cuerpo. Existe un valor especial, el -1, que representa un cuerpo de masa infinita.
- Posición: Coordenadas en cada uno de los 3 ejes.

- Rotación: Rotación en radianes con respecto a cada uno de los 3 ejes.
- Velocidad angular: No se especifica en el momento de su creación.
- Velocidad lineal: No se especifica en el momento de su creación.
- Material: Identificador del material, es opcional.

En particular, el sistema soporta en la actualidad la creación de los siguientes cuerpos:

- Sphere: Representa una esfera, su propiedad específca es su radio. Un ejemplo de la sintáxis que representa una esfera en la creación de un robot es la siguiente:

  <Body Type="Sphere">
  <Body_id>1<Body_id>
  <Material>1</Material>
  <Mass>1.3</Mass>
  <Position_X>1</Position_X>
  <Position_Y>0.3</Position_Y>
  <Position_Z>1.8</Position_Z>
  <Rotation_X>1.570796</Rotation_X>
  <Rotation_Y>0</Rotation_Y>
  <Rotation_Z>0</Rotation_Z>
  <Radius>0.023</Radius>
  </Body>

- Box: Representa un prisma, cuyas dimensiones se especifican en la dirección de cada uno de los 3 ejes. Un ejemplo de la sintáxis que representa un cuerpo de este tipo en la creación de un robot es la siguiente:

  <Body Type="Box">
  <Body_id>2<Body_id>
  <Material>1</Material>
  <Mass>1.2</Mass>
  <Position_X>1</Position_X>
  <Position_Y>0.3</Position_Y>
  <Position_Z>1.8</Position_Z>
  <Rotation_X>0</Rotation_X>
  <Rotation_Y>0</Rotation_Y>
  <Rotation_Z>0</Rotation_Z>

<Size_X>0.075</Size_X>
<Size_Y>0.075</Size_Y>
<Size_Z>0.075</Size_Z>
</Body>

- Cylinder: Representa un cilindro, cuyas dimensiones se especifican mediante el radio y su longitud. Un ejemplo de la sintáxis que representa un cuerpo de este tipo en la creación de un robot es la siguiente:

<Body Type="Cylinder">
<Body_id>3<Body_id>
<Material>1</Material>
<Mass>0.5</Mass>
<Position_X>1</Position_X>
<Position_Y>0.3</Position_Y>
<Position_Z>1.8</Position_Z>
<Rotation_X>0</Rotation_X>
<Rotation_Y>0</Rotation_Y>
<Rotation_Z>0</Rotation_Z>
<Radius>0.023</Radius>
<Length>0.3</Length>
</Body>

- WheelCylinder: Representa un cilindro para ser utilizado como una rueda, cuyas dimensiones se especifican mediante el radio y su longitud. Internamente se representa como una esfera con colisiones con otros cuerpos del mismo tipo deshabilitadas, esto las hace ideales para modelar ruedas en robots móviles. Un ejemplo de la sintáxis que representa un cuerpo de este tipo en la creación de un robot es la siguiente:

<Body Type="WheelCylinder">
<Body_id>4<Body_id>
<Material>2</Material>
<Mass>0.5</Mass>
<Position_X>1</Position_X>
<Position_Y>0.3</Position_Y>
<Position_Z>1.8</Position_Z>
<Rotation_X>0</Rotation_X>
<Rotation_Y>0</Rotation_Y>
<Rotation_Z>0</Rotation_Z>

<Radius>0.023</Radius>
<Length>0.3</Length>
</Body>

Emapalmes

Son componentes que conectan dos cuerpos, aplicando ciertas restricciones en el movimiento relativo de ambos. Poseen algunas propiedades importantes comunes a todos los emaplmes:

- Identificador: Cada emapalme creado debe tener un identificador único en la entidad, o sea, el identificador es local a la entidad que lo encapsula.
- Cuerpos: Identificadores de los dos cuerpos que conecta.

Actualmente el sistema soporta solamente dos tipos de empalmes, que se explican a continuación.

- Fixed: Mantiene fijas las posiciones relativas y orientaciones de los dos cuerpos que conecta. Un ejemplo de la sintáxis que representa un empalme de este tipo en la creación de un robot es la siguiente:

  <Joint Type="FixedJoint">
  <Joint_id>1<Joint_id>
  <Body_1>1</Body_1>
  <Body_2>2</Body_2>
  </Joint>

- WheelJoint: Es ideal para modelar la conexión entre ruedas y el chasis de un robot. Controlando la velocidad de este empalme se pueden mover las ruedas y así controlar al robot. Presenta algunas propiedades específicas importantes que se detallan a continuación.

  - Ejes: Este empalme tiene dos ejes a los cuales se les puede especificar la dirección mediante un vector.

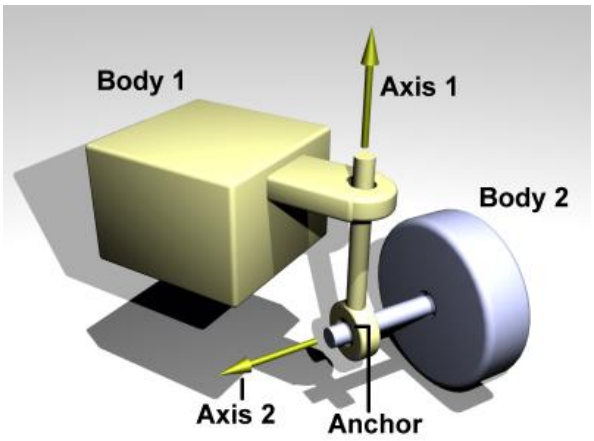

Figura 2: Empalme *WheelJoint* (extraído del sitio www.ode.org).

- Dureza o Hardness: Análogo a lo explicado en la sección 2.3
- Esponjosidad o Bounciness: Análogo a lo explicado en la sección 2.3
- Torque máximo: Es el torque máximo que se le puede aplicar al emaplme para cambiar la velocidad del mismo.

Un ejemplo de la sintáxis que representa un empalme de este tipo en la creación de un robot es la siguiente:

```
<Joint Type="WheelJoint">
  <Joint_id>1<Joint_id>
  <Body_1>1</Body_1>
  <Body_2>2</Body_2>
  <Axis_1_X>1</Axis_1_X>
  <Axis_1_Y>0</Axis_1_Y>
  <Axis_1_Z>0</Axis_1_Z>
  <Axis_2_X>0</Axis_2_X>
  <Axis_2_Y>1</Axis_2_Y>
  <Axis_2_Z>0</Axis_2_Z>
  <Hardness>0.001</Hardness>
  <Bounciness>0.9</Bounciness>
  <Maximum_Torque>10</Maximum_Torque>
</Joint>
```

2.4.1. Robot

Para la creación de una entidad de tipo robot se deben especificar los siguientes parámetros:

- Clave del comando a ejecutar, en este caso es ”Create_Entity”
- Tipo de la entidad a ser creada, en este caso es ”Robot”
- Estructura física del robot, que debe incluir cuerpos y empalmes.
- Los motores, que se asocian a empalmes de tipo ”WheelJoint”, para cada motor se debe especificar el identificador del empalme asociado a el.
- Cuerpo de referencia: Al momento de hacer broadcast, los datos del robot que se trasmiten en realidad son los datos de un cuerpo de referencia que compone al robot. Se debe especificar el identificador de este cuerpo.

La sintáxis de un mensaje que ejecuta este comando es la siguiente.

```
<Control_Request Command="Create_Entity" Type="Robot">
  <Data>
    <Entity_id>3</Entity_id>
    <Structure>
      <Body Type="Box">
        <Body_id>0</Body_id>
        <Material>0</Material>
        <Mass>1.2</Mass>
        <Position_X>0</Position_X>
        <Position_Y>-0.1</Position_Y>
        <Position_Z>0.35</Position_Z>
        <Rotation_X>0</Rotation_X>
        <Rotation_Y>0</Rotation_Y>
        <Rotation_Z>0</Rotation_Z>
        <Size_X>0.075</Size_X>
        <Size_Y>0.075</Size_Y>
        <Size_Z>0.075</Size_Z>
      </Body>
      <Body Type="WheelCylinder">
        <Body_id>1</Body_id>
        <Material>1</Material>
        <Mass>0.1</Mass>
        <Position_X>0</Position_X>
        <Position_Y>-0.132925</Position_Y>
        <Position_Z>0.3305</Position_Z>
        <Rotation_X>1.570796</Rotation_X>
        <Rotation_Y>0</Rotation_Y>
        <Rotation_Z>0</Rotation_Z>
        <Radius>0.023</Radius>
        <Length>0.04</Length>
      </Body>
      <Body Type="WheelCylinder">
        <Body_id>2</Body_id>
```

```
      <Material>1</Material>
      <Mass>0.1</Mass>
      <Position_X>0</Position_X>
      <Position_Y>-0.067075</Position_Y>
      <Position_Z>0.3305</Position_Z>
      <Rotation_X>1.570796</Rotation_X>
      <Rotation_Y>0</Rotation_Y>
      <Rotation_Z>0</Rotation_Z>
      <Radius>0.023</Radius>
      <Length>0.04</Length>
    </Body>
    <Joint Type="WheelJoint">
      <Joint_id>0</Joint_id>
      <Body_1>0</Body_1>
      <Body_2>1</Body_2>
      <Axis_1_X>1</Axis_1_X>
      <Axis_1_Y>0</Axis_1_Y>
      <Axis_1_Z>0</Axis_1_Z>
      <Axis_2_X>0</Axis_2_X>
      <Axis_2_Y>1</Axis_2_Y>
      <Axis_2_Z>0</Axis_2_Z>
      <Hardness>0.001</Hardness>
      <Bounciness>0.9</Bounciness>
      <Maximum_Torque>10</Maximum_Torque>
    </Joint>
    <Joint Type="WheelJoint">
      <Joint_id>1</Joint_id>
      <Body_1>0</Body_1>
      <Body_2>2</Body_2>
      <Axis_1_X>1</Axis_1_X>
      <Axis_1_Y>0</Axis_1_Y>
      <Axis_1_Z>0</Axis_1_Z>
      <Axis_2_X>0</Axis_2_X>
      <Axis_2_Y>1</Axis_2_Y>
      <Axis_2_Z>0</Axis_2_Z>
      <Hardness>0.001</Hardness>
      <Bounciness>0.9</Bounciness>
      <Maximum_Torque>10</Maximum_Torque>
    </Joint>
  </Structure>
  <Motors>
    <Motor_id>0</Motor_id>
    <Motor_id>1</Motor_id>
  </Motors>
  <Reference_Body>0</Reference_Body>
</Data>
```

</Control_Request>

Este mensaje crea una entidad de tipo robot que está compuesto por un cubo que constituye el chasis y dos ruedas a ambos lados, ambas ruedas están unidas al chasis mediante empalmes de tipo "WheelJoint". Además presenta motores asociados a cada uno de estos empalmes.

2.4.2. Campo de juego

Para la creación de una entidad de tipo campo de juego se deben especificar los siguientes parámetros:

- Clave del comando a ejecutar, en este caso es "Create_Entity"
- Tipo de la entidad a ser creada, en este caso es "Field"
- Estructura física del campo de juego, que debe ser una lista de cuerpos.

La sintáxis de un mensaje que ejecuta este comando es la siguiente.

```
<Control_Request Command="Create_Entity" Type="Field">
  <Data>
  <Entity_id>7</Entity_id>
  <Structure>
    <Body Type="Box">
      <Body_id>0</Body_id>
      <Material>0</Material>
      <Mass>-1</Mass>
      <Position_X>0</Position_X>
      <Position_Y>0</Position_Y>
      <Position_Z>0</Position_Z>
      <Rotation_X>0</Rotation_X>
      <Rotation_Y>0</Rotation_Y>
      <Rotation_Z>0</Rotation_Z>
      <Size_X>2.8</Size_X>
      <Size_Y>1.82</Size_Y>
      <Size_Z>0.1</Size_Z>
    </Body>
    <Body Type="Box">
      <Body_id>1</Body_id>
      <Material>0</Material>
      <Mass>-1</Mass>
      <Position_X>0</Position_X>
      <Position_Y>-0.9</Position_Y>
      <Position_Z>0</Position_Z>
      <Rotation_X>0</Rotation_X>
      <Rotation_Y>0</Rotation_Y>
      <Rotation_Z>0</Rotation_Z>
      <Size_X>2.2</Size_X>
```

```
    <Size_Y>0.02</Size_Y>
    <Size_Z>0.4</Size_Z>
  </Body>
  <Body Type="Box">
    <Body_id>2</Body_id>
    <Material>0</Material>
    <Mass>-1</Mass>
    <Position_X>0</Position_X>
    <Position_Y>0.9</Position_Y>
    <Position_Z>0</Position_Z>
    <Rotation_X>0</Rotation_X>
    <Rotation_Y>0</Rotation_Y>
    <Rotation_Z>0</Rotation_Z>
    <Size_X>2.2</Size_X>
    <Size_Y>0.02</Size_Y>
    <Size_Z>0.4</Size_Z>
  </Body>
  <Body Type="Box">
    <Body_id>3</Body_id>
    <Material>0</Material>
    <Mass>-1</Mass>
    <Position_X>1.1</Position_X>
    <Position_Y>-0.56</Position_Y>
    <Position_Z>0</Position_Z>
    <Rotation_X>0</Rotation_X>
    <Rotation_Y>0</Rotation_Y>
    <Rotation_Z>1.5708</Rotation_Z>
    <Size_X>0.7</Size_X>
    <Size_Y>0.02</Size_Y>
    <Size_Z>0.4</Size_Z>
  </Body>
  <Body Type="Box">
    <Body_id>4</Body_id>
    <Material>0</Material>
    <Mass>-1</Mass>
    <Position_X>1.1</Position_X>
    <Position_Y>0.56</Position_Y>
    <Position_Z>0</Position_Z>
    <Rotation_X>0</Rotation_X>
    <Rotation_Y>0</Rotation_Y>
    <Rotation_Z>1.5708</Rotation_Z>
    <Size_X>0.7</Size_X>
    <Size_Y>0.02</Size_Y>
    <Size_Z>0.4</Size_Z>
  </Body>
  <Body Type="Box">
```

```
    <Body_id>5</Body_id>
    <Material>0</Material>
    <Mass>-1</Mass>
    <Position_X>-1.1</Position_X>
    <Position_Y>-0.56</Position_Y>
    <Position_Z>0</Position_Z>
    <Rotation_X>0</Rotation_X>
    <Rotation_Y>0</Rotation_Y>
    <Rotation_Z>1.5708</Rotation_Z>
    <Size_X>0.7</Size_X>
    <Size_Y>0.02</Size_Y>
    <Size_Z>0.4</Size_Z>
  </Body>
  <Body Type="Box">
    <Body_id>6</Body_id>
    <Material>0</Material>
    <Mass>-1</Mass>
    <Position_X>-1.1</Position_X>
    <Position_Y>0.56</Position_Y>
    <Position_Z>0</Position_Z>
    <Rotation_X>0</Rotation_X>
    <Rotation_Y>0</Rotation_Y>
    <Rotation_Z>1.5708</Rotation_Z>
    <Size_X>0.7</Size_X>
    <Size_Y>0.02</Size_Y>
    <Size_Z>0.4</Size_Z>
  </Body>
  <Body Type="Box">
    <Body_id>7</Body_id>
    <Material>0</Material>
    <Mass>-1</Mass>
    <Position_X>1.2</Position_X>
    <Position_Y>0.22</Position_Y>
    <Position_Z>0</Position_Z>
    <Rotation_X>0</Rotation_X>
    <Rotation_Y>0</Rotation_Y>
    <Rotation_Z>0</Rotation_Z>
    <Size_X>0.21</Size_X>
    <Size_Y>0.02</Size_Y>
    <Size_Z>0.4</Size_Z>
  </Body>
  <Body Type="Box">
    <Body_id>8</Body_id>
    <Material>0</Material>
    <Mass>-1</Mass>
    <Position_X>-1.2</Position_X>
```

```
    <Position_Y>0.22</Position_Y>
    <Position_Z>0</Position_Z>
    <Rotation_X>0</Rotation_X>
    <Rotation_Y>0</Rotation_Y>
    <Rotation_Z>0</Rotation_Z>
    <Size_X>0.21</Size_X>
    <Size_Y>0.02</Size_Y>
    <Size_Z>0.4</Size_Z>
  </Body>
  <Body Type="Box">
    <Body_id>9</Body_id>
    <Material>0</Material>
    <Mass>-1</Mass>
    <Position_X>1.3</Position_X>
    <Position_Y>0</Position_Y>
    <Position_Z>0</Position_Z>
    <Rotation_X>0</Rotation_X>
    <Rotation_Y>0</Rotation_Y>
    <Rotation_Z>1.5708</Rotation_Z>
    <Size_X>0.46</Size_X>
    <Size_Y>0.02</Size_Y>
    <Size_Z>0.4</Size_Z>
  </Body>
  <Body Type="Box">
    <Body_id>10</Body_id>
    <Material>0</Material>
    <Mass>-1</Mass>
    <Position_X>-1.3</Position_X>
    <Position_Y>0</Position_Y>
    <Position_Z>0</Position_Z>
    <Rotation_X>0</Rotation_X>
    <Rotation_Y>0</Rotation_Y>
    <Rotation_Z>1.5708</Rotation_Z>
    <Size_X>0.46</Size_X>
    <Size_Y>0.02</Size_Y>
    <Size_Z>0.4</Size_Z>
  </Body>
  <Body Type="Box">
    <Body_id>11</Body_id>
    <Material>0</Material>
    <Mass>-1</Mass>
    <Position_X>1.2</Position_X>
    <Position_Y>-0.22</Position_Y>
    <Position_Z>0</Position_Z>
    <Rotation_X>0</Rotation_X>
    <Rotation_Y>0</Rotation_Y>
```

```
      <Rotation_Z>0</Rotation_Z>
      <Size_X>0.21</Size_X>
      <Size_Y>0.02</Size_Y>
      <Size_Z>0.4</Size_Z>
    </Body>
    <Body Type="Box">
      <Body_id>12</Body_id>
      <Material>0</Material>
      <Mass>-1</Mass>
      <Position_X>-1.2</Position_X>
      <Position_Y>-0.22</Position_Y>
      <Position_Z>0</Position_Z>
      <Rotation_X>0</Rotation_X>
      <Rotation_Y>0</Rotation_Y>
      <Rotation_Z>0</Rotation_Z>
      <Size_X>0.21</Size_X>
      <Size_Y>0.02</Size_Y>
      <Size_Z>0.4</Size_Z>
    </Body>
  </Data>
</Control_Request>
```

Este mensaje crea una entidad de tipo campo de juego formado por una serie de cuerpos de masa infinita (propiedad Mass con valor -1). En este caso se crea entonces un campo de juego con características aptas para realizar competencias de fútbol de robots.

2.4.3. Pelota

Para la creación de una entidad de tipo pelota se deben especificar los siguientes parámetros:

- Clave del comando a ejecutar, en este caso es "Create_Entity"
- Tipo de la entidad a ser creada, en este caso es "Ball"
- Propiedades físicas de la pelota, análogas a la creación de un cuerpo de tipo esfera.

La sintáxis de un mensaje que ejecuta este comando es la siguiente.

```
<Control_Request Command="Create_Entity" Type="Ball">
  <Data>
    <Entity_id>8</Entity_id>
    <Material>1</Material>
    <Mass>0.6</Mass>
    <Position_X>1.2</Position_X>
    <Position_Y>0</Position_Y>
    <Position_Z>0.6</Position_Z>
```

```
      <Rotation_X>0</Rotation_X>
      <Rotation_Y>0</Rotation_Y>
      <Rotation_Z>0</Rotation_Z>
      <Radius>0.03</Radius>
    </Data>
  </Control_Request>
```

Este mensaje crea una entidad de tipo pelota de radio 0.03 metros.

2.5. Inicializar simulación

Una vez iniciado el sistema, para que efectivamente se comienza a avanzar el modelo físico del mismo, es necesario ejecutar este comando. Su sintáxis se muestra a continuación.

```
<Message>
  <Simulation_Message>
    <Control_Request Command = "Initialize_Simulation">
    </Control_Request>
  </Simulation_Message>
</Message>
```

2.6. Especificar modo de paso de tiempo

Existen dos modos de paso de tiempo: normal y paso a paso. En el modo normal siempre que se hayan cumplido las tareas previas se va a avanzar un paso la simulación, en el modo paso a paso se le debe indicar explicitamente al sistema (a través de un mensaje) que se desea avanzar un paso la simulación. Para esto se debe especificar los siguientes parámetros:

- Clave del comando a ejecutar, en este caso es "Set_Step_Mode"
- Modo paso a paso: este parámetro puede tener dos valores: true(activa le modo paso a paso) o false (desactiva el modo paso a paso)

La sintáxis de un mensaje que ejecuta este comando es la siguiente.

```
<Message>
  <Simulation_Message>
    <Control_Request Command = "Set_Step_Mode">
      <Step_Mode>true</Step_Mode>
    </Control_Request>
  </Simulation_Message>
</Message>
```

2.7. Especificar bandera de paso de tiempo

Cuando el modo de paso de tiempo del sistema está en modo paso a paso, para pasar a la siguiente iteración, se debe enviar un mensaje de este tipo para especificar la bandera de paso de tiempo y otorgarle el permiso al simulador

para avanzar. Para esto se debe especificar solamente la clave del comando a ejecutar, que en este caso es "Make_Step".

La sintáxis de un mensaje que ejecuta este comando es la siguiente.

```
<Message>
  <Simulation_Message>
    <Control_Request Command = "Make_Step">
    </Control_Request>
  </Simulation_Message>
</Message>
```

2.8. Especificar modo de avance con ajuste automático

Esta funcionalidad permite pasar un modo de avance de la simulación en el cual se ajustan automáticamente los parámetros de tamaño de paso de simulación y cantidad de pasos por iteración, de forma tal que el tiempo simulado transcurra a velocidad similar al tiempo real.

La sintáxis de un mensaje que ejecuta este comando es la siguiente.

```
<Message>
  <Simulation_Message>
    <Control_Request Command = Set_AutoAdjust_Mode">
      <AutoAdjust_Mode>true</AutoAdjust_Mode>
    </Control_Request>
  </Simulation_Message>
</Message>
```

2.9. Especificar modo de comunicación

El sistema se puede ejecutar en modo sincrónico o asincrónico, y puede pasar de un modo a otro en cualquier momento. A través de este mensaje se puede especificar si el modo de ejecución del sistema es sincrónico o asincrónico. Para esto se deben especificar los siguientes parámetros:

- Clave del comando a ejecutar, en este caso es "Set_Sync_Mode"
- Modo Sincrónico: este parámetro se utiliza para definir si el modo de ejecución es sincrónico o no. Puede tener dos posibles valores: true (modo sincrónico) o false (modo asincrónico)

La sintáxis de un mensaje que ejecuta este comando es la siguiente.

```
<Message>
  <Simulation_Message>
    <Control_Request Command = "Set_Sync_Mode">
      <Sync_Mode>true</Sync_Mode>
    </Control_Request>
  </Simulation_Message>
</Message>
```

2.10. Sincronización con iteración

Cuando un agente se conecta en modo sincrónico y el simulador está funcionando en modo sincrónico, el agente debe notificarle al sistema que ya ha realizado todas las tareas correspondientes a una iteración y está listo para que el simulador avance hacia la siguiente. Esto se hace con un mensaje que ejecuta este comando, sincronizando un agente con determinada iteración. Para realizar esta acción se deben especificar los siguientes parámetros:

- Clave del comando a ejecutar, en este caso es "Set_Sync_Mode"
- Clave del agente que se quiere sincronizar
- Número de iteración con la cual se desea sincronizar

Un ejemplo de mensaje que ejecuta este comando es el siguiente.

```
<Message>
  <Agent_Message Command="Synchronize">
    <Key>Test</Key>
    <Iteration>1234</Iteration>
  </Agent_Message>
</Message>
```

En este caso se va a sincronizar el agente con clave "Test" con la iteración 1234.

2.11. Especificar tiempo de espera en comunicación sincrónica

Cuando el sistema está funcionando en modo sincrónico y existe algún agente conectado en este modo, siempre que se realiza el broadcast en determinada iteración luego se espera que los distintos agentes conectados al sistema envíen un mensaje de sincronización con la iteración para seguir avanzando la simulación. Esto puede resultar un problema cuando alguno de los agentes no está disponible, por lo que se establece que si no se recibe una sincronización de un agente con determinada iteración luego de cierto lapso de tiempo de realizado el broadcast, se avanza la simulación sin seguir esperando una respuesta. Este lapso de tiempo tiene un valor por defecto configurable, pero puede ser modificado durante la ejecución de la simulación.

Para realizar esta acción se deben especificar los siguientes parámetros:

- Clave del comando a ejecutar, en este caso es "Set_Sync_Wait"
- Valor en segundos del tiempo de espera en comunicación sincrónica

Un ejemplo de mensaje que ejecuta este comando es el siguiente.

```
<Message>
  <Simulation_Message>
    <Control_Request Command = "Set_Sync_Wait">
```

```
      <Sync_Wait>0.2</Sync_Wait>
    </Control_Request>
  </Simulation_Message>
</Message>
```

2.12. Especificar tamaño del paso de simulación

A cada ciclo de iteración se avanza el estado del mundo en una cierta cantidad de segundos. Esta variable refleja la cantidad de segundos que se avanza. El rango recomendado se encuentre entre los 0.010 y 0.070 segundos. Cuanto mayor es el valor de esta variable, más rápido se avanza, pero en contrapartida se pierde precisión numérica. Es una variable que debe ser ajustada de acuerdo a la complejidad de la simulación y al hardware en el cual se está ejecutando.

Para realizar esta acción se deben especificar los siguientes parámetros:

- Clave del comando a ejecutar, en este caso es "Set_Timestep".
- Valor en segundos del tamaño del paso de simulación

Un ejemplo de mensaje que ejecuta este comando es el siguiente.

```
<Message>
  <Simulation_Message>
    <Control_Request Command = "Set_Timestep">
      <Timestep>0.3</Timestep>
    </Control_Request>
  </Simulation_Message>
</Message>
```

2.13. Especificar gravedad

Es posible cambiar a cualquier momento la fuerza de gravedad que actúa sobre el modelo físico de la simulación. Para realizar esta acción se deben especificar los siguientes parámetros:

- Clave del comando a ejecutar, en este caso es "Set_World_Gravity".
- Valor dado a la gravedad en m/s^2

Un ejemplo de mensaje que ejecuta este comando es el siguiente.

```
<Message>
  <Simulation_Message>
    <Control_Request Command = "Set_World_Gravity">
      <Gravity>-9.8</Gravity>
    </Control_Request>
  </Simulation_Message>
</Message>
```

2.14. Especificar restricciones de dureza global

Este parámetro denota que tan rígidas son las restricciones que deben cumplir los distintos cuerpos con base a su estado actual y las fuerzas que actúan sobre el. Su valor varía de 0 a 1. Utilizar valores menores implica restricciones más rígidas, contribuyendo para la estabilidad numérica y la velocidad de la simulación. Para realizar esta acción se deben especificar los siguientes parámetros:

- Clave del comando a ejecutar, en este caso es "Set_World_Hardness".
- Valor de este parámetro

Un ejemplo de mensaje que ejecuta este comando es el siguiente.

```
<Message>
  <Simulation_Message>
    <Control_Request Command = "Set_World_Hardness">
      <Hardness>0.9</Hardness>
    </Control_Request>
  </Simulation_Message>
</Message>
```

2.15. Especificar restricciones de esponjosidad global

Este parámetro denota que porcentaje del error introducido en una iteración se intentará corregir en la siguiente usando fuerzas correctivas. Estas fuerzas son controladas por este parametro, que puede tener un valor entre 0 y 1. Especifica que proporción del error se corregirá en la siguiente iteración. Su valor recomendado es entre 0.1 y 0.8. Para realizar esta acción se deben especificar los siguientes parámetros:

- Clave del comando a ejecutar, en este caso es "Set_World_Bounciness".
- Valor de este parámetro

Un ejemplo de mensaje que ejecuta este comando es el siguiente.

```
<Message>
  <Simulation_Message>
    <Control_Request Command = "Set_World_Bounciness">
      <Bounciness>0.2</Bounciness>
    </Control_Request>
  </Simulation_Message>
</Message>
```

2.16. Guardar simulación a archivo

A través de esta funcionalidad se puede almacenar el estado de una simulación en un archivo. No solamente los datos de los distintos objetos físicos se

almacenan, sino también todos los parámetros que regulan la simulación, como el tamaño del paso de tiempo, el modo de ejecución, etc.

Para realizar esta acción se deben especificar los siguientes parámetros:

- Clave del comando a ejecutar, en este caso es "Save".
- Ruta del archivo en el cual se va a almacenar la simulación. Es importante notar que el usuario debe contar con los permisos suficientes para crear un archivo en esa ubicación.

Un ejemplo de mensaje que ejecuta este comando es el siguiente.

```
<Message>
  <Simulation_Message>
    <Control_Request Command = "Save">
      <File>/home/current_user/simulation.xml</File>
    </Control_Request>
  </Simulation_Message>
</Message>
```

2.17. Cargar simulación desde archivo

Se puede cargar el estado de la simulación desde un archivo previamente guardado.

Para realizar esta acción se deben especificar los siguientes parámetros:

- Clave del comando a ejecutar, en este caso es "Load".
- Ruta del archivo desde el cual se va a cargar la simulación. Es importante notar que el usuario debe contar con los permisos suficientes para leer un archivo en esa ubicación.

Un ejemplo de mensaje que ejecuta este comando es el siguiente.

```
<Message>
  <Simulation_Message>
    <Control_Request Command = "Load">
      <File>/home/current_user/simulation.xml</File>
    </Control_Request>
  </Simulation_Message>
</Message>
```

2.18. Especificación de modo rápido

El sistema puede avanzar un paso el estado del modelo físico en dos modos distintos:

- Normal

- Rápido: en este modo se pierde precisión y estabilidad numérica y se gana velocidad; es especialmente útil cuando se modelan realidades muy complejas.

A través de la ejecución de este comando se puede cambiar este modo a cualquier momento. Para realizar esta acción se deben especificar los siguientes parámetros:

- Clave del comando a ejecutar, en este caso es "Set_Fast_Mode".
- Valor para el parámetro de modo rápido, puede ser true (modo rápido) o false (modo normal).

Un ejemplo de mensaje que ejecuta este comando es el siguiente.

```
<Message>
  <Simulation_Message>
    <Control_Request Command = "Set_Fast_Mode">
      <Fast_Mode>true</Fast_Mode>
    </Control_Request>
  </Simulation_Message>
</Message>
```

2.19. Especificación de pasos de tiempo por iteración

Cuando la cantidad de agentes conectados al sistema es muy grande, el tráfico de mensajes puede eventualmente ser un cuello de botella, ya que el sistema podría pasar la gran mayor parte de su tiempo decodificando mensajes y ejecutando comandos. Esta variable especifica cuantos pasos se avanza la simulación física a cada paso de simulación del simulador. Esto implica que el intercambio de mensajes no se hará a cada paso en el modelo físico, sino que se darán tantos pasos como se especifique antes de proceder a la decodificación de mensajes nuevos, ejecución de comandos y broadcast de información. Es probablemente la manera más práctica y directa de avanzar la simulación a mayor velocidad que la velocidad normal. Aumentar este parámetro no implica pérdida de precisión numérica, solamente implica que desde el punto de vista de los agentes conectados al sistema, estos van a recibir información del sistema a cada mayor período de tiempo, y sus comandos no se ejecutarán a cada ciclo de iteración del modelo físico. Por lo tanto, desde el punto de vista de dichos agentes, el paso de simulación es de (Tamaño del paso de simulación)*(Cantidad de pasos por iteración) segundos.

Para realizar esta acción se deben especificar los siguientes parámetros:

- Clave del comando a ejecutar, en este caso es "Set_Steps_Per_Iteration".
- Cantidad de pasos por iteración, su valor por defecto es 1.

Un ejemplo de mensaje que ejecuta este comando es el siguiente.

```
<Message>
  <Simulation_Message>
    <Control_Request Command = "Set_Steps_Per_Iteration">
      <Steps_Per_Iteration>3</Steps_Per_Iteration>
    </Control_Request>
  </Simulation_Message>
</Message>
```

3. Comandos de entidad

Los comandos de entidad tienen como objetivo realizar alguna acción sobre alguna entidad de la simulación. Para esto cada comando de este tipo tiene una entidad asociada a el. Para una entidad de determinado tipo que encapsula cierto comando, existe una instancia de dicho comando para cada instancia de dicha entidad. Al momento de su ejecución, el comando además de tener a su disposición todas las interfaces de los distintos controladores del sistema, puede acceder a operaciones de la interfaz que provee la entidad asociada a el.

3.1. Especificar posición de entidad

A través de este comando es posible mover alguna entidad en la simulación a determinada posición. Entre las entidades existentes en el sistema, solo se pueden mover las de tipo robot y de tipo pelota.

Para realizar esta acción se deben especificar los siguientes parámetros:

- Clave del comando a ejecutar, en este caso es "Set_Entity_Position".
- Identificador de la entidad que se desea mover
- Posición en cada uno de los tres ejes a la cual se va a mover la entidad

Un ejemplo de mensaje que ejecuta este comando es el siguiente.

```
<Message>
  <Simulation_Message>
    <Entity_Request Command = "Set_Entity_Position">
      <Entity_Id>1</Entity_Id>
      <Position_X>1</Position_X>
      <Position_Y>0.5</Position_Y>
      <Position_Z>2</Position_Z>
    </Entity_Request>
  </Simulation_Message>
</Message>
```

En este caso se moverá a la entidad con identificador 1 a la posición (1,0.5,2)

3.2. Controlar motores de entidad

Las entidades de tipo robot pueden tener uno o varios motores. Estos motores en definitiva están ligados a empalmes de tipo "Wheel Joint", controlándolos. Con este comando se puede controlar los motores de una entidad en particular. Para realizar esta acción se deben especificar los siguientes parámetros:

- Clave del comando a ejecutar, en este caso es "Robot_Motor_Control".
- Identificador de la entidad cuyos motores se desea controlar

- Una lista de motores, y para cada uno el identificador de su empalme de tipo "Wheel Joint" y la velocidad que se va a aplicar a ese motor.

Un ejemplo de mensaje que ejecuta este comando es el siguiente.

```
<Message>
  <Simulation_Message>
    <Entity_Request Command = "Robot_Motor_Control">
      <Entity_Id>1</Entity_Id>
      <Motors>
        <Motor>
          <Wheel_Joint_Id>0</Wheel_Joint_Id>
          <Value>3</Value>
        </Motor>
        <Motor>
          <Wheel_Joint_Id>1</Wheel_Joint_Id>
          <Value>-5</Value>
        </Motor>
      </Motors>
    </Entity_Request>
  </Simulation_Message>
</Message>
```

En este caso a la entidad con identificador 1 se le va a asignar al motor asociado al empalme de tipo "Wheel Joint" con identificador 0 la velocidad 3, y al motor asociado al empalme con identificador 1 la velocidad -5.

4. Respuestas

Luego de ejecutado un comando, los agentes pueden recibir una confirmación de la ejecución de un comando. Para esto se puede indicar en el cuerpo del Request que desean que ese comando luego de ejecutado genere un mensaje de respuesta hacia todos los agentes conectados al sistema. Para hacer esto se utiliza la etiqueta "Response". Un ejemplo se puede ver a continuación.

```
<Message>
  <Simulation_Message>
    <Control_Request Command = "Set_Fast_Mode">
      <Fast_Mode>true</Fast_Mode>
      <Response>true</Response>
    </Control_Request>
  </Simulation_Message>
</Message>
```

En este caso cuando se ejecute el comando para pasar el sistema a modo rápido, se generará un mensaje de confirmación que se propagará mediante broadcast a todos los agentes.

Esta etiqueta puede ser especificada en todos los comandos, excepto el registro de agente(sección 2.1) y el de pedido de información de agente (sección 2.2). En el primero el broadcast con la confirmación del registro se hace siempre, y en el segundo se envía información correspondiente al pedido solo al agente que lo solicitó.

5. Broadcast

Luego que se avanza un paso en la iteración, se realiza el broadcast de la información actualizada de esa iteración, enviandole dicha información a todos los agentes a los que le corresponda recibir información en esa iteración. Se puede distinguir 3 partes principales en el mensaje enviado, que se detallan a continuación:

- Número de iteración de la cual corresponde la información
- Información gráfica: contiene los datos de todos los cuerpos físicos de la simulación, con sus posiciones y orientaciones, así como también de sus velocidades angulares y lineales.
- Información de entidades: información proporcionada por las distintas entidades de la simulación. Cada tipo de entidad brinda diferente información. Una entidad de tipo robot brinda información acerca de la posición y orientación de su cuerpo de referencia. Una entidad de tipo pelota brinda información acerca de la posición y orientación del único cuerpo que la compone. Una entidad de tipo campo de juego no brinda ningún tipo de información de este tipo.

Un ejemplo de mensaje de broadcast se muestra a continuación, en esta simulación se modeló una entidad de tipo robot con un cubo y dos ruedas.

```
<Broadcast>
  <Iteration>53</Iteration>
  <DrawData>
    <Body Type="WheelCylinder">
      <Body_id>1</Body_id>
      <Material>1</Material>
      <Mass>0.1</Mass>
      <Length>0.04</Length>
      <Radius>0.023</Radius>
      <Position_X>-0.0174537</Position_X>
      <Position_Y>-0.130084</Position_Y>
      <Position_Z>0.0433577</Position_Z>
      <Rotation_X>-1.28877</Rotation_X>
      <Rotation_Y>-1.38333</Rotation_Y>
      <Rotation_Z>2.94624</Rotation_Z>
      <Linear_Velocity_X>-0.0372574</Linear_Velocity_X>
      <Linear_Velocity_Y>0.00835157</Linear_Velocity_Y>
      <Linear_Velocity_Z>0.00572461</Linear_Velocity_Z>
      <Angular_Velocity_X>0.00356964</Angular_Velocity_X>
      <Angular_Velocity_Y>-1.32355</Angular_Velocity_Y>
      <Angular_Velocity_Z>0.242375</Angular_Velocity_Z>
    </Body>
    <Body Type="Box">
```

```
      <Body_id>0</Body_id>
      <Material>0</Material>
      <Mass>1.2</Mass>
      <Size_X>0.075</Size_X>
      <Size_Y>0.075</Size_Y>
      <Size_Z>0.075</Size_Z>
      <Position_X>-0.0233573</Position_X>
      <Position_Y>-0.0965763</Position_Y>
      <Position_Z>0.0608945</Position_Z>
      <Rotation_X>-0.0522157</Rotation_X>
      <Rotation_Y>-0.160992</Rotation_Y>
      <Rotation_Z>0.0905903</Rotation_Z>
      <Linear_Velocity_X>-0.0494591</Linear_Velocity_X>
      <Linear_Velocity_Y>0.00857543</Linear_Velocity_Y>
      <Linear_Velocity_Z>0.00130333</Linear_Velocity_Z>
      <Angular_Velocity_X>-0.0754428</Angular_Velocity_X>
      <Angular_Velocity_Y>-0.327948</Angular_Velocity_Y>
      <Angular_Velocity_Z>0.191968</Angular_Velocity_Z>
    </Body>
    <Body Type="WheelCylinder">
      <Body_id>2</Body_id>
      <Material>1</Material>
      <Mass>0.1</Mass>
      <Length>0.04</Length>
      <Radius>0.023</Radius>
      <Position_X>-0.022857</Position_X>
      <Position_Y>-0.0645364</Position_Y>
      <Position_Z>0.0399746</Position_Z>
      <Rotation_X>1.32982</Rotation_X>
      <Rotation_Y>1.34944</Rotation_Y>
      <Rotation_Z>-0.153778</Rotation_Z>
      <Linear_Velocity_X>-0.0487662</Linear_Velocity_X>
      <Linear_Velocity_Y>0.00708905</Linear_Velocity_Y>
      <Linear_Velocity_Z>-0.000940901</Linear_Velocity_Z>
      <Angular_Velocity_X>0.160203</Angular_Velocity_X>
      <Angular_Velocity_Y>-3.31479</Angular_Velocity_Y>
      <Angular_Velocity_Z>0.344777</Angular_Velocity_Z>
    </Body>
  </DrawData>
  <Entities_Info>
    <Robot_Entity>
      <Entity_Id>3</Entity_Id>
      <Position>
        <pos_x>-0.0218736</pos_x>
        <pos_y>-0.160992</pos_y>
        <pos_z>0.0608554</pos_z>
```

```
        </Position>
        <Rotation>
          <rot_x>-0.0522157</rot_x>
          <rot_y>-0.151389</rot_y>
          <rot_z>0.0905903</rot_z>
        </Rotation>
      </Robot_Entity>
    </Entities_Info>
</Broadcast>
```

6. Configuración

Existen diversos parámetros de configuración del sistema que se especifican en el archivo "Configuration.data", que debe encontrarse en la misma carpeta que el ejecutable de la aplicación. Los parámetros que pueden especificarse en este archivo tienen un identificador y un valor. Estos parámetros se describen a continuación, agrupados por categoría.

Comunicación

- communicationreceiverport: Indica el puerto en el cual se va a esperar por mensajes de agentes externos.
- communicationreceiverip: Indica la dirección de red en la cual se va a esperar por mensajes de agentes externos. Este parámetro junto al anterior son fundamentales para configurar la comunicación con el núcleo de simulación, ya que indican la dirección de red y puerto mediante los cuales se va a establecer la comunicación con el núcleo de simulación.
- simulationengineip: Indica la dirección de red que se utiliza para la comunicación entre dos módulos del sistema (comunicación y simulación).
- simulationengineport: Indica el puerto que se utiliza para la comunicación entre dos módulos del sistema (comunicación y simulación).
- maximumdatagramsize: Indica el tamaño máximo en bytes que pueden tener los mensajes intercambiados entre los agentes externos y el núcleo de simulación

Propidades generales

- contactpointsammount: Cantidad de puntos de contacto que se procesan a cada posible colisión. Un valor menor implica menos precisión física y mayor velocidad en los cálculos matemáticos.
- defaultresponserequest: Admite los valores "true" o "false". Indica si se va a enviar una respuesta a cada petición en caso de que no se especifique explícitamente en el mensaje recibido.
- defaultstepsize: Valor por defecto de la cantidad del tamaño de paso de simulación.
- syncwait: Valor por defecto del tiempo de espera en comunicación sincrónica.
- defaultphysicsworldbounciness: Valor por defecto de la propiedad global "Bounciness" o esponjosidad.
- defaultphysicsworldhardness: Valor por defecto de la propiedad global "Hardness" o dureza.

Colisiones

Más información acerca de las propiedades que aparecen en esta sección se puede encontrar en 2.3.

- defaulthardnessinteraction: Valor por defecto de la propiedad "Hardness" en las colisiones.
- defaultbouncinessinteraction: Valor por defecto de la propiedad "Bounciness" en las colisiones.
- defaultslip1interaction: Valor por defecto de la propiedad "Slip1" en las colisiones.
- defaultslip2interaction: Valor por defecto de la propiedad "Slip2" en las colisiones.
- defaultfrictioninteraction: Valor por defecto de la propiedad "Friction" en las colisiones.
- defaultgravity: Valor por defecto de la gravedad de la simulación.

Empalmes

Más información acerca de las propiedades que aparecen en esta sección se puede encontrar en 2.4.

- defaultwheeljointhardness: Valor por defecto de la propiedad "Hardness" en los empalmes de tipo "WheelJoint".
- defaultwheeljointhardness: Valor por defecto de la propiedad "Bounciness" en los empalmes de tipo "WheelJoint".
- defaultwheeljointmaxtorque: Valor por defecto de la propiedad "Torque máximo" en los empalmes de tipo "WheelJoint".

Cuerpos

Más información acerca de las propiedades que aparecen en esta sección se puede encontrar en las secciones 2.4 y 2.3.

- defaultbodymaterial: Material por defecto de los cuerpos rígidos.
- defaultbodymass: Masa por defecto de los cuerpos rígidos.

Comandos

Cada comando tiene un identificador configurable como se detalla a continucación.

- commandcreateentitykey: Identificador del comando de creación de entidad, que se describe en la sección 2.4.
- commandrobotcontrolkey: Identificador del comando de control de motores de robot, que se describe en la sección 3.2.
- commandinitializesimulationkey: Identificador del comando de inicialización de simulación, que se describe en la sección 2.5.
- commandsettimestepkey: Identificador del comando de especificación de tamaño del paso de simulación, que se describe en la sección 2.12.
- commandsetstepmodekey: Identificador del comando de especificación del modo de paso de tiempo, que se describe en la sección 2.6.
- commandsetstepflagkey: Identificador del comando de especificación bandera de paso tiempo, que se describe en la sección 2.7.
- commandsynchronizekey: Identificador del comando de sincronización con iteración, que se describe en la sección 2.10.
- commandsetsyncwaitkey: Identificador del comando de especificación del tiempo de espera en comunicación sincrónica, que se describe en la sección 2.11.
- commandsetworldgravitykey: Identificador del comando de especificación de la gravedad en la simulación, que se describe en la sección 2.13.
- commandworldbouncinesskey: Identificador del comando de la propiedad global "Bounciness" o esponjosidad, que se describe en la sección 2.15.
- commandworldhardnesskey: Identificador del comando de la propiedad global "Hardness" o dureza, que se describe en la sección 2.14.
- commandsetentitypositionkey: Identificador del comando de especificación de posición de entidad, que se describe en la sección 3.1.
- commandregistermaterialsinteracionskey: Identificador del comando de registro de materiales y interacciones, que se describe en la sección 2.3.
- commandsavetofilekey: Identificador del comando de almacenamiento de la simulación a archivo, que se describe en la sección 2.16.
- commandloadfromfilekey: Identificador del comando de carga de la simulación desde archivo, que se describe en la sección 2.17.

- commandsetstepsperiterationkey: Identificador del comando de especificación de cantidad de pasos por iteración, que se describe en la sección 2.19.
- commandsetfastmodekey: Identificador del comando especificación de modo rápido, que se describe en la sección 2.18.

7. Extensibilidad

La arquitectura del sistema está fuertemente enfocada en el requerimiento de la extensibilidad. El sistema es altamente extensible, tanto en las funcionalidades que presenta como en las distintas estructuras que utiliza. En el marco de este documento se presentarán algunas maneras de extender al sistema que resultan interesantes.

7.1. Modelo físico

El módulo "Physics Engine" expone una interfaz física genérica. La misma presenta una serie de clases abstractas o interfaces para su utilización. Estas interfaces son implementadas por clases que interactúan con un motor físico concreto. Se puede fácilmente cambiar el motor físico utilizado sustituyendo dichas clases concretas por otras que interactúen con otro motor físico. Básicamente, existen interfaces que representan cada uno de los tipos básicos de objetos soportados y para el controlador general del modelo físico. Los objetos básicos soportados son los diferentes tipos de cuerpos y empalmes que se pueden modelar actualmente en el sistema. Dichos objetos pueden extenderse de modo de soportar otros tipos de cuerpos y empalmes. Se debe crear una interfaz de dicho objeto que se acople a la interfaz genérica presentada e implementar dicha interfaz utilizando una clase concreta que interactúe con el motor físico. Se podría de esta manera nuevos cuerpos, como cuerpos formados por la unión de triángulos, o nuevos empalmes, como el "Hinge" o "Ball and Socket". Estos empalmes y cuerpos de ejemplo son soportados por el motor físico, pero no están representados en la interfaz física genérica propuesta. De esta manera se podría mapear de forma completa todos los cuerpos y empalmes soportados por el motor físico.

Como ejemplo, se mostrará la estructura básica necesaria para extender el sistema agregándole un nuevo tipo de cuerpo. Para esto es necesario implementar una clase que interactúe con el motor físico utilizado y proponer una interfaz genérica para dicho cuerpo. Además se deben implementar algunas operaciones básicas y exponerlas en dicha interfaz, como la serialización de la clase. También se debe crear un datatype correspondiente a dicho cuerpo. Adicionalmente se deben hacer modificaciones menores al controlador del módulo y a algunos comandos para que consideren los cambios realizados. Si una entidad puede utilizar el nuevo tipo de objeto, se debe decodificar correctamente este en el momento de la creación de dicha entidad. La siguiente imagen muestra resumidamente la arquitectura básica.

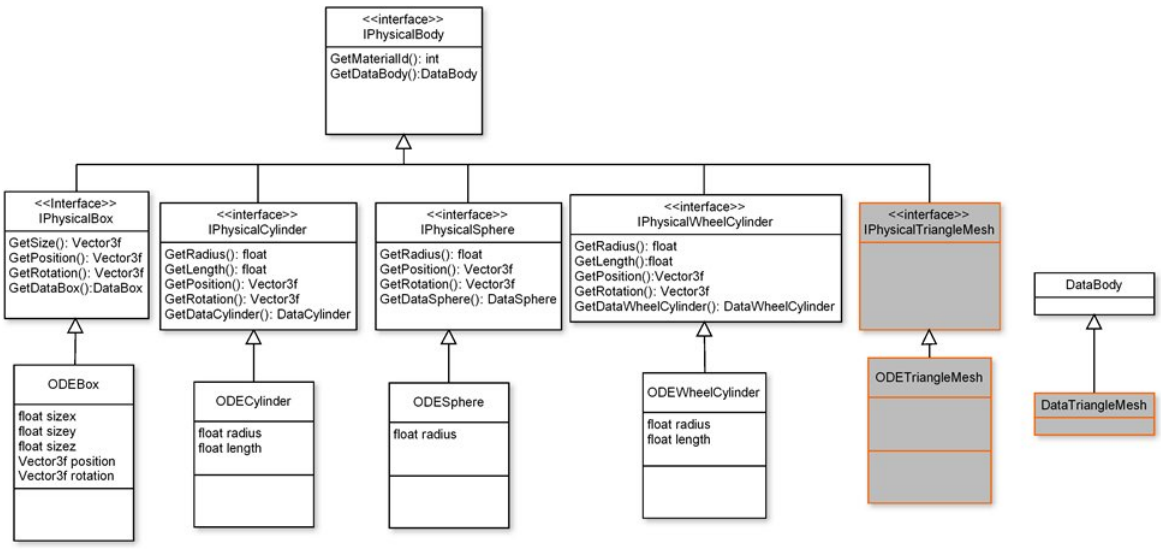

Figura 3: Extensibilidad modelo físico

7.2. Entidades

El sistema provee una interfaz de la cual deben heredar todas las entidades del sistema. Estas deben cumplir con cierta estructura básica, pero existe gran flexibilidad en cuanto a las funcionalidades que puede encapsular una entidad. En particular, sería interesante contar con una entidad sin ningún tipo de restricciones estructurales con algunos comandos básicos, de modo de que pueda ser utilizado para cualquier tipo de entidad.

También para agregar nuevas entidades es necesario hacer algunos cambios menores al decodificador de este módulo, la clase "SimulationEngineEncoderDecoder" para que contemple el nuevo tipo de entidad.

En la siguiente figura se ilustra el mecanismo de extensibilidad de entidades.

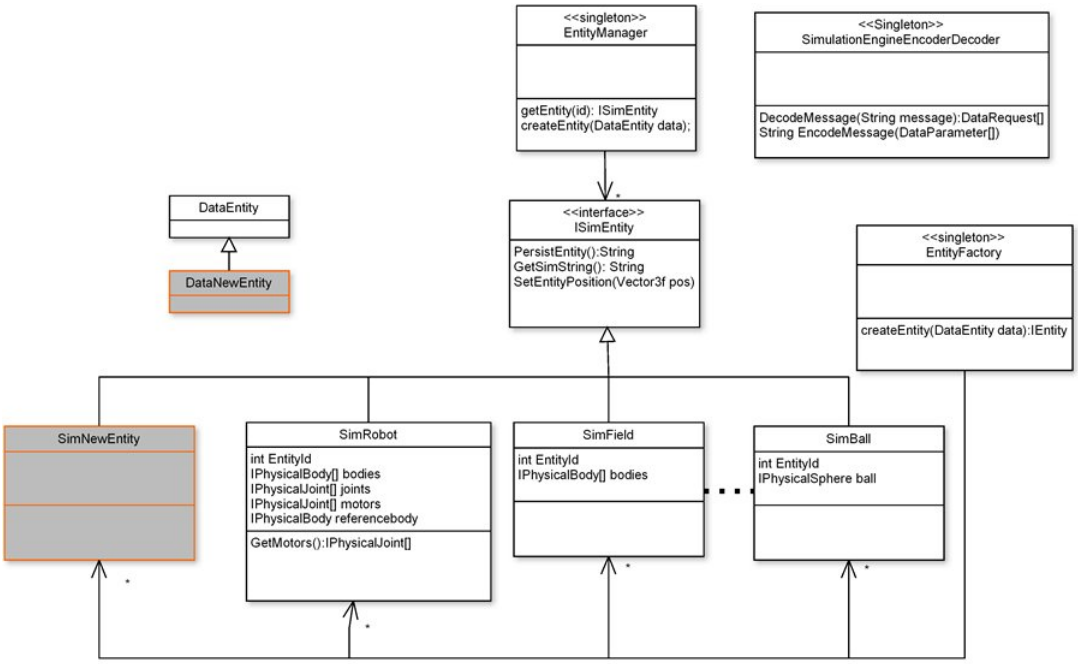

Figura 4: Extensibilidad de entidades

7.3. Comandos

La existencia de comandos le aporta una gran flexibilidad al sistema, y gran parte de las extensiones y nuevas funcionalidades que se podrían considerar, pueden ser resueltas simplemente implementando un nuevo comando. Al agregar una nueva funcionalidad se tienen dos opciones: resolverla utilizando un comando de control o resolverla utilizando un comando asociado a alguna entidad. Simplemente se debe crear uno de estos comandos, implementando la interfaz de comandos y asociarlo al controlador de comandos correspondiente. Existe un controlador de comandos de control y otro controlador de comandos de entidades. Para ejecutar correctamente un comando, debe decodificarse un "Request" encapsulado en un mensaje, y decodificar los distintos parámetros o tipos de datos utilizados para la invocación del comando.

También para agregar nuevas entidades es necesario hacer algunos cambios menores al decodificador de este módulo, la clase "SimulationEngineEncoderDecoder" para que contemple el nuevo tipo de comando.

En la siguiente figura se ilustra el mecanismo de extensibilidad de comandos.

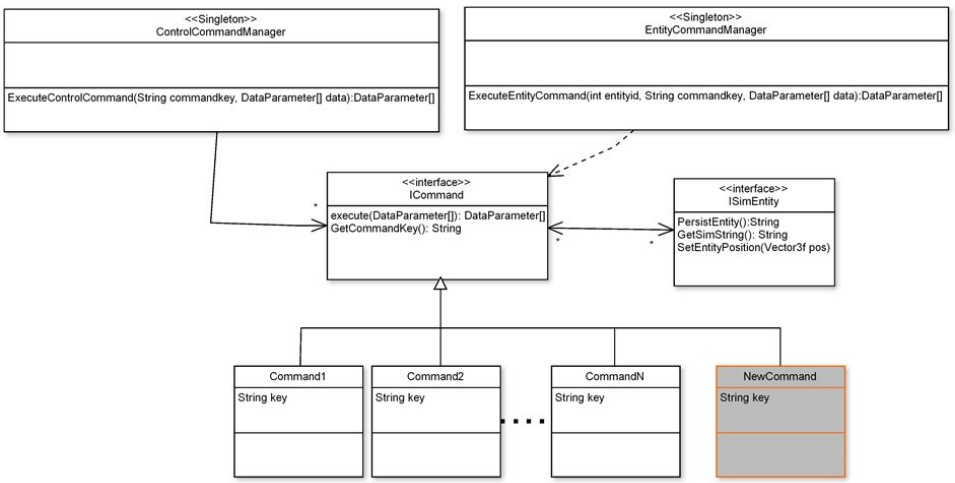

Figura 5: Extensibilidad de comandos

7.4. Tipos de agentes y filtros

Cada tipo de agente que se conecta al sistema, tiene asociado un filtro, que en el momento de enviarle información acerca del estado del mundo determina que información se trasmite y cual no. Esto es especialmente útil para ocultar determinada información, agregarle ruido a los datos, cambiar las unidades de las magnitudes, etc. Cambiar el tipo de filtro asociado a un tipo de entidad es realmente sencillo, así como también crear tipos de agentes nuevos con el filtro que se desee. Además, es necesario realizar modificaciones en el decodificador de este módulo, o sea, la clase "CommunicationDecoder" para que contemple los nuevos tipos de agente.

En el siguiente diagrama se ilustra la estructura de clases y las nuevas extensiones.

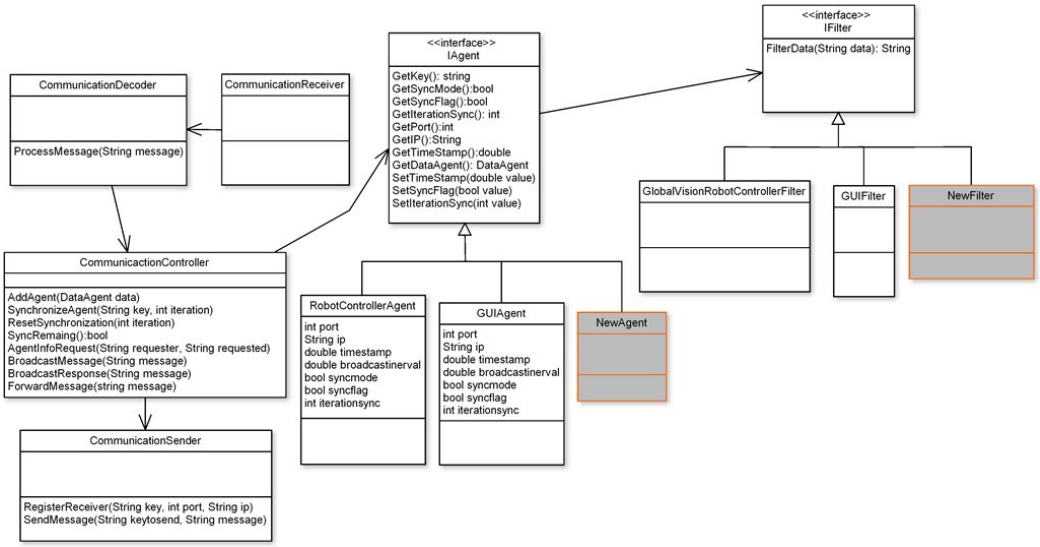

Figura 6: Extensibilidad de tipos de agentes y filtros

8. Interfaz para el envío de comandos

Con el objetivo de de hacer más sencilla la tarea de controlar las distintas funcionalidades del sistema a través del envío de comandos, se desarrolló utilizando el lenguaje Java[?] una aplicación denominada "SimRob Commander", cuya interfaz gráfica permite de manera sencilla enviar mensajes al sistema.

Una captura de pantalla durante la ejecución de esta aplicación se puede ver a continuación.

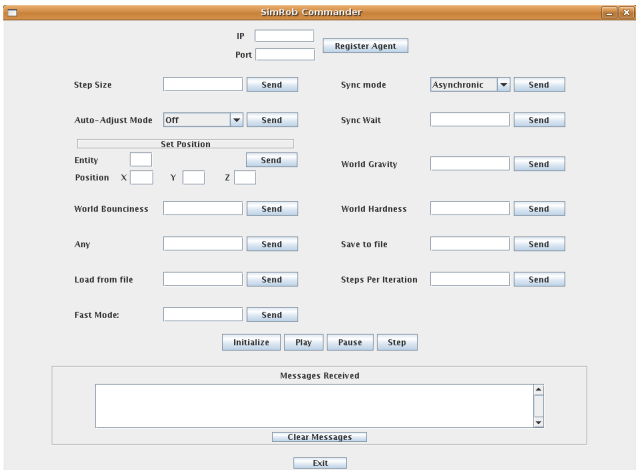

Figura 7: SimRob Commander

Se presentan diversas opciones en la interfaz gráfica, cuyo objetivo es el envío de un mensaje que una vez recibido y decodificado en el núcleo de simulación dispara la ejecución de un comando.

En la parte superior de la pantalla se cuenta con la opción de registrar el agente, en este caso la aplicación "SimRob Commander" en el sistema. Para esto se debe indicar la dirección de red y el puerto en el cual se está ejecutando el sistema. El registro permite comenzar a recibir mensajes de respuesta luego de la ejecución de cada uno de los comandos correspondientes. Estos mensajes una vez que son recibidos, son desplegados en la caja de texto que se encuentra cerca de la parte inferior de la interfaz.

No es necesario el registro del agente para el envío de comandos, sin embargo es necesario ingresar la dirección de red y el puerto en el cual se está ejecutando el sistema para que los mensajes lleguen a su destino correctamente.

Existen diversas funcionalidades en las cuales se debe ingresar algún valor correcto y luego presionar el botón contiguo. Al hacer esto, la aplicación crea un mensaje encapsulando esta información y lo envía. Ejemplos de este tipo

de funcionalidades son la especificación del tamaño del paso de simulación, la solicitud para almacenar la simulación en un archivo, entre otras.

Otras funcionalidades que no requieren el ingreso de ningún valor solamente se ejecutan presionando el botón correspondiente. Ejemplos de estas funcionalidades son las utilizadas para controlar el paso del tiempo cuando el sistema se encuentra en modo paso a paso.

Las funcionalidades presentes y los posibles valores válidos se describen a continuación.

- Registro de Agentes (en pantalla "Register Agent"): Para utilizar esta funcionalidad es necesario especificar los campos que se detallan a continuación y presionar el botón correspondiente.
  - IP: Especifica la dirección de red de escucha del núcleo de simulación. Esta dirección está especificada en el archivo "Configuration.data" con el identificador "communicationreceiverip".
  - Port: Especifica el puerto de escucha del núcleo de simulación. Esta dirección está especificada en el archivo "Configuration.data" con el identificador "communicationreceiverport".
- Especificar tamaño del paso de simulación (en pantalla "Step Size"): Se debe especificar el valor del tamaño del paso de simulación y presionar el botón correspondiente. Los posibles valores son los que se explican en el detalle de esta funcionalidad en la sección 2.12.
- Especificar modo de ajuste automático (en pantalla "Auto Adjust mode"): Se debe especificar el modo de ajuste automático y presionar el botón correspondiente, sus posibles valores son "On" y "Off".
- Especificar posición de entidad (en pantalla "Set Position"): Se debe especificar los campos que se detallan a continuación y presionar el botón correspondiente.
  - Identificador de entidad (en pantalla "Entity"): Valor entero que corresponde al identificador de la entidad que se desea cambiar la posición.
  - X, Y, Z: Coordenadas en cada uno de los 3 ejes cartesianos de la posición donde se desea mover la entidad.
- Especificar esponjosidad global (en pantalla "World Bounciness"): Se debe especificar el valor de la esponjosidad o "Bounciness" global y presionar el botón correspondiente. Los posibles valores son los que se explican en el detalle de esta funcionalidad en la sección 2.15.
- Especificar dureza global (en pantalla "World Hardness"): Se debe especificar el valor de la dureza o "Hardness" global y presionar el botón correspondiente. Los posibles valores son los que se explican en el detalle de esta funcionalidad en la sección 2.14.

- Especificar gravedad (en pantalla "World Gravity"): Se debe especificar el valor de la gravedad y presionar el botón correspondiente. Los posibles valores son los que se explican en el detalle de esta funcionalidad en la sección 2.13.
- Grabar a archivo (en pantalla "Save to file"): Se debe especificar la ruta de un archivo en el cual se va a almacenar la simulación y presionar el botón correspondiente. Es necesario tener permisos de escritura sobre la carpeta que se va a crear el archivo.
- Cargar desde archivo (en pantalla "Load from file"): Se debe especificar la ruta de un archivo desde el cual se va a cargar la simulación y presionar el botón correspondiente. Es necesario tener permisos de escritura sobre la carpeta que se va a crear el archivo.
- Especificar modo de avance rápido (en pantalla "Fast Mode"): Se debe especificar si se desea ejecutar la simulación en modo de avance rápido y presionar el botón correspondiente. Los posibles valores son los que se explican en el detalle de esta funcionalidad en la sección 2.18.
- Especificar modo de comunicación (en pantalla "Sync Mode"): Se debe especificar el modo de comunicación y presionar el botón correspondiente. Los posibles valores son "Synchronic" para modo sincrónico y "Asinchronic" para modo asincrónico.
- Especificar tiempo de espera en comunicación sincrónica (en pantalla "Sync Wait"): Se debe el tiempo de espera en casos de comunicación sincrónica y presionar el botón correspondiente. Este es el período de tiempo que se espera por una respuesta de sincronización de una iteración de un agente al cual se le envió mensajes de difusión, siempre y cuando el modo de comunicación con este agente sea sincrónico. Los posibles valores son los que se explican en el detalle de esta funcionalidad en la sección 2.11.
- Especificar cantidad de pasos por iteración (en pantalla "Steps per Iteration"): Se debe especificar la cantidad de pasos por iteración y presionar el botón correspondiente. Los posibles valores son los que se explican en el detalle de esta funcionalidad en la sección 2.19.
- Botones: Se presentan algunos botones para efectuar las acciones que se detallan a continuación.
    - Botón "Initialize": Inicializa la simulación. Es necesario inicializar la simulación para que el tiempo simulado comience a avanzar por primera vez.
    - Botón "Play": Vuelve a iniciar la simulación una vez que esta estuvo en pausa.
    - Botón "Pause": Pasa la simulación a estado en pausa.

- Botón "Step": Da un paso en la simulación cuando la misma se encuentra en pausa.
- Botón "Clear Messages": Limpia la consola donde se muestran los mensajes de respuesta enviados por el núcleo de simulación.
- Botón "Exit": Cierra la interfaz de envío de comandos.