Universidad de las Ciencias Informáticas

Título: Creador de tareas de comportamiento para los laboratorios virtuales

Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas

Autor: Nallía Iris Ramírez Castro

Tutor: MSc. Leoder Alemañy Socarrás

Co-Tutor: MSc. Orlay García Ducongé

La Habana, Mayo 2012
Año 54 de la Revolución

FRASE

Soy de los que piensan que la ciencia tiene una gran belleza. Un científico (…) no es sólo un técnico: es también un niño colocado ante fenómenos naturales que le impresionan como un cuento de hadas.

Marie Curie

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de este trabajo y autorizo a la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio. Para que así conste firmo la presente a los ___ días del mes ____ del año _____.

_____
Nallía Iris Ramírez Castro
Autor

_____
M.Sc. Leoder Alemañy Socarrás
Tutor

_____
M.Sc. Orlay García Ducongé
Co-Tutor

DATOS DE CONTACTO

Tutor: MSc. Leoder Alemañy Socarrás

Edad: 27 años

Ciudadanía: cubano

Institución: Universidad de las Ciencias Informáticas

Título: Ingeniero en Ciencias Informáticas

Categoría Docente: Profesor Instructor

E-mail: lalemany@uci.cu

Graduado en la Universidad de las Ciencias Informáticas

Tutor: MSc. Orlay García Ducongé

Edad: 27 años

Ciudadanía: cubano

Institución: Universidad de las Ciencias Informáticas

Título: Ingeniero en Ciencias Informáticas

Categoría Docente: Profesor Instructor

E-mail: oducunge@uci.cu

Graduado en la Universidad de las Ciencias Informáticas

AGRADECIMIENTOS

A mis padres y mi familia:

Por apoyarme siempre en todas las decisiones

A José Andrés:

Por compartir su vida conmigo y socorrerme en todo

A mis amigos:

Por darme aliento frente a las vicisitudes

A mis tutores:

Por dedicarme tiempo y ayudarme con la tesis

A mis compañeras del apartamento:

Por soportarme todos los días y reír conmigo a pesar de todo

A todo aquel que aportó su granito de arena en esta travesía

DEDICATORIA

A mis padres:

Son mi sostén y mi sosiego

A mis abuelos:

Por cuidarme donde quiera que estén

A José Andrés, Arabella y Leyannis:

Por quererme como soy y no querer cambiarme

A mis vecinos Melba, Rigo, Alisnaida y Dailer:

Son mi otra familia y puedo contar con ustedes para todo

RESUMEN

Las Tecnologías de la Información y las Comunicaciones han tenido un gran impacto en la sociedad actual, destacándose los beneficios alcanzados en la educación, gracias a disímiles tecnologías y software como los laboratorios virtuales.

La presente investigación tiene como objetivo la confección de un módulo que permita crear tareas de comportamiento, de forma visual, a los objetos de las escenas de los laboratorios virtuales. Para su realización, se utilizará *Extreme Programming* como metodología de software y las siguientes herramientas: *Visual Paradigm for UML*, *Qt Creator* con lenguaje *C++* , motor gráfico OGRE3D y XML como lenguaje de marcas.

Con dicho sistema, el desarrollador podrá crear las tareas de forma visual y los usuarios podrán ejecutar las tareas antes creadas. El módulo beneficiará los laboratorios virtuales que sean desarrollados en el CEDIN, pues automatizará el proceso de asignar comportamientos a los objetos e incrementará el dinamismo de las escenas.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

Las Tecnologías de la Información y las Comunicaciones (TIC) han provocado un gran impacto en el ámbito social, económico, político y cultural del mundo actual, destacándose los innumerables beneficios alcanzados en la educación. Entre los software que más han aportado al ámbito educativo, se encuentra el laboratorio virtual, que es un sistema computacional que simula el ambiente de un laboratorio tradicional y constituye una alternativa a la carencia de algunos materiales educativos en las instituciones.

El Centro de Informática Industrial (CEDIN), presente en la Universidad de las Ciencias Informáticas (UCI), potencia el desarrollo de laboratorios virtuales y ya cuenta con tres productos liberados para Venezuela. Con la perspectiva de futuros proyectos, el CEDIN desea eliminar las deficiencias que se manifestaron en los anteriores laboratorios.

El comportamiento de los objetos en los anteriores laboratorios virtuales que se desarrollaron, se realizaba a código. Este proceso era complejo para los programadores, pues había que realizarlo de forma manual, a través de código fuente, e imaginándose el efecto visual que los cambios provocaban sobre los objetos. Cada cambio en el comportamiento de un objeto, provocaba la re-compilación del código fuente y el reinicio de la aplicación. Producto a todo lo descrito anteriormente, solo se asignaron comportamientos a los objetos que formaban parte de la práctica de laboratorio, dejando fuera a otros elementos de las escenas, como puertas y gavetas.

Entonces, surge el siguiente **problema científico**: ¿cómo asignar comportamientos, de forma visual, a los objetos en las escenas de los laboratorios virtuales?

A partir de la situación planteada, se define como **objeto de estudio**: las tareas de los objetos en gráficos por computadoras; y como **campo de acción:** las tareas de comportamiento de los objetos en los laboratorios virtuales.

Para dar solución a la problemática, se formula el siguiente **objetivo**: desarrollar un módulo que permita la creación de tareas de comportamiento, de forma visual, a los objetos en las escenas de los laboratorios virtuales.

Con la realización de las siguientes **tareas de la investigación**, se pretende materializar el objetivo antes planteado:

- Elaboración del marco teórico para conocer acerca de las tareas que pueden realizar los objetos en los gráficos por computadoras.
- Análisis de la arquitectura de los laboratorios virtuales para acoplar correctamente el sistema desarrollado.
- Selección de la metodología y las herramientas a utilizar para el desarrollo del software.
- Diseño de la aplicación.
- Implementación de la solución propuesta.
- Ejecución de pruebas para comprobar el cumplimiento de todas las funcionalidades requeridas por el cliente.
- Creación del manual de usuario, donde se explicará cómo trabajar con la aplicación.

El presente trabajo hará uso de los siguientes **métodos de investigación:**

Métodos Teóricos:

- **Analítico-Sintético:** se emplea para buscar información acerca del problema propuesto y para extraer los elementos que están relacionados con el objeto de estudio.

- **Modelación:** se usa para representar el conocimiento adquirido durante la investigación y en el diseño de la aplicación.

Métodos Empíricos:

- **Consulta de las fuentes de información:** se emplea en la selección de la información importante y en la elaboración del marco teórico.

- **Consulta de especialistas:** para que las personas calificadas en el tema evalúen la aplicabilidad y utilidad del software desarrollado.

- **Pruebas:** se utiliza para comprobar si la aplicación funciona correctamente.

A continuación se presenta una breve descripción de los capítulos que conforman la investigación:

Capítulo 1: Fundamentación Teórica.

Explica los conceptos importantes y menciona algunas herramientas y metodologías que pueden emplearse en el desarrollo de la aplicación.

Capítulo 2: Descripción de la propuesta de solución.

Expone la metodología y herramientas seleccionadas para el desarrollo y detalla los requisitos funcionales y no funcionales de la aplicación, además de la descripción de las clases, mediante los artefactos definidos por la metodología escogida.

Capítulo 3: Construcción de la propuesta de solución.

Plantea las tareas de ingeniería necesarias para implementar los requisitos funcionales y comprueba que el sistema cuente con todas las funcionalidades requeridas por el cliente.

Anexos: Muestra imágenes de la aplicación desarrollada.

Glosario de Términos: Contiene los términos importantes de la investigación, con el objetivo de facilitar la comprensión del lenguaje utilizado.

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA

Introducción

El objetivo del presente capítulo es abordar los elementos relacionados con la investigación. Primero, se hace referencia a qué son los gráficos por computadoras y qué es la Realidad Virtual (RV). Seguidamente, se mencionan algunos conceptos de laboratorio virtual y se exponen varios de los laboratorios desarrollados en Cuba y en la UCI. Además, se explica qué papel juegan las transformaciones geométricas en la animación 3D y cómo las tareas de comportamiento solucionan la problemática de realizar dichas transformaciones desde la programación. Por último, se muestra un estudio sobre las diferentes herramientas y metodologías que son necesarias para desarrollar el software.

1.1 Gráficos por computadoras

Desde la antigüedad, el hombre ha necesitado representar el entorno y el comportamiento humano en las labores que realiza, sea en el arte, la literatura, etc. El surgimiento de las computadoras dio lugar a nuevas técnicas, como los gráficos por computadoras, con las cuales se obtuvo una nueva oportunidad para plasmar ese realismo.

La gráfica por computadoras no es más que “la síntesis pictórica de objetos reales e imaginarios a partir de modelos representados en la computadora” (1). Cuando se habla de síntesis pictórica, se refiere a la representación de un objeto mediante una imagen. Un ejemplo se muestra en la siguiente imagen (ver Figura 1), donde se observa un ventilador virtual que se parece bastante al real.

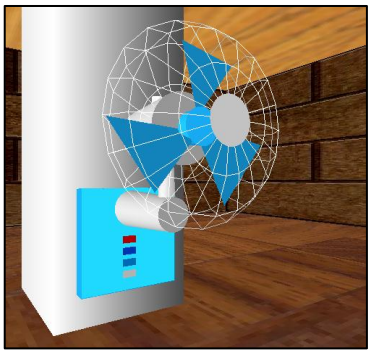


Figura 1. Ventilador virtual.

Actualmente, se aplican en varias esferas de la vida diaria, siendo mayormente utilizados en la simulación del entorno gracias a su capacidad para generar, integrar y cambiar la información visual y espacial del mundo virtual, lo que introduce un nuevo término: la RV.

1.2 Realidad Virtual

Existe un gran número de conceptos asociados a la RV, a continuación se mencionan dos de ellos:

Según la Academia de la Lengua Española, es la “representación de escenas o imágenes de objetos producidas por un sistema informático, que da la sensación de su existencia real” (2).

Jerry Isdale la define como “un camino que tienen los humanos para visualizar, manipular e interactuar con computadoras y con información extremadamente compleja" (3).

Un ejemplo del uso de la RV se observa en las siguientes imágenes, donde a partir de una práctica real (ver Figura 2) para armar una computadora, se diseñó una aplicación que permite realizar la misma práctica, pero de forma virtual (ver Figura 3). En esta última, los objetos, además de verse reales, se comportan como sus homólogos en la realidad, por lo que la RV abarca desde la simulación hasta la forma en que actúan los objetos que se generan en la computadora.

Figura 2. Práctica real.

Figura 3. Práctica virtual.

Su utilidad hoy en día es bastante abarcadora, extendiéndose desde el entretenimiento hasta áreas como la medicina, la comunicación, el transporte, etc. Entre las aplicaciones de RV existentes, se encuentra el laboratorio virtual, el que ha ganado varios seguidores gracias al papel que ha desempeñado en el ámbito educativo e investigativo.

1.2.1 Laboratorio virtual

William Wulf, investigador especializado en informática, acuñó el concepto de laboratorio virtual en 1989, donde expresó lo siguiente:

“Es un centro sin paredes cuyos usuarios pueden investigar sin tener en cuenta su situación geográfica interactuando con los colegas, teniendo acceso a los instrumentos, compartiendo los datos y los recursos informáticos, recurriendo a la información de las bibliotecas electrónicas. Ese entorno se apoya en unos programas informáticos que permiten trabajar en colaboración y simultáneamente a diversas personas desde distintos sitios.” (4)

En una reunión de expertos sobre el tema, organizada por el Instituto Internacional de Física Teórica y Aplicada (IITAP) de la Universidad de Iowa en Estados Unidos, lo definen como:

"Un espacio electrónico de trabajo concebido para la colaboración y la experimentación a distancia con objeto de investigar o realizar otras actividades creativas, y elaborar y difundir resultados mediante tecnologías difundidas de información y comunicación." (4)

Hoy en día existen gran número de laboratorios virtuales en todo el mundo, como el de física aplicada de la universidad de Córdoba, en España, y el de robótica de la universidad del Valle, en Bolivia. En Cuba se han realizado varios laboratorios sobre diferentes temáticas, como el de mecánica de la universidad José Antonio Echeverría y el de física desarrollado por la universidad "Marta Abreu", en la provincia de Villa Clara. A continuación se muestra una imagen del último laboratorio mencionado (ver Figura 4), en la que se observa un experimento sobre dinámica, donde se hace deslizar un objeto circular por una pendiente que choca al final con un muelle.

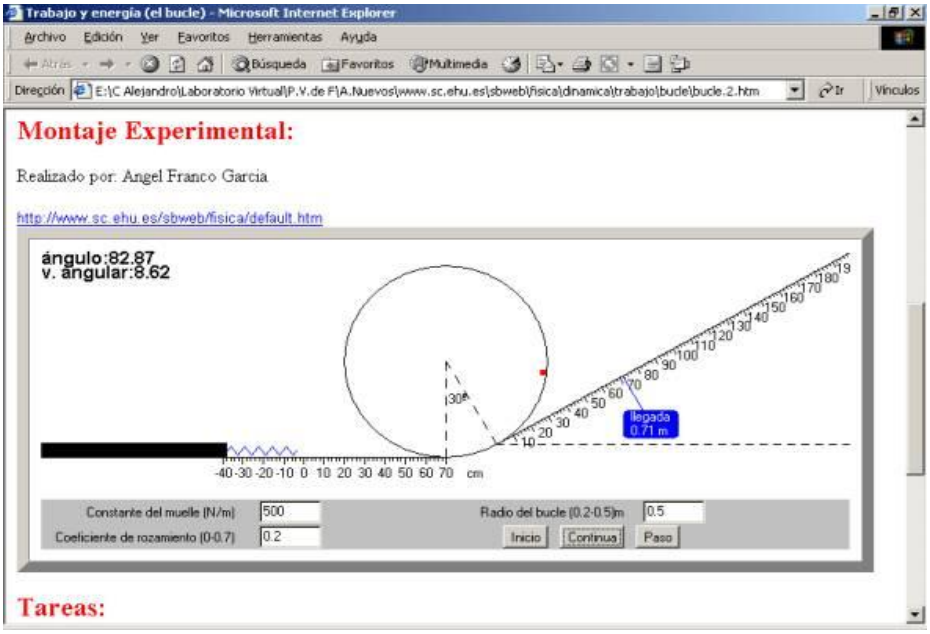


Figura 4. Laboratorio Virtual de Física.

La UCI también ha contribuido al desarrollo de laboratorios virtuales, como los realizados en el CEDIN, que abordan temas informáticos sobre arquitectura de máquinas, diseño de redes y configuración de servicios de red. La Figura 5 muestra una de las prácticas sobre el primer tema mencionado, en la que los usuarios tienen que conectar las piezas externas de la máquina, como: monitor, teclado, bocinas, disco duro, etc.

Figura 5. Laboratorio Virtual de Arquitectura de Máquinas.

Los laboratorios virtuales, al igual que todas las aplicaciones de RV, necesitan que los objetos que se generan en la simulación se vean reales y, además, se comporten lo más realistas posibles. Para lograr que los objetos del mundo virtual puedan realizar las mismas acciones que sus homólogos en el mundo real, se utiliza la animación 3D.

1.3 Animación 3D

La palabra animar significa "dotar de movimiento a cosas inanimadas" (2). En el campo de la informática gráfica, la animación se define como la simulación de movimientos a través de fotogramas*.

La computadora juega un papel fundamental en la producción de animaciones, ya que además de acelerar el proceso de dibujado posibilita la realización de animaciones a partir de modelos tridimensionales, lo que se conoce por animación 3D por ordenador y es bastante utilizada actualmente en la producción de películas y en las aplicaciones de gráficos por computadoras.

Mientras que en la animación tradicional (ver Figura 6) hay que dibujar cada fotograma, en la animación 3D por ordenador se modela cada elemento en 3 dimensiones, se aplican las transformaciones geométricas para modificar su forma o posición y, al igual que en la animación tradicional, se reproducen

* **Fotograma:** es una imagen de una película fotográfica.

estos cambios en una secuencia rápida de fotogramas para dar esa sensación de movimiento (5). Además, gracias a la utilización de luces y sombras se consigue dar mayor realidad a las escenas, como se observa en la Figura 7.

Figura 6. Animación tradicional.

Figura 7. Animación 3D.

A continuación se explican las transformaciones geométricas básicas que son aplicadas a los objetos para realizar animaciones 3D.

1.4 Transformaciones geométricas

Se utilizan para manipular los objetos en entornos sintéticos y están regidas por un sistema de coordenadas, que en el caso de los entornos 3D utilizan 3 ejes: X, Y y Z. Además, se pueden combinar entre ellas para lograr animaciones complejas (1). Las transformaciones básicas son:

1.4.1 Rotar

Para rotar un objeto, se necesita un eje y un ángulo de rotación, ya que definen cuánto girar el elemento alrededor de un punto (6). El siguiente ejemplo (ver Figura 8) muestra un objeto en dos momentos: antes de rotar y después de rotado.

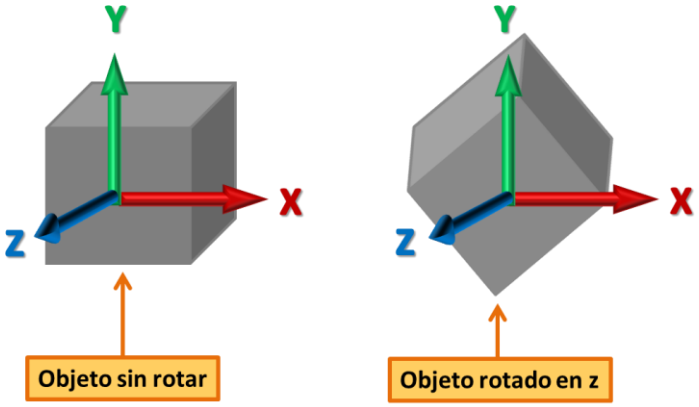


Figura 8. Transformación de rotación.

1.4.2 Escalar

Cuando se habla de escalar un elemento, en RV, se refiere a cambiar el tamaño del mismo (6). Si se utilizan valores mayores a 1 se expande, mientras que valores menores a 1 lo comprimen, como se muestra en la Figura 9. Además, un objeto puede escalarse en uno o varios ejes.

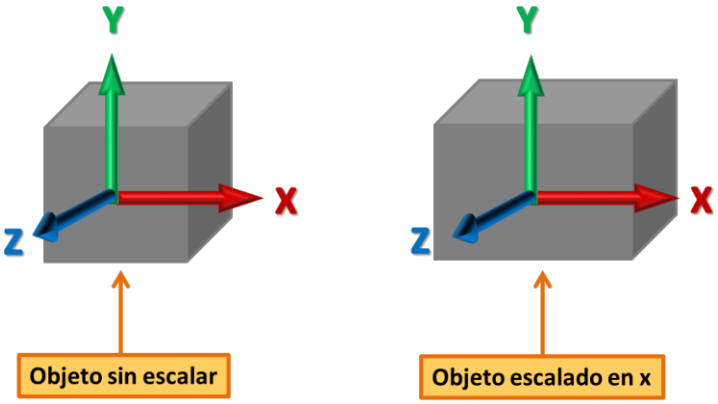


Figura 9. Transformación de escalado.

1.4.3 Trasladar o mover

Mover un cuerpo, en gráficos por computadoras, no es más que la acción de hacer que deje el lugar que ocupa y pase a tomar otro (6), como se observa en la Figura 10.

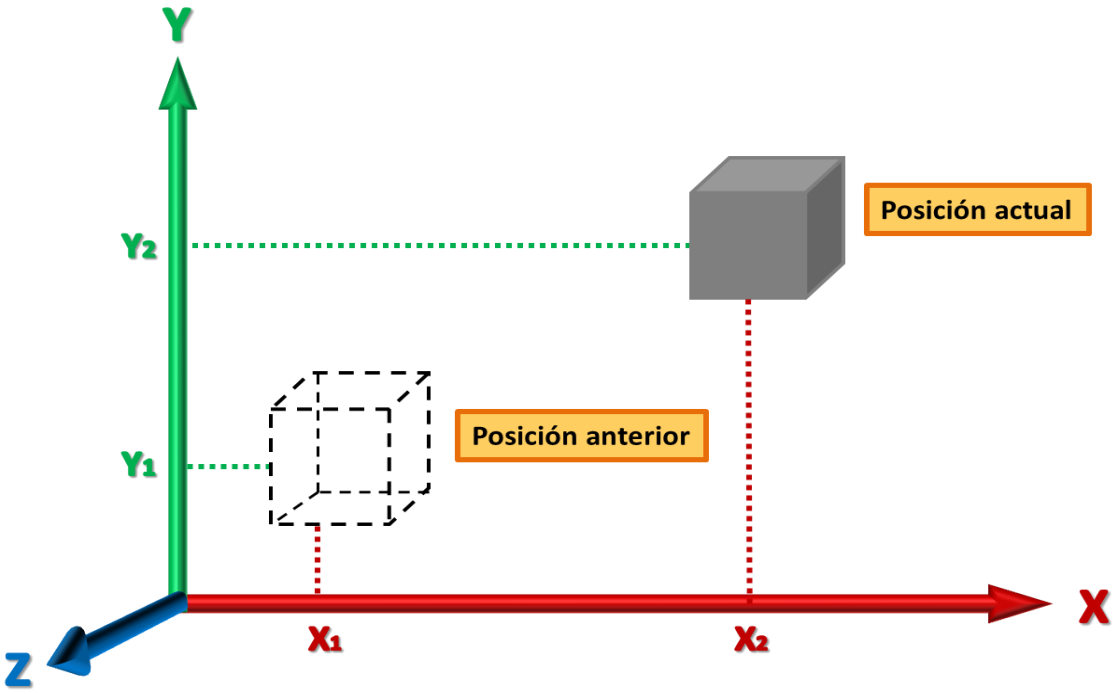


Figura 10. Transformación de traslación.

Existen varias herramientas de diseño como *3DMax*, *Maya*, *Blender*, etc., que permiten realizar las transformaciones mencionadas de forma visual. Pero la animación 3D es un proceso complejo, pues trae consigo la ejecución previa de otras actividades como el diseño y el modelado.

Desde el punto de vista de los programadores, rotar, escalar o trasladar un objeto no es difícil si se realiza a pocos elementos y no se tiene en cuenta las acciones que puedan desencadenarlas. La complejidad aparece cuando es a varios objetos a los que se quiere hacer transformaciones, las que pueden variar en forma y evento desencadenador. Por ello, se necesita automatizar el proceso de realizar dichas transformaciones, para lo que se propone la utilización de las tareas de comportamiento.

1.5 Tarea de comportamiento

Una tarea, de forma general, es "un trabajo que debe hacerse en tiempo limitado" (2), mientras que en la RV es "cualquier actividad realizada por la aplicación" (7), desde las acciones de *rendering*[*] hasta la importación de geometrías.

Los objetos, al igual que la aplicación, realizan varias tareas, como: deformar la malla, cambiar de apariencia, etc. Entre ellas, se encuentran acciones que definen cómo se comportará el objeto ante una determinada situación, las que son llamadas tareas de comportamiento. Estas permiten precisar el evento que desencadenará la acción y el tiempo de ejecución de la misma, dos condiciones que no son posibles manejar en la animación. Además, posibilitan la definición de diferentes tipos de eventos, por ejemplo: teclado, *mouse*, tiempo, posición, etc. Tienen mayor utilidad en elementos que componen la escena, como sillas, puertas, luces, etc., que no necesitan de animaciones complejas. Tienen diferentes clasificaciones, entre los que se encuentran:

- **Movimiento:** reúne las transformaciones de traslación, rotación y escalado. La Figura 11 muestra un ejemplo donde se observa la animación ocurrida cuando se abre una puerta.

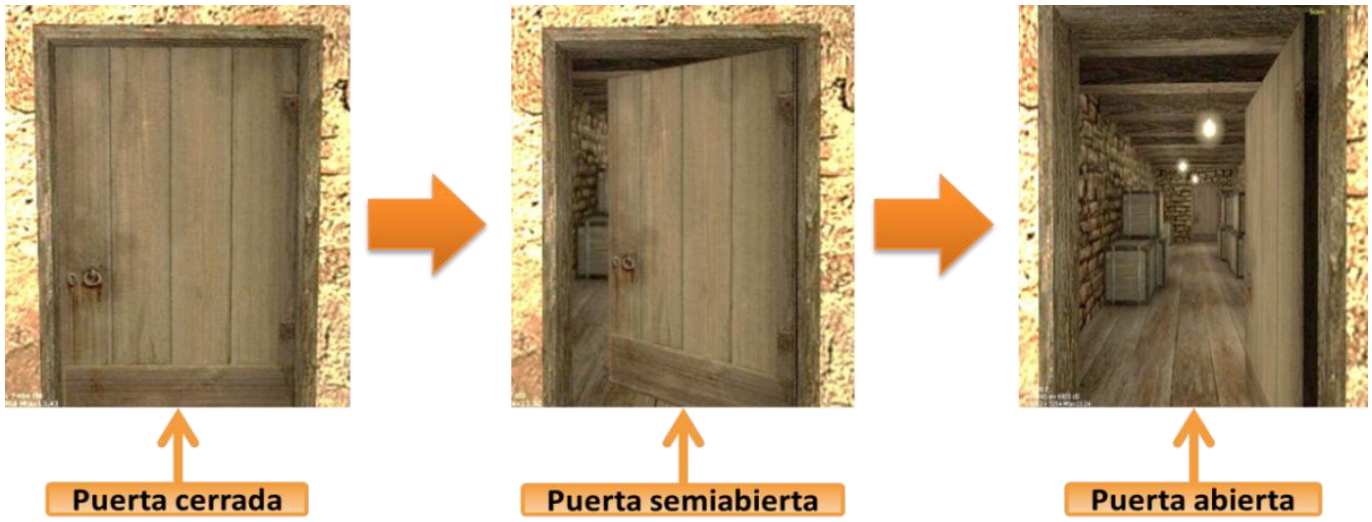


Figura 11. Animación de puerta.

[*] ***Render* o *rendering*:** se le llama al proceso de generar una imagen desde un modelo.

- **Cambio de estado:** permite activar o desactivar objetos para que se muestren solo cuando sea necesario. Un ejemplo son las siguientes imágenes corresponden a una luz que está apagada (ver Figura 12) y se enciende (ver Figura 13).

Figura 12. Escena antes de encender la luz.

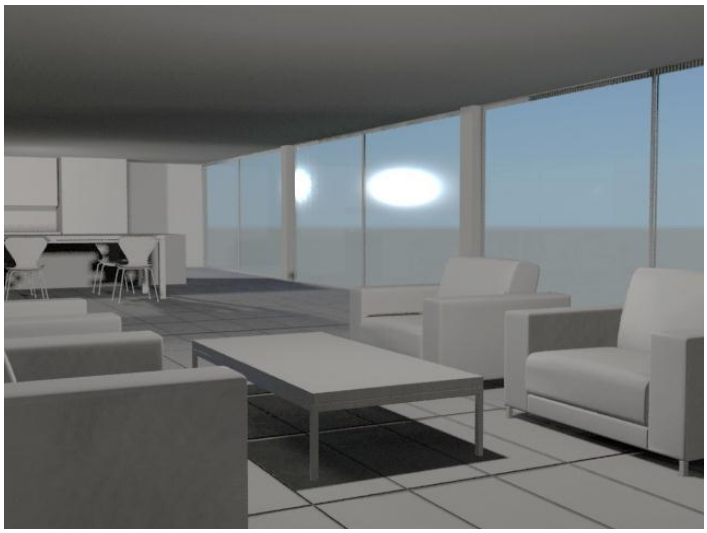


Figura 13. Escena después de encender la luz.

Existen varias herramientas 3D que utilizan las tareas de comportamiento, seguidamente se explican algunas de ellas.

1.5.1 *WorldToolKit* (WTK)

WTK es un avanzado entorno de desarrollo multiplataforma para aplicaciones gráficas 3D en tiempo real con un alto rendimiento, que posee una biblioteca de más de 1000 funciones y herramientas programadas en C. (7)

Permite asignar tareas a los objetos (ver Figura 15), las que se asocian a funciones que pueden provocar cambios de apariencia o darle movimiento y se implementan a través de la clase WTtask (ver Figura 14).

```
class WtTask : public WtBase
{
    WtTask(void *ptr,WTtask_function fptr,float priority = 1.f);
    ~WtTask(void);
    FLAG Add(void);
    WTtask_function GetFunction(void);
    float GetPriority(void);
    FLAG Remove(void);
    FLAG SetPriority(float priority);
};
```

Figura 14. Clase WTtask.

```
int main(int argc, char *argv[])
{
    ...
    /* Create a movable "node" */
    ..
    WTtask_new(node, Rotate_Y, 1.0f);
    ...
}

void Rotate_Y(WTnode *node)
{
    WTmovnode_axisrotation(node, Y, 5.f*PI/180.f);
}
```

Figura 15. Asignar tarea a un objeto.

Entre las principales desventajas que tiene WTK, se encuentran:

- Solamente funciona con OpenGL.
- La licencia comercial para Windows cuesta US$ 795.

1.5.2 3DVIA Studio

3DVIA Studio es un framework destinado a la creación de videojuegos 3D desde el punto de vista de los diseñadores, por lo que la programación es completamente visual y bastante intuitiva. Fue creado por la Dassault Systemes. (8)

Posee un motor de comportamiento que se encarga de realizar el ciclo de procesamiento de las tareas, siendo el componente central de la aplicación. Para conocer cómo se hacen las tareas en 3DVIA Studio se muestra el siguiente proyecto, el que consiste en un ventilador al que se le quiere rotar las aspas. Lo primero, es crear un nuevo comportamiento, que se llamará "fanBehavior" (ver Figura 16).

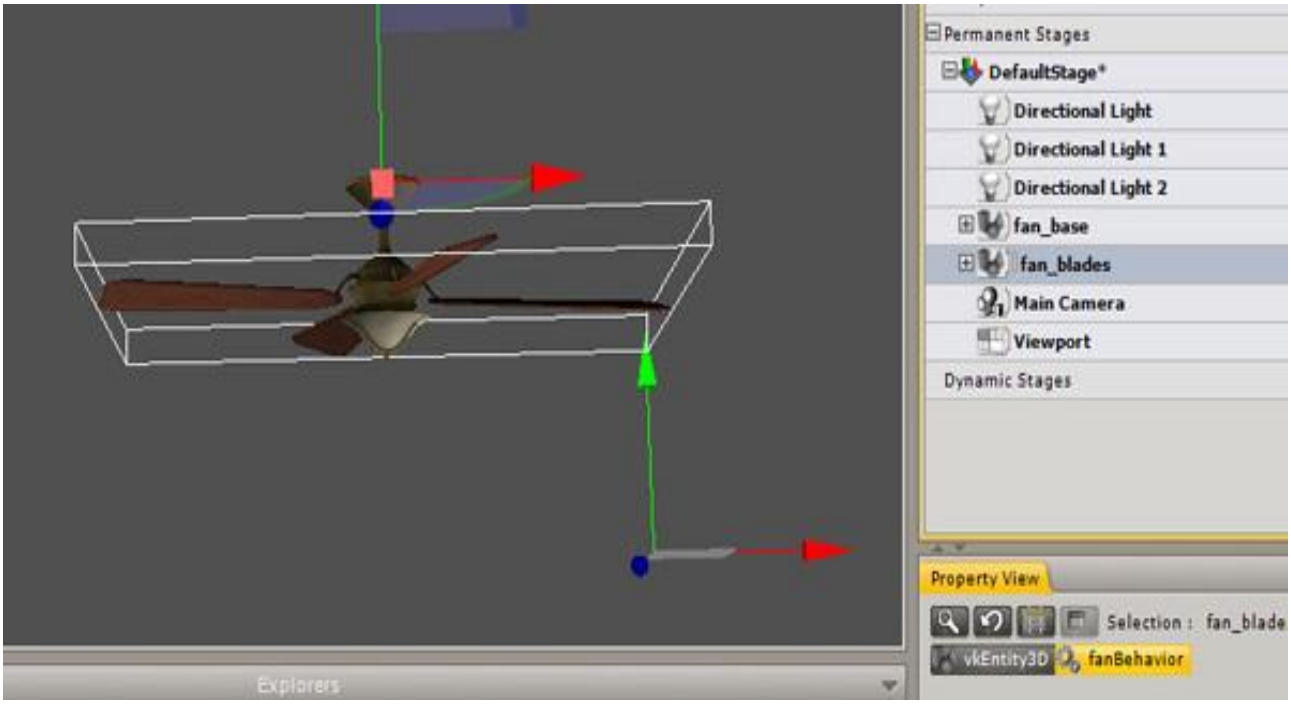


Figura 16. Crear nuevo comportamiento.

Después hay que crear la tarea, en este caso de rotación, especificando el eje y el ángulo de rotación. Por último, se conecta el nodo que representa las aspas del ventilador con la tarea, como se muestra en la siguiente imagen (ver Figura 17).

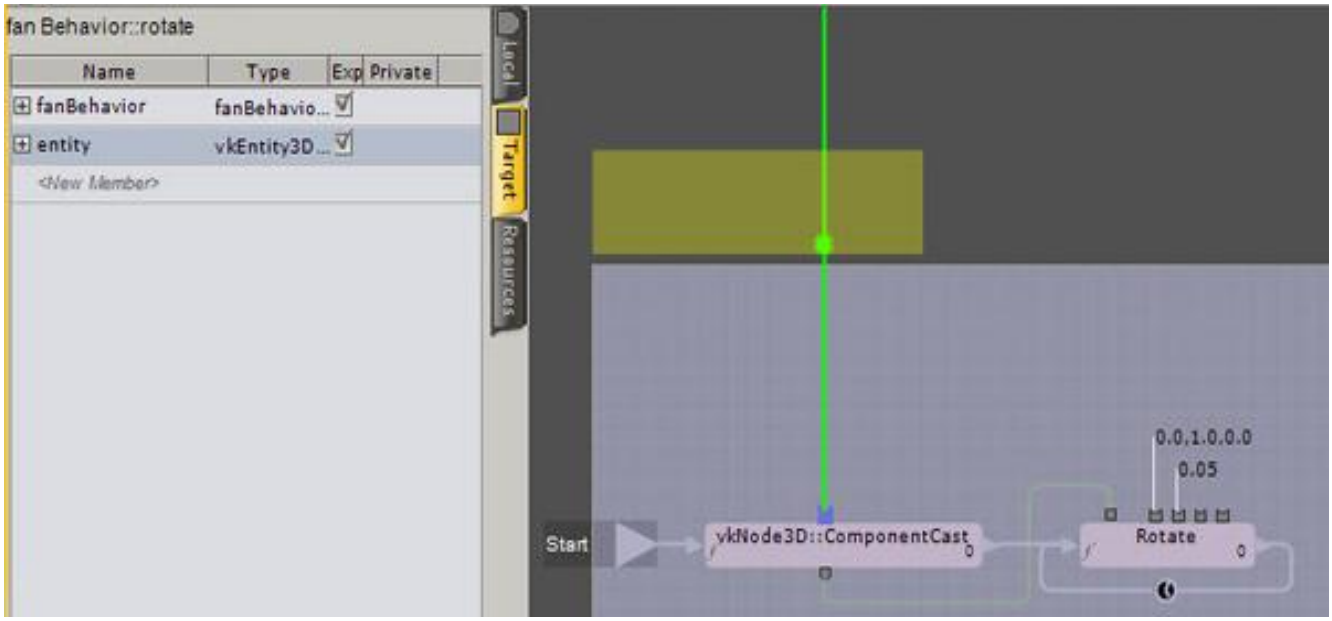


Figura 17. Conectar un nodo con una tarea.

Las desventajas de utilizar 3DVIA Studio son las siguientes:

- Disponible solo para Windows.
- Hay que comprar licencia, ya que se encuentra bajo licencia *Shareware* y después de un tiempo de prueba hay que pagar para poder usarlo.

A continuación se abordan algunas herramientas y metodologías de desarrollo de software que podrían utilizarse en la realización de la futura aplicación.

1.6 Metodologías de desarrollo de software

Una metodología de desarrollo de software es, según Piattini, “un conjunto de procedimientos, técnicas, herramientas y un soporte documental que ayuda a los desarrolladores a realizar nuevos software” (9).

Es una guía que indica qué hacer y cómo actuar en cada momento del ciclo de desarrollo del software, y su finalidad es mantener la eficacia y eficiencia durante todo el proceso de desarrollo para obtener un sistema con calidad. Hoy en día existen varias metodologías, las que se clasifican en dos tipos principales: las metodologías robustas y las metodologías ágiles.

Las robustas se caracterizan por realizar una fuerte planificación del proyecto durante todo el proceso de desarrollo, prestando especial atención en la definición detallada de los procesos a realizar y en las herramientas que se utilizarán, lo que genera una extensa documentación. Además, son eficaces en proyectos de mayor envergadura (10). Ejemplos de metodologías robustas son:

- *Rational Unified Process* (RUP).
- *Microsoft Solutions Framework* (MSF).

Por otra parte, las ágiles están orientadas al equipo de trabajo y no a los procesos o herramientas utilizadas, prestando mayor importancia a la obtención de un producto funcional que a la documentación del mismo y se utilizan en proyectos de menor volumen. Además, se les conoce por la simplicidad de sus reglas y prácticas, su flexibilidad ante los cambios y su ideología de colaboración (10). Entre las metodologías ágiles se encuentran:

- *Extreme Programming* (XP).
- *Crystal Methodologies*.
- SCRUM.

Para facilitar el trabajo a los ingenieros de software se han implementado diversas herramientas que permiten la rápida gestión del mismo, las que reciben el nombre de *Computer Aided Software Engineering* (CASE).

1.7 Herramienta CASE

Una herramienta CASE es “el conjunto de métodos, utilidades y técnicas que facilitan la automatización del ciclo de vida del desarrollo de sistemas de información, completamente o en alguna de sus fases” (11).

No son más que aplicaciones informáticas que permiten al ingeniero de software desarrollar y mantener la documentación del sistema. Automatizan los procesos y tareas relacionadas con el desarrollo de software y reducen los costos. Incrementan la productividad y calidad del software, además de mejorar la gestión de la documentación de la aplicación y del proyecto en general. Potencian la reutilización del software y aseguran una mejor planificación del proyecto. Las más utilizadas son:

- *Rational Rose Enterprise Edition*.
- *Visual Paradigm*.

Después de realizar la ingeniería del software, se pasa a la etapa de construcción, donde el programador tiene que escribir el código, el que tiene que estar en un lenguaje que la máquina pueda interpretar. Estos lenguajes se conocen con el nombre de lenguajes de programación.

1.8 Lenguaje de programación

"Un lenguaje de programación es una técnica estándar de comunicación que permite expresar las instrucciones que han de ser ejecutadas en una computadora. Consiste en un conjunto de reglas sintácticas y semánticas que definen un lenguaje informático." (12)

Se utilizan en la creación de programas para controlar el comportamiento físico y lógico de la máquina, para realizar algoritmos con precisión o como modo de comunicación humana. Permiten al programador especificar sobre qué datos una computadora debe operar, cómo deben ser almacenados y transmitidos y qué acciones tomar bajo ciertas circunstancias. Ejemplo de ellos son: *C*, *C++*, *C#*, *Python*, *Java*, etc.

Mientras escribe el código, el programador necesita de una herramienta que evalúe lo que ha escrito, porque puede haber código que no sea comprensible para la computadora. Para evaluar el código se utilizan los Entorno de Desarrollo Integrado (IDE, por sus siglas en inglés).

1.9 Entorno de desarrollo integrado

Un IDE es un programa informático compuesto por un conjunto de herramientas de programación que puede dedicarse a uno o varios lenguajes de programación (13). Se compone, generalmente, por un editor de código, un compilador, un depurador y un constructor de interfaz gráfica. Existen varios, por ejemplo:

- *Microsoft Visual Studio*.
- *QT Creator*.

Para desarrollar una aplicación de gráficos por computadoras no basta con un IDE, se necesita, además, "algo" que permita simular las escenas y objetos que se han creado, aquí es donde entran en acción los motores gráficos.

1.10 Motor gráfico

Un motor gráfico es una biblioteca de clases que se encarga del proceso de *render* (14). Se caracterizan por estar orientados a los gráficos 2D y 3D, pero en ocasiones también incluyen acciones como la inteligencia artificial, la detección de colisiones, etc. Existe gran variedad de sistemas que son o contienen motores de *rendering*, entre los que se encuentran:

- *3DGameStudio*.
- *Crystal Space*.
- *Object Oriented Graphics Rendering Engine* (OGRE3D).
- *Blender Game Engine*.
- *SceneToolKit* (STK).

Para cualquier aplicación, si se desea guardar los datos generados y, además, se necesita que el texto tenga una estructura definida, se utilizan los lenguajes de marcas.

1.11 Lenguaje de marcas

Un lenguaje de marcado es una forma de codificar un documento, es decir, además de contener el texto, incorpora etiquetas o marcas que estructuran el texto o su presentación y se diferencia del lenguaje de programación en que no posee funciones aritméticas (15). Existen diferentes lenguajes de marcas, como HTML, XML, GML, COLLADA, etc., los que se agrupan en tres tipos principales:

- **Marcado de presentación:** indica el formato del texto y es útil para estructurar la presentación de un documento.

- **Marcado de procedimientos:** enfocado a la presentación del texto y es visible para el usuario que edita el texto.
- **Marcado descriptivo:** utiliza etiquetas para describir los fragmentos de texto, pero sin especificar cómo deben ser representados o en qué orden.

Tienen gran utilidad en la informática, como: páginas web, bases de datos, correo electrónico, videojuegos, etc.

Consideraciones Parciales

Se reunieron los conceptos importantes para la comprensión del tema de investigación y se enfatizó en el uso de las tareas de comportamiento para animar los objetos de la escena, que constituye el centro de la investigación.

CAPÍTULO 2. DESCRIPCIÓN DE LA PROPUESTA DE SOLUCIÓN

Introducción

El capítulo tiene como objetivo valorar las características del módulo a desarrollar, para lo que se determinará el marco conceptual del sistema y se definirá la metodología y herramientas que se utilizarán en la construcción del mismo. Además, se elaborará una propuesta de solución a partir de los requerimientos que plantee el cliente y se realizará el diseño de las clases.

2.1 Marco conceptual de la investigación

Después de brindar un grupo de conceptos en el capítulo anterior, la investigación se desarrollará bajo las siguientes definiciones:

- **RV:**

Se utilizará el concepto dado por C. Manetta y R. Blade en 1995, donde se expone que es “un sistema de computación usado para crear un mundo artificial donde el usuario tiene la impresión de estar en ese mundo y la habilidad de navegar y manipular objetos en él". (16)

- **Laboratorio virtual:**

Se define que un laboratorio virtual es una herramienta informática que simula el ambiente de un laboratorio real, en el que profesores y estudiantes pueden desarrollar prácticas, y tiene como objetivo facilitar la comprensión de temas científicos. Ni sustituye ni compite con el laboratorio tradicional, en cambio puede utilizarse como una extensión del mismo.

- **Tareas de comportamiento:**

Las tareas de comportamiento son las acciones que asignan comportamientos a los objetos. La presente investigación se centrará en las tareas referidas a la rotación, traslación y escalado de los objetos.

2.2 Selección de la metodología y el ambiente de desarrollo

La metodología que guiará el proceso de desarrollo será XP, porque centra las fuerzas en la implementación del software y solo documenta los artefactos más importantes, ya que el sistema está orientado a los programadores de los laboratorios virtuales. Además, se basa en la simplicidad, la comunicación, el reciclado continuo de código y funciona mejor en grupos de desarrollo pequeños. (17)

El ambiente de desarrollo para la realización de la aplicación es el siguiente:

- Para gestionar la documentación del software, se selecciona *Visual Paradigm for UML* como herramienta CASE, pues soporta el ciclo completo de desarrollo y genera el código en una amplia gama de lenguajes. Además, proporciona varios tutoriales, es multiplataforma y muy profesional. (18)

- El lenguaje de programación es *C++*, porque es un lenguaje robusto y muy utilizado en la programación de gráficos por computadoras. También, cuenta con gran número de tutoriales, libros, códigos fuentes, etc. (19)

- Se utilizará *QT Creator* como IDE, ya que funciona perfectamente con *C++* y posee una extensa documentación, es libre y se encuentra funcional en varias plataformas. (20)

- Para el *rendering* de las escenas se utilizará *OGRE3D*, porque está basado en *C++*, es multiplataforma, soporta *OpenGL* y *DirectX*, y es de código abierto. Además, posee una ayuda bastante completa y gran número de ejemplos de apoyo. (21)

- Como lenguaje de marcado se utilizará XML, pues permite compartir la información entre aplicaciones diferentes de forma segura, fiable y fácil, característica que lo ha convertido en un estándar para el intercambio de información estructurada entre diferentes plataformas. (22)

2.3 Modelo de dominio

Un modelo de dominio o modelo conceptual es “una representación visual de las clases conceptuales u objetos del mundo real en un dominio de interés” (23). En otras palabras, explica los conceptos

significativos de un problema, facilitando la interpretación del sistema y el entorno donde está enmarcado. La Figura 18 muestra el modelo de dominio del software a desarrollar.

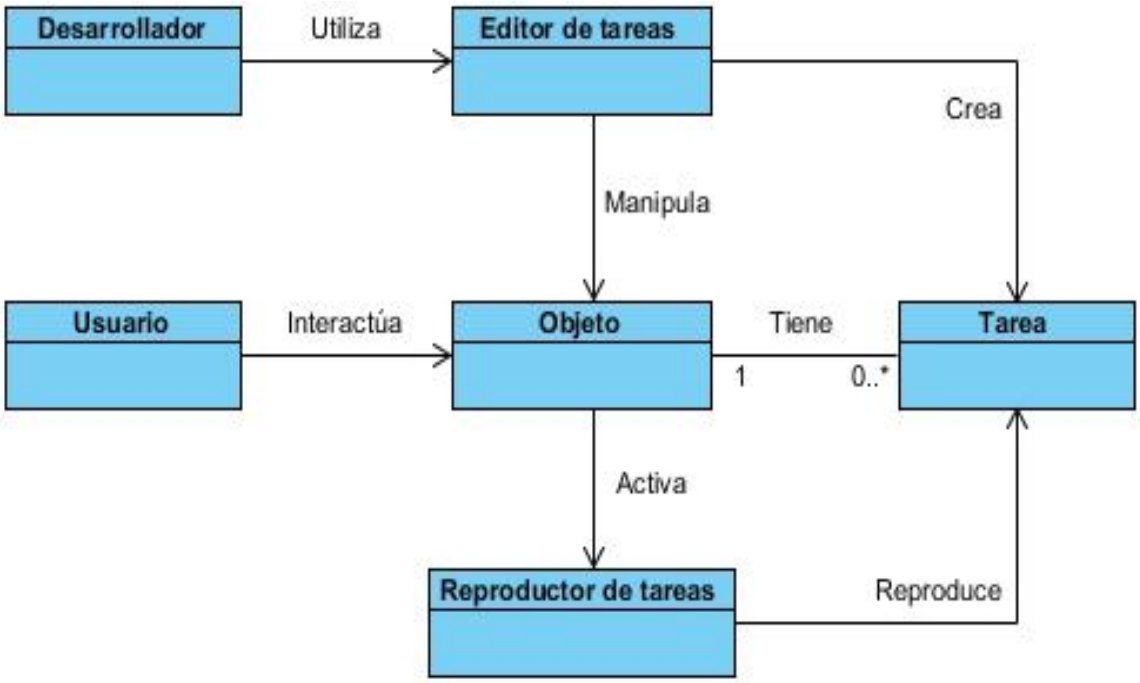


Figura 18. Diagrama del Modelo de dominio.

A continuación se describen los conceptos presentes en el modelo anterior:

- **Desarrollador:** utiliza el Editor de tareas para crear y editar las tareas.
- **Usuario:** interactúa con los objetos de la escena, los que ejecutan las tareas que le fueron creadas.
- **Objeto:** contiene las tareas y activa el Reproductor de tareas cuando el usuario interactúa con él.
- **Tarea:** es la acción que realizará el objeto.
- **Editor de tareas:** es la interfaz que permite al desarrollador crear y editar las tareas mediante la manipulación de los objetos.
- **Reproductor de tareas:** reproduce las tareas creadas.

2.4 Propuesta de solución

Para dar solución a la problemática planteada, se propone realizar un módulo que le permita al desarrollador la creación de tareas de forma visual y que sea integrable a la arquitectura de los laboratorios virtuales del CEDIN. Además, le brindará, al desarrollador, la posibilidad de editar, reordenar, pre-visualizar y reproducir las tareas sin necesidad de ir al código y sin tener que reiniciar la aplicación para observar los cambios. También, los usuarios de los laboratorios podrán ejecutar las tareas que sean creadas anteriormente.

Contará con una interfaz independiente de la principal, la que estará compuesta por varios componentes visuales que permitirán: introducir identificador, listar los eventos, seleccionar el manipulador (herramienta) y establecer el tiempo de ejecución. Además, tendrá un panel para mostrar y editar las tareas que han sido creadas anteriormente. La Figura 19 muestra un prototipo de la interfaz:

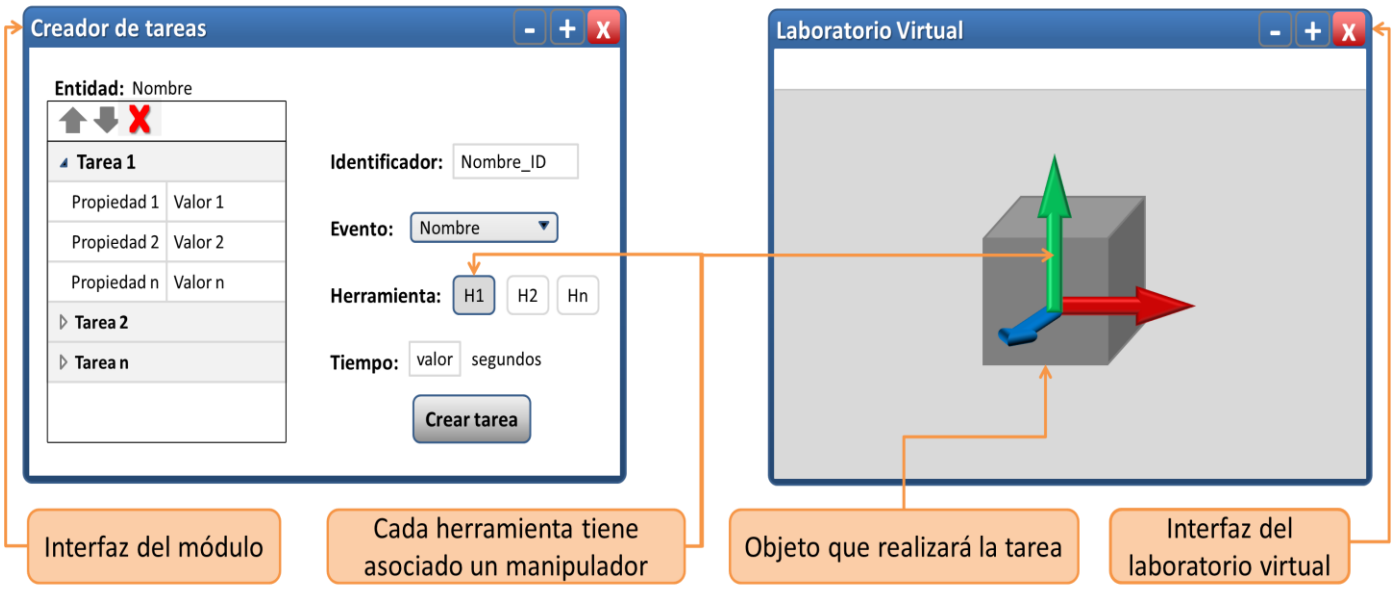


Figura 19. Prototipo del sistema.

2.5 Requisitos del sistema

Los requerimientos son descripciones de los servicios proporcionados por el sistema y sus restricciones operativas (24). Reflejan las necesidades del cliente y las características, especificaciones y

funcionalidades de la aplicación. A continuación se mencionan los requisitos funcionales y no funcionales que definen la aplicación a desarrollar.

2.5.1 Requisitos funcionales

"Los requerimientos funcionales describen lo que el sistema debe hacer" (24), que en otras palabras son las acciones que se pueden realizar. Para el futuro software, se definieron los siguientes requisitos funcionales:

RF1. Crear tarea a un objeto.
RF2. Ejecutar las tareas de un objeto.
RF3. Listar las tareas de un objeto.
RF4. Modificar las propiedades de una tarea.
RF5. Eliminar tarea.
RF6. Reordenar las tareas.
RF7. Pre-visualizar tareas.

2.5.2 Requisitos no funcionales

Los requisitos no funcionales se refieren a las propiedades o cualidades del software (24), las que hacen del mismo un producto atractivo, usable, rápido y confiable. Seguidamente, se explican los requerimientos no funcionales de la aplicación a desarrollar:

- **Soporte.**

Contará con un manual de usuario.

- **Hardware.**

Se utilizará en computadoras que cuenten, como mínimo, 1 Gb de memoria RAM.

- **Software.**

Las computadoras deben tener como sistema operativo: Windows XP, Vista, Seven o alguna distribución GNU/Linux.

2.6 Fase de planificación

Se define el alcance del proyecto, donde el cliente plantea sus necesidades a través de las historias de usuarios. Además, se estima la prioridad y el esfuerzo necesario para desarrollar cada historia de usuario (HU) y se realiza el cronograma de iteraciones de acuerdo a las mismas.

2.6.1 Historias de usuario

Una HU es una parte de una funcionalidad, o una funcionalidad completa, que es de valor para el cliente (17). Su principal diferencia con la descripción de los casos de uso es el nivel de detalle, ya que solamente proporciona información sobre la estimación del riesgo y el tiempo que conllevará su implementación. A continuación se detallan las historias de usuario definidas para la realización del módulo:

Historia de Usuario	
Número: 1	**Usuario:** Desarrollador
Nombre historia: Crear tarea a un objeto.	
Prioridad en el negocio: Alta	**Nivel de complejidad:** Alta
Tiempo de Estimación: 2 semanas	**Iteración asignada:** 1
Descripción: Después de seleccionar un objeto, para crear una tarea se necesita seleccionar el evento que la desencadenará, introducir el tiempo de duración y realizar la transformación deseada. La tarea creada se salva en un fichero XML.	
Observaciones: Las tareas que corresponden al mismo objeto y evento se guardan en el mismo fichero XML y se salvan en el orden que fueron creadas.	

Tabla 1. HU "Crear tarea a un objeto".

Historia de Usuario	
Número: 2	**Usuario:** Usuario del laboratorio virtual
Nombre historia: Ejecutar las tareas de un objeto.	

Prioridad en el negocio: Alta	**Nivel de complejidad:** Alta
Tiempo de Estimación: 2 semanas	**Iteración asignada:** 1
Descripción: Cuando el usuario realice una acción sobre un objeto, se ejecutarán las tareas asignadas al objeto que cumplan con ese evento. Además, las tareas serán ejecutadas en el orden en que aparecen en el fichero.	
Observaciones: Si al objeto no le fueron creadas tareas anteriormente, no habrá ninguna tarea para ejecutar.	

Tabla 2. HU "Ejecutar las tareas de un objeto".

Historia de Usuario	
Número: 3	**Usuario:** Desarrollador
Nombre historia: Listar las tareas de un objeto.	
Prioridad en el negocio: Media	**Nivel de complejidad:** Media
Tiempo de Estimación: 1 semana	**Iteración asignada:** 2
Descripción: Se listarán las tareas de un objeto de acuerdo a un evento. Además, las tareas serán listadas en el mismo orden en que aparecen en el fichero.	
Observaciones: Todas las tareas en la lista pertenecen al mismo evento.	

Tabla 3. HU "Listar las tareas de un objeto".

Historia de Usuario	
Número: 4	**Usuario:** Desarrollador
Nombre historia: Modificar las propiedades de una tarea.	
Prioridad en el negocio: Media	**Nivel de complejidad:** Media
Tiempo de Estimación: 1 semana	**Iteración asignada:** 2
Descripción: Se selecciona una tarea y se modifica la propiedad o propiedades deseadas, introduciendo los valores requeridos.	
Observaciones: La tarea debe existir en el fichero.	

Tabla 4. HU "Modificar las propiedades de una tarea".

Historia de Usuario	
Número: 5	**Usuario:** Desarrollador
Nombre historia: Eliminar tarea.	
Prioridad en el negocio: Media	**Nivel de complejidad:** Baja
Tiempo de Estimación: 0.5 semana	**Iteración asignada:** 2
Descripción: Se selecciona la tarea y se elimina.	
Observaciones: La tarea debe existir en el fichero.	

Tabla 5. HU "Eliminar tarea".

Historia de Usuario	
Número: 6	**Usuario:** Desarrollador
Nombre historia: Reordenar las tareas.	
Prioridad en el negocio: Baja	**Nivel de complejidad:** Baja
Tiempo de Estimación: 0.5 semana	**Iteración asignada:** 3
Descripción: Cambia el orden en que se ejecutan las tareas.	
Observaciones: Reordena el fichero que contiene las tareas.	

Tabla 6. HU "Reordenar las tareas".

Historia de Usuario	
Número: 7	**Usuario:** Desarrollador
Nombre historia: Pre-visualizar tareas.	
Prioridad en el negocio: Baja	**Nivel de complejidad:** Media
Tiempo de Estimación: 0.5 semana	**Iteración asignada:** 3
Descripción: Después de que el desarrollador crea las tareas, puede pre-visualizarlas y observar cómo se verán en la aplicación final.	
Observaciones: Se pre-visualizan todas las tareas del objeto que corresponden al evento seleccionado.	

Tabla 7. HU "Pre-visualizar tareas".

2.6.2 Estimación de esfuerzos por historias de usuario

Al estimar el esfuerzo que se necesita para realizar las historias de usuario, se mide la velocidad con que se desarrollará el software y el progreso existente en la implementación del mismo. A continuación se muestra la estimación del esfuerzo requerido para cada HU:

Historia de usuario	Puntos de Estimación (semanas)
Crear tarea a un objeto.	2
Ejecutar las tareas de un objeto.	2
Listar las tareas de un objeto.	1
Modificar las propiedades de una tarea.	1
Eliminar tarea.	0.5
Reordenar las tareas.	0.5
Pre-visualizar tareas.	0.5

Tabla 8. Estimación de esfuerzos por HU.

2.6.3 Plan de iteraciones

Después de identificar y detallar las historias de usuario, se procede a la elaboración del plan de iteraciones, donde se definen qué historias de usuario serán implementadas en cada iteración. Para el desarrollo del presente software se estableció la realización de tres iteraciones:

- **Iteración 1:** realiza las historias de usuario de mayor importancia para el cliente, obteniendo la primera versión funcional de la aplicación, que sirve para captar nuevos requerimientos y para realizar las pruebas pertinentes.

- **Iteración 2:** se implementan las historias de usuario con prioridad media en el negocio y se corrigen los errores y no conformidades de la anterior iteración. Igualmente, se prueba cada funcionalidad implementada.

- **Iteración 3:** se desarrollan las historias de usuarios faltantes y se prueba el software en su totalidad.

2.6.4 Plan de duración de las iteraciones

El plan de duración de las iteraciones (ver Tabla 9) muestra cuánto se estima que dure cada iteración a realizar, además del orden en que serán implementadas las historias de usuario en cada una de las iteraciones.

Iteración	Historias de usuarios	Duración total (semanas)
1	Crear tarea a un objeto.	4
	Ejecutar las tareas de un objeto.	
2	Listar las tareas de un objeto.	2.5
	Modificar las propiedades de una tarea.	
	Eliminar tarea.	
3	Reordenar las tareas.	1
	Pre-visualizar tareas.	

Tabla 9. Plan de duración de las iteraciones.

2.6.5 Plan de entregas

El plan de entregas (ver Tabla 10) define las historias de usuario que conformarán una entrega.

Historias de usuarios	1ra Iteración Marzo-3ra semana	2ra Iteración Abril-2da semana	3ra Iteración Abril-4ta semana
Crear tarea a un objeto.	V1.0		
Ejecutar las tareas de un objeto.	V1.1		
Listar las tareas de un objeto.		V1.2	
Modificar las propiedades de una tarea.		V1.13	
Eliminar tarea.		V1.4	
Reordenar las tareas.			V1.5
Pre-visualizar tareas.			V1.6

Tabla 10. Plan de entregas.

2.7 Fase de diseño

Se realiza el diseño del sistema mediante las tarjetas CRC (Contenido, Responsabilidad, Colaboración) y se añade un diagrama de clases para facilitar la comprensión del mismo. Además, se describe el estándar de codificación que se usará en la implementación.

2.7.1 Tarjetas CRC

Las tarjetas CRC son fichas en la que se escriben las responsabilidades (funcionalidades) de la clase y los objetos (clases que involucra) con los que colabora para llevar a cabo esas responsabilidades (23). A continuación se detalla cada clase:

Tarjeta CRC	
Clase: TaskEditor	
Responsabilidad	**Colaboración**
Es la clase principal de la aplicación y las funciones que realiza son: • Controlar y actualizar la interfaz principal de la aplicación. • Gestionar los manipuladores: ✓Crear y destruir manipulador. ✓Mostrar manipulador en dependencia de la tarea que se va a realizar. ✓Actualizar posición del manipulador. • Gestionar las tareas del objeto. ✓Crear tarea. ✓Modificar tarea. ✓Eliminar tarea. • Gestionar el estado del objeto. ✓Calcular la transformación del objeto. ✓Actualizar el estado del objeto.	Task TaskReader TaskWriter

Tabla 11. Tarjeta CRC "TaskEditor".

Tarjeta CRC
Clase: TaskPlayer

Responsabilidad	Colaboración
Reproduce las tareas de un objeto en dependencia del evento que se ejecutó.	Task TaskReader TaskEditor

Tabla 12. Tarjeta CRC "TaskPlayer".

Tarjeta CRC	
Clase: TaskReader	
Responsabilidad	**Colaboración**
Hace un análisis de la estructura del fichero XML y llena una lista con las tareas que contiene.	Task

Tabla 13. Tarjeta CRC "TaskReader".

Tarjeta CRC	
Clase: TaskWriter	
Responsabilidad	**Colaboración**
Guarda las tareas en un fichero XML con una estructura definida.	Task

Tabla 14. Tarjeta CRC "TaskWriter".

Tarjeta CRC	
Clase: Task	
Responsabilidad	**Colaboración**
Representa el concepto de tarea en sí y se encarga de: • Crear y destruir la tarea. • Actualizar la tarea: realiza los cálculos de las transformaciones que se realizaron y actualiza el estado en se encuentra (activa o inactiva) y si se ejecutó o no. • Emitir señal después de terminar la ejecución: emite una señal para informar a la siguiente tarea que puede comenzar a ejecutarse.	

Tabla 15. Tarjeta CRC "Task".

2.7.2 Diagrama de clases del diseño

Un diagrama de clases del diseño "representa las especificaciones de las clases e interfaces software" (23). La Figura 20 muestra el diagrama de clases del diseño correspondiente al sistema desarrollado.

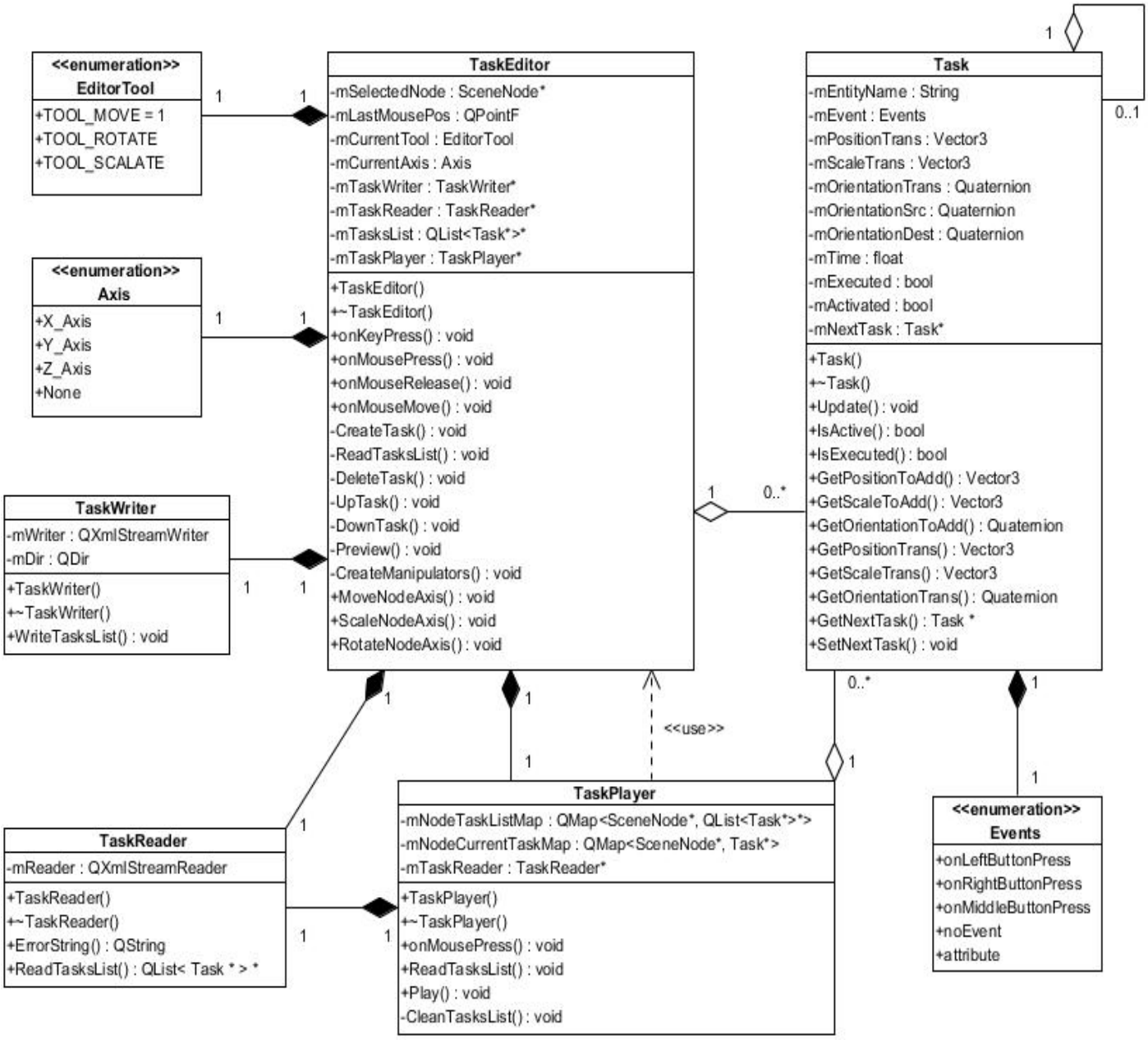


Figura 20. Diagrama de clases del diseño.

2.7.3 Estándar de codificación

Los estándares de codificación o estilos de programación, como también se les llama, definen la estructura y apariencia física del código, lo que facilita la lectura, comprensión y mantenimiento del mismo. Para la implementación del módulo, se utilizó el siguiente estilo de programación:

Sangría: una tabulación equivalente a cuatro espacios. (Ver Figura 21)

```
void Task::SetExecuted( bool executed )
{
    mExecuted = executed;
}
```

Figura 21. Ejemplo de sangría.

Declaración:

- **Variables:** comienzan con la letra m, acrónimo de *my* (en español: mi), para simbolizar que son variables privadas de la clase y todas las palabras que las componen comenzarán con letra inicial mayúscula. (Ver Figura 22)

```
Events mEvent;
Ogre::Vector3 mPositionTrans, mScaleTrans;
float mTime;
```

Figura 22. Ejemplo de declaración de variables.

- **Clases:** todas las palabras que componen el nombre de la clase comenzarán con letra inicial mayúscula. (Ver Figura 23)

```
class TaskEditor
```

Figura 23. Ejemplo de declaración de clases.

- **Funciones:** todas las palabras que contenga el nombre de la función se escribirán con letra inicial mayúscula. (Ver Figura 24)

```
bool IsActive();
void SetActive(bool active);
```

Figura 24. Ejemplo de declaración de funciones.

Condicionales y ciclos: se utilizarán las llaves solo cuando contengan más de una línea de código. A continuación se muestran dos condicionales: una simple (ver Figura 25) y una compuesta (ver Figura 26).

```
if (inpEvent == "onLeftButtonPress")
        return onLeftButtonPress;
```

Figura 25. Ejemplo de condicional simple.

```
if (timeElapsed >= mTime)
{
        mExecuted = true;
        mActivated = false;
        timeElapsed = 0.0f;
}
```

Figura 26. Ejemplo de condicional compuesta.

2.7.4 Estructura del fichero XML

- **Nombre de archivo:** se compone del nombre del objeto, un guión bajo y el evento que desencadena la tarea.
- **Encabezado del fichero XML:** el tipo de documento será “TaskEditor”.
- **Etiquetas compuestas:** cada una equivale a una tarea y se representan con el nombre de *task* en el fichero.
- **Etiquetas simples:** constituyen las propiedades con los atributos de la tarea:
  - ✓ **Entidad:** contiene el nombre del objeto.
  - ✓ **Evento:** tiene el nombre del evento que desencadena la tarea.

- ✓ **Posición:** sus atributos son los valores escalares que representan el desplazamiento del objeto en X, Y y Z.
- ✓ **Escala:** posee los valores escalares que representan cuánto tiene que escalarse el objeto en X, Y y Z.
- ✓ **Orientación:** tiene los valores escalares que representan el vector dirección hacia el que tiene que apuntar el objeto.
- ✓ **Tiempo:** contiene el valor de tiempo, en segundos, que tendrá la tarea para ejecutarse.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE AnimationEditor>
<AnimationEditor version="1.0">
    <task>
        <entityName name="Head"/>
        <events name="onLeftButtonPress"/>
        <positionTrans x="-150" y="1" z="0"/>
        <scaleTrans x="0" y="0" z="0"/>
        <orientationTrans w="1" x="0" y="0" z="0"/>
        <time value="2"/>
    </task>
    .
    .
    Otras tareas
    .
    .
```

Figura 27. Estructura del fichero XML.

Consideraciones Parciales

Se conformó un prototipo del software a partir de los requisitos planteados por el cliente y se describieron las clases que se utilizarán en la confección del mismo, con lo que se sientan las bases para la etapa de implementación.

CAPÍTULO 3. CONSTRUCCIÓN DE LA PROPUESTA DE SOLUCIÓN

Introducción

El objetivo de este capítulo es implementar la propuesta de solución elaborada anteriormente y comprobar que el sistema desarrollado cumple con cada requerimiento planteado por el cliente. Para ello, se llevan a cabo las tareas de ingeniería y se realizan las pruebas de aceptación.

4.1 Fase de implementación

Se realizan las tareas de ingeniería necesarias para materializar cada HU y se chequea el plan de iteraciones para saber si se ha afectado el mismo. El proceso de implementación se muestra en tres iteraciones, como se propuso en el capítulo anterior.

4.1.1 Iteración 1

No. de HU	Historias de Usuario	Tiempo de Implementación (semanas)	
		Estimación	Real
1	Crear tarea a un objeto.	2	2
2	Ejecutar las tareas de un objeto.	2	2
3	Listar las tareas de un objeto.	1	1

Tabla 16. Tiempo de implementación por HU de la iteración 1.

Historias de Usuario	Tareas por HU
Crear tarea a un objeto.	1. Transformar objeto mediante el manipulador. (Ver Tabla 18) 2. Calcular la transformación del objeto entre el estado inicial y final. (Ver Tabla 19) 3. Salvar tarea en el fichero XML. (Ver Tabla 20)

Ejecutar las tareas de un objeto.	1. Cargar la lista de tareas desde el fichero XML. (Ver Tabla 21) 2. Calcular el nuevo estado del objeto según el tiempo transcurrido. (Ver Tabla 22) 3. Actualizar el estado del objeto durante la ejecución de la tarea. (Ver Tabla 23)
Listar las tareas de un objeto.	1. Cargar la lista de tareas desde el fichero XML. (Ver Tabla 21) 2. Mostrar las tareas con sus propiedades. (Ver Tabla 24)

Tabla 17. Tareas por HU en la iteración 1.

Tarea de Ingeniería	
No. de tarea: 1	**No. de HU:** 1
Nombre de la tarea: Transformar objeto mediante el manipulador.	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 0.6
Fecha inicio: 21/2/2012	**Fecha fin:** 24/2/2012
Descripción: El desarrollador escoge el tipo de tarea a realizar y se muestra un manipulador, el que permite escalar, trasladar o rotar el objeto, en dependencia del manipulador escogido.	

Tabla 18. Tarea 1 de la HU 1.

Tarea de Ingeniería	
No. de tarea: 2	**No. de HU:** 1
Nombre de la tarea: Calcular la transformación del objeto entre el estado inicial y final.	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 0.6
Fecha inicio: 25/2/2012	**Fecha fin:** 28/2/2012
Descripción: Se calcula la diferencia de posición, escala y orientación entre los estados final e inicial del objeto luego de que el desarrollador lo haya transformado.	

Tabla 19. Tarea 2 de la HU 1.

Tarea de Ingeniería	
No. de tarea: 3	**No. de HU:** 1
Nombre de la tarea: Salvar tarea en el fichero XML.	

Tipo de tarea: Desarrollo.	**Puntos estimados:** 0.6
Fecha inicio: 29/2/2012	**Fecha fin:** 3/3/2012
Descripción: Guarda las propiedades y los valores de la tarea en el fichero XML según la estructura definida.	

Tabla 20. Tarea 3 de la HU 1.

Tarea de Ingeniería	
No. de tarea: 1	**No. de HU:** 2-3
Nombre de la tarea: Cargar la lista de tareas desde el fichero XML.	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 0.6
Fecha inicio: 4/3/2012	**Fecha fin:** 7/3/2012
Descripción: Se lee el fichero XML y se llena la lista con las tareas existentes.	

Tabla 21. Tarea 1 de la HU 2 y la HU 3.

Tarea de Ingeniería	
No. de tarea: 2	**No. de HU:** 2
Nombre de la tarea: Calcular el nuevo estado del objeto según el tiempo transcurrido.	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 0.6
Fecha inicio: 8/3/2012	**Fecha fin:** 11/3/2012
Descripción: Se realizan los cálculos necesarios para determinar el estado del objeto según el tiempo que ha transcurrido desde que se inició la tarea, teniendo en cuenta el tiempo total de la misma.	

Tabla 22. Tarea 2 de la HU 2.

Tarea de Ingeniería	
No. de tarea: 3	**No. de HU:** 2
Nombre de la tarea: Actualizar el estado del objeto durante la ejecución de la tarea.	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 0.6
Fecha inicio: 12/3/2012	**Fecha fin:** 15/3/2012

Descripción: Se actualiza el estado del objeto en cuanto a posición, escala y orientación, después de realizar los cálculos en la tarea que se está ejecutando.

Tabla 23. Tarea 3 de la HU 2.

Tarea de Ingeniería	
No. de tarea: 2	**No. de HU:** 3
Nombre de la tarea: Mostrar las tareas con sus propiedades.	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 0.5
Fecha inicio: 16/3/2012	**Fecha fin:** 18/3/2012
Descripción: Se muestran las propiedades y valores asociados a cada tarea.	

Tabla 24. Tarea 2 de la HU 3.

4.1.2 Iteración 2

No. de HU	Historias de Usuario	Tiempo de Implementación (semanas)	
		Estimación	Real
1	Modificar las propiedades de una tarea.	1	1
2	Eliminar tarea.	0.5	0.5

Tabla 25. Tiempo de implementación por HU de la iteración 2.

Historias de Usuario	Tareas por HU
Modificar las propiedades de una tarea.	1. Modificar valores de las propiedades. (Ver Tabla 27) 2. Actualizar el estado del objeto. (Ver Tabla 28)
Eliminar tarea.	1. Eliminar la tarea. (Ver Tabla 29) 2. Actualizar orden de ejecución de las tareas. (Ver Tabla 30)

Tabla 26. Tareas por HU en la iteración 2.

Tarea de Ingeniería	
No. de tarea: 1	**No. de HU:** 4
Nombre de la tarea: Modificar valores de las propiedades.	

Tipo de tarea: Desarrollo.	**Puntos estimados:** 0.5
Fecha inicio: 19/3/2012	**Fecha fin:** 21/3/2012
Descripción: Se introducen los valores deseados y se rescribe el fichero XML con los valores actuales.	

Tabla 27. Tarea 1 de la HU 4.

Tarea de Ingeniería	
No. de tarea: 2	**No. de HU:** 4
Nombre de la tarea: Actualizar el estado del objeto.	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 0.5
Fecha inicio: 22/3/2012	**Fecha fin:** 24/3/2012
Descripción: El objeto adopta los nuevos valores en cuanto a posición, escala y orientación.	

Tabla 28. Tarea 2 de la HU 4.

Tarea de Ingeniería	
No. de tarea: 1	**No. de HU:** 5
Nombre de la tarea: Eliminar la tarea.	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 0.25
Fecha inicio: 25/3/2012	**Fecha fin:** 26/3/2012
Descripción: Se selecciona la tarea y se elimina.	

Tabla 29. Tarea 1 de la HU 5.

Tarea de Ingeniería	
No. de tarea: 2	**No. de HU:** 5
Nombre de la tarea: Actualizar orden de ejecución de las tareas.	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 0.25
Fecha inicio: 26/3/2012	**Fecha fin:** 27/3/2012
Descripción: Se actualiza el orden de ejecución de las tareas y se reescribe el fichero XML.	

Tabla 30. Tarea 2 de la HU 5.

4.1.3 Iteración 3

No. de HU	Historias de Usuario	Tiempo de Implementación (semanas)	
		Estimación	Real
1	Reordenar las tareas.	0.5	0.5
2	Pre-visualizar tareas.	0.5	0.5

Tabla 31. Tiempo de implementación por HU de la iteración 3.

Historias de Usuario	Tareas por HU
Reordenar las tareas.	1. Cambiar orden de ejecución de las tareas. (ver Tabla 33)
Pre-visualizar tareas.	1. Mandar a ejecutar las tareas del objeto seleccionado. (ver Tabla 34)

Tabla 32. Tareas por HU en la iteración 3.

Tarea de Ingeniería	
No. de tarea: 1	**No. de HU:** 6
Nombre de la tarea: Cambiar orden de ejecución de las tareas.	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 0.5
Fecha inicio: 28/3/2012	**Fecha fin:** 30/3/2012
Descripción: Se cambia el orden de ejecución de las tareas y se reescribe el fichero XML.	

Tabla 33. Tarea 1 de la HU 6.

Tarea de Ingeniería	
No. de tarea: 1	**No. de HU:** 7
Nombre de la tarea: Mandar a ejecutar las tareas del objeto seleccionado.	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 0.5
Fecha inicio: 31/3/2012	**Fecha fin:** 2/4/2012
Descripción: Permite mostrar una vista previa de todas las tareas del objeto seleccionado.	

Tabla 34. Tarea 1 de la HU 7.

4.2 Descripción del flujo de datos en el sistema

La aplicación consta de dos componentes principales: el Editor de tareas y el Reproductor de tareas. A continuación se describen los elementos que reciben cada uno y el resultado que devuelven (ver Figura 28 y Figura 29).

Figura 28. Flujo de datos del "Editor de tareas".

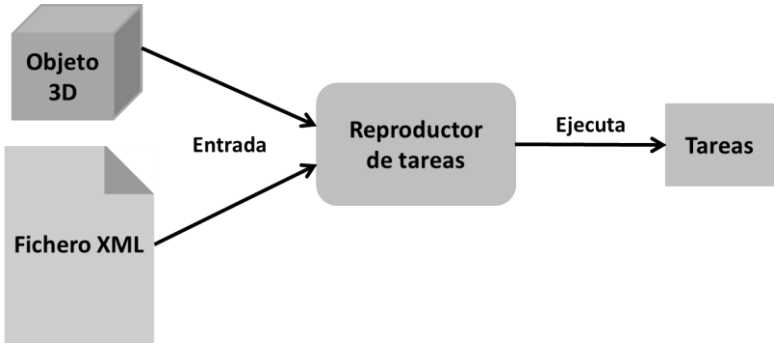


Figura 29. Flujo de datos del "Reproductor de tareas".

4.3 Fase de pruebas

XP propone la realización de las pruebas unitarias y las pruebas de aceptación. Las primeras son desarrolladas por el programador para verificar que el código funciona, mientras que las segundas son utilizadas para probar que la aplicación contiene las funcionalidades deseadas por el cliente.

4.3.1 Pruebas de aceptación

Las pruebas de aceptación son pruebas de caja negra que se crean a partir de las historias de usuario y significan el grado de satisfacción del cliente con el producto desarrollado. Para la presente aplicación, se realizará un caso de prueba por cada HU.

Caso de Prueba de Aceptación	
Código: H1_P1	**HU:** 1
Nombre: Crear tarea a un objeto.	
Descripción: Después de seleccionar un objeto, para crear una tarea se necesita seleccionar el evento que la desencadenará, introducir el tiempo de duración y realizar la transformación deseada. La tarea creada se salva en un fichero XML.	
Condiciones de Ejecución: 1. Seleccionar un objeto. 2. Presionar la tecla que inicia el editor de tareas.	
Entrada/Pasos de Ejecución: 1. Seleccionar el evento. 2. Seleccionar la herramienta correspondiente al tipo de tarea a realizar. 3. Realizar la transformación. 4. Introducir el tiempo de ejecución de la tarea. 5. Presionar el botón “Crear tarea”.	
Resultado Esperado: La tarea debe guardarse en el fichero y mostrarse automáticamente en el panel de tareas.	
Resultado Obtenido: La tarea se mostró en el panel de tareas. (Ver Anexo 2)	
Evaluación de la Prueba: Satisfactoria.	

Tabla 35. Prueba de aceptación “Crear tarea a un objeto”.

Caso de Prueba de Aceptación	
Código: H2_P2	**HU:** 2
Nombre: Ejecutar las tareas de un objeto.	
Descripción: Cuando el usuario realice una acción sobre un objeto, se ejecutarán las tareas asignadas al objeto que cumplan con ese evento. Además, las tareas serán ejecutadas en el orden en que aparecen en el fichero.	

Condiciones de Ejecución: El objeto debe contener tareas que respondan a ese evento.
Entrada/Pasos de Ejecución: Ejecutar un evento sobre un objeto.
Resultado Esperado: Se ejecutarán las tareas que correspondan al evento.
Resultado Obtenido: Después de realizar una acción sobre un objeto, se ejecutaron todas las tareas que le fueron creadas. (Ver Anexo 3)
Evaluación de la Prueba: Satisfactoria.

Tabla 36. Prueba de aceptación "Ejecutar las tareas de un objeto".

Caso de Prueba de Aceptación	
Código: H3_P3	**HU:** 3
Nombre: Listar las tareas de un objeto.	
Descripción: Se listarán las tareas de un objeto de acuerdo a un evento. Además, las tareas serán listadas en el mismo orden en que aparecen en el fichero.	
Condiciones de Ejecución: Seleccionar un objeto que contenga tareas.	
Entrada/Pasos de Ejecución: Seleccionar el evento.	
Resultado Esperado: Se mostrará una tabla con las propiedades y valores de las tareas existentes.	
Resultado Obtenido: Se mostró una tabla con las propiedades y valores de las tareas existentes (ver Anexo 4).	
Evaluación de la Prueba: Satisfactoria.	

Tabla 37. Prueba de aceptación "Listar las tareas de un objeto".

Caso de Prueba de Aceptación	
Código: H4_P4	**HU:** 4
Nombre: Modificar las propiedades de una tarea.	
Descripción: Se selecciona una tarea y se modifica la propiedad o propiedades deseadas, introduciendo los valores requeridos.	
Condiciones de Ejecución: Seleccionar la propiedad de la tarea a modificar.	

Entrada/Pasos de Ejecución: Introducir los valores deseados.
Resultado Esperado: Se modificarán los valores de la propiedad y se actualizará automáticamente el fichero que contiene la tarea.
Resultado Obtenido: Se modificaron los valores y se actualizó el fichero de la tarea (ver Anexo 5).
Evaluación de la Prueba: Satisfactoria.

Tabla 38. Prueba de aceptación "Modificar las propiedades de una tarea".

Caso de Prueba de Aceptación	
Código: H5_P5	**HU:** 5
Nombre: Eliminar tarea.	
Descripción: Se selecciona la tarea y se elimina.	
Condiciones de Ejecución: El objeto debe contener tareas.	
Entrada/Pasos de Ejecución: 1. Seleccionar la tarea. 2. Dar clic en el botón "Eliminar".	
Resultado Esperado: La tarea se eliminará del fichero y del panel de tareas.	
Resultado Obtenido: Se eliminó la tarea (ver Anexo 6).	
Evaluación de la Prueba: Satisfactoria.	

Tabla 39. Prueba de aceptación "Eliminar tarea".

Caso de Prueba de Aceptación	
Código: H6_P6	**HU:** 6
Nombre: Reordenar las tareas.	
Descripción: Cambia el orden en que se ejecutan las tareas.	
Condiciones de Ejecución: El objeto debe contener, al menos, dos tareas.	

Entrada/Pasos de Ejecución: 1. Seleccionar la tarea. 2. Dar clic en el botón “Subir tarea” para que se ejecute antes o presionar el botón “Bajar tarea” para que se ejecute después.
Resultado Esperado: Las tareas se mostrarán y se ejecutarán en el orden deseado.
Resultado Obtenido: Las tareas se mostraron y se ejecutaron en el orden deseado. (Ver Anexo 7)
Evaluación de la Prueba: Satisfactoria.

Tabla 40. Prueba de aceptación “Reordenar las tareas”.

Caso de Prueba de Aceptación	
Código: H7_P7	**HU:** 7
Nombre: Pre-visualizar tareas.	
Descripción: Después de que el desarrollador crea las tareas, puede pre-visualizarlas y observar cómo se verán en la aplicación final.	
Condiciones de Ejecución: 1. Crear al menos una tarea. 2. Seleccionar un evento.	
Entrada/Pasos de Ejecución: Dar clic en el botón “Vista previa”.	
Resultado Esperado: Se ejecutarán todas las tareas de un objeto.	
Resultado Obtenido: Se ejecutaron todas las tareas del objeto. (Ver Anexo 8)	
Evaluación de la Prueba: Satisfactoria.	

Tabla 41. Prueba de aceptación “Reordenar las tareas”.

Consideraciones Parciales

Se implementó el sistema y se efectuaron las pruebas correspondientes, una por cada HU, para comprobar que cumple con los requisitos planteados, por lo que puede ser utilizado en el desarrollo de los futuros laboratorios virtuales.

CONCLUSIONES

Con la realización de la presente investigación, se obtuvo un módulo que permite crear las tareas de comportamiento, de forma visual, a los objetos en las escenas de los laboratorios virtuales. La herramienta automatiza el proceso de asignar comportamientos a los objetos, pues el programador no necesita escribir código, ni reiniciar tras cada cambio realizado. Además, la aplicación posibilita realizar diferentes acciones, como abrir puertas y gavetas, lo que incrementa el dinamismo de las escenas.

RECOMENDACIONES

Para dar continuidad a la investigación, se recomienda:

- Añadir las tareas de cambio de estado.
- Incluir otros eventos, como los de teclado y tiempo.

REFERENCIAS

1. **Iznaga Benítez, Dr. Arsenio M. and Pérez Mallea, MSc. Ivan.** *Fundamentos de la Gráfica por Computadora.* La Habana : Félix Varela, 2006. p. 222. 959-07-0213-9.

2. **Academia de la Lengua Española.** *Diccionario de la Lengua Española.* [Online] http://buscon.rae.es/.

3. **Isdale, Jerry.** *What Is Virtual Reality.* 1998.

4. **IITAP.** *Informe de la reunión de expertos sobre laboratorios virtuales.* París : UNESCO, 2000. p. 64.

5. **Vila, Cristóbal.** *Introducción a la Animación 3D.* [Online] http://www.etereaestudios.com/training_img/intro_3d/intro_3d.htm.

6. **García, Oscar and Guevara, Alex.** *Introducción a la Programación Gráfica con OpenGL.* Barcelona : Escuela Técnica Superior de Ingienería Electrónica e Informática La Salle, 2004. p. 81.

7. **SENSE8 Corporation.** *Manual de WorldToolKit.* s.l. : Engineering Animation, 1999. p. 1064.

8. **Dassault Systemes.** *Sitio oficial de 3DVIA Studio.* [Online] http://www.3ds.com/products/3dvia/3dvia-studio/welcome/.

9. **Fernández Aedo, C. Raúl and Delavaut Romero, Martín E.** *Educación y Tecnología: Un binomio excepcional.* [ed.] Martín Delavaut. p. 247. 9872323038.

10. **Canós, José H., Letelier , Patricio and Penadé, María del Carmen .** *Métodologías Ágiles en el Desarrollo de Software.* s.l. : Universidad Politécnica de Valencia, 2003.

11. **Escobar Pompa, Mailen Edith and Ortiz Azahares, Leydis Andis.** *Análisis y Diseño de un Nodo Virtual de Procesos.* Universidad de las Ciencias Informáticas. La Habana : s.n., 2008. p. 121, Trabajo de Diploma.

12. **Joyanes Aguilar, Luis.** *Fundamentos de Programación, Algoritmos, Estructura de Datos y Objetos.* Tercera. s.l. : McGraw-Hill, 2003. p. 1012. 8-448-13664-0.

13. **Universidad de Oviedo.** *Sitio web de la E.U. de Ingeniería Técnica Informática de Oviedo.* [Online] http://petra.euitio.uniovi.es/~i1667065/HD/documentos/Entornos%20de%20Desarrollo%20Integrado.pdf.

14. **Quintana López, Alfredo and Alfonso Monteagudo, Yasmany.** *Motor de Render Genérico.* La Habana : s.n., 2010. Trabajo de Diploma.

15. *Markup Systems and the Future of Scholarly Text Processing.* **Coombs, James H., Renear, Allen H. and DeRose, Steven J.** 11, New York : Communications of the ACM, 1987, Vol. 30. 0001-0782.

16. **Manetta , C. and Blade, R.** *Glossary of Virtual Reality Terminology.* 2. 1995. Vol. 1.

17. **Beck , Kent and Fowler , Martin .** *Planning Extreme Programming.* 1. 2000. p. 160. ISBN: 0-201-71091-9.

18. **Visual Paradigm International.** *Sitio oficial de Visual Paradigm.* [Online] https://www.visual-paradigm.com/.

19. **Stroustrup, Bjarne.** *El lenguaje de programacion C++. Segunda edición.* Wilmington : Addison-Wesley, 1993. 0-201-60104-4.

20. **Nokia Corporation.** *Sitio oficial de Qt.* [Online] http://qt.nokia.com/.

21. **Ogre Team.** Sitio oficial de OGRE. [Online] http://www.ogre3d.org.

22. **Dick, Kevin.** *XML: a manager's guide.* Segunda. s.l. : Addison-Wesley, 2003. p. 298. 0-201-77006-7.

23. **Larman, Craig.** *UML y Patrones. Una introducción al análisis y diseño orientado a objetos y al proceso unificado.* Segunda. s.l. : Prentice Hall, 2006. p. 590. 8420534382.

24. **Sommerville, Ian.** *Ingeniería del software.* [ed.] Miguel Martín-Romo. [trans.] María I. Alfonso Galipienso y otros. Séptima. Madrid : Pearson Educación, 2005. p. 712. ISBN: 84-7829-074-5.

25. **Duch i Gavaldà, Jordi and Tejedor Navarro, Heliodoro.** *Introducción a los videojuegos.* s.l. : UOC, 2008. pp. 65-66. P07/B0053/02685.

26. **Booch, Grady, Rumbaugh, James and Jacobson, Ivar.** *El Lenguaje Unificado de Modelado.* 2000.

27. **Jacobson, Ivar, Booch, Grady and Rumbaugh, James.** *El Proceso Unificado de Desarrollo de Software.* Madrid : s.n., 2000. p. 435.

28. **Escribano, Gerardo Fernández.** *Introducción a Extreme Programming.* 2002.

29. **Kniberg, Henrik.** *Scrum and XP from the Trenches. How we do Scrum.* 2.1. 2007. p. 168. 978-1-4303-2264.

30. **Rational Software Corporation.** *Rational Suite® Tutorial.* 2001A.04.00 . 2001. 800-024444-000.

31. **Roosendaal, Ton and Selleri, Stefano.** *Guía de Blender. La suit abierta de creación 3D.* 2008.

32. **SENSE8 Corporation.** *Quick Reference Guide WorldToolKit.* 1998. p. 62.

33. **Döllner, Jürgen and Hinrichs, Klaus.** *A Generic Rendering System.* 2002. pp. 6-7. Vol. 8. 1077-2626.

34. **Piattini, M., et al., et al.** *Análisis y diseño detallado de aplicaciones informáticas de gestión.* Madrid : Ra-Ma, 1996. 8478972331.

35. **Letelier, Patricio and Penadés, Mª Carmen.** *Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP).* Valencia : Universidad Politécnica de Valencia. p. 2.

36. **Beck, Kent.** *Extreme Programming Explained. Embrace Change.* s.l. : Pearson Education, 1999. 0201616416.

37. **Wyke, R. Allen, Rehman, Sultan and Leupen, Brad.** *XML Programming (Core Reference).* Redmond : Microsoft Press, 2002. p. 704. 0-7356-1185-8.

38. **Microsoft Corporation.** *¿Qué es Visual Studio?* [Online] http://www.microsoft.com/visualstudio/latam.

39. **The OGRE Team.** *Ogre Wiki - Ogitor.* [En línea] http://www.ogre3d.org/tikiwiki/Ogitor.

40. **Conitec Datensysteme GmbH.** *Sitio oficial de 3DGameStudio.* [Online] http://www.3dgamestudio.com/.

41. **Crystal Space Team.** *Sitio oficial de Crystal Space.* [Online] http://www.crystalspace3d.org/main/Main_Page.

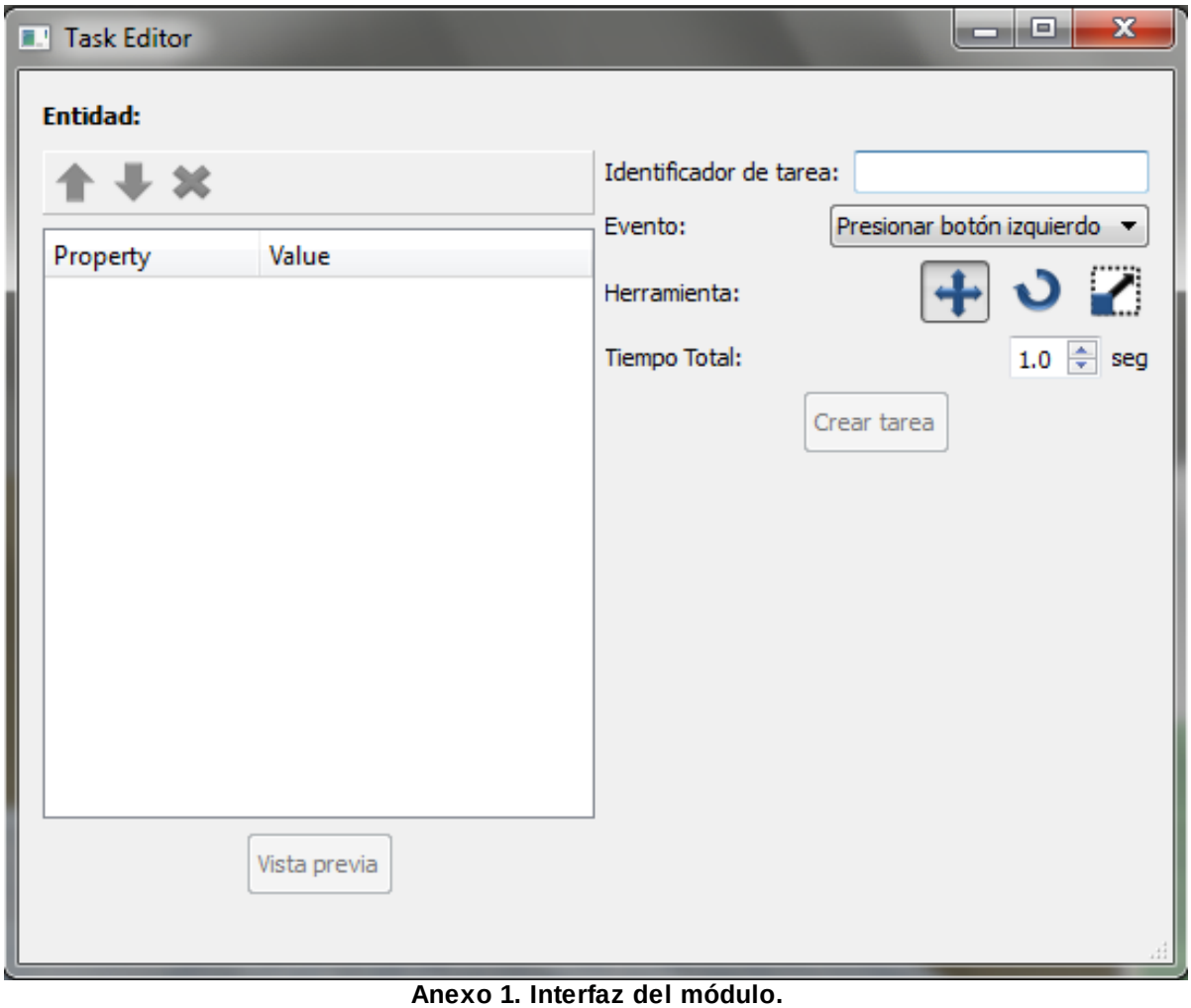
42. **The OGRE Team.** *Sitio web de Ogre3D.* [Online] http://www.ogre3d.org.

43. **Fowler, Martin.** *Is Design Dead?* [Online] http://www.martinfowler.com/articles/designDead.html.

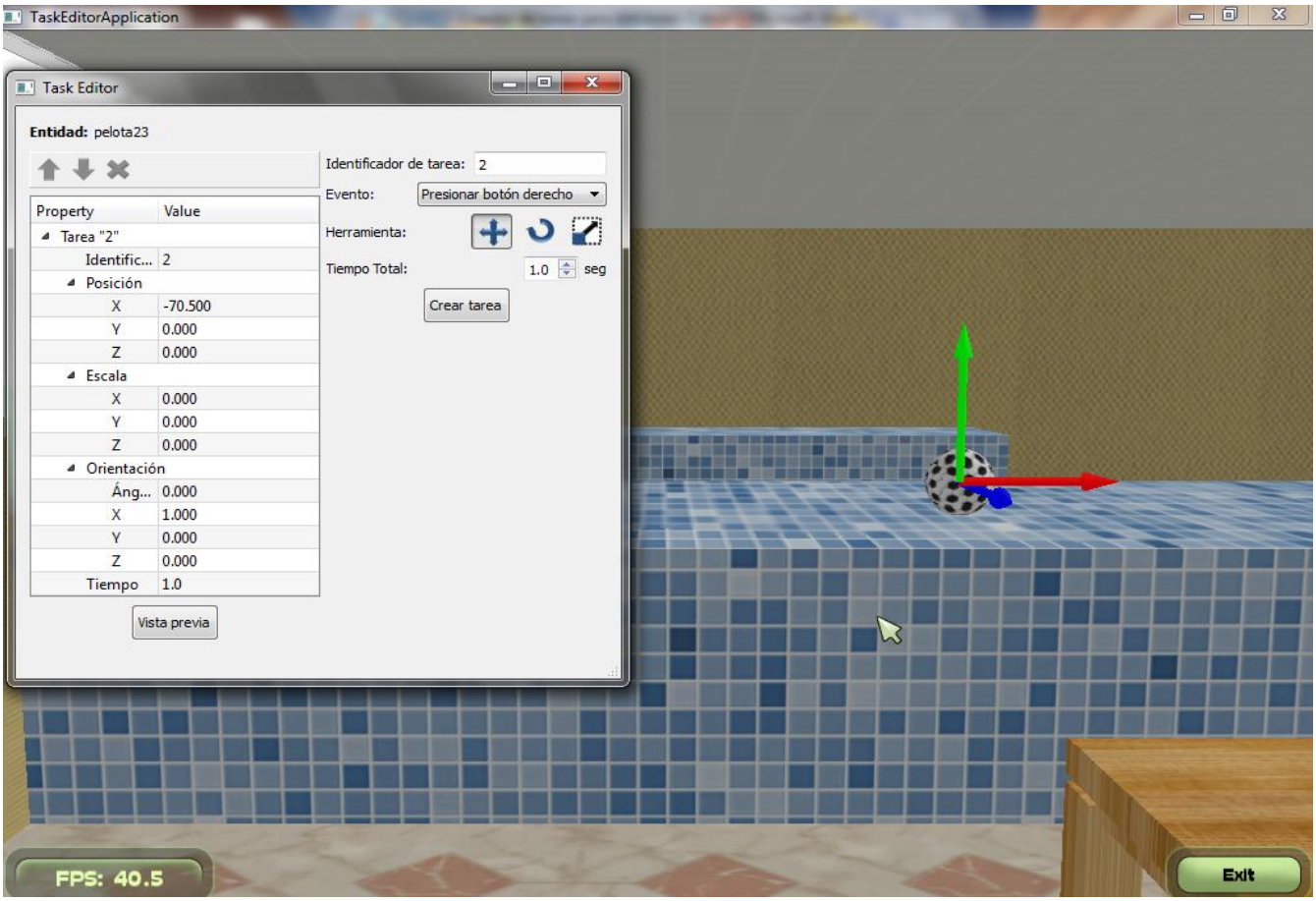
44. **Mendoza, César.** *Animación.* [Online] http://www.mendozajullia.com/papers/Animacion15marzo.pdf.

45. **Racero, Omar.** *Vectores.* [Online] http://www.monografias.com/trabajos35/vectores/vectores.shtml.

ANEXOS

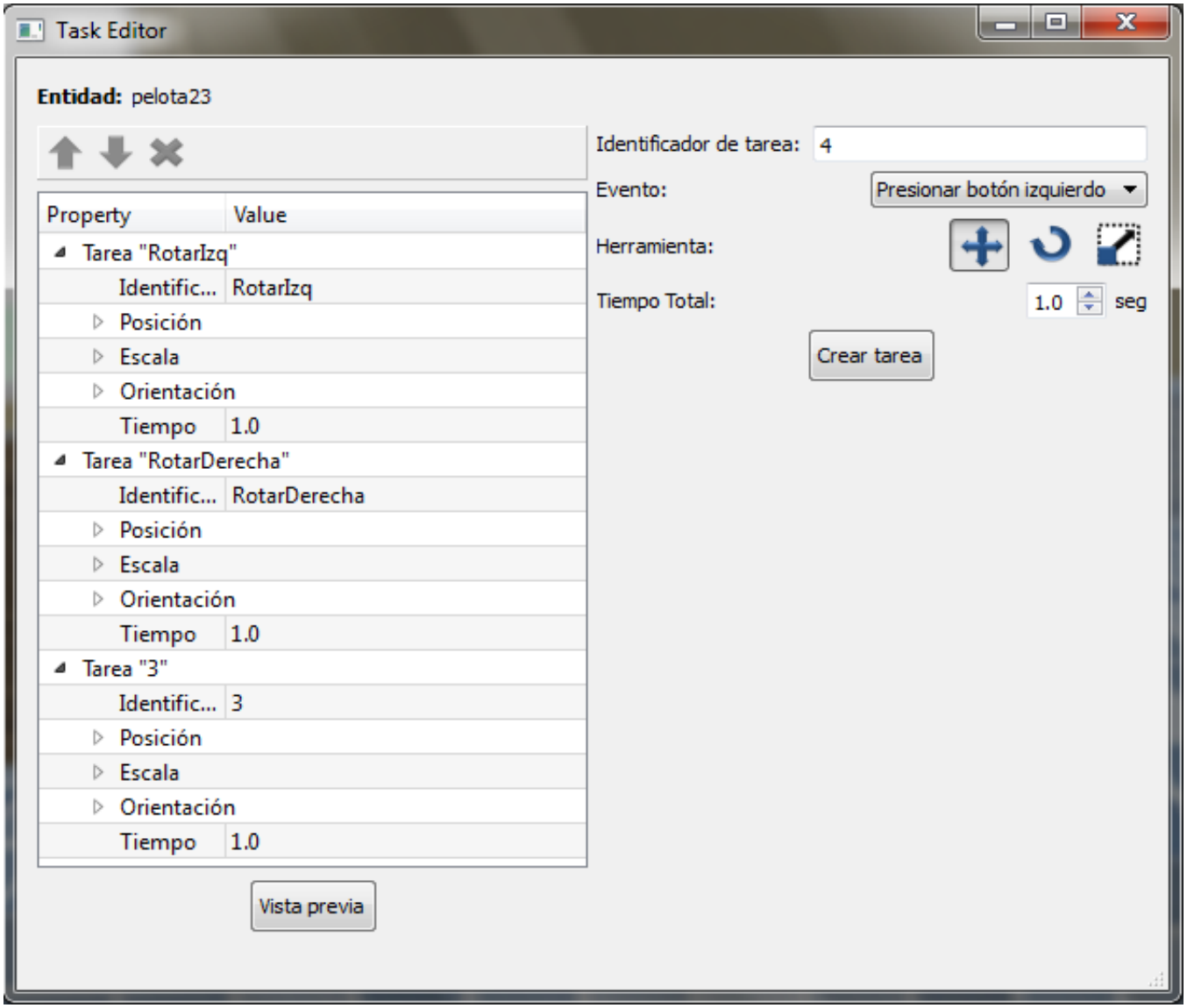


Anexo 1. Interfaz del módulo.

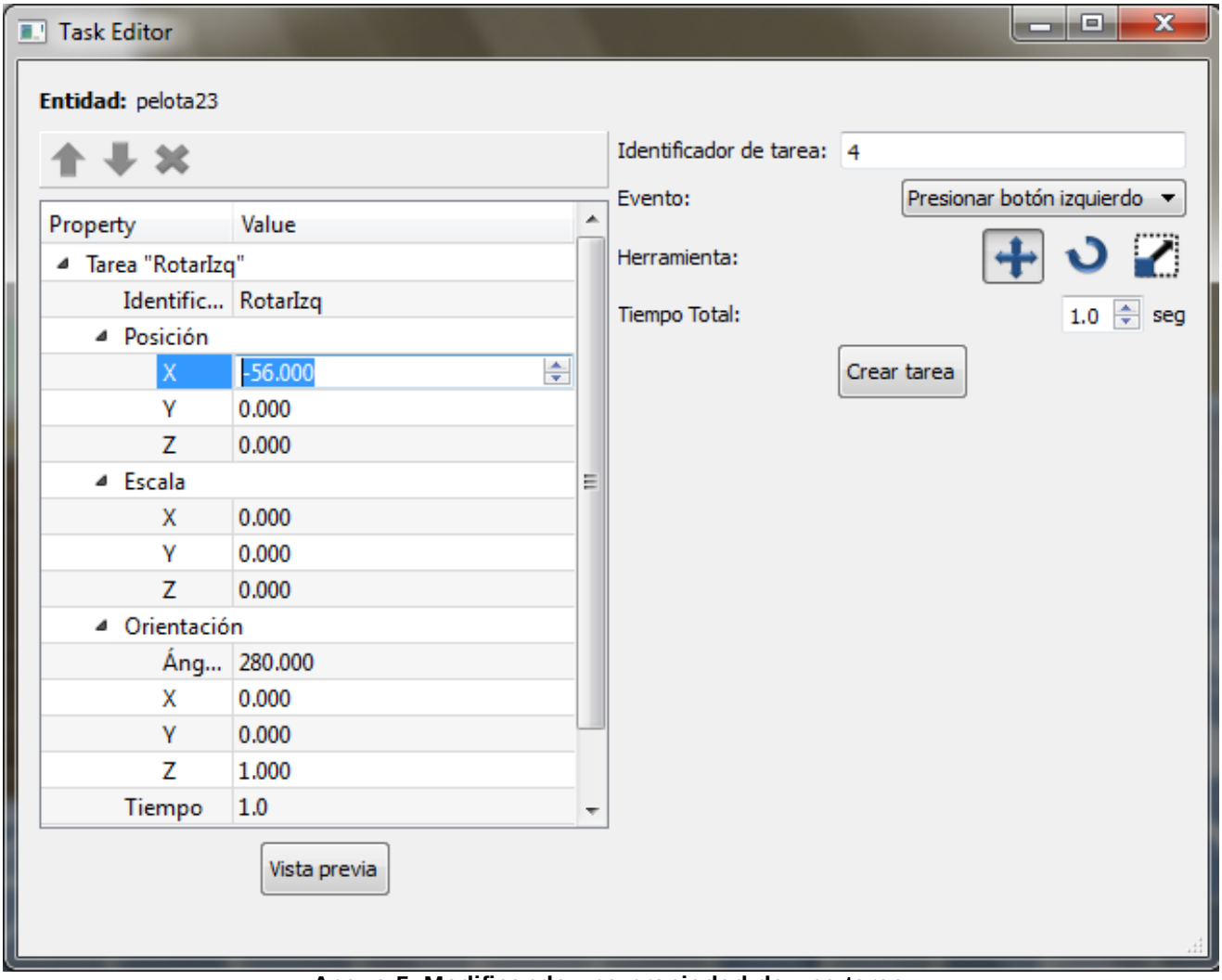


Anexo 2. Objeto con tarea creada.

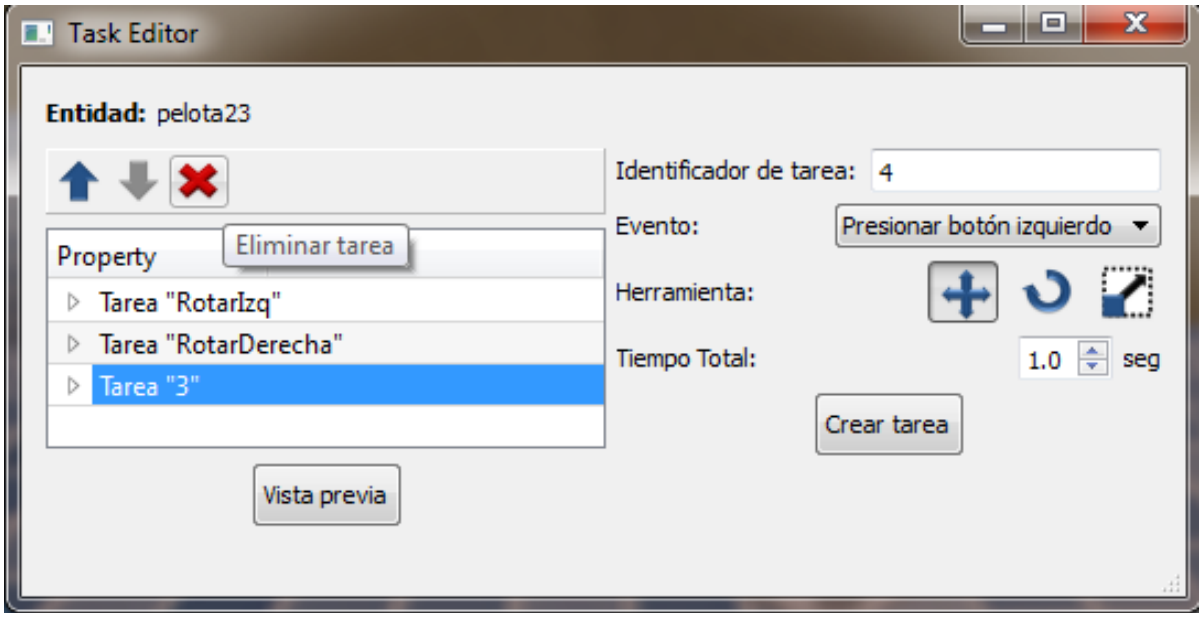
Anexo 3. Objeto ejecutando una tarea.



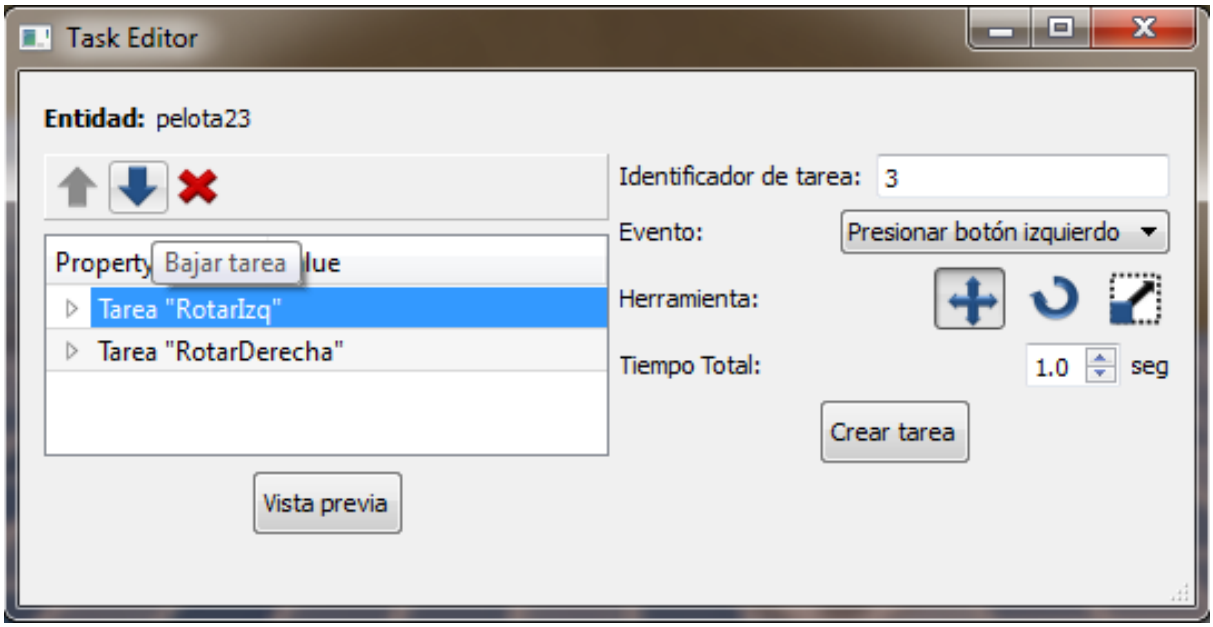
Anexo 4. Panel con el listado de tareas creadas.



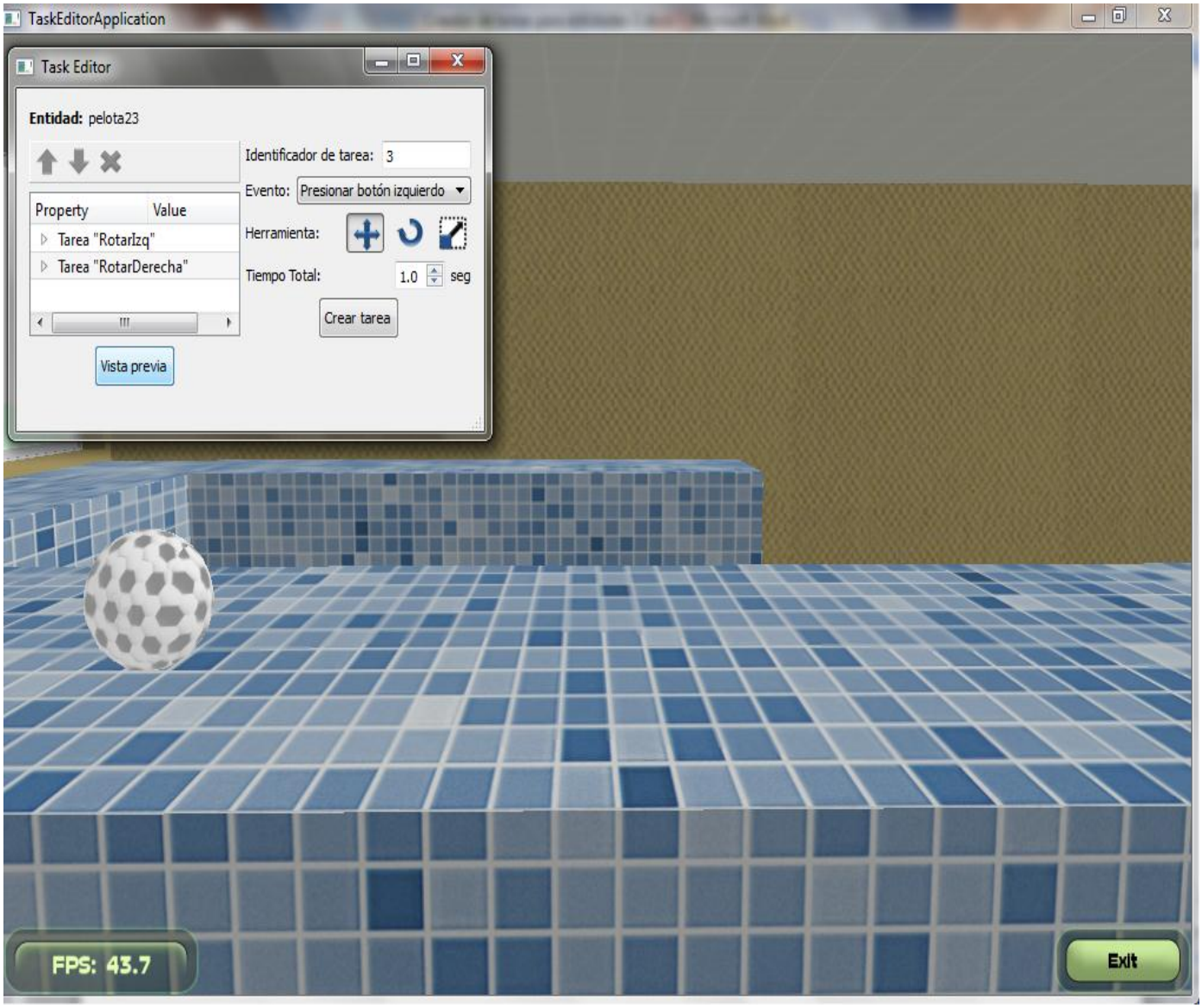
Anexo 5. Modificando una propiedad de una tarea.



Anexo 6. Eliminando una tarea.



Anexo 7. Reordenar las tareas.



Anexo 8. Pre-visualizando las tareas.

GLOSARIO DE TÉRMINOS

API gráfica: Es una interfaz proporcionada por una biblioteca o sistema para poder acceder a las funciones de esta, que no incluye información sobre la implementación de la biblioteca.

Código abierto: es el término con el que se conoce al software distribuido y desarrollado libremente. Además, tiene un punto de vista más orientado a los beneficios prácticos de compartir el código que a las cuestiones morales que defiende el Software libre.

Coordenadas Cartesianas: Un sistema de coordenadas utiliza una o más coordenadas (números) para determinar unívocamente la posición de un punto o de otro objeto geométrico. Si es bidimensional tendrá dos ejes ortogonales (x, y), en caso de los entornos 3D es tridimensional, es decir tres ejes (x, y, z).

***Framework*:** En el desarrollo de software, es una estructura de soporte en la que otro proyecto de software puede ser organizado y desarrollado. Puede incluir soporte de programas, bibliotecas, un lenguaje interpretado, entre otros sistemas para ayudar a desarrollar y unir los diferentes componentes de un proyecto.

Licencia *Shareware*: es una forma de distribuir el software, en la que el usuario puede evaluar de forma gratuita el producto, pero con limitaciones en el tiempo de uso o en algunas de las formas de uso.

Render: Se define como la trasformación de la información de una representación a otra. En gráficos por computadoras, denota el proceso de sintetizado de imágenes, es decir, la traducción de la geometría controlada por atributos gráficos, al medio de la imagen.

Software libre: es la denominación del software que respeta la libertad de los usuarios sobre un producto adquirido y, por tanto, una vez obtenido puede ser usado, copiado, estudiado, modificado, y redistribuido libremente.