https://doi.org/10.17979/spudc.9788497498081.0891

DISEÑO DE UN MOTOR DE TAREAS PARA TERAPIAS DE NEUROREHABILITACIÓN ASISTIDAS POR ROBOTS

Luis D. Lledó, Santiago Ezquerro, Arturo Bertomeu-Motos,
José M. Catalán, Ramón Ñeco, José M. Sabater y Nicolás García-Aracil

Neuroingeniería Biomédica, Universidad Miguel Hernández de Elche,
{llledo, sezquerro, abertomeu, jose.catalan, ramon.neco, j.sabater, nicolas.garcia}@umh.es

Resumen

Este artículo presenta el proceso de planteamiento, análisis, diseño e implementación de un motor de tareas diseñado para facilitar la generación de actividades virtuales destinadas a terapias de neuro-rehabilitación de personas que han sufrido un accidente cerebrovascular, asistidos por dispositivos robóticos. Todo el desarrollo del sistema ha seguido la metodología RUP de diseño de software, utilizando herramientas de libre distribución. Por consiguiente, este un método genera tareas virtuales de manera rápida con costes minimos. Por ultimo, se han implementado tres ejemplos de tareas para comprobar el correcto funcionamiento de los componentes diseñados.

Palabras clave: Realidad Virtual, Programación modular, Robótica de rehabilitación, Ingeniería de software

1 INTRODUCCIÓN

Durante la época de los 90 surgió un nuevo concepto de modulación de software llamado motor de juegos [3] para el desarrollo de videojuegos o aplicaciones interactivas en tiempo real, definiendo rutinas de programación que permiten su diseño, creación y visualización. De esta manera, la arquitectura quedaba divida en módulos fundamentales, como puede ser el sistemas de renderizado visual, el sistema de simulación dinámica o la reproducción de sonidos [2]. La separación de los componentes facilita un paradigma orientado a la reutilización de código con el objetivo de generar aplicaciones interactivas con gráficos en tiempo real, de un mismo tipo sin tener que modificar el núcleo de programación.

Por otro lado, una terapia de rehabilitación está vinculada al tratamiento de personas con la necesidad de recobrar alguna condición o estado que han perdido a causa de alguna lesión u enfermedad, realizando algún tipo de ejercicio o tarea repetitiva dependiendo de la zona afectada. Esto quiere decir que para obtener una completa terapia de rehabilitación basada de entornos virtuales y asistidos por dispositivos robóticos se debería de disponer de un número de diferentes tareas virtuales que pueda realizar un paciente buscando una variedad para implicarlo un poco más dentro de la terapia evitando que procese un sentimiento de frustración. En este punto, cobra sentido la necesidad de utilizar motores de juegos para crear todo este tipo de juegos terapéuticos. La generación de una tarea implica la necesidad de crear un proyecto único con un fichero ejecutable que contenga la configuración de un entorno virtual y la programación de los objetivos que el paciente debe cumplir con los elementos seleccionables. El implementar un proyecto para cada tarea, añadiendo los componentes descritos en el motor de juego y establecer una escena manualmente con sus respectivos modelos físicos es un proceso lento y laborioso.

Por lo tanto, uno de los objetivos de esta tesis es desarrollar un sistema que permita generar ficheros ejecutables con tareas virtuales de manera rápida y adaptable destinadas a pacientes post-ictus durante la realización de terapias de rehabilitación asistidas por dispositivos robóticos. Este proceso permitirá el diseño de actividades que se ajusten a cada usuario, utilizando librerías de software libre. Estos ejercicios consistirán básicamente en trasladar objetos de un lugar a otro o colocar el efector final en ciertas posiciones. De esta manera la parte de programación queda relevada a un segundo plano necesitando la mayor parte del esfuerzo en el diseño de la parte visual. Esta arquitectura software permite optimizar el proceso de creación de tareas para la terapia de manera rápida con el menor coste y mínimos recursos posibles.

2 MATERIALES

2.1 Componentes software

En esta sección se definen los elementos software de libre distribución que se ha utilizado para plantear la arquitectura del motor de tareas propuesto. Para completar la parte visual, se ha utilizado OGRE3D como motor gráfico para renderizar escenas en 3D. Ogre3D facilita el diseño,

desarrollo e implementaci ón de aplicaciones de realidad virtual con entornos bastantes realistas, proporcionando interactividad y ejecución en tiempo real. es un motor de calculo de físicas desarrollado por NVIDIA para simular objetos con un alto grado de realismo. Proporciona algoritmos de detección de colisiones para calcular interacciones entre los elementos de una escena simulada en tiempo real y genera respuestas de dichos elementos virtuales ante la influencia de estas fuerzas, determinando su movimiento y dirección. Para añadir elementos sonoros que concuerden continuamente con eventos producidos durante el ciclo de vida de la tarea se ha utilizado OpenAL. Finalmente, para incorporar interfaces gráficas de usuario se utiliza CEGUI.

2.2 Sistema de neurorehabilitación

Las tareas virtuales obtenidas a partir del proceso de implementación propuesto en este trabajo están orientadas a su uso en un sistema de neurorrehabilitación formado por el dispositivo robótico llamado PUPArm [1]. Este sistema ha sido diseñado y desarrollado por el Grupo de Neuroingeniería Biomédica en la Universidad Miguel Hernández de Elche como un robot de rehabilitación para pacientes con accidente cerebrovascular. En la Figura 1 se puede observar el sistema de neurorrehabilitación.

Figura 1: Sistema de neurorehabilitación basado en el robot PUPArm

El mecanismo robótico consiste en cuatro barras metálicas conectadas como un paralelogramo [4]. Esta estructura proporciona un manipulador planar con movimiento en dos dimensiones. Por consiguiente, el sistema sólo permite el movimiento horizontal de la extremidad superior de los sujetos. Por otra parte, el subsistema de visualización se compone de un monitor de computador con un software desarrollado llamado REVIRE que se encarga de ejecutar las actividades generadas, en coordinación con los movimientos del robot.

3 METODOLOGÍA

Esta seccion se detalla el proceso de planificación, análisis, diseño e implementación del motor de tareas y las dependencias entre todos los componentes. La metodología utilizada para crear este sistema está basada en un desarrollo de programación orientado a objetos [6], lo que permite beneficiarse de las ventajas que aporta este paradigma como la reusabilidad, la modificabilidad, fiabilidad junto con la facilidad y sencillez de mantenimiento. En concreto se ha utilizado la metodología RUP [5] como proceso de de desarrollo de software. Por otro lado, se ha utilizado UML [8] para documentar y describir aspectos o métodos del sistema software. Terapias

3.1 REQUERIMIENTOS

En la fase de inicio se propone definir los objetivos y analizar de manera muy general la arquitectura de software que se quiere producir, identificando los requisitos funcionales y no funcionales para comprobar los requerimientos que tiene que alcanzar el sistema.

3.1.1 Visión general

De manera general, el sistema propuesto tiene por finalidad el cargar unos ficheros de recursos multimedia y organizarlos para generar un fichero ejecutable con la tarea virtual. Esto quiere decir, que el motor de tareas es un plantilla con los componentes interconectados de manera que cada vez que se ejecuta se encarga de decodificar los recursos multimedia y crear una ventana donde muestra todo el contenido virtual con un comportamiento físico y gestión de sonidos. En la Figura 2 se presenta una visión general del objetivo de este apartado y el contexto donde aplica el motor de tareas y cuando se realiza la creación del contenido multimedia.

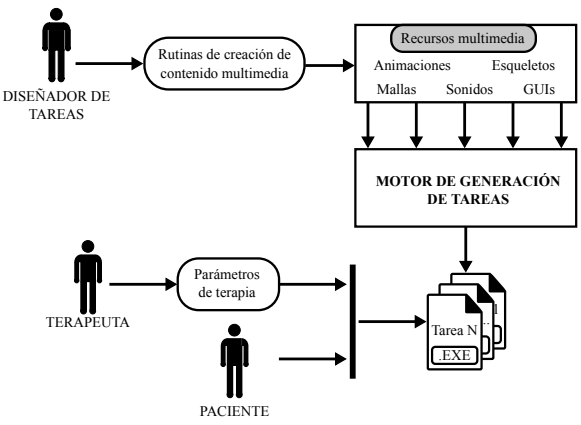

Figura 2: Esquema general del sistema propuesto

Como complemento al desarrollo de un sistema de generación de tareas virtuales se establece un proceso de producción de contenido multimedia para combinarlo con este motor de tareas. Con respecto, a los objetivos de las tareas se debe tener en cuenta las posibilidades que ofrece dispositivo de control, que en este caso son unos dispositivos robóticos. Debido a la estructura mecánica la única interacción que puede realizar el paciente es el movimiento del efector final de punto a otro para realizar las trayectorias. Bajo este fundamento, los objetivos dentro de las tareas se traducen en un desplazamiento de un objeto controlable buscando un objetivo alcanzable.

3.1.2 Requisitos funcionales

Los requisitos funcionales describen la funcionalidad que se espera del sistema software. Las tareas generadas contarán con la siguiente lista de requisitos funcionales: movimiento de un elemento controlable, salir de la tarea en cualquier momento, visualizar una interfaz gráfica con información y establecer objetivos de alcance de zonas o selección de objetos.

3.2 ANÁLISIS

Una vez establecidos los objetivos iniciales con una visión general del formato de sistema, se realiza la fase de análisis para identificar las necesidades que debe cumplir el sistema a partir de los requisitos obtenidos en la sección anterior. En primer lugar, se define el modelo de dominio que contiene los tipos de objetos conceptuales más importantes dentro del contexto del sistema. Después, se realiza una explicación de los casos de uso.

3.2.1 Modelo del dominio

El modelo del dominio representa los conceptos clave dentro del dominio del problema. La información que presenta se puede expresar mediante sentencias, pero de manera visual permite una mejor compresión de los distintos elementos y sus relaciones naturales. Es una representación de los objetos conceptuales mas importantes del sistema y necesarios para entender de forma abstracta el dominio del problema aportando. Se puede apreciar en la Figura 3 el diagrama de dominio de este sistema.

3.2.2 Casos de uso

En este tipo de análisis se definen cuales son los posibles usuarios o actores del sistema y los distintos objetivos que pueden cumplir interactuando en forma de secuencia de acciones. Describe la secuencia de interacciones entre actores y el sistema

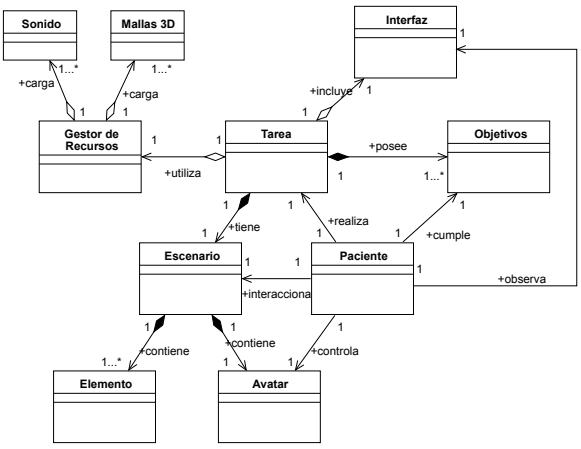

Figura 3: Diagrama conceptual de dominio

para llevar a cabo algún procedimiento, definiendo relaciones de comportamiento. También se ha utilizado la notación UML para realizar el modelado de todas las posibles situaciones que se pueden dar entre los usuarios y el sistema.

En este caso se pueden identificar tres posibles perfiles de usuario: Diseñador de Tareas, Terapeuta y Paciente. El actor principal del sistema será el Paciente siempre supervisado por el Terapeuta rehabilitador. Sin embargo, como también existe proceso de creación de contenidos multimedia se puede definir el rol de Diseñador de Tareas. Por otro lado, los casos de uso del sistema se pueden clasificar en dos grupos: casos de uso del motor de tareas y los casos de uso en el diseño de recursos.

En primer lugar se exponen los casos de uso que puede realizar el Paciente y el Terapeuta en la parte más importante del sistema, que es el motor de tareas. Este subsistema software genera una tarea diferente en función de los elementos virtuales cargados. En consecuencia, todas las tareas generadas tienen el mismo formato de diagrama de casos de uso. En la Figura 4 se muestra el diagrama de casos de uso de una tarea general implementada con el motor de tareas.

El otro grupo de casos de uso que se pueden extraer del sistema son los correspondientes al diseño del contenido multimedia utilizado en el motor de tareas para simular un escenario. Esta parte del sistema se refiere a las rutinas de programación y diseño que debe seguir el Diseñador de Tareas para generar los recursos que se visualizarán durante la ejecución de la tarea.

3.2.3 Modelado del comportamiento

En este apartado se detalla el modelo de comportamiento del sistema diferenciando los dos grupos

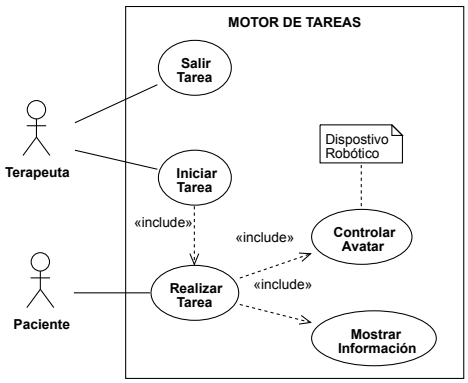

Figura 4: Diagrama de casos de uso general

de casos de uso. Para el caso del motor de tareas se presenta un diagrama de secuencia que se encarga de identificar como se comunican internamente las clases principales para la realización de los casos de uso. Normalmente, se aplica un diagrama de secuencia por cada caso de uso, pero en este caso se pueden representar todos los casos de uso en un diagrama de secuencia.

En la Figura 5 se representa un diagrama de secuencia general del motor de tareas que define los casos de uso: Iniciar Tarea, Salir Tarea, Realizar Tarea, Controlar Avatar y Mostrar Información. Muestra de manera global las clases iniciales y la interaccion entre actores y el sistema.

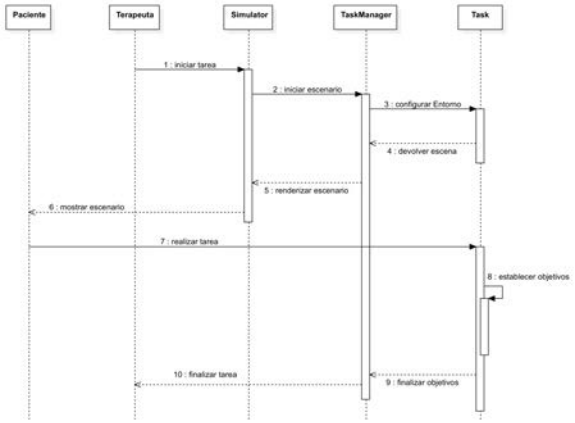

Figura 5: Diagrama de secuencia general

Por otro lado, el proceso de creación de contenido multimedia puede representarse como un diagrama de actividad debido a que no consiste en un subsistema de programación orienta a objetos, sino en una rutina de procedimientos que debe seguir el Diseñador de Tareas para poder generar recursos de una manera correcta que complementen al motor de tareas. La Figura 6 representa el protocolo de creación de contenido de esta parte del sistema en un diagrama de actividades separado en cuatro subactividades dependiendo del tipo de recurso.

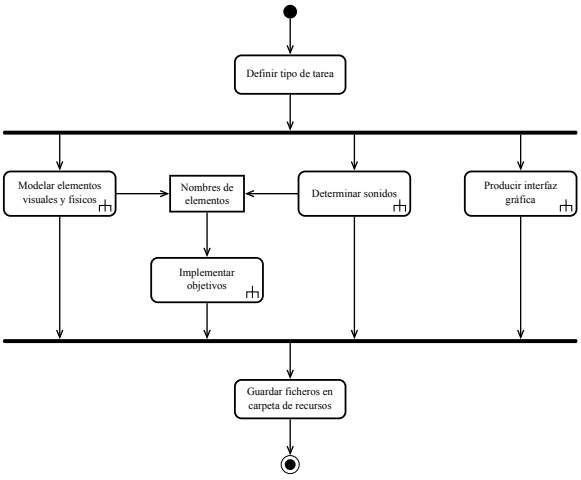

Figura 6: Diagrama de actividades para recursos

3.3 DISEÑO

En esta sección se va a presentar el diseño arquitectural del sistema para crear una estructura software que soporte todos los requisitos, empleando también la notación UML. Esta fase se va a centrar el diseño del motor de tareas, debido a que es la única parte del sistema que posee una arquitectura software.

3.3.1 Arquitectura

En primer lugar, se define un modelo de arquitectura general para obtener un diseño de alto nivel que permitirá organizar el sistema en una serie de módulos y relaciones existentes entre ellos. De este modo, se puede tener una estructuración de las futuras clases que compondrán del motor de tareas. En la Figura 7 se muestra la arquitectura general del motor de tareas diferenciando los componentes principales que servirán de referencia para crear las clases del proyecto. La parte de la creación de contenido no necesita un diseño previo, ya que consiste en utilizar rutinas de programación con herramientas libres.

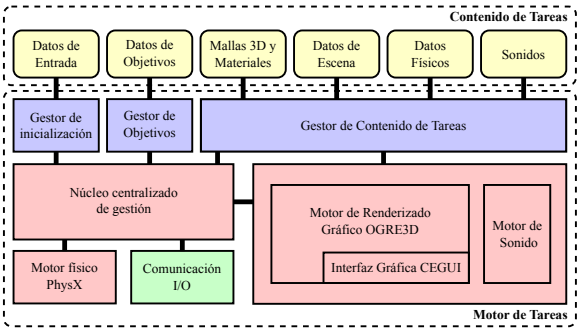

Figura 7: Arquitectura modular del sistema

3.3.2 Diagramas de clases

En este apartado se definen las clases que componen cada uno de los módulos que forman la arquitectura del motor de tareas para presentar los diagramas con una mayor claridad. Con cada diagrama se aportan las clases mostrando los principales métodos y atributos. El primer diagrama mostrado en la Figura 8 define las clases encargadas de inicializar los motores de renderizado gráfico, físico, de sonido y el de renderizado de interfaces dando soporte al resto de clases del proyecto.

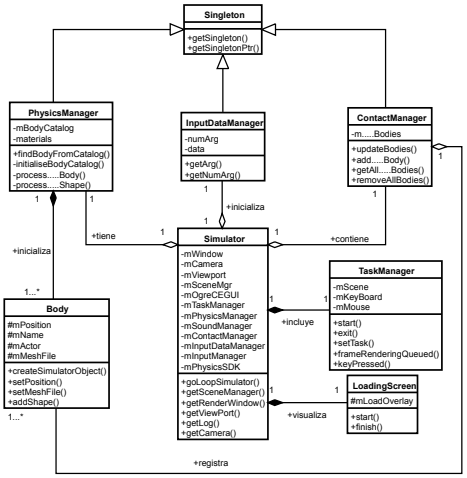

Figura 8: Diagrama de clases de inicialización

En el diagrama de la Figura 9 se puede observar las clases relacionadas con el control del escenario junto con su organizacion y el avatar de control. También se encargan de gestionar la lógica de juego de la tarea para actualizar el estado de los distintos elementos del escenario y gestionar el tratamiento de los eventos externos como la pulsación del teclado o el administrar los datos de entrada del sistema robótico.

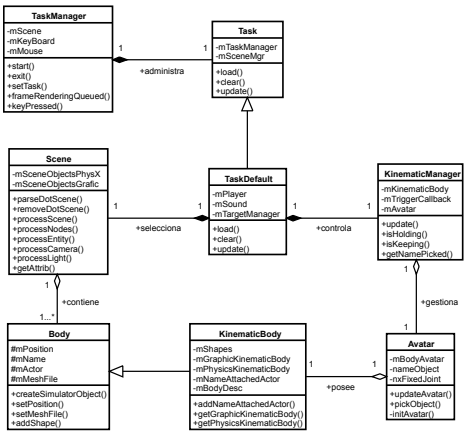

Figura 9: Diagrama de clases de control

El último diagrama (Figura 10) presenta las clases que se ocupan de administrar toda la gestión del cumplimiento de los objetivos, la lectura de ficheros de objetivos y el control del estado de los objetivos. Para completar el diagrama se agrega una comunicación por socket entre el gestor del control cinemático y el dispositivo robótico mediante UDP.

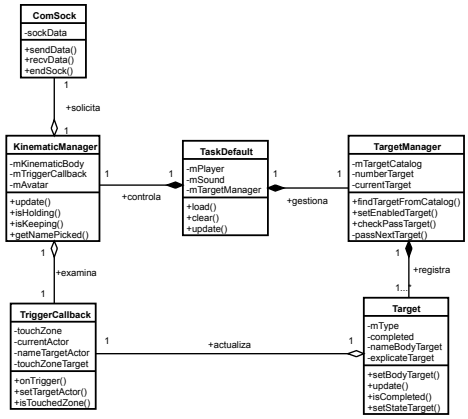

Figura 10: Diagrama de clases de actualización

3.4 IMPLEMENTACIÓN

El proceso de implementación conlleva la elaboración de la arquitectura del sistema en términos de componentes como pueden ser ficheros de código fuente, scripts, ejecutables y similares. En este apartado se van a comentar los puntos y funcionalidades más importantes del proceso de implementación del sistema de manera detallada, evitando la presentación de código.

En este trabajo surgido dos lineas de desarrollo para implementar tareas. Por lo tanto, este apartado se va a dividir en dos secciones dónde se describen los detalles de implementación más significativos de las diferentes lineas: en la primera se detallan instrucciones para la creación de contenido multimedia y la otra trata sobre la generación de ficheros con rutinas de programación para el diseño del motor de tareas.

3.4.1 Creación de contenido multimedia

El contenido que utiliza el motor de tareas incluye mallas, materiales, texturas, animaciones, scripts de datos o sonidos. Todos los modelos 3D del escenario virtual se generan con la herramienta de modelado Blender y junto con el exportador a OGRE3D, proporciona los ficheros necesarios para cargarlos en el motor de renderizado. Además de los objetos, se pueden definir los materiales y texturas que simulan las características superficiales de los objetos. A partir de las mallas visuales, se pueden obtener mallas para la detección de colisiones, también con Blender. En diferentes

scripts se reúnen todas las formas colisionables y las posiciones-orientaciones de todos los elementos visuales. Uno de los elementos clave para dar una funcionalidad de rehabilitación a las tareas es el fichero de datos de objetivos para proporcionar funcionalidad a la parte visual y fisica de la tarea. Por lo tanto, se debe generar otro fichero XML que recoge todos los objetivos que se pueden cumplir durante la relación de una tarea. Además de todos los recursos comentados anteriormente, existe la posibilidad de incorporar contenido adicional en forma de sonidos para aportar una realimentación sonora de cuando se completan los objetivos o incluso una interfaz gráfica encargada de mostrar información adicional que necesita el usuario para completar algún objetivo.

3.4.2 Implementación del motor de tareas

La segunda parte de la sección de implementación del sistema consiste en el desarrollo del motor de tareas. Durante la evolución de este apartado se detallan los aspectos más importantes a nivel de programación de las clases mostrando los diagramas de flujo. El núcleo del motor de tareas está construido en torno a la clase Simulator y StateManager. La clase Game se encarga de declarar e inicializar los principales elementos del programa como. El primer paso a realizar es inicializar todos los componentes que formarán el motor de tareas. De esta acción se encarga la clase Simulator. Otra parte del núcleo del programa es TaskManager, la cual se encarga de gestionar todo el proceso de frames, iniciar el ciclo de simulación, gestionar el estado de la tarea y registrar los eventos de teclado.

En un escenario cada elemento del escenario de la tarea tiene una forma física que actúan dentro de la simulación física y un nodo con una malla 3D para poder visualizar las posibles colisiones generadas. Teniendo en cuenta esto, se ha implementado la clase Body para encapsular los dos elementos y actualiza en pantalla los objetos visuales en función de la simulación de físicas. Por consiguiente, desde la clase TaskDefault se crea una instancia de la clase Scene para completar esta etapa. Scene es la clase encargada de tratar el fichero XML con la organización de la escena y las mallas necesarias, y almacenar un registro de los elementos creados. En la Figura 12 se muestra el diagrama de flujo de la creación de la escena de la tarea.

La clase TaskDefault se encarga de modelar el estado de la simulación de la tarea. Da la orden de cargar el escenario y la interfaz así como la actualización de los objetos que forman parte de los objetos de los datos de información o gestionar el cumplimiento de los objetivos para enviar información sobre los resultados de rendimiento del paciente. Durante cada ciclo de simulacion se realiza comprobación del cumplimiento del objetivo a partir de las clases TargetManager, Target y TriggerCallback. Si existe interfaz de información se actualiza el contenido. KinematicManager también posee la funcionalidad de actualizar el avatar de control y crear una comunicación vía socket entre el sistema de gestión el dispositivo robótico y el motor de tareas, para intercambiar datos de posición. La Figura 12 muestra el diagrama de flujo general de la actualización del sistema.

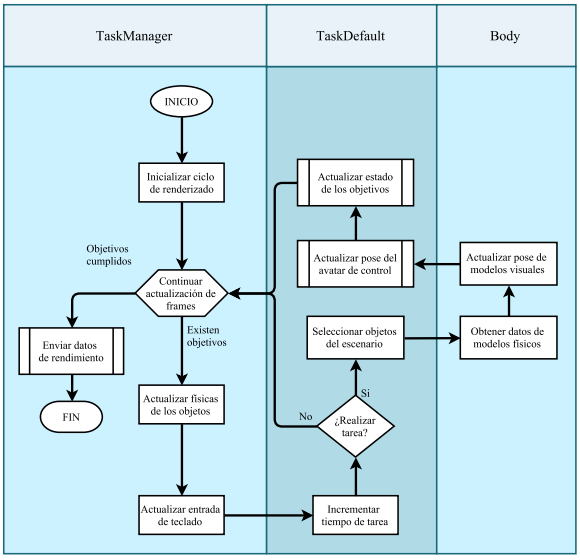

Figura 12: Diagrama de flujo de actualización

4 PRUEBAS DE INTEGRACIÓN

En esta sección se verifica el correcto funcionamiento de los componentes diseñados e implementados para comprobar si son capaces de interactuar entre ellos a partir de las clases software, funcionando en el dispositivo robótico. De esta manera para comprobar la eficacia del sistema software se implementen algunos ejemplos de tareas utilizando la arquitectura. Se desarrollan múltiples entornos de tareas aumentando la cantidad de elementos que forman parte de la escena y la cantidad de objetivos a realizar.

4.1 Ruleta

Esta tarea simula un fábrica de cajas con una vista en prespectiva donde la escena converge al punto central de la pantalla. El escenario gráfico estático consiste en 8 plataformas y un depósito central. Las 8 plataformas están colocadas uniformemente alrededor del depósito, siempre manteniendo la

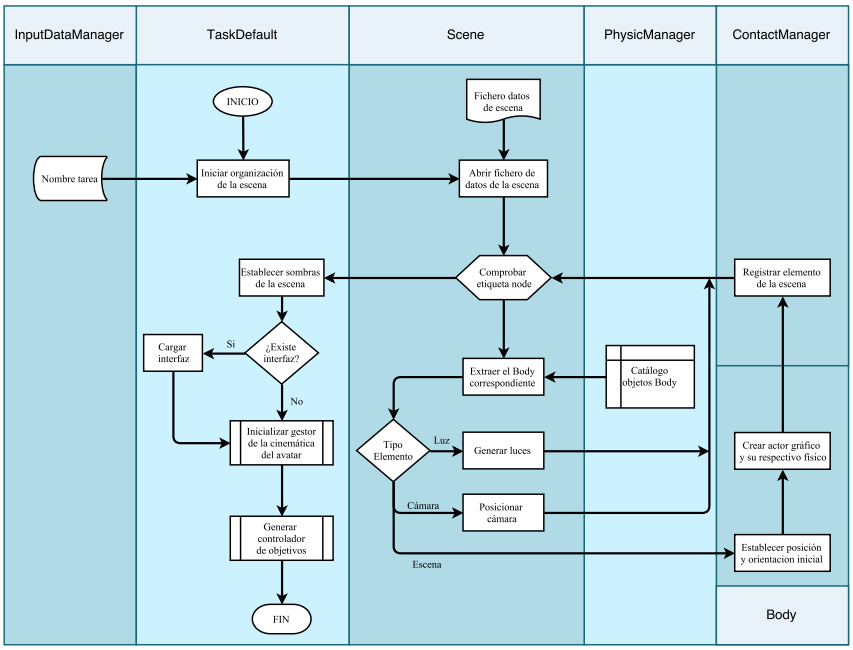

Figura 11: Diagrama de flujo de inicialización de la tarea

misma distancia. Todos los elementos estáticos son soportes para el elementos dinamico representado por una caja, la cual interactua con la herramienta virtual cinemática definida por una llave inglesa. También se ha añadido unos elementos decorativos para simular una especie de fábrica. En la Figura 13 se puede observar una captura de pantalla de la tarea. La estructura del escenario permite realizar dos objetivos principales: recoger la caja y el otro sería soltarla dentro del depósito.

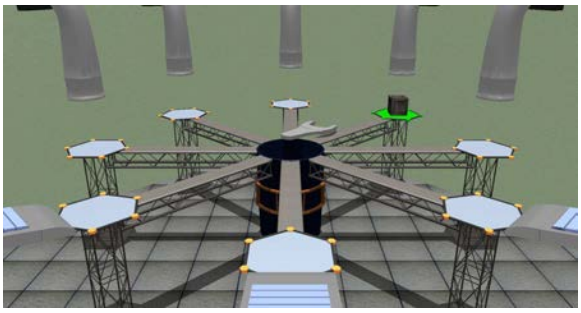

Figura 13: Entorno virtual de una fábrica

4.2 Transporte de elementos

En esta prueba del sistema se ha implementado una tarea con más elementos de interacción para completar los objetivos que en las pruebas anteriores. La tarea consiste en simular el trabajo de un camarero encargado de proporcionar diferentes tipos de bebidas en función del pedido de los cliente. El usuario debe recoger un vaso, rellenarlo con un líquido y depositarlo encima de una bandeja. Por tanto, el escenario gráfico está formada por una mesa, dispensador de bebidas y un dispensador de vasos. Como elementos decorativos se ha añadido una pila de objetos encima de una toalla de cocina y 3 posa-vasos de diferentes colores para indicar el tipo de bebida. El avatar de control es un brazo virtual, el cual interactuará con un vaso con comportamiento dinámico. En esta tarea también se ha añadido un brazo extra con una bandeja a modo de objetivo final a la hora de depositar el vaso. En la Figura 13 se visualiza el escenario gráfico de la tarea implementada.

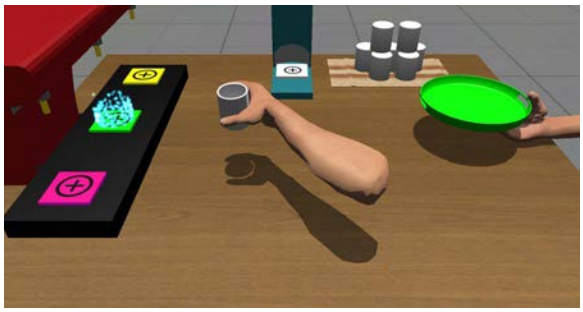

Figura 14: Entorno virtual de selección de bebidas

4.3 Tarea de la vida diaria: Cocina

Con esta prueba se pretende comprobar todas las funcionalidades añadidas durante el proceso de implementación del sistema generando un entorno de AVD [7] basado en una cocina para realizar una tarea sencilla como es preparar un huevo frito. El escenario contiene los siguientes elementos: un huevo, una sartén, una aceitera y un deposito de basura. El mobiliario de la cocina está represen-

tado por una mesa con cajones, una estantería, una vitrocerámica y una tabla para cortar. En esta actividad también se ha utilizado un brazo virtual con la mano abierta. Además, incorporá una interfaz gráfica formada por una caja de texto para informar al usuario del objetivo a cumplir en cada momento y una barra de progreso para indicar el tiempo utilizado durante la realización de la receta. La Figura 15 muestra el escenario virtual completo.

Figura 15: Entorno virtual de una cocina

4.4 CONCLUSIONES

En este trabajo se ha presentado el proceso de captura de requisitos, análisis, diseño, implementación y pruebas de integración para implementar tareas virtuales destinadas a terapias de neuro-rehabilitación del miembro superior asistidas por dispositivos robóticos. El proceso de implementación del sistema ha constado de dos partes: la parte de la producción de contenido multimedia y la generación del código capaz de realizar simulaciones basadas en principios físicos. De esta manera, el sistema se encarga de proporcionar objetivos y movimiento a un escenario virtual, siendo posible crear múltiples tareas sin modificar una sola linea de código. Sin embargo, la configuración u objetivos de algún tipo de tarea puede necesitar incorporar algún nuevo bloque de código para controlar algún tipo de objetivo o animaciones. A pesar de ello, una vez establecido el nuevo bloque, ya se podrían diseñar más tipos de tareas en función de los diversos objetivos que pueden surgir durante el diseño de los escenario y los elementos. Simplemente modificando la clase Target para añadir las nuevas funcionalidades que han surgido durante el desarrollo de las pruebas.

Como resultado se ha obtenido un método de implementación de tareas de manera rápida y de bajo coste, lo que permita minimizar los gastos ocasionados durante el proceso de reducción. Genera entornos con el mismo tipo de objetivos añadiendo simplemente un fichero con etiquetas, las cuales son leídas por los módulos de lectura del sistema para automatizar toda la gestion de objetivos alcanzables. Permite la creación de tareas adaptadas a cada paciente comprobando cuales pueden ser mas beneficiosos para su neuro-rehabilitación motora. Cabe destacar que este sistema ya está funcionando en un hospital de en un hospital de atención a pacientes crónicos y de larga estancia.

Agradecimientos

Este trabajo ha sido financiado por la Comisión Europea a través del proyecto HomeRehab: Echord++ (GA 601116) y por el Ministerio de Economía y Competitividad a través del proyecto DPI2015-70415-C2-2-R.

Referencias

[1] Badesa, F. J., Llinares, A., Morales, R., Garcia-Aracil, N., Sabater, J. M., y Perez-Vidal, C., (2014), "Pneumatic planar rehabilitation robot for post-stroke patients," *Biomedical Engineering: Applications, Basis and Communications*, vol. 26, no. 02.

[2] Fernandez, D. V., Angelina, C. M., *et al.*, *Desarrollo de Videojuegos: Arquitectura del Motor de Videojuegos*. Cursos en Español, (2011).

[3] Gregory, J., *Game engine architecture*. CRC Press, (2009).

[4] Krebs, H. I., Hogan, N., Aisen, M. L., y Volpe, B. T., (1998), "Robot-aided neurorehabilitation," *Rehabilitation Engineering, IEEE Transactions on*, vol. 6, no. 1, pp. 75–87.

[5] Kruchten, P., *The rational unified process: an introduction*. Addison-Wesley Professional, (2004).

[6] Larman, C., "Uml y patrones. una introducción al análisis y diseño orientado a objetos y al proceso unificado. aragón df," (2003).

[7] Mehrholz, J., Platz, T., Kugler, J., y Pohl, M., (2008), "Electromechanical and robot-assisted arm training for improving arm function and activities of daily living after stroke," *Cochrane Database Syst Rev*, vol. 4, no. 4.

[8] Rumbaugh, J., Jacobson, I., y Booch, G., *The Unified Modeling Language Reference Manual*. Pearson Higher Education, (2004).