(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 535 273**

(51) Int. CI.:

B25J 9/16 (2006.01)
B62D 57/032 (2006.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(96) Fecha de presentación y número de la solicitud europea: **22.11.2011 E 11787856 (1)**

(97) Fecha y número de publicación de la concesión europea: **21.01.2015 EP 2651607**

(54) Título: **Robot humanoide dotado de un gestor de sus recursos físicos y virtuales, procedimientos de utilización y de programación**

(30) Prioridad:

17.12.2010 FR 1060755

(45) Fecha de publicación y mención en BOPI de la traducción de la patente:
07.05.2015

(73) Titular/es:

ALDEBARAN ROBOTICS S.A. (100.0%)
168 bis - 170 rue Raymond Losserand
75014 Paris, FR

(72) Inventor/es:

MAZEL, ALEXANDRE;
HOUSSIN, DAVID y
MONCEAUX, JÉRÔME

(74) Agente/Representante:

VALLEJO LÓPEZ, Juan Pedro

DESCRIPCIÓN

Robot humanoide dotado de un gestor de sus recursos físicos y virtuales, procedimientos de utilización y de programación

La presente invención pertenece al campo de los robots humanoides. Más precisamente, se aplica a la gestión de los recursos de dicho robot, principalmente cuando varios usuarios de este robot programan de manera independiente unos comportamientos destinados a ser puestos en práctica por dicho robot.

Puede calificarse a un robot de humanoide a partir del momento en que posee ciertos atributos de aspecto y de funcionalidades del hombre: una cabeza, un tronco, dos brazos, dos manos, dos piernas, dos pies, etc. Junto al aspecto, las funciones que un robot humanoide es capaz de completar dependerán de su capacidad para efectuar movimientos, de hablar y de "razonar". Unos robots humanoides son capaces de marchar, de hacer gestos, con los miembros o con la cabeza. La complejidad de los gestos que son capaces de efectuar se incrementa sin cesar. Unos gestos coordinados entre ellos o coordinados con unas palabras (por ejemplo un brazo levantado, unas palabras amenazantes pronunciadas con un tono ronco), unas señales emitidas por unas partes del robot (principalmente sus LED) pueden ser la expresión de comportamientos del robot que traduce el sus "emociones" (por ejemplo, la cólera). Ciertos robots, principalmente el robot NAO, que es la plataforma en la que se han desarrollado los conceptos y los medios de realización de la presente invención, ofrecen la posibilidad a varios usuarios de desarrollar unos comportamientos de manera independiente uno del otro. Estas posibilidades se mejoraran grandemente con unas herramientas de desarrollo en lenguaje usuario, tales como el programa Chorégraphe. En consecuencia, se plantea el problema de saber cómo asegurar la coherencia de los diferentes comportamientos, programados por unos usuarios diferentes, en el momento de su ejecución, principalmente para evitar los conflictos en la utilización de los recursos materiales y de programación del o de los robots, asegurar la sincronización de la disponibilidad de los recursos que deben utilizarse simultáneamente y generar las prioridades de ejecución de un cierto número de funciones.

Es tradicional en robótica asegurar una protección de los motores e impedir su utilización simultánea por varios comportamientos; es un primer tipo de solución a este problema general de gestión de recursos, pero no se trata en ese caso de una solución general: en particular, este tipo de solución no aporta una respuesta al problema planteado por la sincronización de recursos diferentes, ni al problema de la liberación de los recursos.

Es tradicional para resolver estos problemas de asignación de recursos en el tiempo en otros campos de la tecnología realizar unos mecanismos de optimización en la asignación de los recursos, utilizando principalmente unos algoritmos de tipo de programación lógica por restricciones. Estos algoritmos están adaptados sin embargo a los cálculos de asignación de recursos que no son consumidos inmediatamente porque son ávidos en cuanto a recursos de cálculo. En particular, no se podría utilizar en un robot humanoide dotado de múltiples comportamientos más que para una parte pequeña de éstos.

El documento EP-A1-1 486 300 (Sony Corporation) describe un robot humanoide de ese tipo.

No existe por tanto en la técnica anterior una solución que permita resolver el problema de la asignación de recursos materiales, de programación o de otros tipos a unos procesos independientes. La presente invención resuelve este problema proporcionando una definición jerárquica de los recursos conectada con un mecanismo de resolución local o descentralizado.

Con este fin, la presente invención divulga un robot humanoide adecuado para ejecutar una pluralidad de comportamientos bajo el control de un módulo de gestión integrado a bordo que comprende un submódulo de gestión de una pluralidad de recursos que pueden estar asignados para la ejecución de dichos comportamientos, estando dicho robot caracterizado por que dicha pluralidad de comportamientos y dicha pluralidad de recursos se organizan cada uno en subconjuntos jerárquicos y por que dicho submódulo de gestión de los recursos es adecuado para ser programado para reservar al menos un recurso que pertenece a al menos un subconjunto de recursos para la ejecución de al menos un comportamiento que pertenece a al menos un subconjunto de comportamientos, pudiendo programarse dicha reserva para ser heredada por dicho subconjunto de comportamientos y/o aplicada a dicho subconjunto de recursos.

Ventajosamente, dicho submódulo de gestión de la pluralidad de recursos se programa para responder, cuando un primer comportamiento está en curso de ejecución, a una solicitud de recursos por un segundo comportamiento mediante al menos una de las acciones siguientes: rechazo de la puesta a disposición de dicho recurso a dicho segundo comportamiento, puesta a disposición inmediata de dicho recurso a dicho segundo comportamiento, ejecución de al menos una acción suplementaria, posteriormente puesta a disposición de dicho recurso a dicho segundo comportamiento, puesta en pausa del primer comportamiento en curso de ejecución, posteriormente puesta a disposición de dicho recurso a dicho segundo comportamiento.

Ventajosamente, dicho submódulo de gestión de la pluralidad de recursos se programa para que, cuando se ejecuta la acción de rechazo de puesta a disposición de dicho recurso requerido por dicho segundo comportamiento, dicho

segundo comportamiento ejecute una nueva solicitud de puesta a disposición según una frecuencia predefinida hasta expiración de un período de expiración igualmente predefinido.

Ventajosamente, dicho submódulo de gestión de la pluralidad de recursos se programa para que, cuando al menos dos comportamientos ejecutan una solicitud de puesta a disposición de un mismo recurso, el primero de los al menos dos comportamientos sea prioritario en la reserva de dicho mismo recurso cuando se libera este último.

Ventajosamente, dicho submódulo de gestión de la pluralidad de recursos se programa para que, cuando al menos un comportamiento ejecuta una solicitud de puesta a disposición de al menos dos recursos, dichos al menos dos recursos se reagrupen en un nuevo recurso tratado como recurso único, ordenándose dicho reagrupamiento de manera única para todos los comportamientos que ejecutan unas solicitudes de recursos idénticas.

Ventajosamente, dicho submódulo de gestión de la pluralidad de recursos se programa para que, cuando un comportamiento padre reserva al menos un recurso sin utilizarlo, dicho al menos un recurso pueda utilizarse por parte de un comportamiento hijo de dicho comportamiento padre.

Ventajosamente, dicho submódulo de gestión de la pluralidad de recursos se programa para que, cuando un primer comportamiento hijo utiliza un recurso reservado por dicho comportamiento padre, un segundo comportamiento hijo del mismo padre deba, para poder utilizar dicho recurso, efectuar una solicitud de puesta a disposición de dicho recurso ante el primer comportamiento hijo.

Ventajosamente, dicho submódulo de gestión de la pluralidad de recursos se programa para que, cuando un comportamiento padre libera un recurso del que tiene la reserva, dicho recurso deba liberarse por todo comportamiento hijo de dicho padre.

Ventajosamente, dicho submódulo de gestión de la pluralidad de recursos se programa para, cuando un subconjunto de recursos tiene una disponibilidad reducida, realizar una reserva de dichos recursos para un comportamiento virtual definido en función de dicha reducción de disponibilidad.

Ventajosamente, dicho submódulo de gestión de la pluralidad de recursos se programa para liberar dicha reserva en función de la evolución del estado de dicha reducción de disponibilidad.

Ventajosamente, la pluralidad de recursos comprende al menos una variable de estado, debiendo estar dicha al menos una variable en un estado predefinido para ser reservada, siendo necesaria dicha reserva para la ejecución de al menos un comportamiento.

Ventajosamente, la pluralidad de recursos comprende al menos un recurso acumulativo y por que una tasa de utilización de este se asigna a los comportamientos programados para utilizar dicho al menos un recurso.

Ventajosamente, la pluralidad de recursos comprende al menos un recurso esencial y un recurso no esencial, no pudiéndose ejecutar ningún comportamiento que deba acudir a al menos un recurso esencial si dicho recurso esencial no está disponible y pudiéndose ejecutar si debe acudir a dicho al menos un recurso no esencial, incluso si este no está disponible.

La invención divulga igualmente un procedimiento de gestión de los recursos de un robot humanoide adecuado para ejecutar una pluralidad de comportamientos bajo el control de un módulo de gestión integrado a bordo, comprendiendo dicho procedimiento una etapa de reserva de una pluralidad de recursos que pueden asignarse para la ejecución de dichos comportamientos, estando dicho procedimiento caracterizado por que dicha pluralidad de comportamientos y dicha pluralidad de recursos se organizan cada uno en subconjuntos jerárquicos y por que dicha etapa de reserva asigna al menos un recurso que pertenece a al menos un subconjunto de recursos para la ejecución de al menos un comportamiento que pertenece a al menos un subconjunto de comportamientos, pudiendo programarse dicha reserva para ser heredada por dicho subconjunto de comportamientos y/o aplicada a dicho subconjunto de recursos.

La invención divulga igualmente un programa de ordenador que comprende unas instrucciones de código de programa que permiten la ejecución de dicho procedimiento cuando el programa se ejecuta en un ordenador, estando adaptado dicho programa para permitir la gestión de los recursos de un robot humanoide adecuado para ejecutar una pluralidad de comportamientos bajo el control de un subprograma de gestión integrado a bordo, comprendiendo dicho subprograma un módulo adecuado para ejecutar una función de reserva de una pluralidad de recursos que pueden asignarse para la ejecución de dichos comportamientos, estando dicho programa caracterizado por que dicha pluralidad de comportamientos y dicha pluralidad de recursos se organizan cada uno en subconjuntos jerárquicos y por que dicha función de reserva asigna al menos un recurso que pertenece a al menos un subconjunto de recursos para la ejecución de al menos un comportamiento que pertenece a al menos un subconjunto de comportamientos, pudiendo programarse dicha reserva para ser heredada por dicho subconjunto de comportamientos y/o aplicada a dicho subconjunto de recursos.

Ventajosamente, dichos subconjuntos de comportamientos y de recursos se definen como clases de objetos.

La invención divulga también un procedimiento de desarrollo de un módulo de gestión de los recursos de un robot humanoide y un programa de ordenador para realizar dicho procedimiento.

La invención ofrece además la ventaja de evitar las situaciones de bloqueo que son frecuentes en los sistemas informáticos de procesos que se ejecutan en paralelo en un mismo procesador. Permite igualmente gestionar unos tipos de recursos muy diferentes, comprendidos en ellos los recursos virtuales que corresponden a unos estados, en el que la congelación permite evitar el lanzamiento de comportamientos que serían susceptibles de crear una situación de bloqueo o de peligro para el robot.

La invención se comprenderá mejor y surgirán sus diferentes características y ventajas de la descripción a continuación de varios ejemplos de realización y de sus figuras adjuntas en las que:

- la figura 1 es un esquema de la arquitectura funcional según la cual se pone en práctica la invención en varios de sus modos de realización;
- las figuras 2a y 2b son unas vistas de pantalla de un programa informático que permite la puesta en práctica de la invención en varios de sus modos de realización;
- las figuras 3a y 3b representan unas modalidades de gestión de recursos, respectivamente bajo la forma de listas de recursos disponibles y bajo la forma de programación gráfica mediante cajas, en varios modos de realización de la invención;
- las figuras 4a y 4b representan un mecanismo de sincronización de los recursos de diferentes tipos, respectivamente bajo la forma de organigrama funcional y bajo la forma de menú gráfico desplegable, en varios modos de realización de la invención;
- las figuras 5a y 5b representan un mecanismo de reserva de recursos críticos, respectivamente bajo la forma de organigrama funcional y bajo la forma de menú gráfico desplegable, en varios modos de realización de la invención;
- las figuras 6a y 6b representan un mecanismo de herencia de una reserva de recursos, respectivamente bajo la forma de menú gráfico desplegable y bajo la forma de programación gráfica por cajas, en varios modos de realización de la invención;
- las figuras 7a, 7b y 7c ilustran la problemática de la gestión de los recursos ausentes, respectivamente representando un robot completo, robot tronco y una lista de recursos disponibles/indisponibles, en varios modos de realización de la invención;
- las figuras 8a a 8e ilustran varias modalidades de resolución de conflictos, respectivamente bajo la forma de organigrama funcional (8a) y bajo la forma de menús gráficos desplegables que corresponden a diferentes opciones de salida de un conflicto (8b a 8e), en varios modos de realización de la invención;
- la figura 9 ilustra la utilización de los recursos de estado bajo la forma de menú gráfico desplegable, en varios modos de realización de la invención;
- la figura 10 representa un organigrama funcional de la utilización de un recurso acumulativo compartido, en varios modos de realización de la invención.

La figura 1 es un esquema de la arquitectura funcional según la que se pone en práctica la invención en varios de sus modos de realización. La arquitectura física subyacente está constituida por un robot humanoide 110, del que se da a continuación un ejemplo de realización y, eventualmente, de una estación de trabajo remota 150 equipada con un programa informático de creación de comportamientos 160.

Es conocido un robot humanoide de la marca comercial NAO™. Un robot de ese tipo se ha divulgado principalmente en la solicitud de patente WO 2009/124951 publicada el 15/10/2009. Esta plataforma ha servido de base a las mejoras que han conducido a la presente invención. En lo que sigue de la descripción, puede designarse indiferentemente a este robot humanoide bajo esta apelación genérica o bajo su marca comercial NAO™, sin que se altere la generalidad de la referencia. Este robot comprende alrededor de dos docenas de tarjetas electrónicas de control de captadores y de actuadores que controlan las articulaciones. Una articulación tiene normalmente al menos dos grados de libertad y por tanto dos motores. Cada motor es controlado en ángulo. La articulación comprende igualmente varios captadores de posición, particularmente unos MRE (Magnetic Rotary Encoder). La tarjeta electrónica de control comprende un microcontrolador comercial. Este puede ser por ejemplo un DSPIC™ de la compañía Microchip. Es una MCU de 16 bits conectada a un DSP. Esta MCU tiene un ciclo de trabajo en bucle de un milisegundo. El robot puede comprender igualmente otros tipos de actuadores, particularmente unos LED (diodos electroluminiscentes) cuyo color e intensidad pueden traducir las emociones del robot. Este puede comprender igualmente otros tipos de captadores de posición, principalmente una central inercial, unos FSR (Captadores de presión contra el suelo).

La cabeza comprende la inteligencia del robot, principalmente la tarjeta que ejecuta las funciones de alto nivel que permiten al robot cumplir las misiones que se le asignan. El procesador de la tarjeta puede ser un procesador x86 comercial. Se elegirá de manera preferida un procesador de bajo consumo tal como el Geode™ de la compañía AMD (32 bits, 500 MHz). La tarjeta comprende igualmente un conjunto de memorias RAM y flash. Esta tarjeta genera igualmente las comunicaciones del robot con el exterior (servidor de comportamientos, otros robots…),

normalmente sobre una capa de transmisión WiFi, WiMax, eventualmente en una red pública de comunicaciones móviles de datos con unos protocolos estándar eventualmente encapsulados en un VPN. El procesador está controlado normalmente por un sistema operativo estándar lo que permite utilizar los lenguajes de alto nivel usuales (C, C++, Python, etc.) o lenguajes específicos de la inteligencia artificial como URBI (lenguaje de programación especializado en la robótica) para la programación de las funciones de alto nivel.

Otra tarjeta se aloja en el tronco del robot. Es ahí donde se sitúa el calculador que asegura la transmisión a las tarjetas de control de las articulaciones de las órdenes calculadas por la tarjeta de la cabeza. El calculador de esta tarjeta es igualmente un procesador comercial. Puede ser ventajosamente un procesador de 32 bits el tipo ARM 9™ con reloj a 100 MHz. El tipo de procesador, su posición central, próximo al botón de marcha/parada, su conexión al control de la alimentación hacen de él una herramienta bien adaptada para la gestión de la alimentación del robot (modo vigilia, parada de urgencia, etc.). La tarjeta comprende igualmente un conjunto de memorias RAM y flash.

La inteligencia del robot se implanta según una arquitectura de programación específica, que constituye un verdadero sistema operativo y de gestión de las interfaces del robot, 120, denominado NAOQI. Esta arquitectura se ha divulgado principalmente en la solicitud de patente WO 2009/124955 publicada el 15/10/2009. Los módulos del programa que componen esta arquitectura comprenden las funciones del sistema de gestión de las comunicaciones entre un robot y un PC o un emplazamiento distante y de intercambio de programas que proporcionan la infraestructura de programación necesaria para la puesta en práctica de la presente invención. Esta arquitectura comprende las funciones necesarias para la gestión de las funciones de base del robot, tales como moverse, accionar un miembro, interpretar las señales de un captador o enviar unas señales a través de un actuador. Permite igualmente al usuario del robot implantar en la memoria integrada a bordo de este unos programas de software de control de la ejecución por el robot de comportamientos 130, 1310, 1320, que pueden ser unos gestos, comprendidos en ellos unos gestos coordinados complejos tal como pasos de danza, palabras o emociones, siendo estas últimas en general unas combinaciones de gestos, de palabras y de emisión de señales por unos actuadores tales como unos LED.

Para la ejecución de estos comportamientos, el robot deberá acudir a unos recursos 140, 1410, 1420. La presente invención trata sobre las modalidades de gestión de dichos recursos para permitir una ejecución no conflictiva de los comportamientos. Se pone en práctica en lo esencial en el módulo de gestión de los recursos o “Resource manager server”, 1210, describiéndose el funcionamiento detallado de este módulo en lo que sigue de la presente descripción.

Es posible programar los comportamientos a ejecutar por el robot sobre un terminal remoto tal como la estación de trabajo 150, utilizando no importa qué programa cuyo lenguaje objetivo será C, C++, Python o Urbi. Para poner en práctica la presente invención, se utilizará sin embargo ventajosamente la arquitectura funcional específica de edición y de programación de los comportamientos de un robot humanoide. Una arquitectura de ese tipo se ha descrito mediante la solicitud de patente PCT/EP2010/057111 presentada el 25/05/2010. El programa informático de edición y de programación de los comportamientos de un robot humanoide, 160, que permite implementar dicha arquitectura se denomina comercialmente Chorégraphe™, y puede designarse indiferentemente por su nombre genérico o por su nombre comercial, sin alterar la generalidad de las referencias. Esta arquitectura es particularmente eficaz para generar unos comportamientos de robot porque permite realizar fácilmente una articulación de las órdenes desencadenadas por unos eventos con su dimensión temporal. Las órdenes desencadenadas por unos eventos se representan en Chorégraphe mediante unas Boxes o “Cajas” o “Cajas de órdenes”. Una caja es una estructura de programación arborescente que puede comprender uno o varios de los elementos siguientes que se definen a continuación:

- Un “Timeline” o eje temporal de tramas;
- Un “Diagram” o diagrama de flujo;
- Un Script.

Las Cajas de órdenes están normalmente unidas entre sí mediante unas conexiones que transmiten con frecuencia una información de eventos de una caja a la otra. Cualquier caja está unida directamente o indirectamente a una “Caja Raíz” o Root que inicializa escenario de comportamiento/movimiento del robot.

Un eje temporal de Tramas representa la restricción temporal a la que se someten los comportamientos y los movimientos del robot definidos en la Caja en la que dicho Eje temporal de Tramas o Timeline se inserta. El Timeline realiza de ese modo la sincronización de los comportamientos y movimientos de la Caja. Se divide en frames (Tramas) a los que se asocia una velocidad de desarrollo definida en número de Tramas por segundo o Frames Per Second (FPS). La FPS de cada Timeline es parametrizable por el usuario. Por defecto, la FPS puede fijarse a un valor dado, por ejemplo 15 FPS.

Una Timeline puede comprender: i) una o varias Behavior Layers o “Capas de comportamiento) que comprende cada una, una o varias Behavior Key Frames o “Tramas principales de comportamiento”, que pueden comprender por sí mismas uno o varios Diagrams o “Diagramas de flujo”, que son de hecho los conjuntos de Cajas que pueden igualmente unirse directamente con la Caja de nivel superior, sin pasar por una Capa de comportamiento ni una

Timeline; ii) una o varias Motion Layers o “Capas de movimiento”, en que cada una comprende una o varias Motion Key Frames o “Tramas principales de movimiento” que puede comprender una o varias Motion Screens o “Pantallas de movimiento”.

Un Diagrama de flujo es un conjunto de Cajas conectadas entre sí, como se detalla más adelante. Cada una de las Cajas puede a su vez comprender otras Timeline con las que se relacionan nuevas Capas de comportamiento o de movimiento.

Un Script es un programa directamente ejecutable por el robot. En el marco de la presente invención, los Scripts se escriben de manera preferida en lenguaje C++. Una caja que comprende un Script no comprende ningún otro elemento.

El software puede implantarse en un PC o en otra plataforma de tipo ordenador personal que utilice un sistema operativo, Windows™, Mac™ o Linux™.

En el marco de la presente invención, pueden considerarse unas pantallas específicas de Chorégraphe, que puede considerarse como un módulo de Chorégraphe denominado “Resource manager client”, 1610 que permite la programación de la reglas de gestión de los recursos de robots humanoides tales como NAO generando unos scripts que se integrarán en el módulo Resource manager server, 1210. La concepción de las Cajas de Chorégraphe, que pueden estar contenidas unas en las otras, es particularmente ventajosa para la implementación de la presente invención. En efecto, el comportamiento de más alto nivel, 130 de la figura 1, al que puede calificarse de comportamiento padre, podrá comprender por ejemplo dos comportamientos hijos, 1310, 1320, que podrán codificarse gráficamente (arrastrando y soltando Cajas preprogramadas y conexiones entre ellas) mediante unas Cajas que estarán incluidas en la Caja del comportamiento padre. Los recursos elementales 1410, 1420, que designan por ejemplo el recurso Pierna izquierda y el recurso Pierna derecha, pueden agruparse en un recurso único Miembros inferiores, 140. La Pierna izquierda y la Pierna derecha son por sí mismos unos recursos compuestos que comprende cada uno un muslo, una rodilla, una pierna, un tobillo, un pie, pudiendo cada uno de estos subconjuntos comprender por sí mismo uno o varios motores, uno o varios captadores, uno o varios actuadores. Chorégraphe gestiona mediante el desplazamiento de las Timeline la sincronización de los recursos suponiendo que estarán disponibles aquellos de los que tienen necesidad los comportamientos. Chorégraphe, en el Resource manager server de la presente invención, no gestiona la puesta a disposición de los recursos.

Los conflictos de puesta a disposición de los recursos de bajo nivel tales como los motores pueden resolverse mediante unos algoritmos clásicos de tipo “exclusión mutua” o mutex. Dichos algoritmos tienen la característica común de excluir la toma de un recurso gestionado de esta manera por dos procesos simultáneos. Los algoritmos de este tipo difieren principalmente en los mecanismos de gestión de las prioridades y de las esperas. Dichos algoritmos no permiten sin embargo resolver fácilmente unos conflictos de más alto nivel, cuando unos procesos deben acceder a unos recursos compuestos. Por otro lado, no puede concebirse una solución a este problema con el recurso a unos algoritmos de optimización de asignación de recursos, teniendo en cuenta la constante de tiempo necesariamente muy corta de los comportamientos de un robot para que tenga un aspecto humano, el consumo de CPU de dichos algoritmos y la debilidad de los recursos de cálculo integrados en el robot.

Otra dificultad del problema de gestión de recursos resuelto en el marco de la presente invención es que varios usuarios pueden programar unos comportamientos independientemente unos de los otros e implantar cada uno de sus comportamientos en un mismo robot (por supuesto con la condición de que dispongan de los derechos para hacer esto).

Como se detalla a todo lo largo de la descripción, la arquitectura de los comportamientos y de los recursos y las funciones de gestión de las relaciones entre estas entidades permiten principalmente, en el marco de la presente invención: evitar los conflictos de recursos (dos comportamientos no pueden utilizar el mismo motor al mismo tiempo, los comportamientos que manipulan una misma pelota no pueden ejecutarse al mismo tiempo); sincronizar la utilización de los recursos diferentes (para decir buenos días, se utiliza un brazo y el audio; estos recursos deben estar disponibles al mismo tiempo para tener un comportamiento coherente); gestionar los diferentes modelos de robots (por ejemplo, un robot sin brazos, con unas piernas, con unas ruedas, etc.) y un robot que tenga unos órganos averiados o funcionando en modo degradado (economía de energía); evitar los deadlock o situaciones de bloqueo sobre la atribución de los recursos en un sistema multi-thread (en el que se ejecutan varios procesos en paralelo); disponer de un mecanismo de resolución de conflictos; asegurar la disponibilidad de los recursos para los comportamientos críticos; proteger el robot cualesquiera que sean los comportamientos (no caer por ejemplo).

El Resource manager server 1210 asegura la disponibilidad de los recursos. Tiene una estructura de datos jerárquica en el marco de un desarrollo objeto: un objeto o un comportamiento padre puede reservar unos recursos, los comportamientos hijo tienen asegurado entonces obtener los recursos. Permite igualmente resolver rápidamente los conflictos: la resolución es descentralizada/local lo que permite superar la complejidad matemática y algorítmica (tiempos de cálculo) de un motor de resolución de restricciones. Quien genera el conflicto resuelve el conflicto, el Resource manager client no realiza ningún cálculo.

El Resource manager server comprende unos mecanismos que permiten evitar los bloqueos/deadlock en un sistema multithread (por ejemplo, gestión unitaria de un grupo de recursos –es decir todos los motores– como un simple recurso –es decir un motor–. El sistema es igualmente genérico: contrariamente a otros sistemas en los que la funcionalidad y los recursos están imbricados, el Resource manager puede gestionar no importa qué tipo de recursos y no está ligado a una funcionalidad. Puede compararse el enfoque tradicional con unos semáforos/mutex que protegen un dato preciso en una funcionalidad sin permitir un nivel suplementario de actuación como es el caso del Resource manager de la presente invención (sincronización de estos diferentes recursos).

Las figuras 2a y 2b son unas vistas de pantallas de un programa informático que permite la implementación de la invención en varios de sus modos de realización.

En la figura 2a se representa una pantalla de Chorégraphe. El módulo de Chorégraphe 160 carga sobre el robot los comportamientos 130, 1310, 1320 así como los recursos 140, 1410, 1420 utilizados por dichos comportamientos, habiendo sido definidas previamente las reglas de reserva de dichos recursos en el Resource manager client, 1610, según unas modalidades que se detallarán a continuación en la descripción. Un Resource manager server, 1210, es el correspondiente sobre el robot del Resource manager client.

Cuando Chorégraphe hace referencia a los recursos, los solicita al servidor del robot (o a un robot simulado en un PC). Con cada ejecución, el código de Chorégraphe se transfiere sobre el robot.

Un archivo de inicialización representado en la figura 2b lista el conjunto de los recursos del robot. El robot comprende ventajosamente un servidor web, 1220, que, bajo demanda del cliente, 1610, devuelve la lista de los recursos a la estación 150, pudiendo visualizarse dicha lista en el Resource manager client, 1610. Dicha lista puede visualizarse mediante Chorégraphe (Chorégraphe cliente del robot) o un navegador web (página web cliente del robot). Se observará sin embargo que la invención puede implementarse sin embargo sin recurso a Chorégraphe, ni a un módulo de Resource manager client, 1610. En este caso, la estación servidor 150 envía los comportamientos al robot, siendo interpretados dichos comportamientos por el módulo NAOQI, 120, y se proporciona la lista de los recursos en el servidor, 1220, siendo efectuada la jerarquización de los comportamientos y los reagrupamientos de los recursos en el Resource manager server, 1210.

De ese modo el Resource manager server, 1210, es un módulo del robot, subconjunto del NAOQI, que define el conjunto de los recursos disponibles. Un comportamiento contiene unas acciones (movimiento, audio, vídeo). Se codifica en C++ o Python, o se realiza gráficamente (encapsulado de Python o C++). El servidor web del robot puede devolver sobre un navegador la lista de los recursos así como los comportamientos que los utilizan y aquellos que esperan a la disponibilidad de un recurso. Siendo jerárquicos los recursos, el usuario puede ocupar fácilmente el alto, el bajo, una pierna del robot sin conocer el detalle de los motores situados en cada una de estas partes.

Una API (Application Programming Interface) permite utilizar el Resource manager 1210 en C++, en Python (C++ envuelto) o en el lenguaje gráfico (encapsulado del C++ o del Python) y generar las órdenes siguientes:

- Tomar o esperar un recurso único, o sea
  void waitForResource (const string& ressourceName, const string& ownerName, const string& callbackName, const int& timeoutSeconds);
- Tomar o esperar un recurso único dando unas informaciones jerárquicas entre los comportamientos (en lo que sigue de la descripción, los comportamientos que reservan unos recursos se llaman “Owners”) o sea
  void waitForLocalResourcesTree (const vector<string>& ressourceName, const class AL::ALPtr<class AL::ALHierarchyOwner>& treeOwnerPtr, const string& callbackName, const int& timeoutSeconds);
- liberar un recurso, o sea
  void releaseResource (const string& resourceName, const string& ownerName).

Las figuras 3a y 3b representan unas modalidades de gestión de recursos, respectivamente bajo la forma de listas de recursos disponibles y bajo la forma de programación gráfica por cajas, en varios modos de realización de la invención.

Estas figuras ilustran el principio de los tratamientos aplicados a la gestión de los conflictos sobre un recurso simple.

En el caso de un recurso simple (un motor por ejemplo), se trata de evitar cualquier deadlock. Solo se protege la lista de los recursos con la ayuda de un algoritmo del tipo mutex y este mutex se utiliza en la adición o la supresión del recurso. El conflicto en sí (espera a la liberación de un recurso) no está protegido aunque sea thread-safe. Los mutex son un sistema de protección de la librería estándar con el fin de convertir a una aplicación en thread-safe (accesible en un sistema paralelo con varios thread). Por el contrario, no impiden los deadlock. Los presentes inventores utilizan únicamente los mutex con el fin de proteger la lista de los recursos. La resolución del conflicto no acude al sistema de bloqueo que son los mutex sino que es sin embargo thread-safe.

La resolución de conflictos en la reserva de recursos se realiza mediante la aplicación de los principios siguientes: i) en un sistema multi-thread, dos Owners no pueden tomar el mismo recurso al mismo tiempo; se protege por tanto la

solicitud mediante un try-lock que es una tentativa no bloqueante de adquirir un recurso enclavado por un mutex; ii) cuando dos Owners solicitan el mismo recurso exactamente al mismo tiempo (definido por el planificador del sistema operativo), todos los Owners deben resolver el conflicto pero solo el primero obtiene el recurso cuando está libre (primero en llegar, primero servido).

Un ejemplo de este tipo se ilustra mediante la lista de los recursos de la figura 3a y las Cajas y la vista de la pantalla de definición de las relaciones de dependencias de Chorégraphe de la figura 3b.

En este ejemplo MoveHead1 posee el recurso. MoveHead2 espera al recurso. Un MoveHead3 esperaría igualmente al recurso pero MoveHead2 lo obtendría el primero (primero en llegar y primero servido). La asignación de recursos se realiza mediante un smart-pointer (puntero inteligente que comprende un código encapsulado ejecutado sobre los recursos apuntados lo que evita la utilización de un mutex (la asignación debe ser atómica para el sistema operativo).

Si dos thread ejecutan A = B (o en el caso de la aplicación ilustrada, OwnerRecursoA = nuevo_Owner), entonces la línea de código A = B debe protegerse mediante un mutex, si no la aplicación se interrumpe en el momento en el que se ejecuten dos threads. Ahora bien no se desea utilizar mutex para resolver este conflicto. Sin embargo, en ciertos casos, A = B no tiene necesidad de ser protegido, particularmente, si A y B son unos tipos atómicos (de tipo entero, flotante, etc.) que no pueden provocar jamás una interrupción de A = B por otro thread (y provocar una parada de la aplicación). Esto es por lo que en OwnerRecursoA = nuevo_Owner, OwnerRecursoA y nuevo_Owner son unos enteros (unos punteros). De ese modo, en ausencia de mutex, la presente aplicación no puede bloquearse (deadlock).

El algoritmo es entonces del tipo:

Esperar recurso-cabeza
Detección de las solicitudes simultáneas mediante un try-lock
resolución del conflicto (espera o timeout)
si soy el primero entrante después del try-lock:
puntero-sobre-mi-antiguo-owner = puntero-sobre-mi-nuevo-owner, si no, resolver aún el conflicto.

En Chorégraphe, se utilizan unas cajas Move-Head. Los dos movimientos de cabeza son paralelos y cada uno de entre ellos referencia a los mismos motores. En este caso, la opción lock significa que los otros solicitantes serán puestos en espera dentro del límite de su timeout.

Las figuras 4a y 4b representan un mecanismo de sincronización de recursos de diferentes tipos, respectivamente bajo la forma de organigrama funcional y bajo la forma de menú gráfico desplegable, en varios modos de realización de la invención.

Estas figuras ilustran el caso en el que un mismo Owner (comportamiento, objeto) solicita varios recursos que deben estar disponibles para que pueda ejecutarse. Un Owner puede ser una caja Chorégraphe pero igualmente un objeto C++ (en el sentido de orientación a objetos) que posee igualmente unas propiedades jerárquicas.

El objeto o el comportamiento están a la espera o previamente tienen reservado un recurso. Cuando un Owner solicita varios recursos, se crea un nuevo recurso único que referencia los sub-recursos. Este nuevo recurso permite tratar de la misma manera cada recurso (un motor por ejemplo) y el grupo de recursos (mismo timeout). Este mecanismo, ilustrado por el organigrama funcional de la figura 4a, permite sobre todo detener la espera de todos los recursos si uno solo no está disponible. El procedimiento de la invención evita de ese modo los bloqueos (deadlock).

Por el contrario, el algoritmo siguiente no funcionaría:

tomar (Lista_recursos)

Para cada recurso
solicitar_y_tomar el recurso

Como se ilustra en la figura 4b, en Chorégraphe, se procesa la selección de varios recursos lo que conduce a acudir al método waitResources (lista de los recursos, owner, callback, timeout) y a la generación del nuevo recurso único que referencia la lista de los recursos inicialmente seleccionados.

Además, la lista de recursos se ordena ventajosamente con el fin de evitar ciertos timeout inútiles; en el ejemplo ilustrado en la figura 4b, en el que el recurso A es el procesamiento de la señal de audio (Audio-TTS) y el recurso B un movimiento de oscilación del brazo izquierdo (Arm-Left arm-ShoulderPitch):

Owner 1 (thread1) solicita A+B: Owner 1 coge A y espera B,
Owner 2 (thread2) solicita B+A: Owner 2 coge B y espera A.

Sin una clasificación de los recursos, no se envía ninguna orden al robot, con una clasificación:

Owner 1 (thread1) solicita A+B: Owner 1 coge A, Owner 1 coge B, lo que permite la ejecución de la orden.
Owner 2 (thread2) solicita A+B: Owner 2 espera A hasta el timeout, o hasta la liberación del recurso A+B por el Owner 1.

Las figuras 5a y 5b representan un mecanismo de reserva de recursos críticos, respectivamente bajo la forma de organigrama funcional y bajo la forma de menú gráfico desplegable, en varios modos de realización de la invención.

Ciertos recursos deben estar disponibles para unos comportamientos críticos (recarga de la batería por ejemplo). Esta disponibilidad se asegura mediante una herencia de objetos de los recursos. Un objeto o comportamiento padre puede reservar unos recursos. Estos recursos estarán entonces disponibles para los hijos, por el contrario, los hijos se disputarán estos recursos.

Como se ilustra en el organigrama funcional de la figura 5a, el Owner 1 no utiliza forzosamente los recursos, les ocupa para los objetos o comportamientos hijo. El algoritmo del gestor de recursos impone a todo hijo devolver sus recursos cuando él mismo los libera. Dicho de otra manera, resolver un conflicto con un hijo obliga, si el hijo libera estos recursos, a resolver el conflicto igualmente con el padre.

Como se ilustra en la figura 5b, en Chorégraphe, se crea una Caja padre de tipo Diagrama de flujo (objeto jerárquico para el servidor gestor de recursos o Caja que contiene unas Cajas):

Las figuras 6a y 6b representan un mecanismo de herencia de una reserva de recursos, respectivamente bajo la forma de menú gráfico desplegable y bajo la forma de programación gráfica por cajas, en varios modos de realización de la invención.

En el caso de la implementación de la invención ilustrado por estas figuras, el padre Owner 1 reserva un recurso A+B (A = TTS y B = Arms-LeftArm-ShoulderPitch).

Las Cajas (comportamientos Owner 2, Owner 3) encapsuladas en el padre Owner 1 tienen este recurso compuesto a su disposición y el primero que lo requiere puede utilizarlo inmediatamente.

Las figuras 7a, 7b y 7c ilustran la problemática de la gestión de los recursos ausentes, respectivamente representando un robot completo, un robot tronco y una lista de recursos disponibles/indisponibles, en varios modos de realización de la invención.

El Gestor de recursos puede gestionar los modelos de robots congelando los recursos que no existen. En un modo ventajoso de la invención, es posible de ese modo realizar unos comportamientos independientes del modelo de robot. Una empresa no puede en efecto rehacer el conjunto de los comportamientos de sus clientes si el robot cambia.

En el caso de un modelo de robot que disponga de todos las partes de un robot completo (todos sus miembros en particular, tal como se ilustra en la figura 7a), el Resource manager reconoce que el modelo de robot está completo y no implementa el mecanismo de congelación de recursos.

En el caso del modelo de robot de tronco sin piernas, tal como se ilustra en la figura 7b, después de la inicialización de la aplicación, se obtiene la lista de los recursos representada en la figura 7c, en la que los recursos correspondientes a las piernas ausentes del modelo de robot tronco están congelados (es decir no disponibles para no importa qué Owner). Se reservan para un comportamiento tal que LHipYawPitch ->ALFrameManager_0x9425af8_root_ RobotModel_3eS trong0 (1ª línea del subconjunto de Legs de la lista de recursos de la figura 7c). Este comportamiento puede calificarse de artificiosidad o de comportamiento virtual en la medida en que no tiene vocación de ejecutarse.

El mismo mecanismo de congelación, total o parcial de recursos, puede aplicarse para gestionar las averías de ciertas partes del robot o prevenirlas. Supóngase en particular que un motor de la cabeza se calienta, un comportamiento hotEngine ocupará entonces al motor reduciendo la energía enviada hasta que el motor se refrigere. Igualmente, si se rompe un brazo, un comportamiento brokenArm ocupará al motor roto con el fin de que ningún otro comportamiento lo utilice.

Por extensión de este concepto de gestión de los recursos ausentes o deficientes, pueden definirse los recursos esenciales/no esenciales para un comportamiento. La disponibilidad de los recursos declarados como esenciales para un comportamiento es imperativa para la ejecución de dicho comportamiento. Por el contrario, dicho comportamiento se podrá ejecutar a pesar de la indisponibilidad de recursos declarados como no esenciales. Por ejemplo, si un comportamiento es un Saludo que comprende normalmente la combinación de un mensaje vocal “Buenos días” (Recurso A = audio), de un gesto del brazo derecho (Recurso B = RightArm) y de dos pasos hacia adelante (Recurso C = LeftLeg; Recurso D = RightLeg), y los recursos A y B son declarados esenciales y los

recursos C y D son declarados como no esenciales, el comportamiento Saludo se podrá ejecutar incluso si los recursos C y D no están disponibles porque, por ejemplo, el robot está en el curso de una marcha.

Si se utiliza el concepto de recursos esenciales/no esenciales, un comportamiento para el que unos recursos se definen como no esenciales y que están ausentes o deficientes se podrá ejecutar, mientras que un comportamiento para el que los mismos recursos se declaran esenciales no podrá ejecutarse. Si un comportamiento acude a la vez a unos recursos esenciales y a unos recursos no esenciales, se ejecutará, en función de la disponibilidad o no de los recursos no esenciales, de manera diferente.

Pero pueden también, para obtener la realización de la misma función, definirse todos los recursos como que son esenciales mientras se divide un comportamiento en subcomportamientos. Por defecto, los subcomportamientos no se ejecutan si están indisponibles unos recursos. Por defecto igualmente, el comportamiento global (padre) se ejecutará siempre. Pero puede modificarse la acción por defecto a nivel del comportamiento padre y de los subcomportamientos (o comportamientos hijo) como se ha explicado más arriba.

Las figuras 8a a 8e ilustran varias modalidades de resolución de conflictos, respectivamente bajo la forma de organigrama funcional (8a) y bajo la forma de menús gráficos desplegables que corresponden a diferentes opciones de salida de un conflicto (8b a 8e), en varios modos de realización de la invención.

Como se ha ilustrado en la figura 8a, el enfoque multi-agente/distribuido permite superar los algoritmos complejos y poco compatibles con el tiempo real. El objeto o el comportamiento resuelve por sí mismo el conflicto de manera local. O bien rechaza la solicitud de recursos por el o los otros comportamientos que lo soliciten (caso ilustrado en la figura 8b). O bien libera inmediatamente los recursos para que puedan ser utilizados por otro comportamiento que lo solicite (caso ilustrado en la figura 8c). O bien ejecuta previamente una parte del comportamiento que le permita ponerse en posición de estabilidad antes de liberar dichos recursos (caso ilustrado en la figura 8d; por ejemplo, cuando un movimiento está en curso, termina este movimiento hasta la próxima posición de estabilidad). O bien el comportamiento titular de los recursos se pone en pausa y libera los recursos solicitados, durando la pausa hasta que el otro comportamiento que toma los recursos los libera (caso ilustrado en la figura 8e). En todos los casos en los que los recursos no son liberados inmediatamente, el comportamiento solicitante se pone en posición de espera de recursos hasta un timeout cuya duración puede predefinirse. En todos los casos, se evita el deadlock.

La notificación se realiza mediante un sistema de callback. En C++ o en Python: si un owner posee un recurso, se le notifica la solicitud en un callback. En Chorégraphe, se notifica al comportamiento la solicitud en un procedimiento de callback proporcionando una ayuda al usuario proponiendo unas soluciones de resolución en este procedimiento de callback (del que el usuario no tiene necesidad de conocer la existencia):

* La notificación provoca una parada del comportamiento que posee el recurso
* La notificación no hace nada, el comportamiento que la posee la guarda.
* La notificación pone en pausa el comportamiento que posee el recurso.
* La notificación acude a una función del usuario que hace lo que desea (por ejemplo se pone en una posición estable antes de liberar el recurso). Más precisamente, en Chorégraphe, las opciones siguientes son programables por un usuario:

- Lock: tomar los recursos sin liberarlos si son solicitados por otro comportamiento (figura 8b);
- Stop on demand: tomar los recursos y liberarlos si son solicitados (figura 8c);
- Callback on demand: permite efectuar no importa qué código de usuario en caso de conflicto de recursos; por ejemplo en el caso de una danza, poner el robot en posición estable antes de detener la danza y liberar los recursos;
- Pause on demand: permite poner en pausa un comportamiento en caso de conflicto de recursos y retomar el comportamiento en el que estaba si los recursos son libres de nuevo.

Por defecto los comportamientos guardan los recursos. Aquellos que esperan un recurso irán por tanto a esperar el fin del comportamiento.

La figura 9 ilustra la utilización de los recursos de estado bajo la forma de menú gráfico desplegable, en varios modos de realización de la invención; esta figura ilustra la utilización, en unos modos de realización de la invención, de un tipo particular de recurso, denominado Recurso de Estado.

Evitar unos conflictos de utilización de los motores, del audio y del video es insuficiente para proteger a un robot humanoide, en particular contra riesgos de caída que puedan implicar averías o roturas. Existen unos mecanismos de protección de nivel más bajo, tales como el procedimiento de gestión de las caídas divulgado por la solicitud de patente FR/ 1056798 presentada por el solicitante. Es ventajoso sin embargo poder prevenir las caídas. Por ejemplo, un corte de los motores mientras que el robot marcha tiene como consecuencia hacer caer al robot. Para prevenir este tipo de circunstancias, según la invención, es posible definir un recurso de estado cuya disponibilidad, definida por el estado de actividad de dicho recurso de estado es necesaria para la ejecución de ciertos comportamientos. Un recurso de estado es compartido (pueden ocuparle varios objetos o comportamientos). Solo está disponible un único

estado del recurso (el estado activado). Si el estado no está disponible, el recurso responde al mismo mecanismo descrito anteriormente (espera del recurso hasta un timeout y notificación a los que ocupan el recurso). En el caso ilustrado en la figura 9, el recurso de estado es Sit, es decir que el robot debe estar en posición Sit (o sentado) para que ciertos comportamientos (principalmente cortar todos los motores) puedan ejecutarse. En tanto que este recurso no esté disponible, el comportamiento “cortar todos los motores” no puede ejecutarse (esta desactivado durante el timeout fijado para este comportamiento).

En Chorégraphe, es posible por tanto seleccionar unos Recursos de Estado. Es posible generalizar el ejemplo de la figura 9 a otros estados, tales como: camera1/camera2 (el robot puede estar dotado de varias cámaras de campos diferentes, cuya utilización no sea pertinente más que en ciertas posiciones), conectado a la red/no conectado (ciertas funciones necesitan la utilización de una energía incompatible con los recursos autónomos del robot), etc.

La figura 10 representa un organigrama funcional de la utilización de un recurso acumulativo compartido, en varios modos de realización de la invención.

Esta figura ilustra la utilización, en unos modos de realización de la invención, de un tipo particular de recurso, denominado Recurso Acumulativo.

Este tipo de recurso encuentra aplicación en particular para optimizar la gestión de la CPU del robot. Los diferentes usuarios no pueden lanzar todos los comportamientos, pueden hacerlo en función de la disponibilidad de los recursos (recurso simple tal como un motor), de la posición del robot (Recurso de Estado) y de la CPU (Recurso Acumulativo), de la red (Recurso Acumulativo).

Cada comportamiento declara la parte de la CPU que ocupa, siendo esta CPU un recurso compartido como los recursos de tipo Recurso de Estado. Su utilización está limitada sin embargo al 100%. En el caso ilustrado en la figura 10, Owner 1 declara utilizar el 10% del Recurso Acumulativo A y Owner 2 declara utilizar el 90% del mismo Recurso Acumulativo A. Owner 3 espera la disponibilidad del Recurso Acumulativo A hasta su timeout. Se notifica a Owner 1 y Owner 2 sobre la solicitud de Owner 3 y puede responder a la solicitud mediante una de las opciones indicadas anteriormente en el presente documento.

La invención puede implementarse combinando los diferentes ejemplos de realización representados en las figuras y descritos de manera independiente en la descripción. De ese modo, pueden definirse varios comportamientos o subcomportamientos reales o virtuales y acudir de modo combinado a unos recursos simples, unos Recursos de Estado, unos Recursos Acumulativos, que pueden ser unos recursos esenciales o no esenciales.

Los modos de realización de la invención se han descrito en el caso de un único robot. Es posible sin embargo generalizar la aplicación a unos casos en los que varios robots, que comunican entre sí, son conducidos a compartir unos recursos. De ese modo:

- si se utilizan los recursos de un robot, entonces no importa qué robot (cliente) del sistema puede interrogar al gestor de recursos de los otros robots (servidor) para tomar un recurso;
- los robots pueden también utilizar unos recursos virtuales: una bola por ejemplo; en este caso, un único robot servidor debe referenciar el recurso bola; los otros robots (clientes) pueden tomar entonces este recurso ante el robot servidor.

Se da a continuación un ejemplo ilustrativo del segundo caso:

- El robot 1 desea tomar la bola (solicitud al robot servidor);
- El robot 1 no puede tomar la bola (timeout) porque otro robot posee el recurso bola.
- De ese modo, dos robots del mismo equipo no intentan por tanto disputarse la misma bola.

Los ejemplos descritos anteriormente se dan a título de ilustración de modos de realización de la invención. No limitan de ninguna manera el campo de la invención que se define por las reivindicaciones que siguen:

REIVINDICACIONES

1. Robot humanoide adecuado para ejecutar una pluralidad de comportamientos (130, 1310, 1320) bajo el control de un módulo de gestión integrado a bordo (120) que comprende un submódulo de gestión (1210) de una pluralidad de recursos (140, 1410, 1420) que pueden estar asignados para la ejecución de dichos comportamientos, estando dicho robot **caracterizado por que** dicha pluralidad de comportamientos y dicha pluralidad de recursos están organizados cada uno en subconjuntos jerárquicos y **por que** dicho submódulo de gestión de los recursos es adecuado para ser programado para reservar al menos un recurso que pertenece a al menos un subconjunto de recursos para la ejecución de al menos un comportamiento que pertenece a al menos un subconjunto de comportamientos, pudiendo programarse dicha reserva para ser heredada por dicho subconjunto de comportamientos y/o aplicada a dicho subconjunto de recursos.

2. Robot humanoide según la reivindicación 1, **caracterizado por que** dicho submódulo de gestión de la pluralidad de recursos está programado para responder, cuando un primer comportamiento está en curso de ejecución, a una solicitud de recurso por un segundo comportamiento mediante al menos una de las acciones siguientes: rechazo de puesta a disposición de dicho recurso a dicho segundo comportamiento, puesta a disposición inmediata de dicho recurso a dicho segundo comportamiento, ejecución de al menos una acción suplementaria, posteriormente puesta a disposición de dicho recurso a dicho segundo comportamiento, puesta en pausa del primer comportamiento en curso de ejecución, posteriormente puesta a disposición de dicho recurso a dicho segundo comportamiento.

3. Robot humanoide según la reivindicación 2, **caracterizado por que** dicho submódulo de gestión de la pluralidad de recursos está programado para que, cuando se ejecuta la acción de rechazo de puesta a disposición de dicho recurso requerido por dicho segundo comportamiento, dicho segundo comportamiento ejecute una nueva solicitud de puesta a disposición según una frecuencia predefinida hasta expiración de un período de expiración igualmente predefinido.

4. Robot humanoide según una de las reivindicaciones 1 a 3, **caracterizado por que** dicho submódulo de gestión de la pluralidad de recursos está programado para que, cuando al menos dos comportamientos ejecutan una solicitud de puesta a disposición de un mismo recurso, el primero de los al menos dos comportamientos sea prioritario en la reserva de dicho mismo recurso cuando se libera este último.

5. Robot humanoide según una de las reivindicaciones 1 a 4, **caracterizado por que** dicho submódulo de gestión de la pluralidad de recursos está programado para que, cuando al menos un comportamiento ejecuta una solicitud de puesta a disposición de al menos dos recursos, dichos al menos dos recursos se reagrupen en un nuevo recurso tratado como recurso único, ordenándose dicho reagrupamiento de manera única para todos los comportamientos que ejecutan solicitudes de recursos idénticas.

6. Robot humanoide según una de las reivindicaciones 1 a 5, **caracterizado por que** dicho submódulo de gestión de la pluralidad de recursos está programado para que, cuando un comportamiento padre reserva al menos un recurso sin utilizarlo, dicho al menos un recurso pueda utilizarse por parte de un comportamiento hijo de dicho comportamiento padre.

7. Robot humanoide según la reivindicación 6, **caracterizado por que** dicho submódulo de gestión de la pluralidad de recursos está programado para que, cuando un primer comportamiento hijo utiliza un recurso reservado por dicho comportamiento padre, un segundo comportamiento hijo del mismo padre deba, para poder utilizar dicho recurso, efectuar una solicitud de puesta a disposición de dicho recurso ante el primer comportamiento hijo.

8. Robot humanoide según una de las reivindicaciones 6 a 7, **caracterizado por que** dicho submódulo de gestión de la pluralidad de recursos está programado para que, cuando un comportamiento padre libera un recurso del que tiene la reserva, dicho recurso deba liberarse por todo comportamiento hijo de dicho padre.

9. Robot humanoide según una de las reivindicaciones 1 a 8, **caracterizado por que** dicho submódulo de gestión de la pluralidad de recursos está programado para, cuando un subconjunto de recursos tiene una disponibilidad reducida, realizar una reserva de dichos recursos para un comportamiento virtual definido en función de dicha reducción de disponibilidad.

10. Robot humanoide según la reivindicación 9, **caracterizado por que** dicho submódulo de gestión de la pluralidad de recursos está programado para liberar dicha reserva en función de la evolución del estado de dicha reducción de disponibilidad.

11. Robot humanoide según una de las reivindicaciones 1 a 10, **caracterizado por que** la pluralidad de recursos comprende al menos una variable de estado, debiendo estar dicha al menos una variable en un estado predefinido para ser reservada, siendo necesaria dicha reserva para la ejecución de al menos un comportamiento.

12. Robot humanoide según una de las reivindicaciones 1 a 11, **caracterizado por que** la pluralidad de recursos comprende al menos un recurso acumulativo y **por que** una tasa de utilización de este se asigna a los

comportamientos programados para utilizar dicho al menos un recurso.

13. Robot humanoide según una de las reivindicaciones 1 a 12, **caracterizado por que** la pluralidad de recursos comprende al menos un recurso esencial y un recurso no esencial, no pudiéndose ejecutar ningún comportamiento que deba acudir a al menos un recurso esencial si dicho recurso esencial no está disponible y pudiéndose ejecutar si debe acudir a dicho al menos un recurso no esencial, incluso si este no está disponible.

14. Procedimiento de gestión de los recursos de un robot humanoide adecuado para ejecutar una pluralidad de comportamientos (130, 1310, 1320) bajo el control de un módulo de gestión integrado a bordo (120), comprendiendo dicho procedimiento una etapa de reserva de una pluralidad de recursos (140, 1410, 1420) que pueden asignarse para la ejecución de dichos comportamientos, estando dicho procedimiento **caracterizado por que** dicha pluralidad de comportamientos y dicha pluralidad de recursos están organizanados cada uno en subconjuntos jerárquicos y **por que** dicha etapa de reserva asigna al menos un recurso que pertenece a al menos un subconjunto de recursos para la ejecución de al menos un comportamiento que pertenece a al menos un subconjunto de comportamientos, pudiendo programarse dicha reserva para ser heredada por dicho subconjunto de comportamientos y/o aplicada a dicho subconjunto de recursos.

15. Programa de ordenador que comprende unas instrucciones de código de programa que permiten la ejecución de un procedimiento según la reivindicación 14 cuando el programa se ejecuta en un ordenador, estando adaptado dicho programa para permitir la gestión de los recursos de un robot humanoide adecuado para ejecutar una pluralidad de comportamientos (130, 1310, 1320) bajo el control de un subprograma de gestión integrado a bordo (120), comprendiendo dicho subprograma un módulo (1210) adecuado para ejecutar una función de reserva de una pluralidad de recursos (140, 1410, 1420) que pueden asignarse para la ejecución de dichos comportamientos, estando dicho programa **caracterizado por que** dicha pluralidad de comportamientos y dicha pluralidad de recursos están organizados cada uno en subconjuntos jerárquicos y **por que** dicha función de reserva asigna al menos un recurso que pertenece a al menos un subconjunto de recursos para la ejecución de al menos un comportamiento que pertenece a al menos un subconjunto de comportamientos, pudiendo programarse dicha reserva para ser heredada por dicho subconjunto de comportamientos y/o aplicada a dicho subconjunto de recursos.

16. Programa de ordenador según la reivindicación 15, **caracterizado por que** dichos subconjuntos de comportamientos y de recursos se definen como clases de objetos.

17. Procedimiento de desarrollo de un módulo de gestión (1610) de los recursos de un robot humanoide adecuado para ejecutar una pluralidad de comportamientos (130, 1310, 1320) bajo el control de un subprograma de gestión integrado a bordo (120), siendo adecuado dicho subprograma para ejecutar una función de reserva de una pluralidad de recursos (140, 1410, 1420) que pueden asignarse para la ejecución de dichos comportamientos, estando dicho procedimiento **caracterizado por que** dicha pluralidad de comportamientos y dicha pluralidad de recursos están organizados cada uno en subconjuntos jerárquicos y **por que** dicha función de reserva asigna al menos un recurso que pertenece a al menos un subconjunto de recursos para la ejecución de al menos un comportamiento que pertenece a al menos un subconjunto de comportamientos, pudiendo programarse dicha reserva para ser heredada por dicho subconjunto de comportamientos y/o aplicada a dicho subconjunto de recursos.

18. Programa de ordenador que comprende unas instrucciones de código de programa que permiten la ejecución del procedimiento según la reivindicación 17 cuando el programa se ejecuta en un ordenador, estando adaptado dicho programa para permitir, mediante la interacción con un módulo remoto de gestión (1610) de los recursos de un robot humanoide adecuado para ejecutar una pluralidad de comportamientos (130, 1310, 1320), la parametrización de un subprograma de gestión integrado a bordo (120) de dicho robot, comprendiendo dicho subprograma un módulo (1210) adecuado para ejecutar una función de reserva de una pluralidad de recursos (140, 1410, 1420) que pueden asignarse para la ejecución de dichos comportamientos, estando dicho programa **caracterizado por que** dicha pluralidad de comportamientos y dicha pluralidad de recursos están organizados cada uno en subconjuntos jerárquicos y **por que** dicha función de reserva asigna al menos un recurso que pertenece a al menos un subconjunto de recursos para la ejecución de al menos un comportamiento que pertenece a al menos un subconjunto de comportamientos, pudiendo programarse dicha reserva para ser heredada por dicho subconjunto de comportamientos y/o aplicada a dicho subconjunto de recursos.

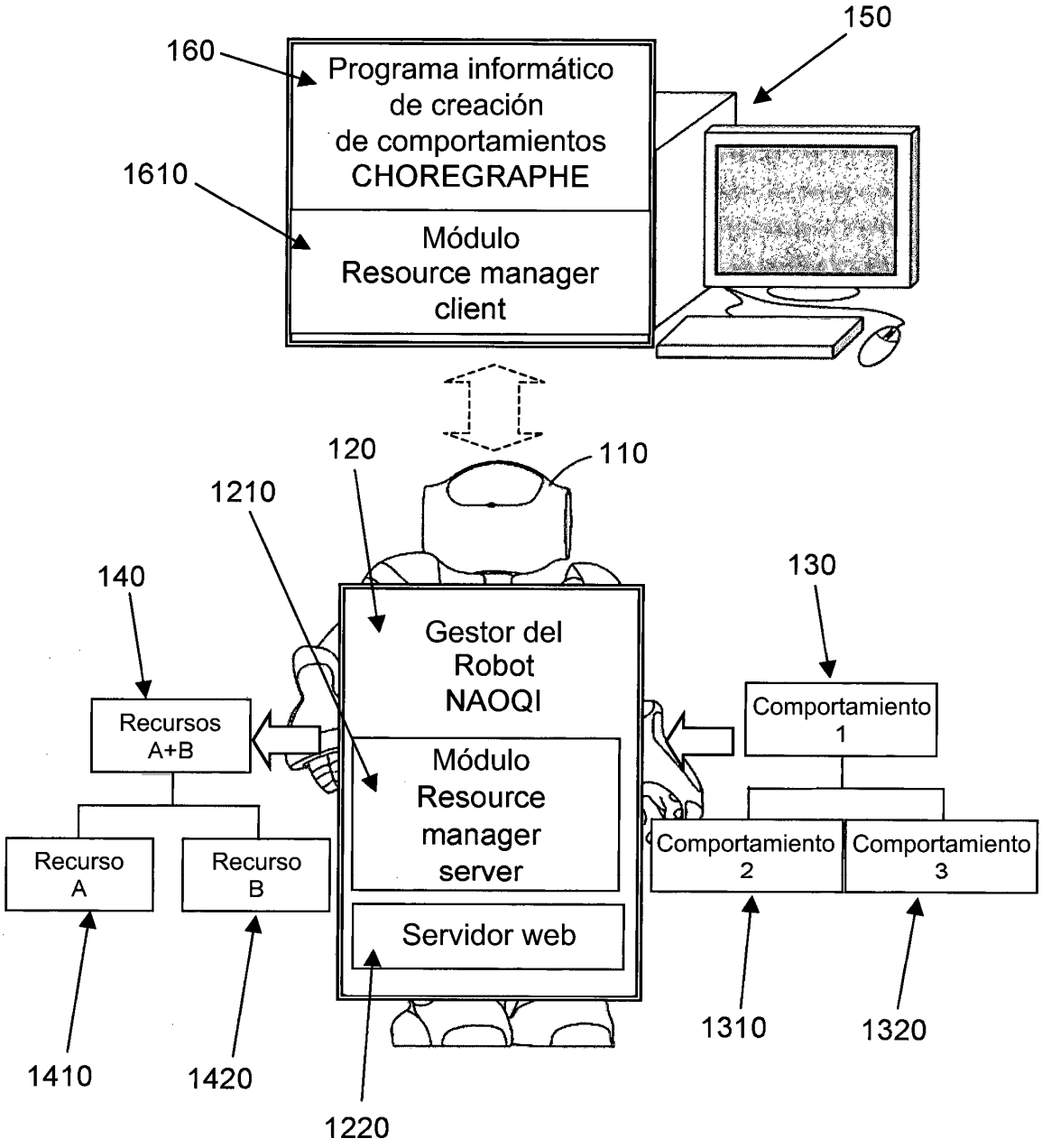
150
160
Programa informático de creación de comportamientos CHOREGRAPHE
1610
Módulo Resource manager client
120
110
1210
140
130
Gestor del Robot NAOQI
Recursos A+B
Comportamiento 1
Módulo Resource manager server
Recurso A
Recurso B
Comportamiento 2
Comportamiento 3
Servidor web
1410
1420
1220
1310
1320

FIG.1

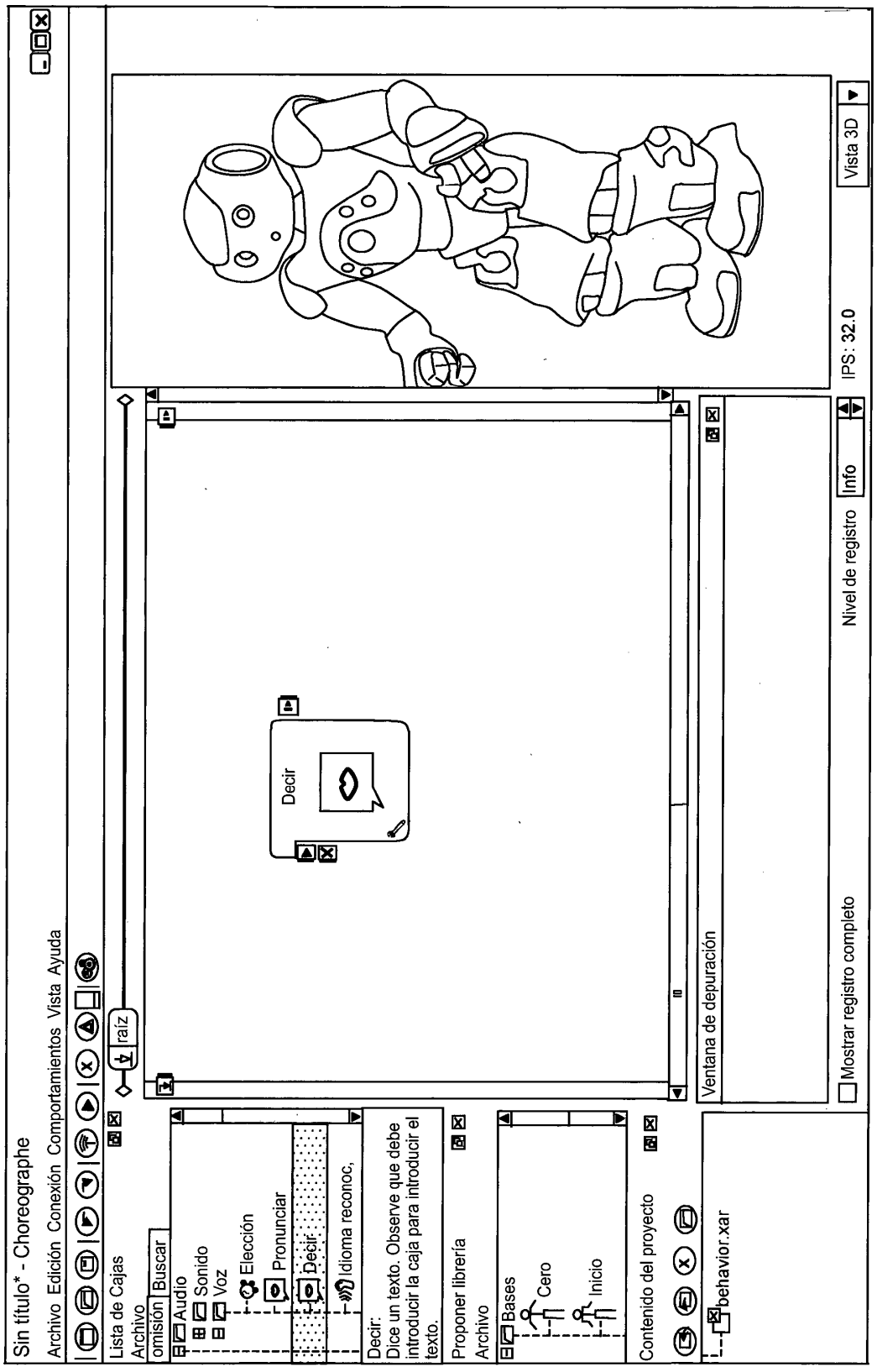
Sin título* - Choreographe
Archivo Edición Conexión Comportamientos Vista Ayuda
Lista de Cajas
Archivo
omisión
Buscar
Audio
Sonido
Voz
Elección
Pronunciar
Decir
Idioma reconoc,
Decir:
Dice un texto. Observe que debe introducir la caja para introducir el texto.
Proponer librería
Archivo
Bases
Cero
Inicio
Contenido del proyecto
behavior.xar
raíz
Decir
Ventana de depuración
Mostrar registro completo
Nivel de registro
Info
IPS: 32.0
Vista 3D

FIG.2a

-----root
-------State
---------Stand Waiter:
---------Sit (Activated) Waiter:
---------Back Waiter:
-------Leds
---------Left eye
---------Right eye
---------Left ear
---------Right ear
---------Chest
---------Head
---------Left foot
---------Right foot
-------Audio
---------TTS
---------Microphone
-------All motors
---------Head
---------- HeadYaw
----------HeadPitch
--------Arms
----------Left arm
------------LShoulderPitch
------------LShoulderRoll
------------LElbowRoll
------------LElbowYaw
------------LWristYaw
------------ LHand
----------Right arm
------------RShoulderPitch
------------RShoulderRoll
------------RElbowRoll
------------RElbowYaw
------------RWristYaw
------------RHand
---------Legs
----------LHipYawPitch
----------Left leg
------------LHipRoll
------------LHipPitch
------------LKneePitch
------------LAnklePitch
------------LAnkleRoll
----------Right leg
------------RHipRoll

FIG.2b

root
State
Stand Waiter:
Sit (Activated) Waiter:
Back Waiter:
Leds
Left eye
Right eye
Left ear
Right ear
Chest
Head
Left foot
Right foot
Audio
TTS
Microphone
All motors
Head
HeadYaw ->ALFrameManager_0x947f000_root_MoveHead_1 eStrong ALFrameManager_0x947f000_root_MoveHead_2 1
HeadPitch ->ALFrameManager_0x947f000_root_MoveHead_1 eStrong ALFrameManager_0x947f000_root_MoveHead_2 1
Arms
Left arm
LShoulderPitch
LShoulderRoll
LElbowRoll
LElbowYaw
LWristYaw
LHand

FIG.3a

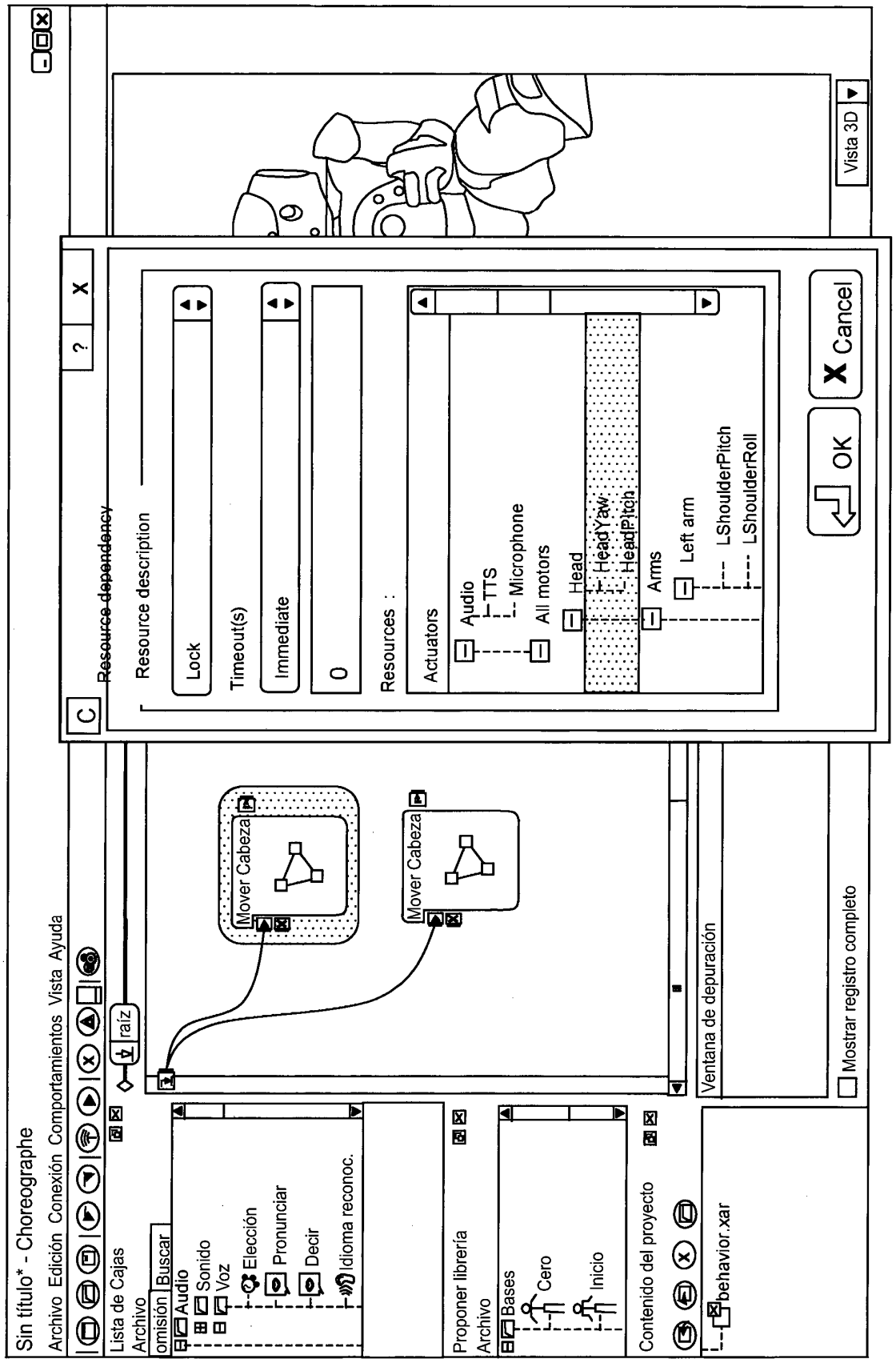
Sin título* - Choreographe
Archivo Edición Conexión Comportamientos Vista Ayuda
Lista de Cajas
Archivo
omisión Buscar
Audio
Sonido
Voz
Elección
Pronunciar
Decir
Idioma reconoc.
raiz
Mover Cabeza
Mover Cabeza
Resource dependency
Resource description
Lock
Timeout(s)
Immediate
0
Resources :
Actuators
Audio
TTS
Microphone
All motors
Head
HeadYaw
HeadPitch
Arms
Left arm
LShoulderPitch
LShoulderRoll
OK
Cancel
Vista 3D
Proponer librería
Archivo
Bases
Cero
Inicio
Contenido del proyecto
behavior.xar
Ventana de depuración
Mostrar registro completo

FIG.3b

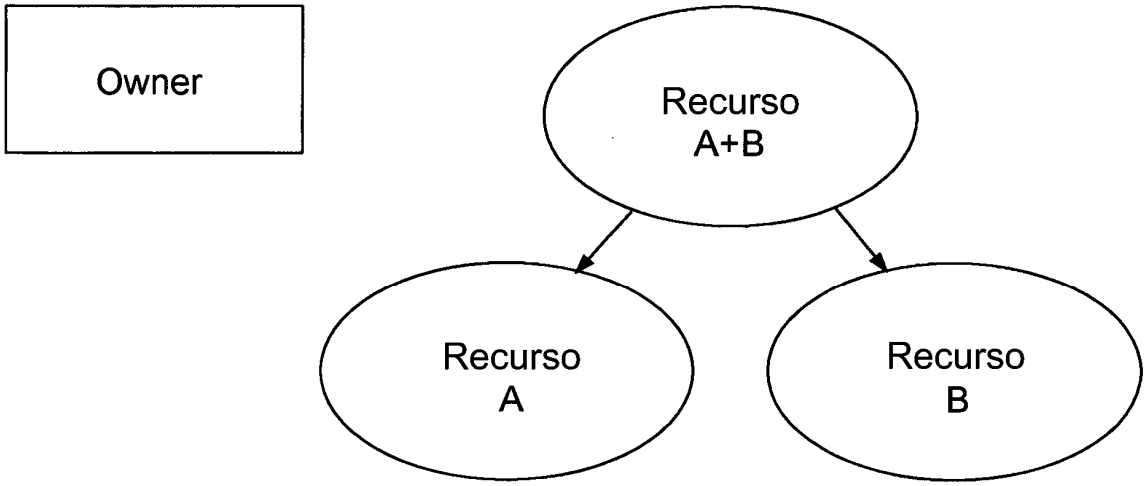
Owner
Recurso A+B
Recurso A
Recurso B

FIG.4a

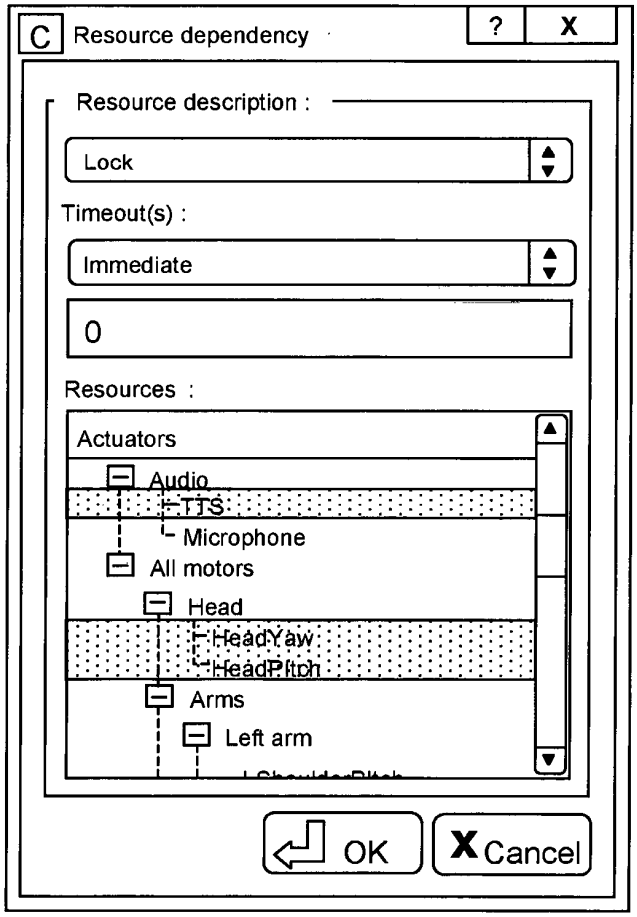
C
Resource dependency
?
X
Resource description :
Lock
Timeout(s) :
Immediate
0
Resources :
Actuators
Audio
TTS
Microphone
All motors
Head
HeadYaw
HeadPitch
Arms
Left arm
OK
Cancel

FIG.4b

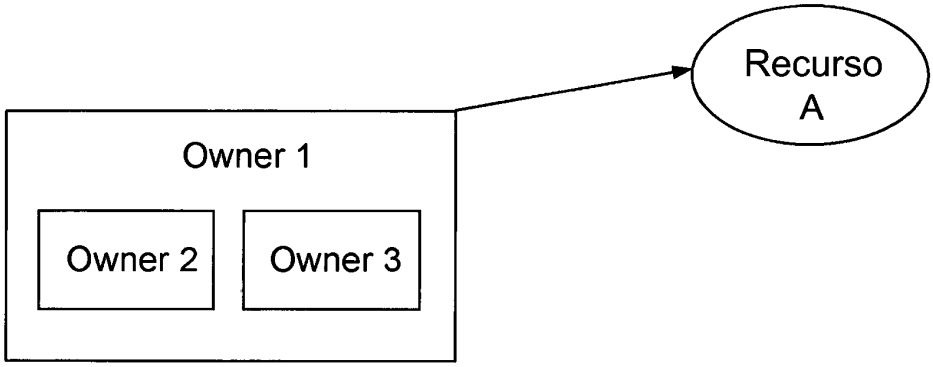
Recurso
A
Owner 1
Owner 2
Owner 3

FIG.5a

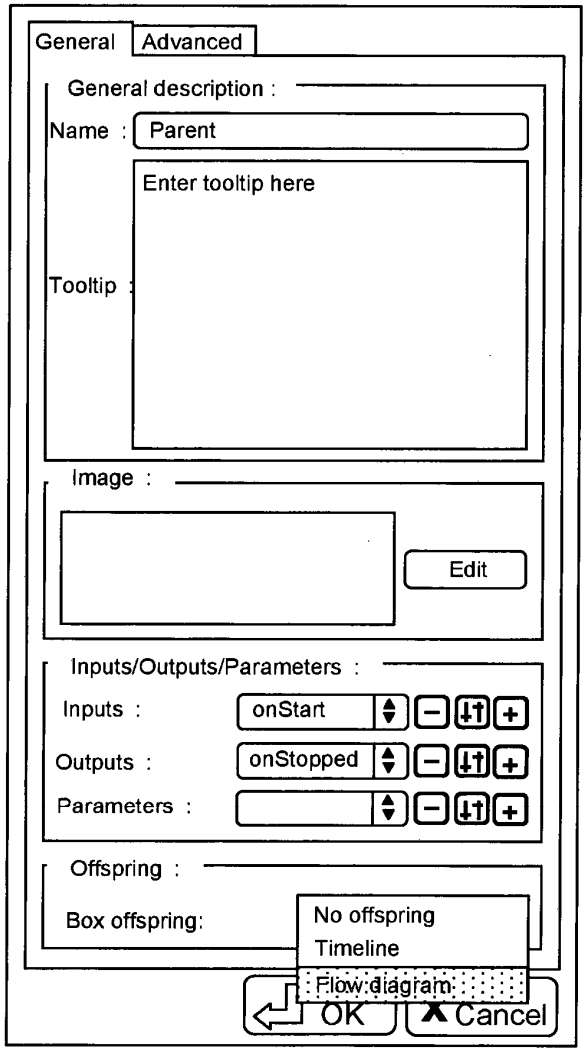
General
Advanced
General description :
Name :
Parent
Enter tooltip here
Tooltip :
Image :
Edit
Inputs/Outputs/Parameters :
Inputs :
onStart
Outputs :
onStopped
Parameters :
Offspring :
Box offspring:
No offspring
Timeline
Flow diagram
OK
Cancel

FIG.5b

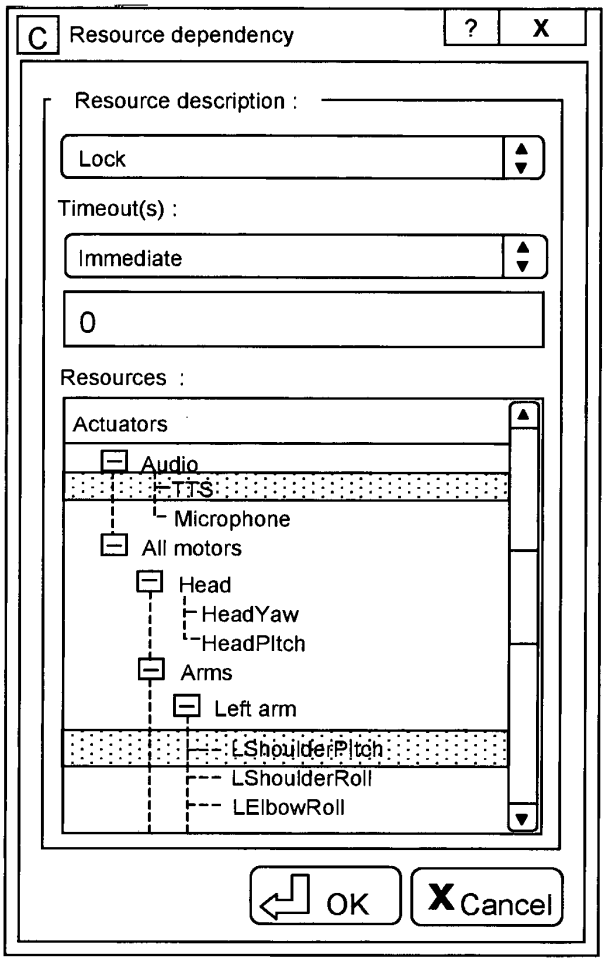
Resource dependency
?
X
Resource description :
Lock
Timeout(s) :
Immediate
0
Resources :
Actuators
Audio
TTS
Microphone
All motors
Head
HeadYaw
HeadPitch
Arms
Left arm
LShoulderPitch
LShoulderRoll
LElbowRoll
OK
Cancel

FIG.6a

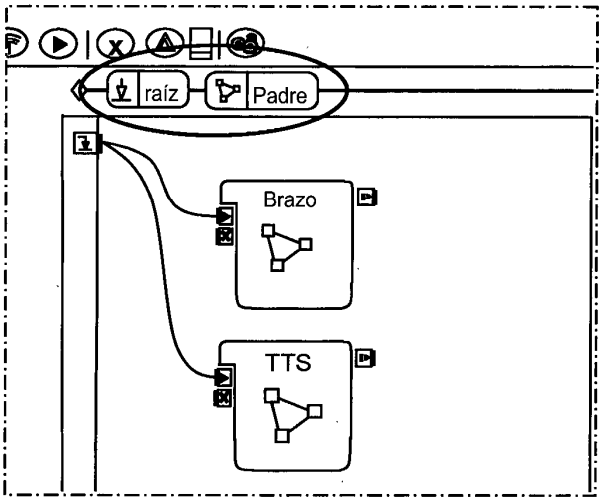
raíz
Padre
Brazo
TTS

FIG.6b

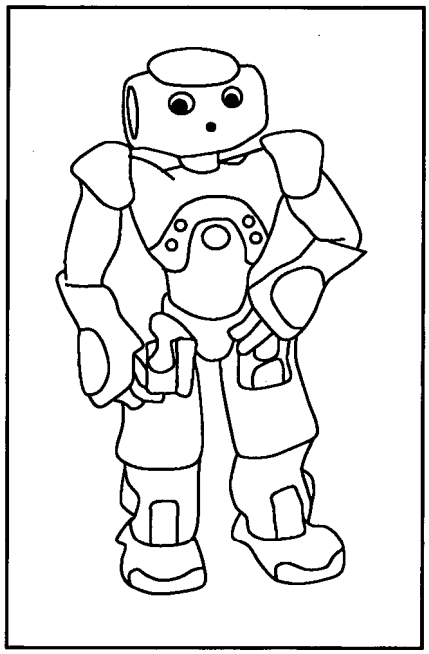

FIG.7a

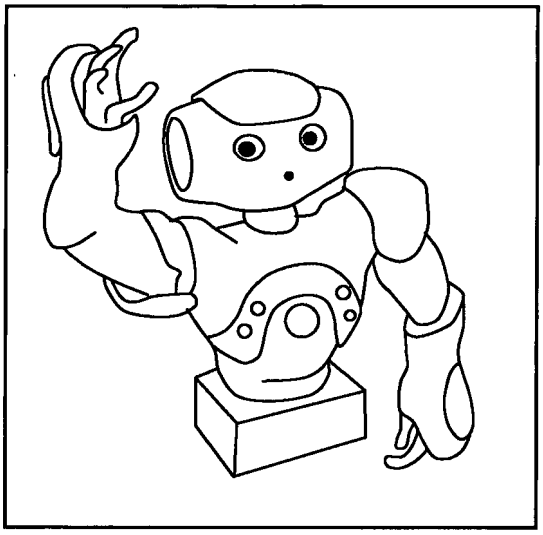

FIG.7b

---------Right foot
-------Audio
---------TTS
---------Microphone
-------All motors
---------Head
---------- HeadYaw
----------·HeadPitch
--------Arms
----------·Left arm
------------·LShoulderPitch
------------·LShoulderRoll
------------·LElbowRoll
------------·LElbowYaw
------------·LWristYaw
------------ LHand
----------·Right arm
------------·RShoulderPitch
------------·RShoulderRoll
------------·RElbowRoll
------------·RElbowYaw
------------·RWristYaw
------------·RHand
---------Legs
----------·LHipYawPitch ->ALFrameManager_0x9425af8_root_RobotModel_3eStrong0
----------·Left leg
------------·LHipRoll ->ALFrameManager_0x9425af8_root_RobotModel_3eStrong0
------------·LHipPitch ->ALFrameManager_0x9425af8_root_RobotModel_3eStrong0
------------· LKneePitch ->ALFrameManager_0x9425af8_root_RobotModel_3eStrong0
------------· LAnklePitch ->ALFrameManager_0x9425af8_root_RobotModel_3eStrong0
------------· LAnkleRoll ->ALFrameManager_0x9425af8_root_RobotModel_3eStrong0
----------·Right leg
------------· RHipRoll ->ALFrameManager_0x9425af8_root_RobotModel_3eStrong0
------------· RHipPitchl ->ALFrameManager_0x9425af8_root_RobotModel_3eStrong0
------------· RKneePitch ->ALFrameManager_0x9425af8_root_RobotModel_3eStrong0
------------· RAnklePitch ->ALFrameManager_0x9425af8_root_RobotModel_3eStrong0
------------· RAnkleRoll ->ALFrameManager_0x9425af8_root_RobotModel_3eStrong0

FIG.7c

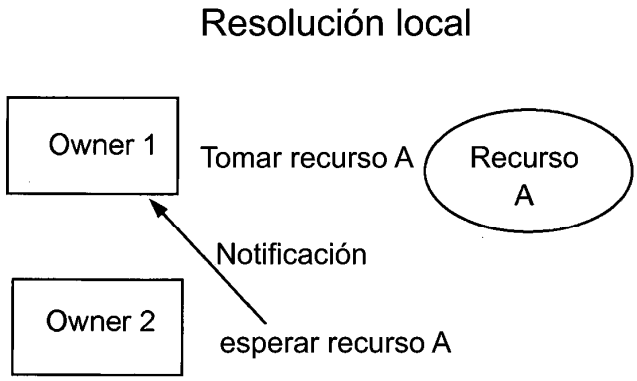

FIG.8a

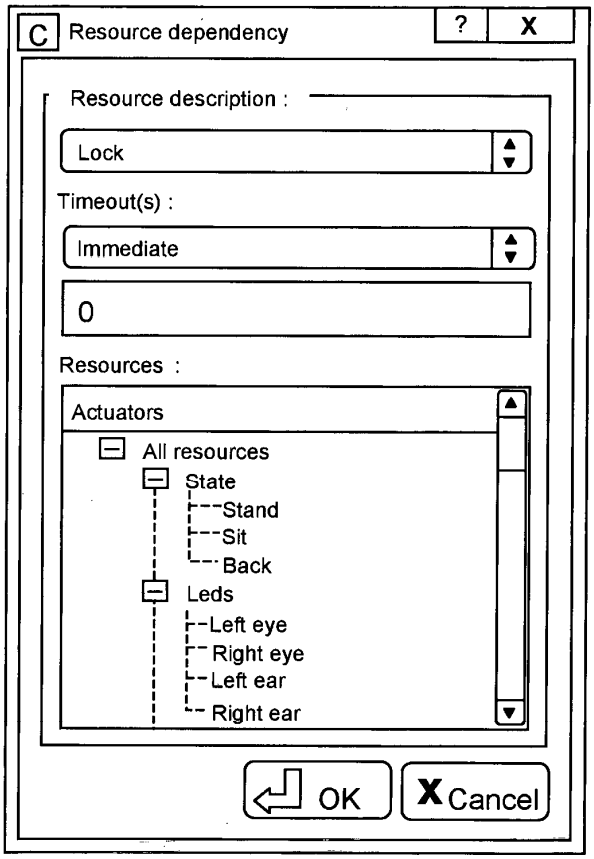

FIG.8b

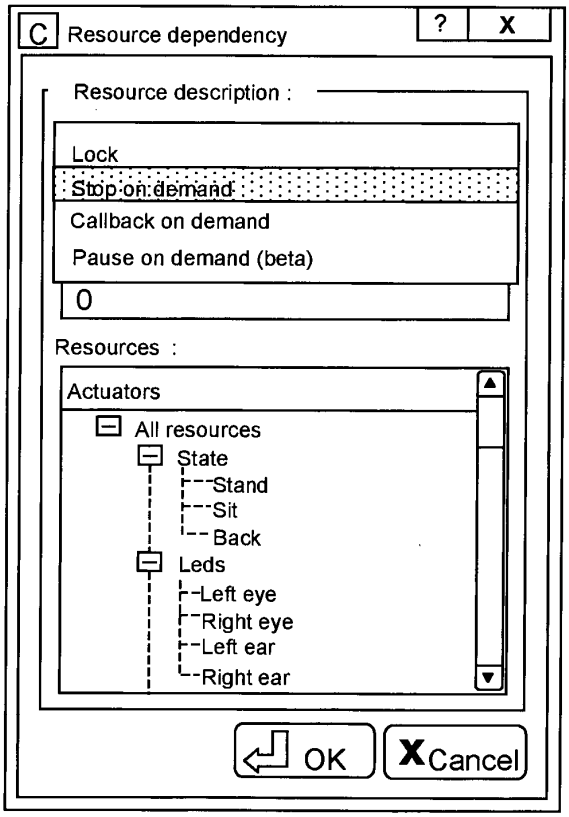

FIG.8c

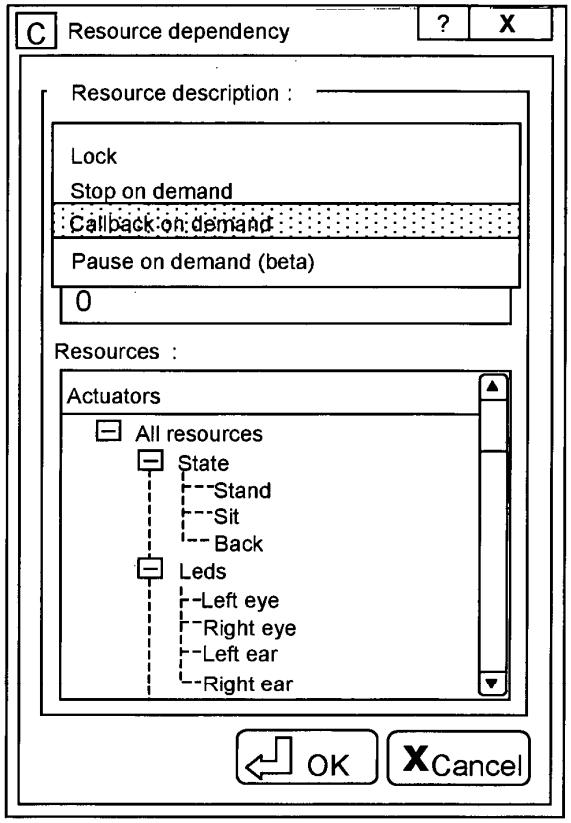
C
Resource dependency
?
X
Resource description :
Lock
Stop on demand
Callback on demand
Pause on demand (beta)
0
Resources :
Actuators
All resources
State
Stand
Sit
Back
Leds
Left eye
Right eye
Left ear
Right ear
OK
Cancel

FIG.8d

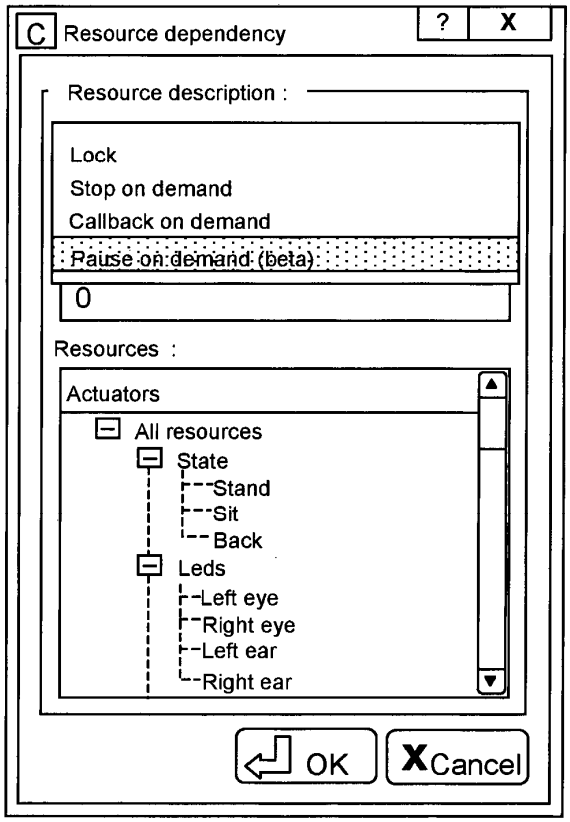
C
Resource dependency
?
X
Resource description :
Lock
Stop on demand
Callback on demand
Pause on demand (beta)
0
Resources :
Actuators
All resources
State
Stand
Sit
Back
Leds
Left eye
Right eye
Left ear
Right ear
OK
Cancel

FIG.8e

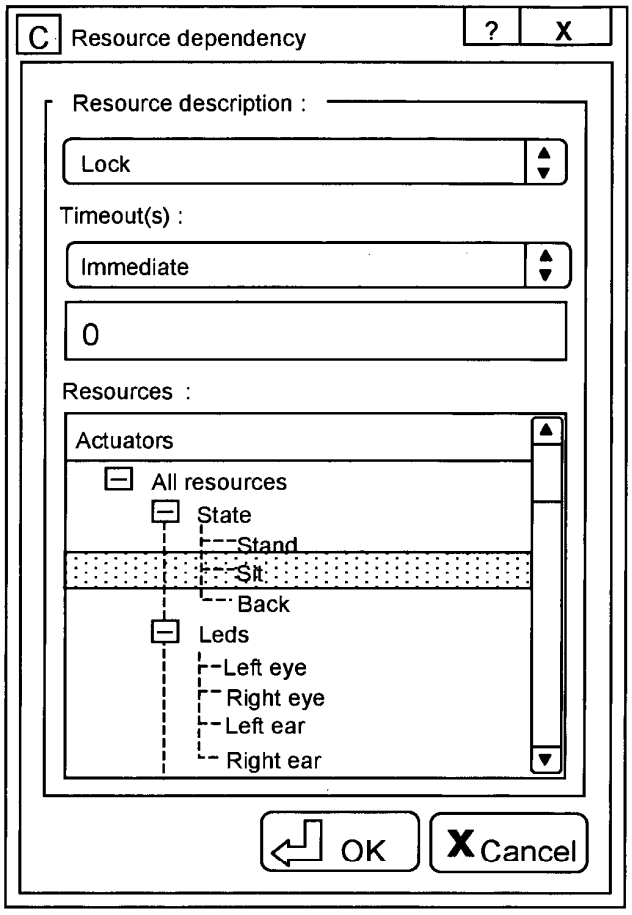
Resource dependency
?
X
Resource description :
Lock
Timeout(s) :
Immediate
0
Resources :
Actuators
All resources
State
Stand
Sit
Back
Leds
Left eye
Right eye
Left ear
Right ear
OK
Cancel

FIG.9

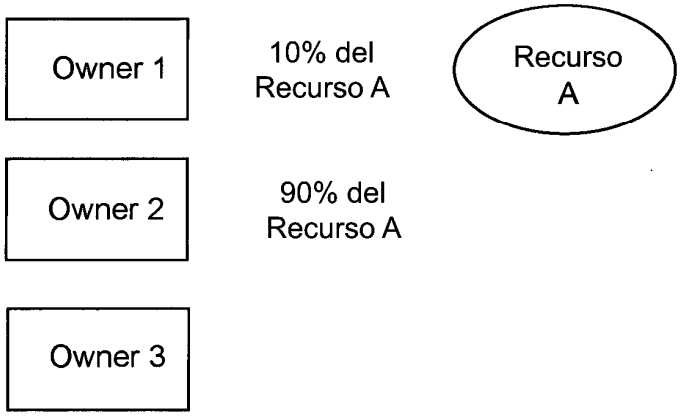
Owner 1
10% del Recurso A
Recurso A
Owner 2
90% del Recurso A
Owner 3

FIG.10