ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

Ingeniería Técnica en Informática de Sistemas

BIBLIOTECA OPENGL 2D

Realizado por
José Manuel Barroso Galindo
75779778E

Dirigido por
Pablo Neira Ayuso

Departamento
Lenguajes y Sistemas Informáticos

Sevilla, Noviembre del 2010

Índice

Capítulo 1: Introducción

En este capítulo encontraremos :

– Una **perspectiva histórica** sobre la computación gráfica, llegando a discutir el problema que hay en la actualidad con la representación de gráficos 2D.
– El **propósito general de la biblioteca**, una solución ante el problema de los gráficos 2D.
– La importancia de **dotar a la biblioteca de mayor interés** mediante funcionalidades adicionales al simple procesamiento gráfico.
– La **motivación** por la cual he elegido hacer esta biblioteca como Proyecto final de carrera.
– Los **objetivos** que perseguiré durante el desarrollo de la biblioteca.
– Un análisis sobre algunas de las **bibliotecas similares** a ésta, para comprender sus puntos débiles e intentar sobreponerme a ellos. .

1.1 Aprovechando la Computación Gráfica : Perspectiva Histórica.

1.1.1 Los inicios

Desde los primeros ordenadores, que representaban el resultado de su tarea computacional mediante destellos de bombillas que se encendían y apagaban, o mediante limitadas impresiones en papel continuo, hasta hoy, la evolución tecnológica ha sido asombrosa. Poco tardó en aparecer como medio para mostrar la información al usuario el primer monitor, un periférico que hoy en día va ligado de manera inseparable al concepto de ordenador. Pero la pronta llegada del monitor no significa que desde entonces hayan habido pocos cambios en la manera de generar las imágenes para dicho dispositivo. Todo lo contrario. Las limitadas representaciones de carácteres monocromos, dieron paso a la libre manipulación de los píxeles[1] con tres componentes de color, que conformaban imágenes a resoluciones[2] cada vez mayores.

El enlace entre la CPU[3] y el monitor en un primer momento empezó siendo un adaptador de vídeo que contenía poco más que el búfer de memoria que se debía volcar al monitor y una interfaz digital que adaptaba la señal a una salida analógica. Con el tiempo se tuvo que convertir en una unidad de procesamiento independiente, contando por tanto con muchos más componentes digitales. Dicha unidad, llamada GPU[4], era necesaria para llevar a cabo las operaciones gráficas – en un principio de apoyo - cada vez más exigentes, y que hasta entonces eran realizadas exclusivamente por la CPU. El desarrollo de las GPU, fue esencial no sólo para mejorar el rendimiento de las CPU, que a partir de entonces se liberarían de gran parte del costoso trabajo gráfico. También lo fue para posibilitar una especialización del hardware, que un procesador de propósito más general no se podía permitir. Es por ello, que en adelante la capacidad gráfica evolucionó a un ritmo mucho más acelerado de lo que venía siendo normal hasta ese momento.

1.1.2 La victoria de las tres dimensiones

Ciertas tareas, como procesadores de texto, clientes de mensajería o en general, la mayoría de aplicaciones de escritorio, cuentan típicamente con una representación visual en dos dimensiones, siendo ello más que suficiente para aportar información al usuario. Otras sin embargo, como simuladores de vuelo, exigen una manera de mostrar los datos de manera más fiel a la realidad. Debido a que virtualizar un espacio 3D y representarlo gráficamente, es una tarea bastante exigente computacionalmente, inicialmente sufrió un lento desarrollo.

1 Píxel : Cada celda que conforma la superficie de la pantalla. Un píxel sólo puede emitir un único color. La combinación de muchos de ellos es lo que da la sensación de representar una imagen.

2 Resolución de Imagen : Indica cuánto detalle puede observarse en una imagen. A mayor resolución, mayor nitidez de imagen, al estar definida por un mayor número de pixels.

3 **C**entral **P**rocessing **U**nit : Unidad central de procesamiento.

4 **G**raphics **P**rocessing **U**nit : Unidad de procesamiento Gráfico.

Pero con el paso de los años, tras superar numerosas barreras tecnológicas, surgieron nuevas GPU pensadas específicamente para soportar la renderización 3D. Dicha clase de GPU recientemente se popularizó para su uso en ordenadores personales debido a la demanda de videojuegos más realistas por parte de los consumidores. Como consecuencia, el mercado se llenó de novedosas GPU's especialmente diseñadas para el tratamiento 3D, y se olvidó de dar soporte a la aceleración 2D en los mismos productos.

1.1.3 **El problema de las 2D**

Hoy en día, tras el éxito comercial que trajeron las 3D al mercado de las Tarjetas Gráficas[5] durante la década de los 90, las GPU se han convertido en procesadores muy potentes, con un alto grado de paralelismo y de segmentación, y dedicados a realizar operaciones en punto flotante. Bajo esta circunstancia nos alejamos un poco de la funcionalidad originaria de las unidades gráficas que procesaban imágenes planas, y es por ello que en muchas ocasiones es común encontrarse como la manipulación de entornos gráficos 2D, lo lleva a cabo la CPU y no la GPU.

Dicha situación además choca con una tendencia surgida en los últimos años que procura un mayor refinamiento visual de las interfaces gráficas, por ejemplo, en sistemas operativos. Las interfaces pioneras en ese sentido han acaparado la atención de los usuarios, abriendo una nueva vía rentable para la innovación en interfaces visuales. De ahí, que haya actualmente un interés renovado en tratar de solucionar esta situación donde la aceleración 2D apenas se utiliza. Aunque por el momento no hay muy buenas herramientas para lograr tal cometido.

1.2 La Biblioteca. Su propósito general

Este proyecto busca servir de apoyo al usuario, para realizar una aplicación **interactiva**[6] y de **tiempo real**[7], que aproveche la GPU de manera eficaz para representar por pantalla escenas 2D. Para ello, se ofrecerá un marco de trabajo preparado para tratar imágenes con la mayor libertad posible. Que estará integrado en un sistema preparado para procesar los estímulos externos que llamamos eventos.

El secreto de aprovechar eficazmente una GPU actual - típicamente enfocada para el renderizado 3D - cuando se trata de imágenes planas, consiste en lo siguiente. Se han de tratar las imágenes como texturas de objetos pertenecientes a un espacio

5 Tarjeta Gráfica : Adaptador de vídeo que incluye la GPU, que proporciona la salida que se usa el monitor, y que se conecta directamente a la placa base del ordenador.

6 Aplicación Interactiva : Aquella pensada para que tenga que el usuario tenga un papel activo en si misma, durante su ejecución.

7 Aplicación de Tiempo Real : Aquella que se propone ofrecer respuestas al usuario en un tiempo reducido.

virtual tridimensional. Dicho espacio virtual es recreado gracias a una API[8] 3D - como lo puede ser OpenGL, la API que usaremos en este proyecto – de tal manera, que si le aplicamos las proyecciones adecuadas, el resultado obtenido puede ser visualmente idéntico al que hubiéramos logrado ahorrándonos el tratamiento 3D que se ha comentado. Por tanto, el factor diferenciador entre el procesamiento puramente 2D que lleva a cabo la CPU, y el que realizamos mediante a una adaptación a un entorno 3D a su vez, tratado por la GPU, es únicamente un reparto diferente de las tareas de renderización donde la solución que cuenta con una GPU lógicamente realiza su propósito de una manera mucho más rápida.

> ¿Qué es OpenGL?
>
> OpenGL es una especificación abierta muy popularizada. Cuenta con numerosas implementaciones enfocadas a usar el hardware gráfico para la gran mayoría de las plataformas existentes en el mercado. Hoy, incluso en el desarrollo de aplicaciones 2D, es una de las mejores opciones si se desea tener una buen rendimiento visual.

Resumiendo. Este proyecto afronta la tarea de aprovechar la GPU de manera eficaz para renderizar – representar por la pantalla – escenas 2D, y para ello voy a emplear OpenGL, una API 3D ya existente que nos facilita el acceso a las opciones propias de la tecnología gráfica moderna.

1.3 Dotando de más atractivo a la Biblioteca.

Si una herramienta de este tipo se limitara a las tareas de renderización más simples, no sería una herramienta completa, y por tanto no habría usuarios dispuestos a usarla. Para incrementar el potencial gráfico de esta biblioteca, además se planean añadir diversas primitivas gráficas – operaciones de dibujado en pantalla – y efectos visuales que dotarán de mayor interés a la biblioteca.

Por otro lado, dejando al margen el concepto gráfico de la biblioteca, también se planea darle por otros medios más valor añadido. Se buscará ofrecer mecanismos de control que suelan ser necesarios en el tipo de aplicaciones que son desarrolladas a partir de una biblioteca como esta. Es decir, nos centraremos en ofrecer soluciones de cara al desarrollo de aplicaciones interactivas[9] usando la biblioteca SDL[10] como apoyo.

El resultado es una biblioteca de video 2D genérica, usada por aplicaciones interactivas tales como un videojuego o el entorno de escritorio de un sistema

8 **A**pplication **P**rogramming **I**nterface : Interfaz de programación de aplicaciones. Se llama así a un conjunto de funciones que ofrece cierta biblioteca para desarrollar software.

9 Aplicación Interactiva : Aquella que racciona a una serie de estímulos provenientes del usuario.

10 **S**imple **D**irectMedia **L**ayer : Una biblioteca que ofrece mecanismos de control de eventos de E/S y otras funciones multimedia, cuyo punto fuerte radica en ser muy portable.

operativo, y orientada a frames[11]. Es decir, el ritmo de ejecución de la aplicación se ve determinado por la tasa de frames por segundo elegida por el desarrollador, siendo por tanto dependiente la etapa lógica de la aplicación de la etapa de renderización.

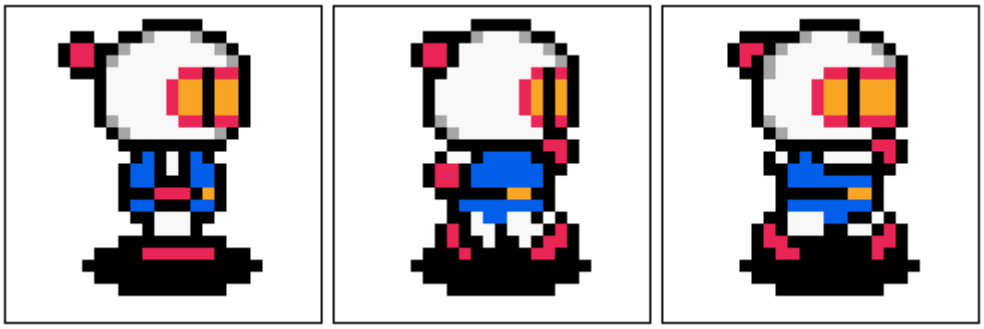


Figura 1.1 – Cuadro de animación.

> ¿Para qué sirve orientar la Biblioteca a Frames?
>
> Entendiendo el progreso de la aplicación como una sucesión de frames, es más intuitivo cualquier tarea de animación gráfica ya que la separación entre pasos de animación acompaña al ritmo de ejecución de la aplicación.

1.4 Objetivos

• Crear una biblioteca gráfica multiplataforma, probada en Windows y GNU/Linux y lista para su uso directo con C++.

• Renderizar escenas 2D haciendo uso de la aceleración por hardware – tarjeta gráfica – de una manera eficiente – sin malgastar recursos.

• Aprovechar las facilidades que nos da el espacio virtual 3D propio de OpenGL para la elaboración de diferentes efectos gráficos.

• Integrar la biblioteca con una serie de plantillas para establecer un marco de trabajo más amplio, y que abarque otra serie de funcionalidades relacionadas con la parte gráfica.

11 Frame: Un estado concreto y estático de la pantalla recibe éste nombre. Cada vez que se produce un cambio de frame, se identifica con el comiento y finalización de una etapa de la ejecución

1.5 Motivación Personal

Durante años he experimentado en mi tiempo libre con bibliotecas o lenguajes de programación que tenían como propósito facilitar la tarea de desarrollar un videojuego. Desde mi primer contacto con con estas herramientas, de la mano de Div Games Studio[12], he sido consciente como el desarrollo de un videojuego no es para nada trivial, si no un trabajo de diseño que requiere una gran madurez en muchas ramas dispares de la ingeniería del software, tales como la programación en tiempo real, la programación orientada a eventos, o la programación gráfica.

En este último campo noté como las herramientas que solía usar tenían importantes carencias.

1.6 Trabajos Relacionados

Desde mi punto de vista, la biblioteca competirá en el "mercado" en 4 frentes :

- **Portabilidad** : Un objetivo principal de este trabajo será el tener una aplicación que se ejecute en al menos dos de las familias de sistemas más utilizadas a la hora de desarrollar software. Windows y Linux. Es común hoy en día encontrarse con buenos productos de este tipo que tienen la penalización de estar enfocados sólo a un SO concreto.

- **Eficacia** : Es importante hacer un buen uso de los recursos gráficos. Actualmente hay numerosas bibliotecas optimizadas para realizar este tipo de tareas de una manera muy eficiente, y evidentemente contra estas herramientas no busco rivalizar, ya que la complejidad que adquiriría el proyecto sería descomunal. Busco una eficiencia suficiente para que el salto cualitativo entre una biblioteca que renderiza por software y la mía, sea grande.

- **Nivel de Complejidad para el Usuario** : No tendría sentido realizar esta biblioteca si no fuese para aportar la comodidad de uso que no ofrecen otras famosas APIs como pueden ser las que uso de base (OpenGL y SDL). Sólo voy a analizar bibliotecas que tengan un nivel de abstracción elevado.

- **Funcionalidad** : Dotar de funcionalidades gráficas adicionales, así como evitar obstaculizar el control de los recursos con limitaciones no necesarias, hace más interesante una biblioteca de este tipo al programador.

12 DIV : Más adelante se hará un balance sobre la utilidad de este lenguaje. http://es.wikipedia.org/wiki/DIV

A continuación, voy muestrear muchos de los productos que cumplen el propósito de este proyecto y compararlos con él según los criterios que anteriormente he descrito:

Nombre	XNA Game Studio
Descripción	Biblioteca orientada para el desarrollo de videojuegos amateur para el entorno de desarrollo .NET
Licencia	Privativo - EULA
Criterios	
Portabilidad	Sólo sistemas de la familia Windows y Xbox 360
Eficacia	Uso de Direct3D muy refinado
Complejidad	Establece un interfaz de POO con numerosas facilidades y funcionalidades adicionales para facilitar la tarea del desarrollo de software tanto en la parte gráfica como en el resto.
Funcionalidad	Posee una gran gama de funcionalidades gráficas tanto para representar entornos 3D como 2D, e incluso mixtos
Conclusión	Muy útil para lanzar tus primeros proyectos comerciales para plataformas Microsoft, y para desarrolladores amateurs con suficiente base de programación. Pero su portabilidad y su costo son grandes problemas para este segundo perfil de usuarios.

Nombre	BlitzMax
Descripción	Es un lenguaje en sí mismo. Derivado de Basic y pensado para el desarrollo de videojuegos.
Licencia	Privativo
Criterios	
Portabilidad	Mac PPC, Mac Intel, Windows x86 y GNU/Linux x86
Eficacia	Utiliza OpenGL en su etapa de renderización
Complejidad	Puede utilizar OpenGL en su etapa de renderización en determinadas plataformas. Pero su modo de renderizado por defecto es por Software.
Funcionalidad	El lenguaje es poco flexible en general, y muy costoso para trabajar con estructuras de datos complejas. Aunque tiene funciones muy prácticas para trabajar en espacios 2D, incluso con efectos propios de la tecnología 3D (luces, transparencias, zoom, etc..)
Conclusión	Buena opción para proyectos de no demasiada envergadura, sobre todo si se puede disponer del módulo de OpenGL.

Nombre	Div/Fenix/Bennu		
Descripción	Es en sí mismo un lenguaje orientado a procesos, los cuales se ejecutan de forma paralela. Pensado para introducir a un usuario con pocos conocimientos de programación al desarrollo de videojuegos.		
Licencia Div	Privativo	Licencia Fenix/Bennu	GPL2
Criterios			
Portabilidad	Windows, GNU/Linux, GP32, PSP, etc...		
Eficacia	El proceso de renderizado va por soft. Usa internamente SDL		
Complejidad	Su metodología es muy rígida pero simple. La naturaleza paralela de los procesos hace más sencillo el control de la aplicación a usuarios poco experimentados en el diseño de software		
Funcionalidad	Al igual que BlitzMax, es un lenguaje exclusivo y por tanto tiene muchas limitaciones. Siendo aún más grave en lo relativo a las estructuras de datos. Pero tiene utilidades muy prácticas para trabajar con entornos 2D avanzados, siendo lo más llamativo el motor de Scroll, de Modo7, y de Blending que incluye de serie. Dichos motores no obstante, son muy restrictivos.		
Conclusión	Ideal para desarrolladores noveles o con poca experiencia. No apto para grandes proyectos, o para recrear un apartado visual técnicamente exigente en cuanto a recursos del sistema.		

Nombre	PyGame
Descripción	Biblioteca para Python pensada para el desarrollo de videojuegos.
Licencia	LGPL
Criterios	
Portabilidad	Windows, GNU/Linux, BSD, Mac, Nokia, BeOS...
Eficacia	Todo el proceso de renderizado va por software. Usa internamente SDL
Complejidad	Cuenta con todas las ventajas propias de Python y simplifica en gran parte los mecanismos de SDL
Funcionalidad	No conserva todas las opciones posibles con SDL enfocadas a tener un control más eficaz del programa, y tampoco añade suficientes técnicas adicionales (collision perfect[13], y poco más)
Conclusión	Gran herramienta para programadores de Python, pero una opción no muy a tener en cuenta si no te importa el lenguaje huésped.

13 Collision Perfect : Algoritmo que detecta con total precisión el solapamiento de imágenes.

Nombre	IndieLib
Descripción	Biblioteca para Visual C++, especializada principalmente en el desarrollo de videojuegos
Licencia	LGPL
Criterios	
Portabilidad	Depende de DirectX, por lo que sólo trabaja en plataformas Windows
Eficacia	Uso razonablemente bueno de Direct3D
Complejidad	Engloba Video, Timing y manejo de eventos. Como XNA, es una interfaz que usa POO, y abstrae con una lógica sencilla y breve todo el manejo del Hardware.
Funcionalidad	Se especializa en 2D, y posee numerosas features dentro de ese marco. También permite para usar de forma mixta la renderización 3D (aunque de manera mucho más sencilla). Establece plantillas de apoyo XML para el almacenamiento de estructuras de datos, que se integran con la librería para técnicas de animaciones, colisiones, etc..
Conclusión	No es tan completa ni eficaz como XNA pero es una alternativa libre para plataformas Windows muy funcional.

Capítulo 2 : Planificación

En este capítulo encontraremos :

– Estimaciones sobre el **costo temporal** del desarrollo de éste proyecto.
– El **modelo COCOMO**, una forma para deducir el coste económico a partir del coste temporal.

2.1 Estimaciones

La planificación trata sobre la estimación de tiempo y esfuerzo durante las fases de desarrollo, con el fin de pronosticar el tiempo que se va a emplear en el proyecto, y evaluar el ritmo que se está siguiendo conforme se va avanzando. También tiene sentido de cara a establecer plazos de entrega ante un hipotético cliente.

Para ello, dividimos la totalidad del proyecto en una serie de actividades diferenciadas que se han de realizar. Y a continuación se realizan dos estimaciones de tiempo a cada actividad, una inicial, al comienzo del proyecto y otra final, cuando ya se han acabado dichas actividades.

Una vez que se cuenta con ambas estimaciones, se puede a proceder a calcular el *Error Relativo :*

$$Error\ Relativo = \frac{A - E}{A}$$, donde A es la estimación final y E, la estimación inicial.

ACTIVIDADES	ESTIMACIÓN INICIAL	ESTIMACIÓN FINAL	ERROR RELATIVO
Busqueda de Documentación	250 Horas	80 Horas	-68,00%
Introducción y Planificación	30 Horas	40 Horas	25,00%
Elicitación de Recursos	30 Horas	110 Horas	72,72%
Análisis de Requisitos	30 Horas	40 Horas	25,00%
Diseño del Sistema	150 Horas	200 Horas	25,00%
Prototipo del Diseño	60 Horas	40 Horas	-33,33%
Instalación de la Plataforma	2 Horas	2 Horas	0,00%
Implementación	100 Horas	300 Horas	33,33%
Pruebas	40 Horas	15 Horas	-37,50%
Rediseño	60 Horas	140 Horas	57,14%
Presentación	12 Horas	12 Horas	0,00%
Revisión y Documentación final	12 Horas	15 Horas	20,00%
TOTAL	776 Horas	1094 Horas	29,07%
Error Relativo Medio			9,94%

Cuadro 2.1 : Planificación del Proyecto15

2.2 El modelo COCOMO

El Modelo Constructivo de Costes (**Co**nstructive **Co**st **Mo**del en Inglés) es un modelo de estimación de costes de software que incluye 3 submodelos, donde cada uno ofrece un nivel de detalle y aproximación cada vez mayor, a medida que avanza el proceso de desarrollo de software : básico, intermedio y detallado.

Fue desarrollado por Barry W. Boehm a finales de los 70 y comienzos de los 80, exponiéndolo detalladamente en su libro "Software Engineering Economics" (Prentice-Hall, 1981).

Pertenece a la categoría de modelos de subestimaciones basados en estimaciones matemáticas. Está orientado a la magnitud del producto final, midiendo el tamaño del proyecto en líneas de código principalmente. Por lo que puede presentar algunas deficiencias:

- Los resultados no son proporcionales a las tareas de gestión ya que no tiene en cuenta los recursos necesarios para realizar las tareas.
- Se puede desviar la reealidad si indica mal el porcentaje de comentarios en las líneas de código.
- Es un tanto subjetivo, puesto que está basado en estimaciones y parámetros que pueden ser "vistos" de distinta manera por distintos analistas que usen el método.
- Se miden los costes del producto, de acuerdo a su tamaño y otras características, pero no la productividad.
- La medición por líneas de código no es válida en la orientación a objetos.
- Utilizar esta herramienta puede resultar un poco complicada, en comparación con otros métodos (que también sólo estiman).

Teniendo en cuenta estas características sólo obtendremos una estimación aproximada del coste real del proyecto.

Fuente : http://es.wikipedia.org/wiki/COCOMO

2.3 Aplicando el modelo COCOMO

La función básica que utilizan los tres modelos (básico, intermedio y detallado) es:

$E = a(Kl)b \cdot m(X)$, donde :

- *a* y *b* son constantes con valores definidos en cada submodelo.
- *Kl* son las líneas de código (en miles), en este caso 15.
- m(X) es un multiplicador que depende de 15 atributos.
- Y el resultado se mide en salarios/mes y horas-hombre.

A la vez, cada submodelo también se divide en **modos** que representan el tipo de proyecto, y puede ser:

- ***modo orgánico***: un pequeño grupo de programadores experimentados desarrollan software en un entorno familiar. El tamaño del software varía de unas pocas miles de líneas (tamaño pequeño) a unas decenas de miles de líneas (medio).
- ***modo semilibre*** o ***semiencajado***: corresponde a un esquema intermedio entre el orgánico y el rígido; el grupo de desarrollo puede incluir una mezcla de personas experimentadas y no experimentadas.
- modo rígido o empotrado: el proyecto tiene unas fuertes restricciones, que pueden estar relacionadas con la funcionalidad o técnicas. El problema a resolver es único y es difícil basarse en la experiencia, puesto que puede no haberla.

Dado que sólo se va a emplear una variable para la estimación (la línea de código), se empleará el tipo de modelo COCOMO básico, ya que es un modelo univariable estático.

Además éste proyecto será considerado como software orgánico, pues posee mucho menos de 50.000 líneas de código. Se utiliza para obtener una primera aproximación rápida del esfuerzo, y hace uso de la siguiente tabla de constantes para calcular distintos aspectos de costes:

MODO	a	b	c	d
Orgánico	2.40	1.05	2.50	0.38
Semilibre	3.00	1.12	2.50	0.35
Rígido	3.60	1.20	2.50	0.32

Cuadro 2.2 : Tabla de modelos de estimación

Estos valores son para las fórmulas:

- Personas necesarias por mes para llevar adelante el proyecto (**MM**) = a*(Klb)
- Tiempo de desarrollo del proyecto (**TDEV**) = c*(MMd)
- Personas necesarias para realizar el proyecto (**CosteH**) = MM/TDEV
- Costo total del proyecto (**CosteM**) = CosteH * *Salario medio entre los programadores y analistas*.

Se puede observar que a medida que aumenta la complejidad del proyecto (modo), las constantes aumentan de 2.4 a 3.6, que corresponde a un incremento del esfuerzo del personal. Hay que utilizar con mucho cuidado el modelo básico puesto que se obvian muchas características del entorno.

La fórmula que aplicaremos es la del tipo orgánico y el método básico, puesto que estos son los modelos en los que cuadra nuestro proyecto.

A continuación se muestra la resolución de las fórmulas anteriormente descritas :

$$(MM) = 2{,}4 \cdot 15^{1{,}05} = 41{,}22 \; personas/mes$$

$$(TDEV) = 2{,}5 \cdot 41{,}22^{0{,}38} = 10{,}27 \; meses$$

$$(CosteH) = \frac{41{,}22}{10{,}27} = 4{,}01 \; personas$$

$$(CosteM) = 4{,}01 \cdot 1000 € = 4010 € \; mensuales$$

Por tanto, el costo total tras los 10,27 meses de desarrollo es de **41.1182** € sin incluir materiales, sólo coste humano.

Realmente se trata de un coste muy relativo, ya que este modelo omite la complejidad que pueda tener cada línea de código o la repetición de las mismas.

Capítulo 3 : Elicitación de Requisitos

En este capítulo encontraremos :

– Una tabla descriptiva con los **participantes** implicados en el desarrollo del proyecto.
– Una lista de **objetivos** a seguir durante el desarrollo de la biblioteca.
– Un conjunto de tipos de **requisitos** con los que analizaremos qué tipo de datos podremos llegar a manejar.
– Una serie de **restricciones** que establecen una lógica de negocio sobre el modelo de requisitos anteriormente formulado.
– Una lista de **casos de uso** que muestran las interacciones de un usuario con la biblioteca.
– Un compendio de **requisitos no funcionales** que trataremos de seguir durante el desarrollo.

3.1. Visión general del Sistema

3.1.1. Participantes del Proyecto

Organización	Departamento de Lenguajes y Sistemas Informáticos
Dirección	Avda. Reina Mercedes s/n. 41012 Sevilla Escuela Técnica Superior de Ingeniería Informática
Teléfono	(+34) 954 900 858
Fax	(+34) 954 900 858
Comentarios	E-mail: buzon@lsi.us.es Web: www.lsi.us.es

Cuadro 3.1 : Departamento de Lenguajes y Sistemas Informáticos.

Participante	José Manuel Barroso Galindo
Organización	Freelance
Rol	Desarrollador del proyecto
Es desarrollador	Sí
Es cliente	Sí
Es usuario	Sí
Comentarios	Alumno de Ingeniería Técnica en Informática en Sistemas por la Escuela Técnica Superior de Ingeniería Informática de la Universidad de Sevilla encargado del desarrollo de la biblioteca.

Cuadro 3.2 : José Manuel Barroso Galindo.

Participante	Pablo Neira Ayuso
Organización	Departamento de Lenguajes y Sistemas Informáticos
Rol	Tutor
Es desarrollador	No
Es cliente	No
Es usuario	No
Comentarios	Profesor titular del Departamento de Lenguajes y Sistemas Informáticos por la Universidad de Sevilla, encargado de la supervisión y control de desarrollo completo de la biblioteca.

Cuadro 3.3 : Pablo Neira Ayuso.

3.1.2. Objetivos del Sistema

OBJ-0001	**Inicialización de Contexto de Vídeo**
Versión	1.0 (14/06/2009)
Autores	• José Manuel Barroso Galindo
Fuentes	• Pablo Neira Ayuso
Descripción	El sistema deberá *inicializar el subsistema de vídeo adaptándose a la configuración de la máquina huésped. Será necesario para la comunicación de la aplicación con el hardware gráfico.*
Subobjetivos	Ninguno
Importancia	vital
Urgencia	inmediatamente
Estado	validado
Estabilidad	alta
Comentarios	Ninguno

OBJ-0002	**Importación de fuentes de recursos externos**
Versión	1.0 (14/06/2009)
Autores	• José Manuel Barroso Galindo
Fuentes	• Pablo Neira Ayuso
Descripción	El sistema deberá *dar la posibilidad de incorporar formatos gráficos y/o estructuras de configuración que serán interpretadas para su uso por la aplicación.*
Subobjetivos	• **[OBJ-0003] Importar formatos gráficos**: El sistema deberá *aceptar imágenes de diversos formatos gráficos.* • **[OBJ-0004] Importar estructuras de configuración**: El sistema deberá *aceptar plantillas que doten de mayor información a las imágenes importadas.*
Importancia	importante
Urgencia	inmediatamente
Estado	validado
Estabilidad	alta

Comentarios	Ninguno

OBJ-0005	**Soporte de Renderización**
Versión	1.0 (14/06/2009)
Autores	• José Manuel Barroso Galindo
Fuentes	• Pablo Neira Ayuso
Descripción	El sistema deberá *procesar la información gráfica del estado actual del sistema, y en base a él, representar en pantalla el resultado.*
Subobjetivos	Ninguno
Importancia	vital
Urgencia	inmediatamente
Estado	validado
Estabilidad	alta
Comentarios	Ninguno

OBJ-0006	**Ofrecer mecanismos para la manipulación de los eventos de Entrada/Salida**
Versión	1.0 (14/06/2009)
Autores	• José Manuel Barroso Galindo
Fuentes	• Pablo Neira Ayuso
Descripción	El sistema deberá *recopilar eventos de E/S provenientes de dispositivos como el Ratón, el Teclado, etc, y poner a disposición de la aplicación el acceso a su información con plena libertad.*
Subobjetivos	Ninguno
Importancia	importante
Urgencia	hay presión
Estado	validado
Estabilidad	alta
Comentarios	Ninguno

OBJ-0007	**Soporte de efectos gráficos adicionales**
Versión	1.0 (14/06/2009)
Autores	• José Manuel Barroso Galindo

Fuentes	• Pablo Neira Ayuso
Descripción	El sistema deberá *aprovechar las capacidades de OpenGL para manejar operaciones básicas de dibujado (Primitivas Gráficas) u otras operaciones especiales.*
Subobjetivos	• **[OBJ-0008] Primitivas Gráficas**: El sistema deberá *ofrecer la posibilidad de dibujar directamente puntos, cuadrados, circulos, polígonos o carácteres.* • **[OBJ-0009] Efectos Especiales**: El sistema deberá *ofrecer funcionalidades que modifiquen las propiedades de cualquier elemento que vaya a ser renderizado. Propiedades como canales de color, perspectiva, geometría, etc..*
Importancia	importante
Urgencia	puede esperar
Estado	validado
Estabilidad	alta
Comentarios	Ninguno

OBJ-0010	**Soporte de temporización**
Versión	1.0 (14/06/2009)
Autores	• José Manuel Barroso Galindo
Fuentes	• Pablo Neira Ayuso
Descripción	El sistema deberá *ofrecer mecanismos para que el usuario pueda ajustar el ritmo de la ejecución.*
Subobjetivos	Ninguno
Importancia	importante
Urgencia	inmediatamente
Estado	validado
Estabilidad	alta
Comentarios	Ninguno

OBJ-0011	**Soporte de métodos Ayudantes**
Versión	1.0 (14/06/2009)

Autores	• José Manuel Barroso Galindo
Fuentes	• Pablo Neira Ayuso
Descripción	El sistema deberá *contar con numerosos métodos y plantillas que ayudarán a que el usuario controle la biblioteca.*
Subobjetivos	• **[OBJ-0012] Métodos Ayudantes**: El sistema deberá *ofrecer rutinas que ayuden a gestionar los recursos de la biblioteca y a controlar sus funcionalidades.* • **[OBJ-0013] Plantillas**: El sistema deberá *poner a disposición del usuario plantillas que faciliten el uso de las funcionalidades de la biblioteca y que establezcan un flujo de trabajo ópticmo, trantado así de disminuir el trabajo del usuario.*
Importancia	importante
Urgencia	inmediatamente
Estado	validado
Estabilidad	alta
Comentarios	Ninguno

OBJ-0014	**Alto grado de Encapsulación**
Versión	1.0 (14/06/2009)
Autores	• José Manuel Barroso Galindo
Fuentes	• Pablo Neira Ayuso
Descripción	El sistema deberá *presentar todos los recursos que la biblioteca ofrece al usuario encapsulados en objetos, para adaptarse lo más fielmente posible al paradigma de la programación orientada a objetos, y así disfrutar de sus ventajas.*
Subobjetivos	Ninguno
Importancia	quedaría bien
Urgencia	puede esperar
Estado	en construcción
Estabilidad	alta
Comentarios	Ninguno

3.2. Catálogo de Requisitos del Sistema

3.2.1. Requisitos de Información

3

IRQ-0002	Sprite	
Versión	1.0 (14/06/2009)	
Autores	• José Manuel Barroso Galindo	
Fuentes	• Pablo Neira Ayuso	
Dependencias	• [OBJ-0004] Importar estructuras de configuración • [IRQ-0001] Texto • [OBJ-0014] Alto grado de Encapsulación	
Descripción	El sistema deberá almacenar la información correspondiente a *las imágenes que provienen de un recurso gráfico externo, para su uso en la etapa de renderización si es requerido.* En concreto:	
Datos específicos	• nombre • punto central • rectángulos de área • alto • ancho • bits por pixel • referencia a los pixels	
Tiempo de vida	**Medio**	**Máximo**
	1 hora(s)	1 día(s)
Ocurrencias simultáneas	**Medio**	**Máximo**
	1000	48
Importancia	importante	
Urgencia	inmediatamente	
Estado	validado	
Estabilidad	alta	
Comentarios	Ninguno	

CRQ-0001	Formato de Imágenes Soportado

Versión	1.0 (14/06/2009)
Autores	• José Manuel Barroso Galindo
Fuentes	• Pablo Neira Ayuso
Dependencias	• [IRQ-0002] Sprite
Descripción	La información almacenada por el sistema deberá satisfacer la siguiente restricción: *Sólo se admitirán los siguientes formatos :* *BMP, GIF, JPEG, LBM, PCX, PNG, PNM, TGA, TIFF, XCF, XPM.*
Importancia	importante
Urgencia	inmediatamente
Estado	validado
Estabilidad	alta
Comentarios	Ninguno

IRQ-0003	**Conjunto de Sprites**	
Versión	1.0 (14/06/2009)	
Autores	• José Manuel Barroso Galindo	
Fuentes	• Pablo Neira Ayuso	
Dependencias	• [IRQ-0002] Sprite • [OBJ-0014] Alto grado de Encapsulación	
Descripción	El sistema deberá almacenar la información correspondiente a *el conjunto de sprites procedentes de la misma plantilla, y deberá agruparlos en un mismo objeto contenedor.* En concreto:	
Datos específicos	• descriptor de la plantilla de sprites • modo • lista de sprites	
Tiempo de vida	**Medio**	**Máximo**
	1 hora(s)	1 día(s)
Ocurrencias simultáneas	**Medio**	**Máximo**
	8	100
Importancia	importante	
Urgencia	inmediatamente	

Estado	validado
Estabilidad	alta
Comentarios	Un descriptor es uno o varios parámetros que permiten importar un objeto desde un medio externo, como por ejemplo el nombre de un fichero junto a un valor, de manera que conformen una clave.

CRQ-0002	**Unicidad en el descriptor de los conjuntos de sprites**
Versión	1.0 (14/06/2009)
Autores	• José Manuel Barroso Galindo
Fuentes	• Pablo Neira Ayuso
Dependencias	• [IRQ-0003] Conjunto de Sprites
Descripción	La información almacenada por el sistema deberá satisfacer la siguiente restricción: *No podrán cargarse varios Conjuntos de Sprites diferentes si estos van a tener el mismo descriptor.*
Importancia	importante
Urgencia	inmediatamente
Estado	validado
Estabilidad	alta
Comentarios	Ninguno

IRQ-0004	**Fuente**
Versión	1.0 (14/06/2009)
Autores	• José Manuel Barroso Galindo
Fuentes	• Pablo Neira Ayuso
Dependencias	• [OBJ-0008] Primitivas Gráficas • [IRQ-0001] Texto • [OBJ-0014] Alto grado de Encapsulación
Descripción	El sistema deberá almacenar la información correspondiente a *las fuentes tipográficas, junto con sus caracteres específicos que son a su vez imágenes.* En concreto:
Datos específicos	• descriptor de la fuente • datos de la fuente • lista de imágenes de los carácteres

	• tamaño	
Tiempo de vida	**Medio**	**Máximo**
	1 hora(s)	1 día(s)
Ocurrencias simultáneas	**Medio**	**Máximo**
	32	3
Importancia	importante	
Urgencia	inmediatamente	
Estado	validado	
Estabilidad	alta	
Comentarios	Ninguno	

IRQ-0001	**Texto**	
Versión	1.0 (14/06/2009)	
Autores	• José Manuel Barroso Galindo	
Fuentes	• Pablo Neira Ayuso	
Dependencias	• [OBJ-0003] Importar formatos gráficos • [OBJ-0014] Alto grado de Encapsulación • [OBJ-0005] Soporte de Renderización	
Descripción	El sistema deberá almacenar la información correspondiente a *la lista de carácteres que se van a obtener de la fuente para usarlos en la etapa de renderización para representar texto. Junto con información adicional si es requerido.* En concreto:	
Datos específicos	• fuente • secuencia de carácteres • posición • tipo de texto • atributos según el tipo	
Tiempo de vida	**Medio**	**Máximo**
	1 hora(s)	1 día(s)
Ocurrencias simultáneas	**Medio**	**Máximo**
	128	1024
Importancia	importante	
Urgencia	inmediatamente	

Estado	validado
Estabilidad	alta
Comentarios	Ninguno

CRQ-0003	**Formato de Fuentes Soportadas**
Versión	1.0 (14/06/2009)
Autores	• José Manuel Barroso Galindo
Fuentes	• Pablo Neira Ayuso
Dependencias	• [IRQ-0004] Fuente
Descripción	La información almacenada por el sistema deberá satisfacer la siguiente restricción: *Sólo se admitirán los siguientes formatos :* *TTF*
Importancia	importante
Urgencia	inmediatamente
Estado	validado
Estabilidad	alta
Comentarios	Ninguno

CRQ-0004	**Unicidad del descriptor de las Fuentes.**
Versión	1.0 (14/06/2009)
Autores	• José Manuel Barroso Galindo
Fuentes	• Pablo Neira Ayuso
Dependencias	• [IRQ-0004] Fuente
Descripción	La información almacenada por el sistema deberá satisfacer la siguiente restricción: *No se permitirá la carga de 2 fuentes cuyos descriptores sean coincidentes.*
Importancia	importante
Urgencia	inmediatamente
Estado	validado
Estabilidad	alta
Comentarios	Ninguno

IRQ-0005	Contexto de Vídeo	
Versión	1.0 (14/06/2009)	
Autores	• José Manuel Barroso Galindo	
Fuentes	• Pablo Neira Ayuso	
Dependencias	• [OBJ-0001] Inicialización de Contexto de Vídeo • [OBJ-0012] Métodos Ayudantes	
Descripción	El sistema deberá almacenar la información correspondiente a *la configuración del contexto de vídeo creado.* En concreto:	
Datos específicos	• recursos reservados • ancho en pixeles • alto en pixeles • bits por pixel • escalado base en anchura • escalado base en altura • efectos gráficos (pantalla completa, sincronización vertical, etc...)	
Tiempo de vida	**Medio**	**Máximo**
	1 día(s)	1 hora(s)
Ocurrencias simultáneas	**Medio**	**Máximo**
	1	1
Importancia	vital	
Urgencia	inmediatamente	
Estado	validado	
Estabilidad	alta	
Comentarios	Ninguno	

CRQ-0005	Configuración de Vídeo Soportada
Versión	1.0 (14/06/2009)
Autores	• José Manuel Barroso Galindo
Fuentes	• Pablo Neira Ayuso
Dependencias	• [IRQ-0005] Contexto de Vídeo
Descripción	La información almacenada por el sistema deberá satisfacer la siguiente restricción: *Deberá comprobarse que la máquina se puede*

	configurar con los datos escogidos, y si no, se intentará utilizar su configuración nativa.
Importancia	importante
Urgencia	hay presión
Estado	validado
Estabilidad	alta
Comentarios	Ninguno

IRQ-0006	**Evento**	
Versión	1.0 (14/06/2009)	
Autores	• José Manuel Barroso Galindo	
Fuentes	• Pablo Neira Ayuso	
Dependencias	• [OBJ-0006] Ofrecer mecanismos para la manipulación de los eventos de Entrada/Salida • [OBJ-0014] Alto grado de Encapsulación	
Descripción	El sistema deberá almacenar la información correspondiente a *las interacciones de entrada salida provenientes de los dispositivos estándar.* En concreto:	
Datos específicos	• tipo • información del estímulo	
Tiempo de vida	**Medio**	**Máximo**
	50 milisegundo(s)	1 segundo(s)
Ocurrencias simultáneas	**Medio**	**Máximo**
	3	128
Importancia	importante	
Urgencia	inmediatamente	
Estado	validado	
Estabilidad	alta	
Comentarios	Ninguno	

CRQ-0010	**Dispositivos de E/S Soportados**
Versión	1.0 (15/06/2009)
Autores	• José Manuel Barroso Galindo

Fuentes	• Pablo Neira Ayuso
Dependencias	• [IRQ-0006] Evento
Descripción	La información almacenada por el sistema deberá satisfacer la siguiente restricción: *Sólo se consideraran los siguienes estímulos :* *Pulsación de Teclado* *Movimiento y pulsación de Ratón* *Pulsación de un Gamepad.* *Cambio de la ventana de la Aplicación.* *Eventos definidos por el Usuario.*
Importancia	importante
Urgencia	inmediatamente
Estado	validado
Estabilidad	alta
Comentarios	Ninguno

IRQ-0012	**Manejador de Eventos**	
Versión	1.0 (13/07/2009)	
Autores	• José Manuel Barroso Galindo	
Fuentes	• Pablo Neira Ayuso	
Dependencias	• [IRQ-0006] Evento	
Descripción	El sistema deberá almacenar la información correspondiente a *la serie de rutinas especificadas por el usuario que deberán reaccionar ante el procesamiento de un evento de un tipo específico, invocando aquellas otras rutinas que consideren necesarias.* En concreto:	
Datos específicos	Ninguno	
Tiempo de vida	**Medio**	**Máximo**
	1 hora(s)	1 día(s)
Ocurrencias simultáneas	**Medio**	**Máximo**
	6	30
Importancia	importante	
Urgencia	inmediatamente	
Estado	validado	
Estabilidad	alta	
Comentarios	Ninguno	

IRQ-0007	**Motor**	
Versión	1.0 (14/06/2009)	
Autores	• José Manuel Barroso Galindo	
Fuentes	• Pablo Neira Ayuso	
Dependencias	• [OBJ-0006] Ofrecer mecanismos para la manipulación de los eventos de Entrada/Salida • [OBJ-0010] Soporte de temporización • [IRQ-0006] Evento • [OBJ-0014] Alto grado de Encapsulación • [OBJ-0013] Plantillas • [OBJ-0012] Métodos Ayudantes	
Descripción	El sistema deberá almacenar la información correspondiente a *al motor al cual se podrá delega el control de la ejecución. La biblioteca gestionará los motores almacenados, procesando cuando se solicite el motor pertinente según su prioridad y la petición del usuario. El motor que se esté procesando por la biblioteca en un momento determinado pasa a ser el motor activo.* En concreto:	
Datos específicos	• lista de eventos • lista de manejadores de eventos • lista de cámaras • prioridad	
Tiempo de vida	**Medio**	**Máximo**
	1 hora(s)	1 día(s)
Ocurrencias simultáneas	**Medio**	**Máximo**
	1	10
Importancia	vital	
Urgencia	inmediatamente	
Estado	validado	
Estabilidad	alta	
Comentarios	La no presencia de eventos en el sistema, es contemplado por el manejador de eventos como un tipo específico de evento - espera - en el cual se llama al proceso de renderización y a otras tareas independientes.	

CRQ-0011	**Único Motor activo en un momento determinado**
Versión	1.0 (13/07/2009)

Autores	• José Manuel Barroso Galindo
Fuentes	• Pablo Neira Ayuso
Dependencias	Ninguno
Descripción	La información almacenada por el sistema deberá satisfacer la siguiente restricción: *Sólo puede haber un único motor activo en un instante determinado. Que no haya ningún motor activo también es posible.*
Importancia	vital
Urgencia	inmediatamente
Estado	validado
Estabilidad	media
Comentarios	Ninguno

IRQ-0008	**Universo**	
Versión	1.0 (14/06/2009)	
Autores	• José Manuel Barroso Galindo	
Fuentes	• Pablo Neira Ayuso	
Dependencias	• [IRQ-0009] Actor • [OBJ-0014] Alto grado de Encapsulación • [OBJ-0013] Plantillas • [OBJ-0012] Métodos Ayudantes	
Descripción	El sistema deberá almacenar la información correspondiente a *el sistema de coordenadas independiente del contexto de vídeo, bajo el cual actúan unas reglas de operabilidad arbitrarias. Y al conjunto de actores y figurantes que están contenidos en él, los cuales podrán ser representados en la fase de renderización en función de dichas reglas de operabilidad.* En concreto:	
Datos específicos	• descriptor • sistema de coordenadas • lista de actores contenidos • conjunto de figurantes • recursos cargados • efectos gráficos (temblor, cambio de color, etc...)	
Tiempo de vida	**Medio**	**Máximo**

	20 minuto(s)	12 hora(s)
Ocurrencias simultáneas	**Medio**	**Máximo**
	2	32
Importancia	importante	
Urgencia	inmediatamente	
Estado	validado	
Estabilidad	alta	
Comentarios	Ninguno	

IRQ-0009	**Actor**	
Versión	1.0 (14/06/2009)	
Autores	• José Manuel Barroso Galindo	
Fuentes	• Pablo Neira Ayuso	
Dependencias	• [IRQ-0008] Universo • [OBJ-0014] Alto grado de Encapsulación	
Descripción	El sistema deberá almacenar la información correspondiente a *todo elemento que necesita operar siempre de acuerdo a las reglas de algún Universo, siempre que tenga la posibilidad de cambiar de Universo huésped, y de ser objetivo de una o varias cámaras.* En concreto:	
Datos específicos	• universo huésped • coordenadas • lista de sprites asociados • nombre característico • efectos gráficos (espejado, rotación, estiramiento...)	
Tiempo de vida	**Medio**	**Máximo**
	20 minuto(s)	1 día(s)
Ocurrencias simultáneas	**Medio**	**Máximo**
	10	50000
Importancia	importante	
Urgencia	inmediatamente	
Estado	validado	
Estabilidad	alta	
Comentarios	Ninguno	

CRQ-0006	Reciprocidad entre el Universo huesped y el Actor
Versión	1.0 (14/06/2009)
Autores	• José Manuel Barroso Galindo
Fuentes	• Pablo Neira Ayuso
Dependencias	• [IRQ-0009] Actor • [IRQ-0008] Universo
Descripción	La información almacenada por el sistema deberá satisfacer la siguiente restricción: *Varios Universos no pueden incluir al mismo actor en el mismo momento. Por tanto, el actor debe tener un único Universo huesped simultáneamente.*
Importancia	vital
Urgencia	inmediatamente
Estado	validado
Estabilidad	alta
Comentarios	Ninguno

CRQ-0007	Compatibilidad entre el Actor y el Universo huésped
Versión	1.0 (14/06/2009)
Autores	• José Manuel Barroso Galindo
Fuentes	• Pablo Neira Ayuso
Dependencias	• [IRQ-0009] Actor • [IRQ-0008] Universo
Descripción	La información almacenada por el sistema deberá satisfacer la siguiente restricción: *Las coordenadas que usa el Actor deben ser compatibles con el sistema de coordenadas de su Universo huésped. Y su manipulación debe respetar las reglas de operabilidad del mismo.*
Importancia	vital
Urgencia	inmediatamente
Estado	validado
Estabilidad	alta
Comentarios	Ninguno

IRQ-0010	Figurante

Versión	1.0 (14/06/2009)	
Autores	• José Manuel Barroso Galindo	
Fuentes	• Pablo Neira Ayuso	
Dependencias	• [IRQ-0003] Conjunto de Sprites • [IRQ-0002] Sprite	
Descripción	El sistema deberá almacenar la información correspondiente a *todo elemento que está ligado exclusivamente a un Universo, con presencia en su sistema de coordenadas, y que puede ser representado gráficamente.* En concreto:	
Datos específicos	• coordenadas • lista de sprites asociados	
Tiempo de vida	**Medio**	**Máximo**
	20 minuto(s)	12 hora(s)
Ocurrencias simultáneas	**Medio**	**Máximo**
	32	2
Importancia	quedaría bien	
Urgencia	puede esperar	
Estado	validado	
Estabilidad	alta	
Comentarios	Ninguno	

CRQ-0008	**Compatibilidad entre los Figurantes y el Universo**
Versión	1.0 (15/06/2009)
Autores	• José Manuel Barroso Galindo
Fuentes	• Pablo Neira Ayuso
Dependencias	• [IRQ-0008] Universo • [IRQ-0010] Figurante
Descripción	La información almacenada por el sistema deberá satisfacer la siguiente restricción: *Las coordenadas de los figurantes deben ser compatibles con el sistema de coordenadas de su universo.*
Importancia	quedaría bien
Urgencia	puede esperar

Estado	validado
Estabilidad	alta
Comentarios	Ninguno

IRQ-0011	**Cámara**	
Versión	1.0 (14/06/2009)	
Autores	• José Manuel Barroso Galindo	
Fuentes	• Pablo Neira Ayuso	
Dependencias	• [IRQ-0009] Actor • [IRQ-0008] Universo • [OBJ-0005] Soporte de Renderización • [OBJ-0014] Alto grado de Encapsulación • [OBJ-0013] Plantillas • [OBJ-0012] Métodos Ayudantes	
Descripción	El sistema deberá almacenar la información correspondiente a *al elemento que determina los objetos a renderizar, según el entorno del Actor objetivo y de acuerdo a la información del Universo huésped de dicho actor.* En concreto:	
Datos específicos	• actor objetivo • universo huésped • coordenadas • disposición en la pantalla • efectos gráficos (zoom, iluminación...)	
Tiempo de vida	**Medio**	**Máximo**
	1 hora(s)	1 día(s)
Ocurrencias simultáneas	**Medio**	**Máximo**
	1	32
Importancia	importante	
Urgencia	inmediatamente	
Estado	validado	
Estabilidad	alta	
Comentarios	Ninguno	

3.2.2. Requisitos Funcionales

3.2.2.1. Diagramas de Casos de Uso

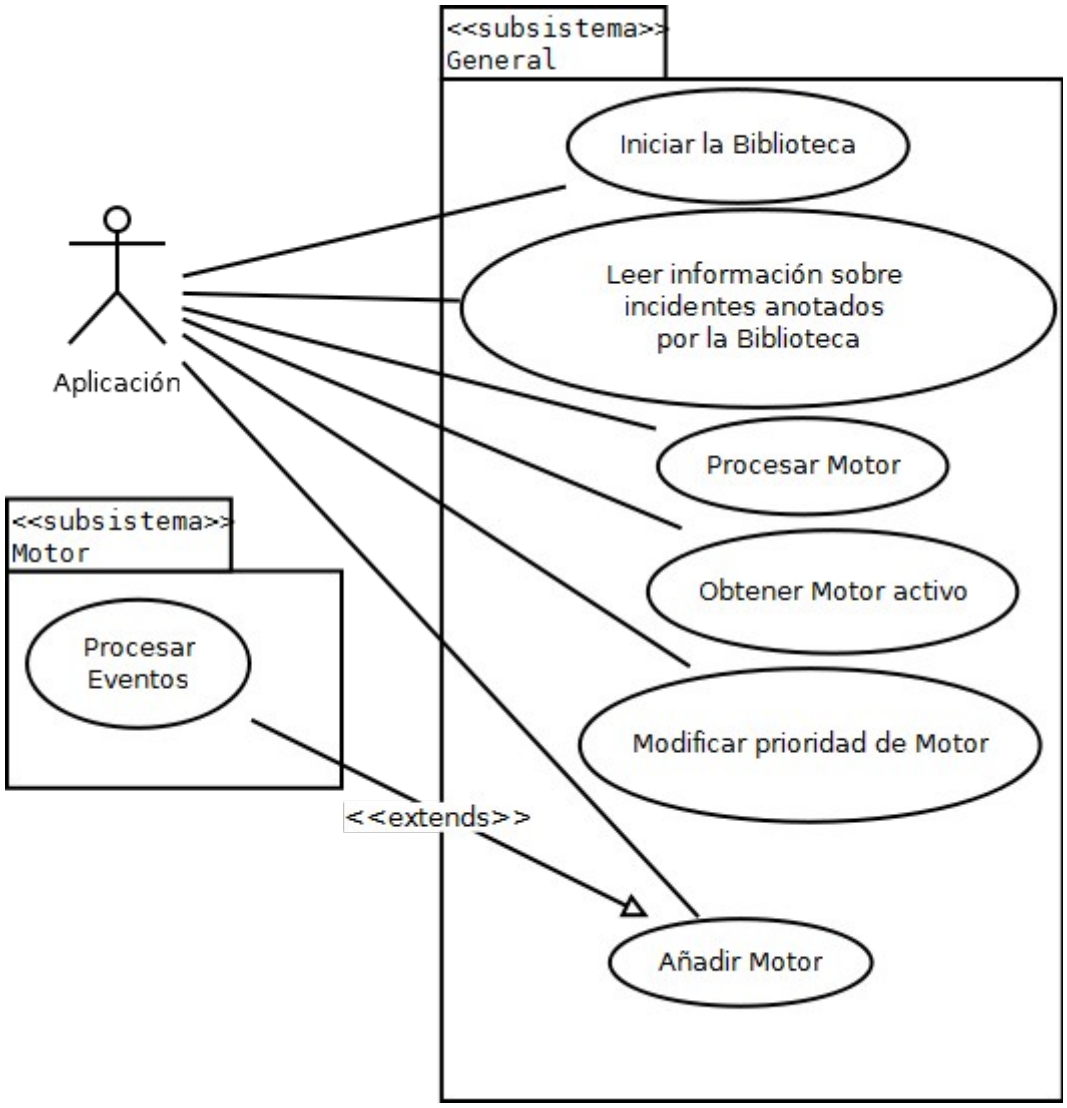


Figura 3.1 : Subsistema General

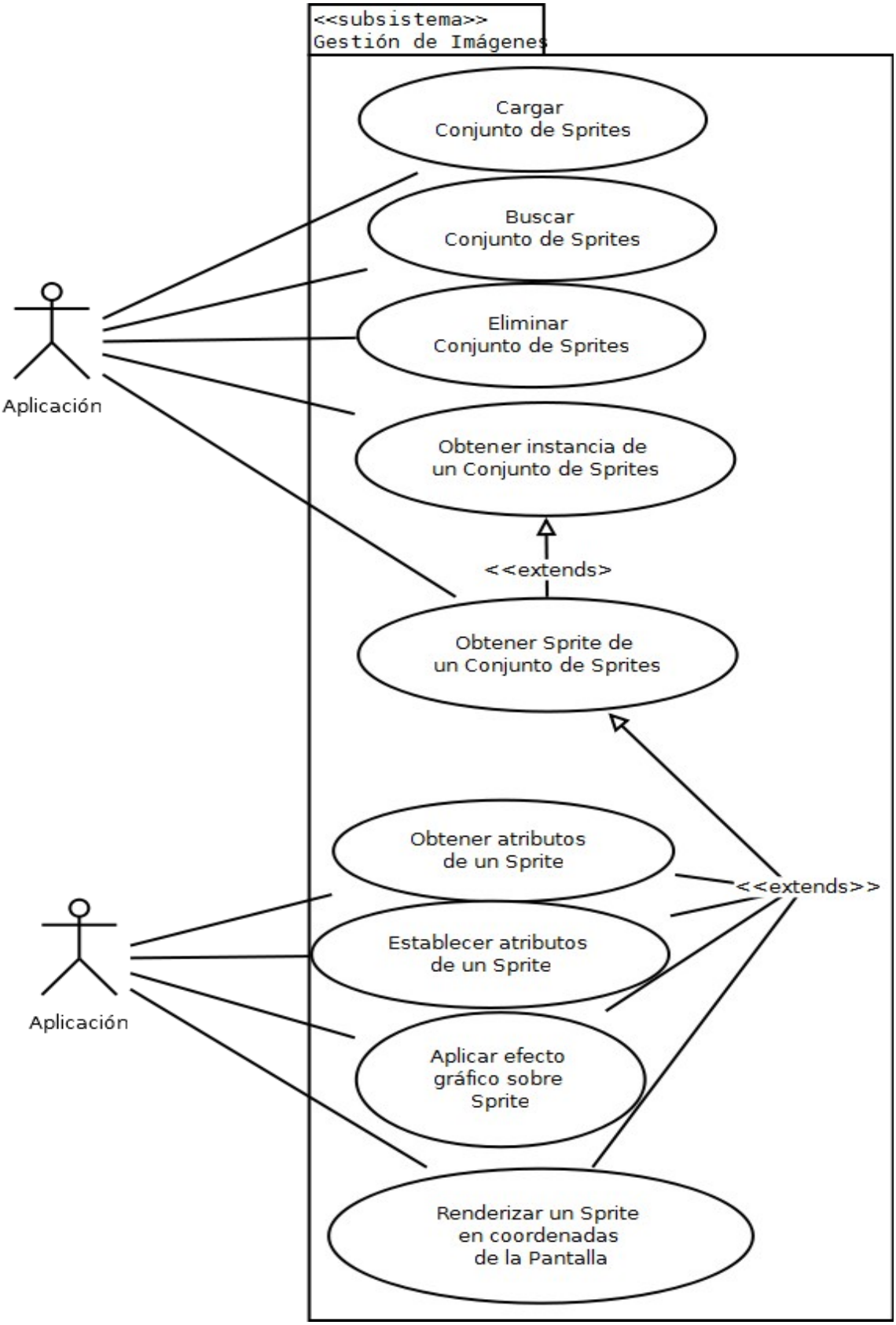


Figura 3.2 : Subsistema Gestión de Imágenes

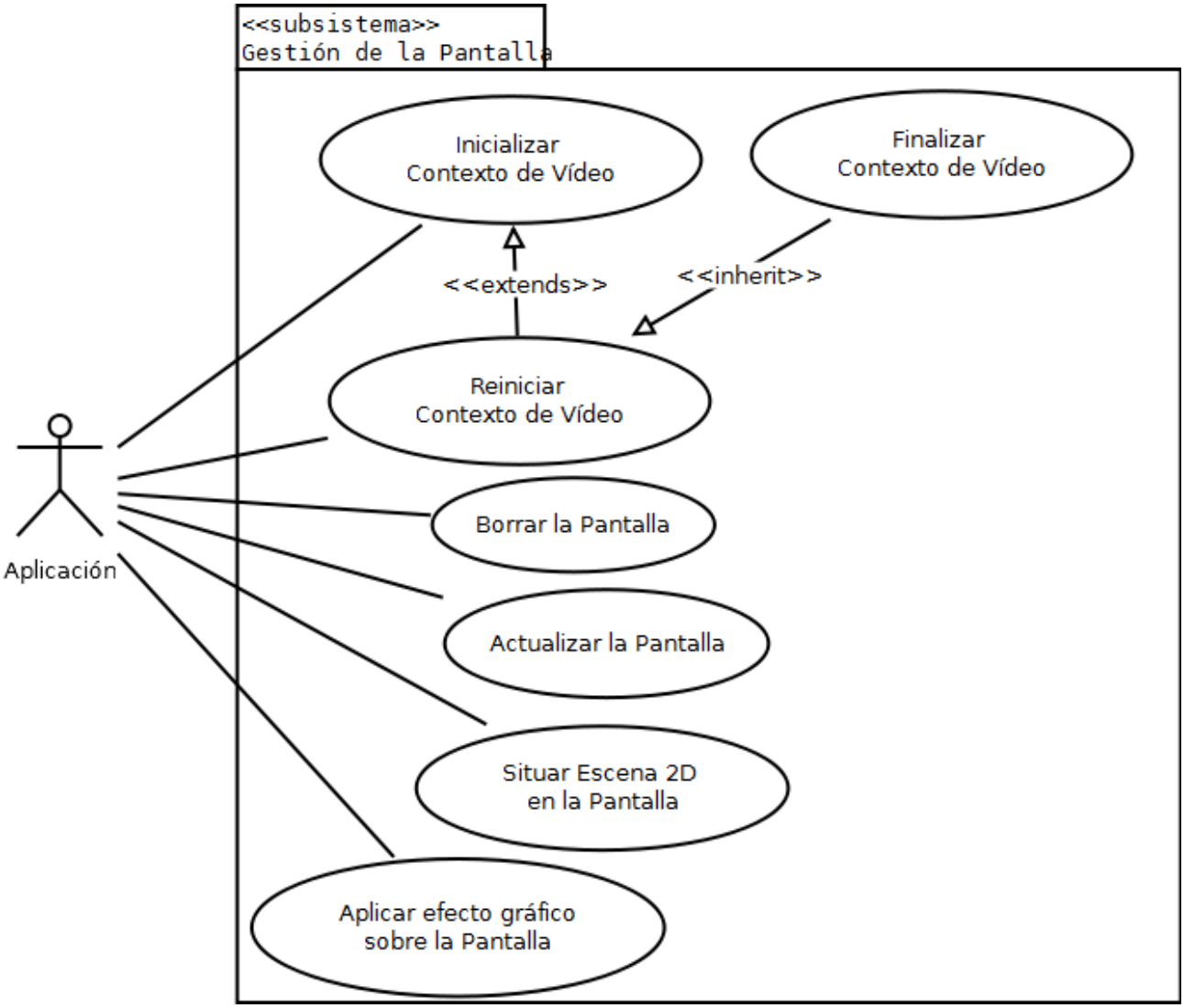


Figura 3.3 : Subsistema Gestión de la Pantalla

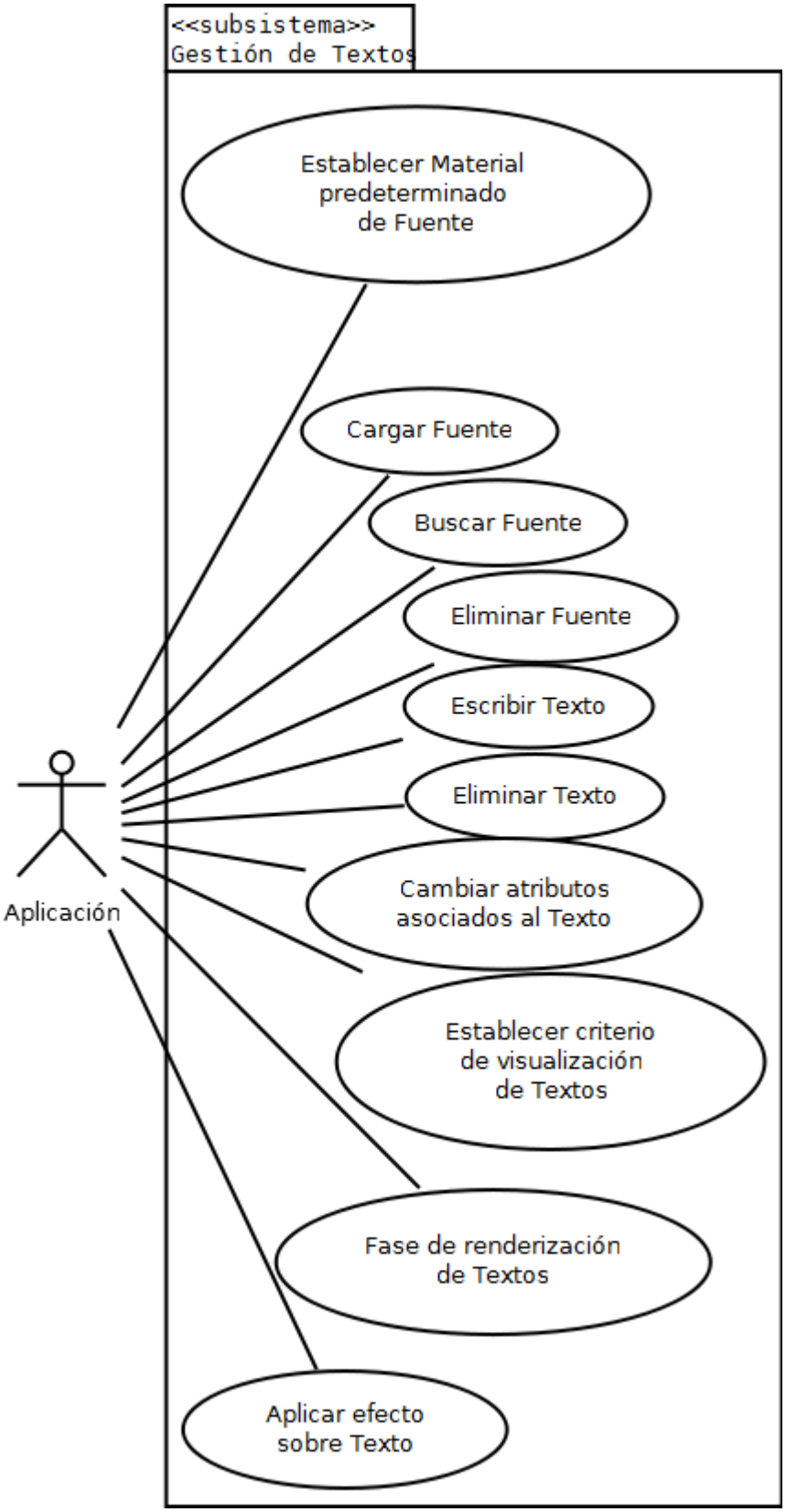


Figura 3.4 : Subsistema Gestión de Textos

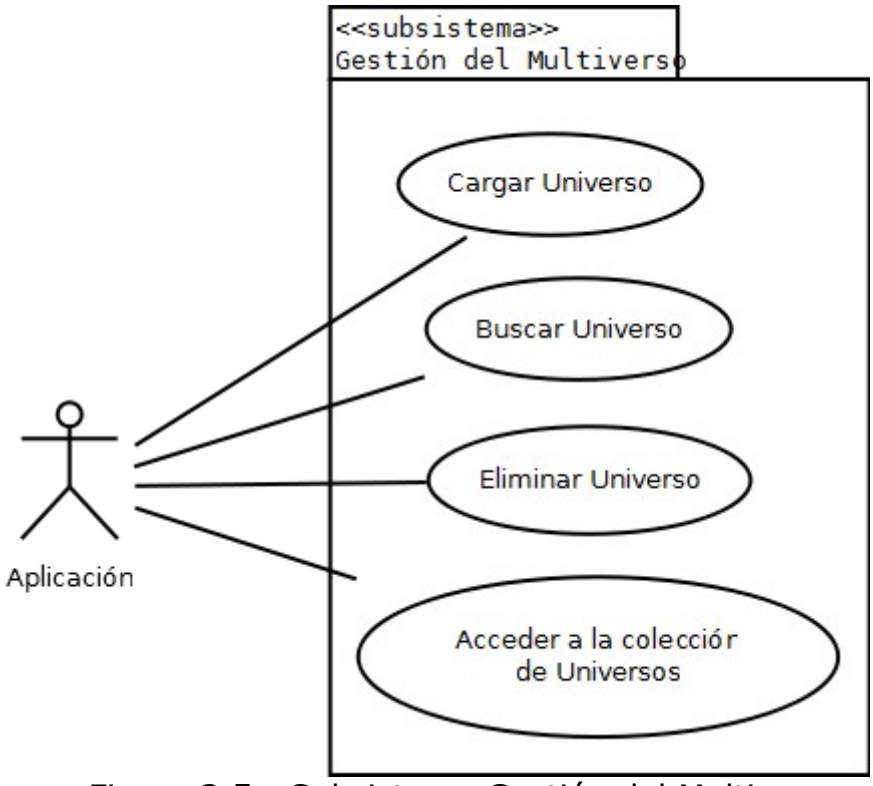


Figura 3.5 : Subsistema Gestión del Multiverso

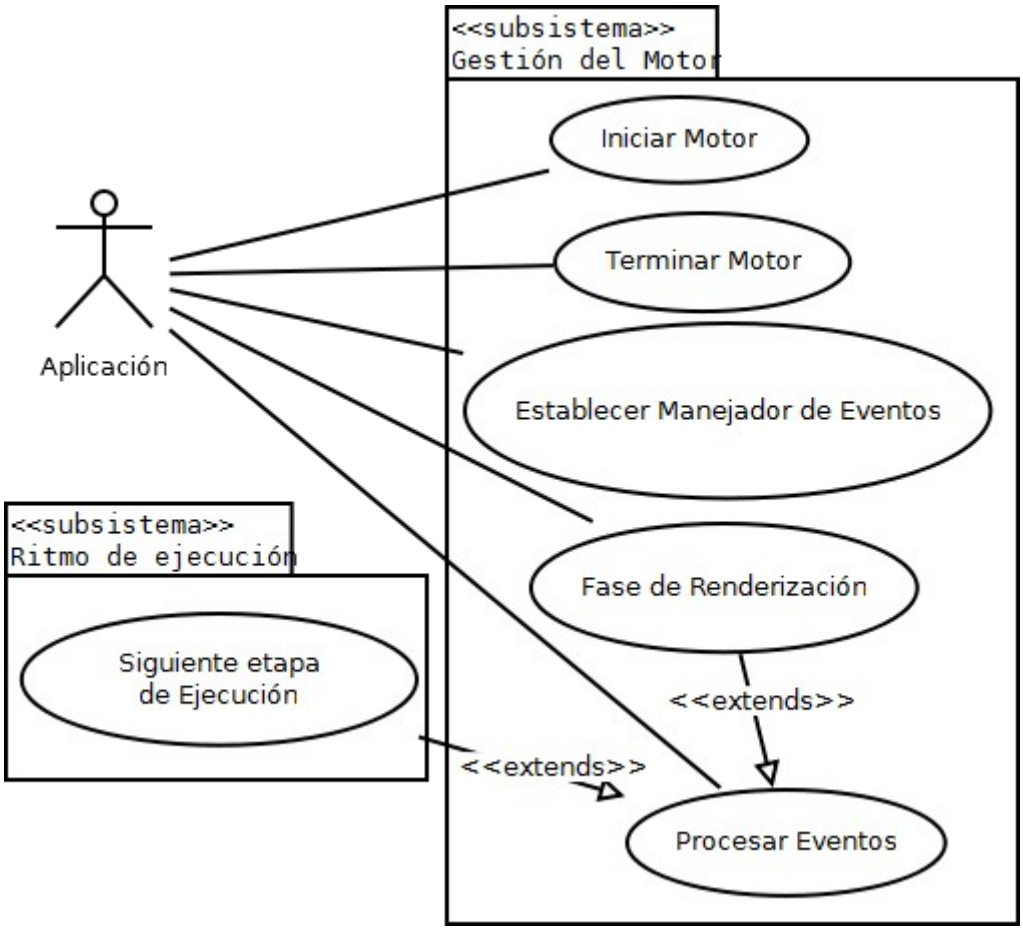


Figura 3.6 : Subsistema Gestión del Motor

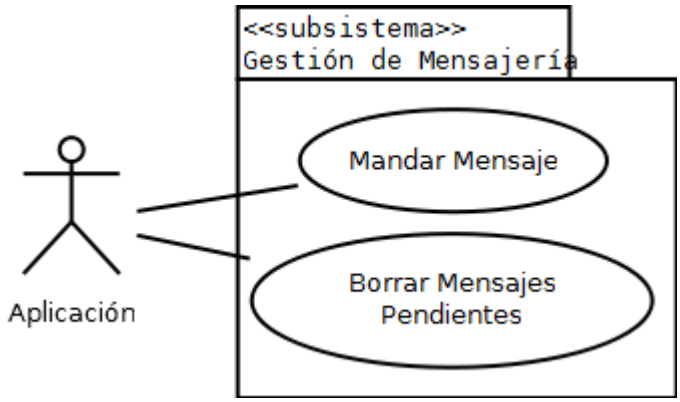


Figura 3.7 : Subsistema Gestión de Mensajería

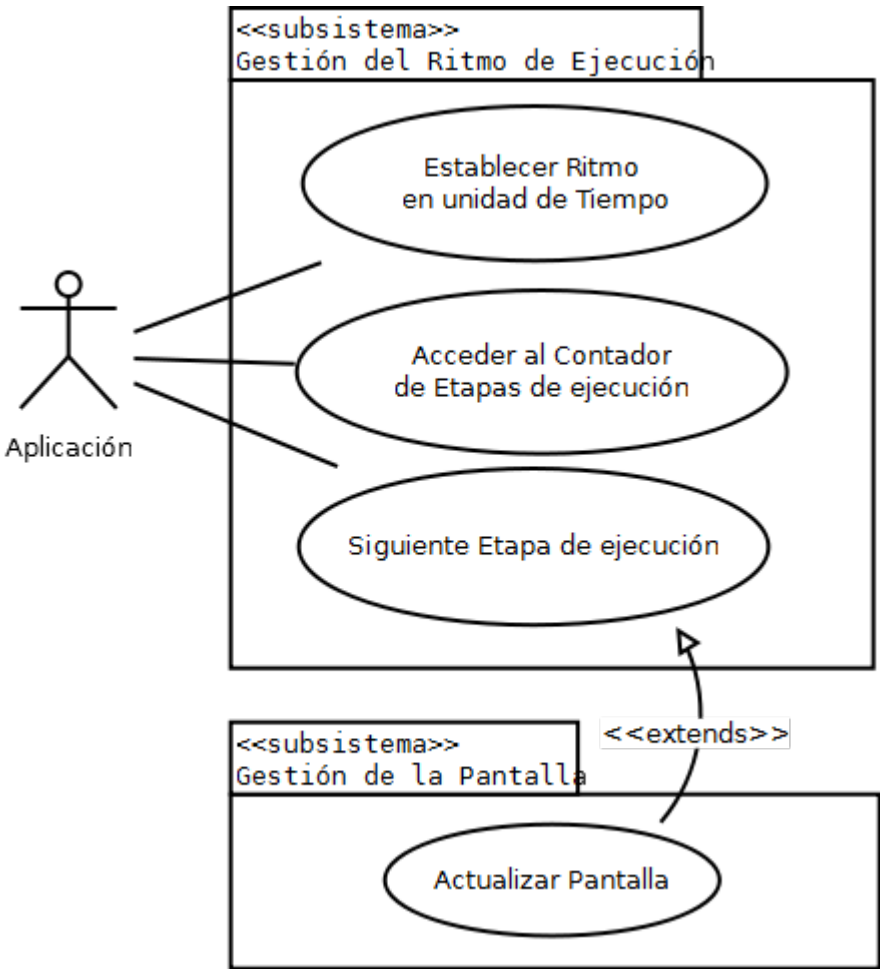


Figura 3.8 : Subsistema Gestión del Ritmo de Ejecución

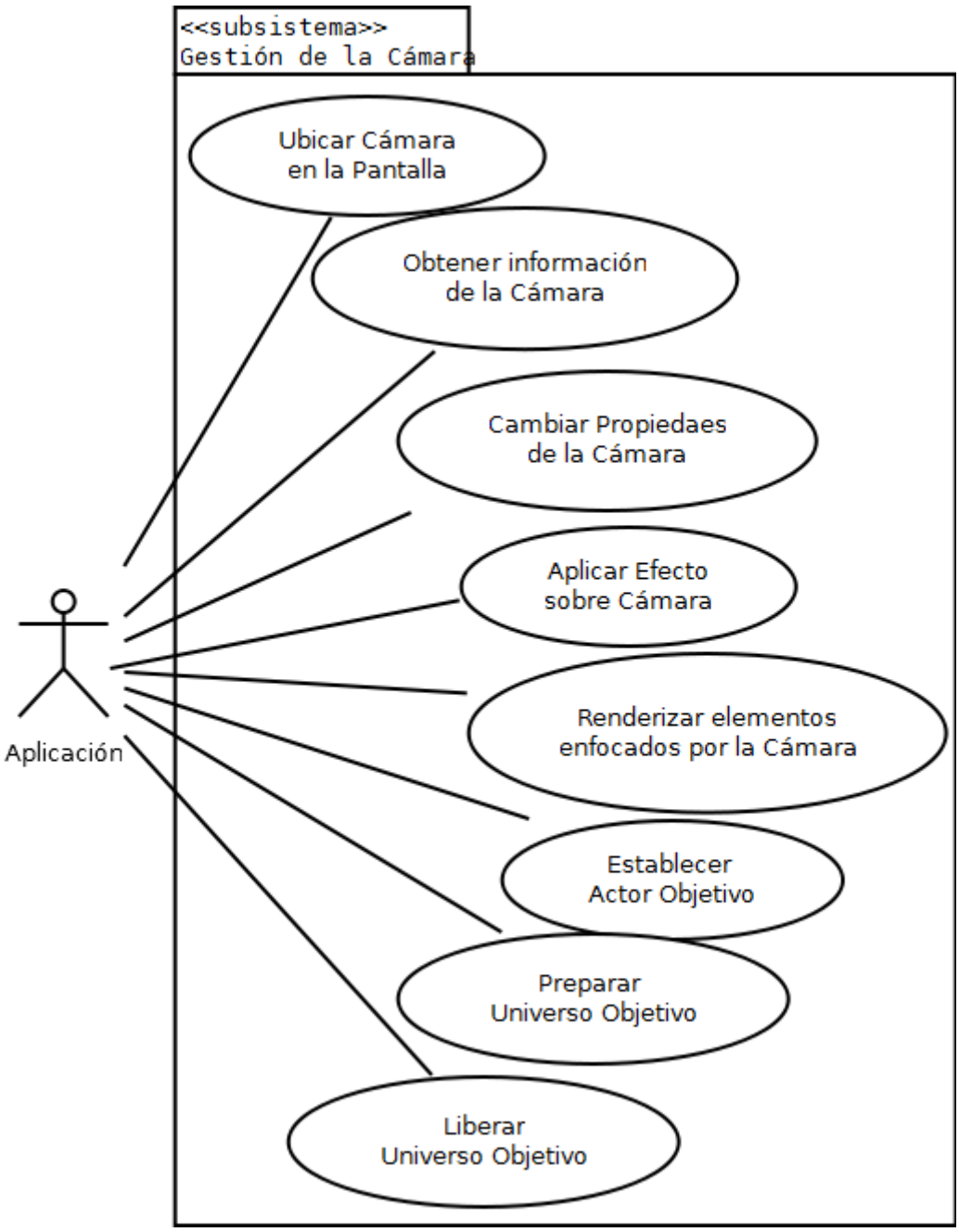


Figura 3.9 : Subsistema Gestión de la Cámara

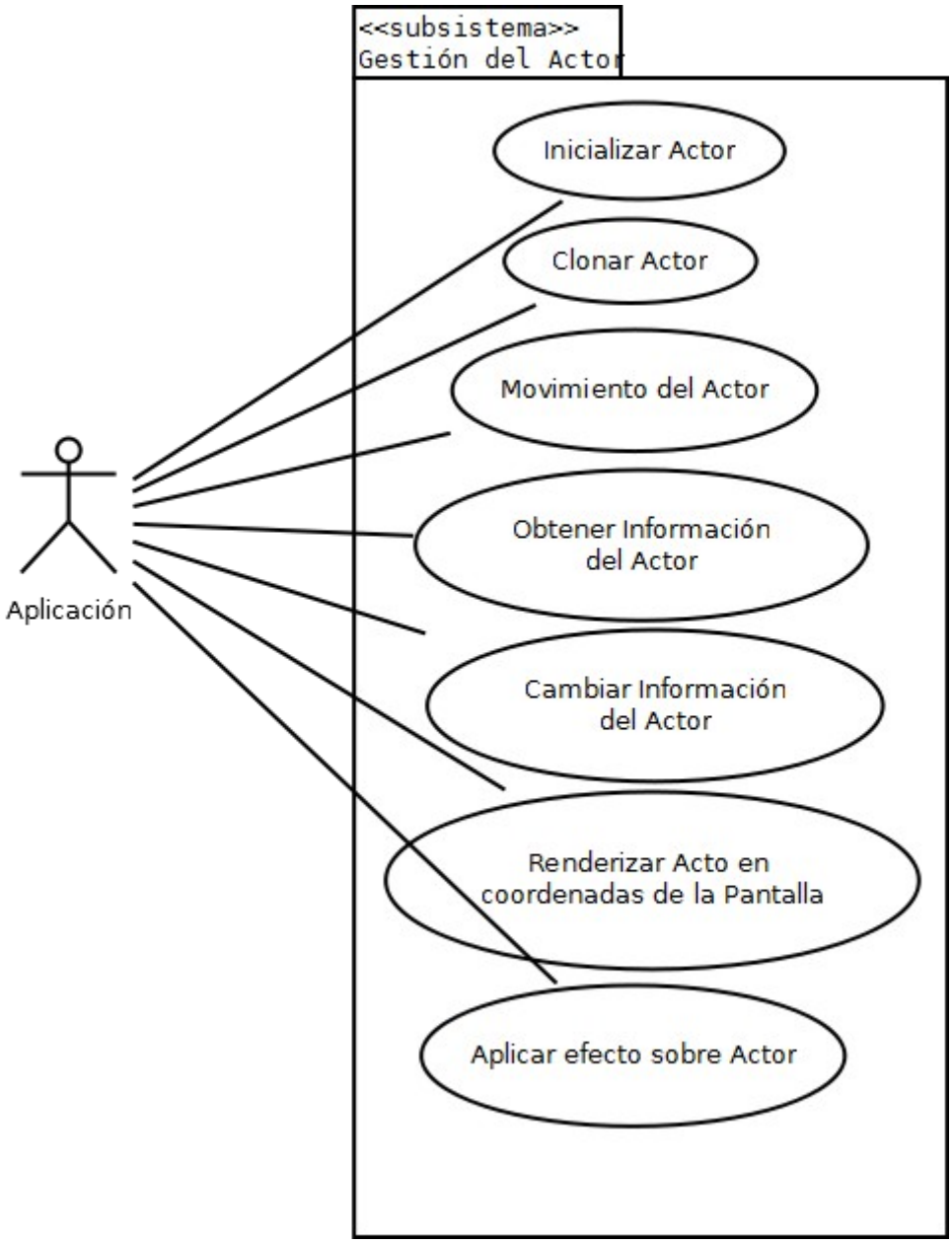


Figura 3.10 : Subsistema Gestión del Actor

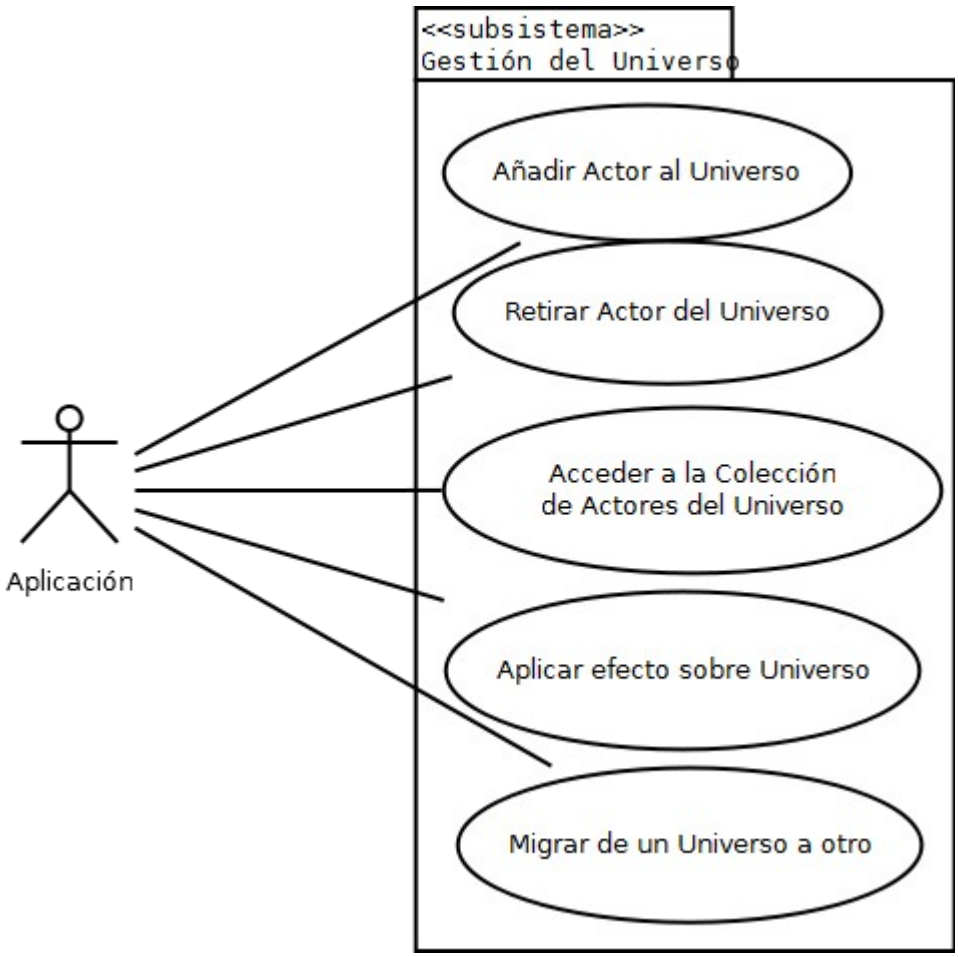


Figura 3.11 : Subsistema Gestión del Universo

3.2.2.2. Definición del Actor

ACT-0001	**Aplicación**
Versión	1.0 (15/06/2009)
Autores	• José Manuel Barroso Galindo
Fuentes	• Pablo Neira Ayuso
Descripción	Este actor representa *a la aplicación que hace uso de la Biblioteca.*
Comentarios	Ninguno

3.2.2.3. Casos de Uso del Sistema

3.2.2.3.1. Gestión General y de Motores

UC-0001	Iniciar la Biblioteca	
Versión	1.0 (15/06/2009)	
Autores	• José Manuel Barroso Galindo	
Fuentes	• Pablo Neira Ayuso	
Dependencias	• [OBJ-0001] Inicialización de Contexto de Vídeo • [OBJ-0013] Plantillas	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando *se decida empezar a hacer uso de la biblioteca* .	
Precondición	La biblioteca no ha sido inicializada.	
Secuencia normal	**Paso**	**Acción**
	1	El actor Aplicación (ACT-0001) *llama a la rutina de inicialización de la biblioteca, indicando información sobre la inicialización de la pantalla, u otros datos de inicialización, si lo cree conveniente.*
	2	El sistema *prepara el programa para que se pueda hacer uso de las funcionalidades de la biblioteca. A continuación devuelve ÉXITO.*
Postcondición	La biblioteca ha sido inicializada.	
Excepciones	**Paso**	**Acción**
	-	-
Importancia	vital	
Urgencia	inmediatamente	
Estado	validado	
Estabilidad	alta	
Comentarios	Ninguno	

UC-0044	Añadir Motor
Versión	1.0 (01/07/2009)
Autores	• José Manuel Barroso Galindo

Fuentes	• Pablo Neira Ayuso	
Dependencias	• [IRQ-0007] Motor • [OBJ-0011] Soporte de métodos Ayudantes • [OBJ-0013] Plantillas • [OBJ-0014] Alto grado de Encapsulación	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando *la aplicación desee añadir un Motor a la bilblioteca.*	
Precondición	La biblioteca ha sido inicializada.	
Secuencia normal	**Paso**	**Acción**
	1	El actor Aplicación (ACT-0001) *crea la instancia de un Motor y la transmite a la biblioteca.*
	2	El sistema *lo registra añadiéndolo a una colección de Motores.*
Postcondición	Se ha añadido al gestor de motores de la biblioteca un nuevo motor.	
Excepciones	**Paso**	**Acción**
	2	Si *el Motor no ha podido ser registrado por la biblioteca correctamente,* el sistema *devuelve FRACASO*, a continuación este caso de uso *queda sin efecto*
Importancia	vital	
Urgencia	inmediatamente	
Estado	validado	
Estabilidad	alta	
Comentarios	Ninguno	

UC-0002	**Procesar Motores**	
Versión	1.0 (15/06/2009)	
Autores	• José Manuel Barroso Galindo	
Fuentes	• Pablo Neira Ayuso	
Dependencias	• [IRQ-0007] Motor • [OBJ-0006] Ofrecer mecanismos para la manipulación de los eventos de Entrada/Salida	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando *se decida empezar a hacer uso de los Motores añadidos por la aplicación.*	
Precondición	La biblioteca ha sido inicializada.	
Secuencia normal	**Paso**	**Acción**

	1	El actor Aplicación (ACT-0001) *solicita a la biblioteca que procese los motores.*
	2	El sistema *selecciona un Motor de entre los motores registrados por la biblioteca según su prioridad y lo procesa. El Motor seleccionado pasa a ser el Motor activo.*
	3	Se realiza el caso de uso Procesar Eventos (UC-0051)
	4	El sistema *al acabar de procesar el Motor activo, se vuelve al Paso 2.*
Postcondición		
Excepciones	**Paso**	**Acción**
	2	Si *en el momento que se ha invocado este caso de uso la biblioteca no ha sido inicializada,* el sistema *devuelve FRACASO*, a continuación este caso de uso *queda sin efecto*
	2	Si *el motor seleccionado por cualquier motivo no es apto para ser procesado,* el sistema *devuelve FRACASO*, a continuación este caso de uso *continúa*
	2	Si *en el momento que se ha invocado este caso de uso, no había ningún Motor registrado por la biblioteca,* el sistema *devuelve ÉXITO*, a continuación este caso de uso *continúa*
Importancia	importante	
Urgencia	inmediatamente	
Estado	validado	
Estabilidad	alta	
Comentarios	Ninguno	

UC-0045	**Modificar prioridad de Motor**	
Versión	1.0 (01/07/2009)	
Autores	• José Manuel Barroso Galindo	
Fuentes	• Pablo Neira Ayuso	
Dependencias	• [IRQ-0007] Motor • [OBJ-0012] Métodos Ayudantes	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando *se quiera alterar la prioridad de alguno de los Motores.*	
Precondición		
Secuencia normal	**Paso**	**Acción**
	1	El actor Aplicación (ACT-0001) *indica la instancia del Motor*

		cuya prioridad quiera cambiar y el valor de prioridad deseado.
	2	El sistema *cumple dicho cambio y devuelve ÉXITO.*
Postcondición	Puede que se haya modificado la prioridad de un Motor.	
Excepciones	**Paso**	**Acción**
	2	Si *la instancia es nula o su prioridad no se puede modificar,* el sistema *devuelve FRACASO*, a continuación este caso de uso *queda sin efecto*
	2	Si *la biblioteca no se ha inicializado anteriormente,* el sistema *devuelve FRACASO*, a continuación este caso de uso *queda sin efecto*
Importancia	importante	
Urgencia	inmediatamente	
Estado	validado	
Estabilidad	alta	
Comentarios	Ninguno	

UC-0046	**Obtener Motor activo**	
Versión	1.0 (01/07/2009)	
Autores	• José Manuel Barroso Galindo	
Fuentes	• Pablo Neira Ayuso	
Dependencias	• [IRQ-0007] Motor • [OBJ-0012] Métodos Ayudantes	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando *se desee acceder al Motor activo, es decir, el Motor que está siendo procesado en este momento.*	
Precondición	La biblioteca ha sido inicializada.	
Secuencia normal	**Paso**	**Acción**
	1	El actor Aplicación (ACT-0001) *solicita la instancia del Motor activo.*
	2	El sistema *devuelve el Motor activo.*
Postcondición		
Excepciones	**Paso**	**Acción**
	2	Si *no hay ningún Motor activo en este momento,* el sistema *devuelve FRACASO*, a continuación este caso de uso *queda sin efecto*

Importancia	vital
Urgencia	inmediatamente
Estado	validado
Estabilidad	alta
Comentarios	Ninguno

UC-0105	**Retirar Motor**	
Versión	1.0 (02/11/2010)	
Autores	• José Manuel Barroso Galindo	
Fuentes	• Pablo Neira Ayuso	
Dependencias	• [IRQ-0007] Motor • [OBJ-0012] Métodos Ayudantes	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando *la aplicación desee retirar un Motor registrado por la bilblioteca.*	
Precondición	El Motor especificado está registrado por la biblioteca.	
Secuencia normal	**Paso**	**Acción**
	1	El actor Aplicación (ACT-0001) *indica la instancia de motor que quiera que se desregistre.*
	2	El sistema *lo desregistra borrándolo de la colección de Motores.*
Postcondición	El Motor especificado ya no está registrado por la biblioteca.	
Excepciones	**Paso**	**Acción**
	2	Si *el Motor no ha podido ser desregistrado por la biblioteca correctamente,* el sistema *devuelve FRACASO*, a continuación este caso de uso *queda sin efecto*
Importancia	quedaría bien	
Urgencia	puede esperar	
Estado	validado	
Estabilidad	alta	
Comentarios	Ninguno	

UC-0043	**Leer información sobre incidentes anotados por la Biblioteca**
Versión	1.0 (01/07/2009)
Autores	• José Manuel Barroso Galindo

Fuentes	• Pablo Neira Ayuso	
Dependencias	• [OBJ-0012] Métodos Ayudantes	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando *el usuario solicite conocer que errores, advertencias, o cualquier otro tipo de incidente hay registrado por la biblioteca como consecuencia de la actuación del usuario.*	
Precondición	La biblioteca ha sido inicializada.	
Secuencia normal	**Paso**	**Acción**
	1	El actor Aplicación (ACT-0001) *reclama información sobre los incidentes registrados por la Biblioteca.*
	2	El sistema *satisface la petición, devolviendo ÉXITO si la consulta ha producido algún tipo de información, o FRACASO si no.*
Postcondición	La información consultada se libera.	
Excepciones	**Paso**	**Acción**
	-	-
Importancia	importante	
Urgencia	inmediatamente	
Estado	validado	
Estabilidad	alta	
Comentarios	Ninguno	

3.2.2.3.2. Gestión de Imágenes

UC-0004	**Cargar Conjunto de Sprites**
Versión	1.0 (15/06/2009)
Autores	• José Manuel Barroso Galindo
Fuentes	• Pablo Neira Ayuso
Dependencias	• [IRQ-0003] Conjunto de Sprites • [OBJ-0002] Importación de fuentes de recursos externos • [OBJ-0014] Alto grado de Encapsulación

Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando *la aplicación solicite la importación de un Conjunto de Sprites.*	
Precondición	El Contexto de Vídeo debe estar inicializado.	
Secuencia normal	**Paso**	**Acción**
	1	El actor Aplicación (ACT-0001) *solicita la carga de un Conjunto de Sprites indicando un descriptor.*
	2	El sistema *devuelve el identificador correspondiente al Conjunto de Sprites obtenido a través del descriptor indicado.* (*Un identificador es un parámetro que identifica al objeto de forma unívoca.*)
Postcondición	El Conjunto de Sprites correspondiente está almacenado.	
Excepciones	**Paso**	**Acción**
	2	Si *se falla al intentar procesar el descriptor,* el sistema *devuelve FRACASO*, a continuación este caso de uso *queda sin efecto*
Importancia	importante	
Urgencia	inmediatamente	
Estado	validado	
Estabilidad	alta	
Comentarios	Ninguno	

UC-0003	**Buscar Conjunto de Sprites**	
Versión	1.0 (15/06/2009)	
Autores	• José Manuel Barroso Galindo	
Fuentes	• Pablo Neira Ayuso	
Dependencias	• [IRQ-0003] Conjunto de Sprites • [OBJ-0012] Métodos Ayudantes	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando *la aplicación requiera el identificador de un Conjunto de Sprites particular.*	
Precondición	Hay algún Conjunto de Sprites almacenado.	
Secuencia normal	**Paso**	**Acción**
	1	El actor Aplicación (ACT-0001) *solicita conocer el identificador de un Conjunto de Sprites indicando un descriptor, o la instancia del Conjunto de Sprites.*

	2	El sistema *busca el Conjuntos de Sprites que tenga el descriptor indicado o sea la instancia indicada, y devuelve su identificador.*
Postcondición		
Excepciones	**Paso**	**Acción**
	2	Si *no se encuentra ningún resultado,* el sistema *devuelve FRACASO*, a continuación este caso de uso *continúa*
Importancia	importante	
Urgencia	inmediatamente	
Estado	validado	
Estabilidad	alta	
Comentarios	Ninguno	

UC-0005	**Eliminar Conjunto de Sprites**	
Versión	1.0 (15/06/2009)	
Autores	• José Manuel Barroso Galindo	
Fuentes	• Pablo Neira Ayuso	
Dependencias	• [IRQ-0003] Conjunto de Sprites • [OBJ-0012] Métodos Ayudantes	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando *se comunique el cese de uso de un Cojunto de Sprites, un subconjunto, o todos.*	
Precondición	El Conjunto de Sprites con el identificador dado debe estar almacenado.	
Secuencia normal	**Paso**	**Acción**
	1	El actor Aplicación (ACT-0001) *solicita descargar un Conjunto de Sprites, indicando su identificador.*
	2	El sistema *procesa la petición para el Conjunto de Sprites correspondiente, borrándolo de la memoria si es pertinente. A continuación devuelve ÉXITO.*
Postcondición	El Conjunto de Sprites puede haber sido desalmacenado y borrado de la memoria.	
Excepciones	**Paso**	**Acción**
	2	Si *no localiza el Conjunto de Sprites requerido,* el sistema *devuelve FRACASO*, a continuación este caso de uso *queda sin efecto*
Importancia	importante	

Urgencia	inmediatamente
Estado	validado
Estabilidad	alta
Comentarios	Ninguno

UC-0007	**Obtener instancia de un Conjunto de Sprites**	
Versión	1.0 (17/06/2009)	
Autores	• José Manuel Barroso Galindo	
Fuentes	• Pablo Neira Ayuso	
Dependencias	• [IRQ-0003] Conjunto de Sprites • [OBJ-0012] Métodos Ayudantes	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando *la aplicación quiera utilizar un Conjunto de Sprites ya almacenado.* o durante la realización de los siguientes casos de uso: [UC-0098] Obtener Sprite de un Conjunto de Sprites	
Precondición	Hay almacenado un Conjunto de Sprites asociado al identificador.	
Secuencia normal	**Paso**	**Acción**
	1	El actor Aplicación (ACT-0001) *solicita el Conjunto de Sprites asociado a un identificador dado.*
	2	El sistema *devuelve el Conjunto de Sprites correspondiente.*
Postcondición		
Excepciones	**Paso**	**Acción**
	2	Si *no encuentra ningún Conjunto de Sprites asociado a dicho identificador,* el sistema *devuelve FRACASO*, a continuación este caso de uso *queda sin efecto*
Importancia	importante	
Urgencia	inmediatamente	
Estado	validado	
Estabilidad	alta	
Comentarios	Ninguno	

UC-0098	**Obtener Sprite de un Conjunto de Sprites**
Versión	1.0 (13/07/2009)
Autores	• José Manuel Barroso Galindo

Fuentes	• Pablo Neira Ayuso	
Dependencias	• [IRQ-0002] Sprite • [OBJ-0014] Alto grado de Encapsulación • [OBJ-0012] Métodos Ayudantes • [IRQ-0003] Conjunto de Sprites	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando *la aplicación solicite el uso de un Sprite perteneciente a un conjunto.* o durante la realización de los siguientes casos de uso: [UC-0100] Aplicar efecto sobre Sprite, [UC-0101] Obtener información de un Sprite, [UC-0102] Cambiar información de un Sprite, [UC-0103] Renderizar un Sprite en coordenadas de la Pantalla	
Precondición	El Contexto de Vídeo debe estar inicializado.	
Secuencia normal	**Paso**	**Acción**
	1	Se realiza el caso de uso Obtener instancia de un Conjunto de Sprites (UC-0007)
	2	El actor Aplicación (ACT-0001) *solicita un Sprite indicando su identificador en el conjunto de Sprites que lo contiene.*
	3	El sistema *devuelve al usuario una instancia válida para el Sprite requerido.*
Postcondición		
Excepciones	**Paso**	**Acción**
	1	Si *este caso de uso ha fallado,* el sistema *devuelve FRACASO*, a continuación este caso de uso *queda sin efecto*
	3	Si *el identificador no es válido,* el sistema *devuelve FRACASO*, a continuación este caso de uso *queda sin efecto*
Importancia	importante	
Urgencia	inmediatamente	
Estado	validado	
Estabilidad	alta	
Comentarios	Ninguno	

UC-0101	**Obtener información de un Sprite**
Versión	1.0 (20/07/2009)
Autores	• José Manuel Barroso Galindo
Fuentes	• Pablo Neira Ayuso

Dependencias	• [IRQ-0002] Sprite • [OBJ-0012] Métodos Ayudantes	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando *la aplicación solicite información de un Sprite perteneciente a un conjunto.*	
Precondición	El Contexto de Vídeo debe estar inicializado.	
Secuencia normal	**Paso**	**Acción**
	1	Se realiza el caso de uso Obtener Sprite de un Conjunto de Sprites (UC-0098)
	2	El actor Aplicación (ACT-0001) *solicita información determinada sobre un Sprite.*
	3	El sistema *satisface la petición.*
Postcondición		
Excepciones	**Paso**	**Acción**
	1	Si *este caso de uso ha fallado,* el sistema *devuelve FRACASO*, a continuación este caso de uso *queda sin efecto*
	3	Si *se falla al intentar acceder a la información especificada,* el sistema *devuelve FRACASO*, a continuación este caso de uso *queda sin efecto*
Importancia	importante	
Urgencia	inmediatamente	
Estado	validado	
Estabilidad	alta	
Comentarios	Ninguno	

UC-0102	**Cambiar información de un Sprite**	
Versión	1.0 (20/07/2009)	
Autores	• José Manuel Barroso Galindo	
Fuentes	• Pablo Neira Ayuso	
Dependencias	• [IRQ-0002] Sprite • [OBJ-0012] Métodos Ayudantes	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando *la aplicación solicite cambiar información de un Sprite perteneciente a un conjunto.*	
Precondición	El Contexto de Vídeo debe estar inicializado.	
Secuencia normal	**Paso**	**Acción**

	1	Se realiza el caso de uso Obtener Sprite de un Conjunto de Sprites (UC-0098)
	2	El actor Aplicación (ACT-0001) *solicita modificar la información de un Sprite, indicando el cambio deseado.*
	3	El sistema *satisface dicha petición. A continuación, devuelve ÉXITO.*
Postcondición	La información del Sprite ha cambiado.	
Excepciones	**Paso**	**Acción**
	1	Si *este caso de uso ha fallado,* el sistema *devuelve FRACASO*, a continuación este caso de uso *queda sin efecto*
	3	Si *se falla al intentar cambiar la información especificada,* el sistema *devuelve FRACASO*, a continuación este caso de uso *queda sin efecto*
Importancia	importante	
Urgencia	inmediatamente	
Estado	validado	
Estabilidad	alta	
Comentarios	Ninguno	

UC-0100	**Aplicar efecto sobre Sprite**	
Versión	1.0 (13/07/2009)	
Autores	• José Manuel Barroso Galindo	
Fuentes	• Pablo Neira Ayuso	
Dependencias	• [IRQ-0002] Sprite • [OBJ-0007] Soporte de efectos gráficos adicionales	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando *se quiera alterar la representación gráfica del Sprite. Dicho efecto puede consistir en cambiar los valores de color, aplicar transparencias, zooms y rotaciones, así como traslaciones o espejados.*	
Precondición	El Contexto de Vídeo debe estar inicializado.	
Secuencia normal	**Paso**	**Acción**
	1	Se realiza el caso de uso Obtener Sprite de un Conjunto de Sprites (UC-0098)
	2	El actor Aplicación (ACT-0001) *especifica el efecto gráfico que quiere que sufra el Sprite en la siguiente fase de renderización.*

	3	El sistema *satisface dicha petición, y en la fase de renderización de los elementos que usen el Sprite, aplicará los efectos gráficos especificados. A continuación devuelve ÉXITO.*
Postcondición	El Sprite registra un efecto gráfico para la siguiente fase de renderización.	
Excepciones	**Paso**	**Acción**
	1	Si *este caso de uso ha fallado,* el sistema *devuelve FRACASO*, a continuación este caso de uso *queda sin efecto*
	3	Si *el efecto gráfico no es aplicable,* el sistema *devuelve FRACASO*, a continuación este caso de uso *queda sin efecto*
Importancia	quedaría bien	
Urgencia	hay presión	
Estado	validado	
Estabilidad	alta	
Comentarios	Ninguno	

UC-0103	**Renderizar un Sprite en coordenadas de la Pantalla**	
Versión	1.0 (20/07/2009)	
Autores	• José Manuel Barroso Galindo	
Fuentes	• Pablo Neira Ayuso	
Dependencias	• [IRQ-0002] Sprite • [OBJ-0005] Soporte de Renderización	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando *se quiera introducir dicho Sprite en el siguiente renderizado, de acuerdo con unas coordenadas de la pantalla indicadas.*	
Precondición	El Contexto de Vídeo debe estar inicializado.	
Secuencia normal	**Paso**	**Acción**
	1	Se realiza el caso de uso Obtener Sprite de un Conjunto de Sprites (UC-0098)
	2	El actor Aplicación (ACT-0001) *especifica las coordenadas en las cuales desea representar el Sprite en Pantalla.*
	3	El sistema *llama a la correspondiente fase de renderización del Sprite. A continuación devuelve ÉXITO.*
Postcondición		
Excepciones	**Paso**	**Acción**
	1	Si *este caso de uso ha fallado,* el sistema *devuelve FRACASO*, a

		continuación este caso de uso *queda sin efecto*
	3	Si *durante este proceso ha ocurrido algún error,* el sistema *devuelve FRACASO*, a continuación este caso de uso *queda sin efecto*
Importancia	quedaría bien	
Urgencia	hay presión	
Estado	validado	
Estabilidad	alta	
Comentarios	Ninguno	

3.2.2.3.3. Gestión de la Pantalla

UC-0011	**Inicializar Contexto de Vídeo**	
Versión	1.0 (17/06/2009)	
Autores	• José Manuel Barroso Galindo	
Fuentes	• Pablo Neira Ayuso	
Dependencias	• [IRQ-0005] Contexto de Vídeo • [OBJ-0001] Inicialización de Contexto de Vídeo	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando *la aplicación considera necesario inicializar el contexto de vídeo* o durante la realización de los siguientes casos de uso: [UC-0013] Reiniciar Contexto de Vídeo	
Precondición	No está inicializado el Contexto de Vídeo.	
Secuencia normal	**Paso**	**Acción**
	1	El actor Aplicación (ACT-0001) *solicita iniciar el Contexto de Vídeo indicando las dimensiones de la resolución a adoptar, la profundidad de bits del color, si se desea entrar en modo pantalla completa o ventana, si se va a hacer uso del doble buffer o no, etc..*
	2	El sistema *crea el Contexto de Vídeo intentando satisfacer las características. A continuación devuelve ÉXITO.*
Postcondición	Está inicializado el Contexto de Vídeo.	
Excepciones	**Paso**	**Acción**
	2	Si *no es capaz de satisfacer las características requeridas,* el sistema *crea el Contexto de Vídeo con una configuración por*

		defecto, y a continuación devuelve FRACASO, a continuación este caso de uso *continúa*
Importancia	importante	
Urgencia	inmediatamente	
Estado	validado	
Estabilidad	alta	
Comentarios	Ninguno	

UC-0012	**Finalizar Contexto de Vídeo**	
Versión	1.0 (17/06/2009)	
Autores	• José Manuel Barroso Galindo	
Fuentes	• Pablo Neira Ayuso	
Dependencias	• [IRQ-0005] Contexto de Vídeo	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso abstracto durante la realización de los siguientes casos de uso: [UC-0013] Reiniciar Contexto de Vídeo	
Precondición	El Contexto de Vídeo está inicializado.	
Secuencia normal	**Paso**	**Acción**
	1	El actor Aplicación (ACT-0001) *solicita finalizar el Contexto de Vídeo. A continuación devuelve ÉXITO*
	2	El sistema *elimina el Contexto de Vídeo.*
Postcondición	El Contexto de Vídeo no está inicializado.	
Excepciones	**Paso**	**Acción**
	-	-
Importancia	importante	
Urgencia	inmediatamente	
Estado	validado	
Estabilidad	media	
Comentarios	Ninguno	

UC-0013	**Reiniciar Contexto de Vídeo**
Versión	1.0 (17/06/2009)
Autores	• José Manuel Barroso Galindo

Fuentes	• Pablo Neira Ayuso	
Dependencias	• [IRQ-0005] Contexto de Vídeo • [OBJ-0001] Inicialización de Contexto de Vídeo	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando *la aplicación considera necesario cambiar algún atributo del Contexto de Vídeo.*	
Precondición	Está inicializado el Contexto de Vídeo.	
Secuencia normal	**Paso**	**Acción**
	1	El actor Aplicación (ACT-0001) *solicita nuevos requisitos para el Contexto de Vídeo. Dimensiones de la resolución a adoptar, la profundidad de bits del color, si se desea entrar en modo pantalla completa o ventana y si se va a hacer uso del doble buffer o no.*
	2	Se realiza el caso de uso Finalizar Contexto de Vídeo (UC-0012)
	3	Se realiza el caso de uso Inicializar Contexto de Vídeo (UC-0011)
Postcondición		
Excepciones	**Paso**	**Acción**
	2 y 3	Si *se ha devuelto FRACASO en dicho caso de uso,* el sistema *devuelve FRACASO*, a continuación este caso de uso *continúa*
Importancia	importante	
Urgencia	inmediatamente	
Estado	validado	
Estabilidad	alta	
Comentarios	Ninguno	

UC-0018	**Actualizar la Pantalla**
Versión	1.0 (17/06/2009)
Autores	• José Manuel Barroso Galindo
Fuentes	• Pablo Neira Ayuso
Dependencias	• [OBJ-0005] Soporte de Renderización
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando *la aplicación solicite refrescar la pantalla, osea actualizar en la pantalla todos los elementos gráficos del sistema,* o durante la

	realización de los siguientes casos de uso: [UC-0064] Siguiente Etapa de ejecución	
Precondición	Hay un Contexto de Vídeo inicializado.	
Secuencia normal	**Paso**	**Acción**
	1	El actor Aplicación (ACT-0001) *solicita actualizar la Pantalla.*
	2	El sistema *muestra en pantalla el resultado de la actual fase de renderización y se prepara para la siguiente fase de renderización.*
Postcondición	La Pantalla está actualizada.	
Excepciones	**Paso**	**Acción**
	-	-
Importancia	vital	
Urgencia	inmediatamente	
Estado	validado	
Estabilidad	alta	
Comentarios	Ninguno	

UC-0019	**Borrar la Pantalla**	
Versión	1.0 (17/06/2009)	
Autores	• José Manuel Barroso Galindo	
Fuentes	• Pablo Neira Ayuso	
Dependencias	• [OBJ-0012] Métodos Ayudantes	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando *la aplicación solicite borrar la pantalla.*	
Precondición	Hay un Contexto de Vídeo inicializado.	
Secuencia normal	**Paso**	**Acción**
	1	El actor Aplicación (ACT-0001) *solicita borrar la Pantalla.*
	2	El sistema *ignora toda la información gráfica que había sido renderizada desde la última fase de renderización.*
Postcondición	Toda la información gráfica especificada desde la última fase de renderización se ha borrado.	
Excepciones	**Paso**	**Acción**
	-	-
Importancia	importante	

Urgencia	inmediatamente
Estado	validado
Estabilidad	alta
Comentarios	Ninguno

UC-0020	**Situar Escena 2D en la Pantalla**	
Versión	1.0 (17/06/2009)	
Autores	• José Manuel Barroso Galindo	
Fuentes	• Pablo Neira Ayuso	
Dependencias	• [OBJ-0012] Métodos Ayudantes	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando *la aplicación indica un marco dentro de la Pantalla en el que la información gráfica que se especificará a continuación se va a representar durante la actual fase de renderización,*	
Precondición	Hay un Contexto de Vídeo inicializado.	
Secuencia normal	**Paso**	**Acción**
	1	El actor Aplicación (ACT-0001) *solicita establecer un nuevo marco para especificar una Escena 2D, indicando la posición en la pantalla, sus dimensiones, y el escalado.*
	2	El sistema *fija el marco de manera acorde a los parámetros requeridos, y tiene en cuenta el escalado base del Contexto de Vídeo que se está usando.*
Postcondición	Toda la información gráfica especificada a partir de entonces se hará en base al marco de la escena establecido.	
Excepciones	**Paso**	**Acción**
	2	Si *los parametros indicados son inválidos,* el sistema *registra el error*, a continuación este caso de uso *queda sin efecto*
Importancia	importante	
Urgencia	inmediatamente	
Estado	validado	
Estabilidad	alta	
Comentarios	Ninguno	

UC-0097	**Aplicar efecto sobre Pantalla**
Versión	1.0 (07/07/2009)

Autores	• José Manuel Barroso Galindo	
Fuentes	• Pablo Neira Ayuso	
Dependencias	• [OBJ-0007] Soporte de efectos gráficos adicionales	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando *se quiera alterar la representación de todo lo que se va a visualizar por la pantalla. Dicho efecto puede consistir en variar los valores de color, desfigurar o recortar la imagen como resultado de un efecto de transición de escena, desplazar la imagen como resultado de un efecto de temblor, ampliar la imagen de acuerdo a un efecto de zoom, etc...*	
Precondición	El Contexto de Vídeo ha sido inicializado.	
Secuencia normal	**Paso**	**Acción**
	1	El actor Aplicación (ACT-0001) *especifica el efecto gráfico que quiere producir en la siguiente fase de renderización.*
	2	El sistema *satisface dicha petición, y de cara a la siguiente actualización aplicará los efectos gráficos esperados. A continuación devuelve ÉXITO.*
Postcondición	La Pantalla registra un efecto para la siguiente fase de renderización.	
Excepciones	**Paso**	**Acción**
	2	Si *el efecto gráfico no es aplicable,* el sistema *devuelve FRACASO*, a continuación este caso de uso *queda sin efecto*
Importancia	quedaría bien	
Urgencia	hay presión	
Estado	validado	
Estabilidad	alta	
Comentarios	Ninguno	

3.2.2.3.4. Gestión de Textos

UC-0021	**Establecer Material predeterminado de Fuente**
Versión	1.0 (17/06/2009)
Autores	• José Manuel Barroso Galindo
Fuentes	• Pablo Neira Ayuso

Dependencias	• [IRQ-0004] Fuente • [OBJ-0008] Primitivas Gráficas	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando *se quiera especificar el material a utilizar para las próximas Fuentes a cargar. Con material englobamos características gráficas tales como tamaño, color, etc..*	
Precondición		
Secuencia normal	**Paso**	**Acción**
	1	El actor Aplicación (ACT-0001) *especifica un material válido para la siguiente o siguientes Fuentes a cargar.*
	2	El sistema *declara el material especificado como material predeterminado de Fuente, y devuelve el material anterior como resultado.*
Postcondición	El valor de material predeterminado pasa a ser el especificado.	
Excepciones	**Paso**	**Acción**
	2	Si *el material especificado no es válido,* el sistema *devuelve FRACASO*, a continuación este caso de uso *queda sin efecto*
Importancia	importante	
Urgencia	inmediatamente	
Estado	validado	
Estabilidad	alta	
Comentarios	Con material englobamos características gráficas tales como tamaño, color, etc..	

UC-0024	**Cargar Fuente**	
Versión	1.0 (17/06/2009)	
Autores	• José Manuel Barroso Galindo	
Fuentes	• Pablo Neira Ayuso	
Dependencias	• [IRQ-0004] Fuente • [OBJ-0012] Métodos Ayudantes	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando *la aplicación solicite la importación de una fuente.*	
Precondición	El Contexto de Vídeo está inicializado.	
Secuencia normal	**Paso**	**Acción**
	1	El actor Aplicación (ACT-0001) *solicita la carga de una Fuente indicando su descriptor.*

	2	El sistema *devuelve al usuario un identificador que está asociado a la fuente requerida.* (*Un identificador es un parámetro que identifica al objeto de forma unívoca.*)
Postcondición	La Fuente correspondiente está almacenada y con un identificador asociado.	
Excepciones	**Paso**	**Acción**
	2	Si *no se ha podido referenciar una fuente con dicho nombre porque no es válido,* el sistema *devuelve FRACASO*, a continuación este caso de uso *queda sin efecto*
Importancia	importante	
Urgencia	inmediatamente	
Estado	validado	
Estabilidad	alta	
Comentarios	Ninguno	

UC-0023	**Buscar Fuente**	
Versión	1.0 (17/06/2009)	
Autores	• José Manuel Barroso Galindo	
Fuentes	• Pablo Neira Ayuso	
Dependencias	• [IRQ-0004] Fuente • [OBJ-0012] Métodos Ayudantes	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando *la aplicación requiera el identificador de una fuente concreta.*	
Precondición		
Secuencia normal	**Paso**	**Acción**
	1	El actor Aplicación (ACT-0001) *solicita conocer el identificador de una Fuente indicando su descriptor, el identificador de un texto que la use, u otros atributos que tengan una correspondencia con una única fuente.*
	2	El sistema *busca la Fuente que satisface la petición, y devuelve su identificador.*
Postcondición		
Excepciones	**Paso**	**Acción**
	2	Si *no se encuentra ningún resultado,* el sistema *devuelve FRACASO*, a continuación este caso de uso *continúa*

Importancia	importante
Urgencia	inmediatamente
Estado	validado
Estabilidad	alta
Comentarios	Ninguno

UC-0025	**Eliminar Fuente**	
Versión	1.0 (17/06/2009)	
Autores	• José Manuel Barroso Galindo	
Fuentes	• Pablo Neira Ayuso	
Dependencias	• [IRQ-0004] Fuente • [OBJ-0012] Métodos Ayudantes	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando *se comunique el cese del uso de una Fuente.*	
Precondición	La Fuente con el identificador dado debe estar almacenado.	
Secuencia normal	**Paso**	**Acción**
	1	El actor Aplicación (ACT-0001) *solicita descargar una Fuente, indicando su identificador.*
	2	El sistema *procesa la petición para la Fuente correspondiente, borrándola de memoria si es pertinente. A continuación, devuelve ÉXITO.*
Postcondición	La Fuente puede haber sido desalmacenada y borrada de la memoria.	
Excepciones	**Paso**	**Acción**
	2	Si *la búsqueda de la Fuente almacenada no da ningún resultado,* el sistema *devuelve FRACASO*, a continuación este caso de uso *queda sin efecto*
Importancia	importante	
Urgencia	inmediatamente	
Estado	validado	
Estabilidad	alta	
Comentarios	Ninguno	

UC-0026	**Cargar Texto**
Versión	1.0 (17/06/2009)
Autores	• José Manuel Barroso Galindo

Fuentes	• Pablo Neira Ayuso
Dependencias	• [IRQ-0001] Texto • [OBJ-0008] Primitivas Gráficas
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando *solicite por parte del usuario representar cualquier tipo de texto en la pantalla. Dicha representación se hará efectiva durante la fase de renderización de texto.*
Precondición	El Contexto de Vídeo se ha inicializado.

Secuencia normal	**Paso**	**Acción**
	1	El actor Aplicación (ACT-0001) *solicita mostrar texto en un formato determinado, indicando el identificador de fuente elegido y con unos valores de posición en la pantalla específicos.*
	2	El sistema *crea el texto correspondiente, y devuelve el identificador asociado.* (*Un identificador es un parámetro que identifica al objeto de forma unívoca.*)

Postcondición	Se ha almacenado un nuevo texto al que se le ha asociado un identificador.

Excepciones	**Paso**	**Acción**
	2	Si *la cadena de texto no contiene ningún carácter, sólo contiene carácteres sin representación gráfica o tiene algún carácter conflictivo,* el sistema *devuelve FRACASO*, a continuación este caso de uso *queda sin efecto*
	2	Si *el identificador de la fuente no tiene correspondencia con ninguna fuente almacenada,* el sistema *devuelve FRACASO*, a continuación este caso de uso *queda sin efecto*

Importancia	importante
Urgencia	inmediatamente
Estado	validado
Estabilidad	alta
Comentarios	Ninguno

UC-0027	**Eliminar Texto**
Versión	1.0 (17/06/2009)
Autores	• José Manuel Barroso Galindo
Fuentes	• Pablo Neira Ayuso

Dependencias	• [IRQ-0001] Texto • [OBJ-0012] Métodos Ayudantes	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando *solicite eliminar un texto determinado.*	
Precondición		
Secuencia normal	**Paso**	**Acción**
	1	El actor Aplicación (ACT-0001) *solicita eliminar el texto con el identificador dado o un conjunto de textos en base a un criterio especificado.*
	2	El sistema *elimina el texto o textos especificados.*
Postcondición	Se ha borrado algún texto con caracter inmediato o de cara a la siguiente fase de renderización.	
Excepciones	**Paso**	**Acción**
	2	Si *se ha optado eliminar un texto a través de un identificador que no tiene correspondencia con ningún texto almacenado,* el sistema *devuelve FRACASO*, a continuación este caso de uso *queda sin efecto*
Importancia	importante	
Urgencia	inmediatamente	
Estado	validado	
Estabilidad	alta	
Comentarios	Ninguno	

UC-0030	**Establecer criterio de visualización de Textos**	
Versión	1.0 (18/06/2009)	
Autores	• José Manuel Barroso Galindo	
Fuentes	• Pablo Neira Ayuso	
Dependencias	• [IRQ-0001] Texto • [OBJ-0012] Métodos Ayudantes	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando *se desee que los textos especificados mediante su identificador sólo sean visibles en una determinada zona de la pantalla.*	
Precondición	Se ha inicializado el Contexto de Vídeo.	
Secuencia normal	**Paso**	**Acción**

	1	El actor Aplicación (ACT-0001) *establece un área de la pantalla y el criterio de visualización deseado.*
	2	El sistema *almacena el área especificada junto al criterio y los selecciona para su uso a continuación.*
	3	El actor Aplicación (ACT-0001) *indica los identificadores de texto sobre los que tendrá efecto el criterio de visualización.*
	4	El sistema *mostrará los textos con las restricciones visuales requeridas en la siguiente fase de renderización del texto.*
Postcondición	Se han asociado los Textos indicados a la nueva Área, que ahora está almacenada.	
Excepciones	**Paso**	**Acción**
	2	Si *la posición y la dimensión del área requerida, sitúa a dicha área completamente fuera de la pantalla,* el sistema *devuelve FRACASO*, a continuación este caso de uso *queda sin efecto*
	4	Si *el identificador de texto no tiene correspondencia con ningún texto almacenado,* el sistema *devuelve FRACASO*, a continuación este caso de uso *queda sin efecto*
Importancia	importante	
Urgencia	inmediatamente	
Estado	validado	
Estabilidad	alta	
Comentarios	Ninguno	

UC-0031	**Cambiar Atributos asociados al Texto**	
Versión	1.0 (18/06/2009)	
Autores	• José Manuel Barroso Galindo	
Fuentes	• Pablo Neira Ayuso	
Dependencias	• [IRQ-0001] Texto • [OBJ-0012] Métodos Ayudantes	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando *se quiera cambiar la información y características de un texto especificado por su identificador.*	
Precondición		
Secuencia normal	**Paso**	**Acción**
	1	El actor Aplicación (ACT-0001) *solicita cambiar el texto del identificador dado, especificando los nuevos valores que ha de adoptar en los campos que van a cambiar.*

	2	El sistema *ejecuta dicha petición y devuelve ÉXITO.*
Postcondición	El texto almacenado indicado ha sido movido.	
Excepciones	**Paso**	**Acción**
	2	Si *el identificador del texto no se corresponde con ningún texto,* el sistema *devuelve FRACASO*, a continuación este caso de uso *queda sin efecto*
	2	Si *se ha especificado que un valor no válido para un atributo,* el sistema *devuelve FRACASO*, a continuación este caso de uso *queda sin efecto*
Importancia	quedaría bien	
Urgencia	puede esperar	
Estado	validado	
Estabilidad	alta	
Comentarios	Ninguno	

UC-0029	**Fase de Renderización de Textos**	
Versión	1.0 (18/06/2009)	
Autores	• José Manuel Barroso Galindo	
Fuentes	• Pablo Neira Ayuso	
Dependencias	• [OBJ-0005] Soporte de Renderización • [OBJ-0013] Plantillas	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando *se solicite iniciar la fase de renderización de textos.*	
Precondición	Se ha inicializado el Contexto de Vídeo.	
Secuencia normal	**Paso**	**Acción**
	1	El actor Aplicación (ACT-0001) *solicita iniciar la fase de renderización de texto.*
	2	El sistema *representa los textos cargados durante el procesado del Motor activo, de acuerdo a su información característica.*
	3	Si *hay algún texto que no ha sido invocado durante el procesamiento de ningún Motor,* el sistema *los dibuja del mismo modo, pero quedando por encima de los carácteres anteriores.*
Postcondición	Se han actualizado aquellos textos que hayan cambiado.	
Excepciones	No	
Importancia	vital	

Urgencia	inmediatamente
Estado	validado
Estabilidad	alta
Comentarios	Ninguno

UC-0096	**Aplicar efecto sobre Texto**	
Versión	1.0 (07/07/2009)	
Autores	• José Manuel Barroso Galindo	
Fuentes	• Pablo Neira Ayuso	
Dependencias	• [OBJ-0007] Soporte de efectos gráficos adicionales	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando *se quiera alterar la representación gráfica del Texto. Dicho efecto puede consistir en alterar los materiales de la fuente de los textos, aplicar transparencias, zooms y rotaciones, así como traslaciones o espejados.*	
Precondición	El texto del identificador dado existe.	
Secuencia normal	**Paso**	**Acción**
	1	El actor Aplicación (ACT-0001) *especifica el efecto gráfico que quiere que sufra en la siguiente fase de renderización.*
	2	El sistema *satisface dicha petición, y en la etapa de renderización de Texto aplicará los efectos gráficos esperados. A continuación devuelve ÉXITO.*
Postcondición	El Texto registra un efecto gráfico para la siguiente fase de renderización.	
Excepciones	**Paso**	**Acción**
	2	Si *el identificador no corresponde con ningún texto,* el sistema *devuelve FRACASO*, a continuación este caso de uso *queda sin efecto*
	2	Si *el efecto gráfico no es aplicable,* el sistema *devuelve FRACASO*, a continuación este caso de uso *queda sin efecto*
Importancia	quedaría bien	
Urgencia	hay presión	
Estado	validado	
Estabilidad	alta	
Comentarios	Ninguno	

3.2.2.3.5. Gestión del Multiverso

UC-0035	**Cargar Universo**	
Versión	1.0 (01/07/2009)	
Autores	• José Manuel Barroso Galindo	
Fuentes	• Pablo Neira Ayuso	
Dependencias	• [IRQ-0008] Universo • [OBJ-0013] Plantillas • [OBJ-0014] Alto grado de Encapsulación	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando *el usuario solicite hacer uso de un Universo.*	
Precondición	Hay algún Motor activo.	
Secuencia normal	**Paso**	**Acción**
	1	El actor Aplicación (ACT-0001) *crea una nueva instancia de un Universo con un descriptor determinado y queda asociada al Motor activo.*
	2	El sistema *devuelve al usuario la instancia definitiva convenientemente inicializada que deberá usar el usuario cada vez que quiera operar con el Universo.*
Postcondición	El Universo correspondiente está almacenado en la colección de Universos correspondiente al Motor activo	
Excepciones	**Paso**	**Acción**
	2	Si *la instancia del universo que se ha pasado es nula o errónea,* el sistema *devuelve FRACASO*, a continuación este caso de uso *queda sin efecto*
	2	Si *el descriptor es erróneo,* el sistema *devuelve FRACASO*, a continuación este caso de uso *queda sin efecto*
Importancia	importante	
Urgencia	inmediatamente	
Estado	validado	
Estabilidad	alta	
Comentarios	Ninguno	

UC-0036	**Buscar Universo**

Versión	1.0 (01/07/2009)	
Autores	• José Manuel Barroso Galindo	
Fuentes	• Pablo Neira Ayuso	
Dependencias	• [IRQ-0008] Universo • [OBJ-0012] Métodos Ayudantes	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando *el usuario solicite conocer la instancia de el universo que coincida con el descriptor indicado.*	
Precondición	Hay algún Motor activo.	
Secuencia normal	**Paso**	**Acción**
	1	El actor Aplicación (ACT-0001) *indica un identificador unívoco.*
	2	El sistema *busca el Universo apropiado en la colección de universos asignada al motor activo y lo devuelve.*
Postcondición		
Excepciones	**Paso**	**Acción**
	2	Si *no encuentra ningún universo como resultado de la búsqueda,* el sistema *devuelve FRACASO*, a continuación este caso de uso *continúa*
	2	Si *algún valor de entrada es incorrecto,* el sistema *devuelve FRACASO*, a continuación este caso de uso *queda sin efecto*
Importancia	importante	
Urgencia	inmediatamente	
Estado	validado	
Estabilidad	alta	
Comentarios	Ninguno	

UC-0037	**Eliminar Universo**
Versión	1.0 (01/07/2009)
Autores	• José Manuel Barroso Galindo
Fuentes	• Pablo Neira Ayuso
Dependencias	• [IRQ-0008] Universo • [OBJ-0012] Métodos Ayudantes

Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando *la aplicación usuaria desee eliminar un Universo.*	
Precondición	Hay un Motor activo.	
Secuencia normal	**Paso**	**Acción**
	1	El actor Aplicación (ACT-0001) *indica la instancia de un universo que quiera que se borre.*
	2	Si *se ha encontrado dicho Universo en la colección de universos asociada al Motor activo,* el sistema *borra dicho Universo de la colección, lo elimina, y se devuelve ÉXITO.*
Postcondición	El universo ha sido removido de la colección, y se ha eliminado de la memoria.	
Excepciones	**Paso**	**Acción**
	2	Si *la instancia del universo que se ha pasado es nula, errónea, o simplemente no pertenece a la colección de universos asociada al motor activo,* el sistema *devuelve FRACASO*, a continuación este caso de uso *queda sin efecto*
Importancia	importante	
Urgencia	hay presión	
Estado	validado	
Estabilidad	alta	
Comentarios	Ninguno	

UC-0041	**Acceder a la Colección de Universos del Motor activo**
Versión	1.0 (01/07/2009)
Autores	• José Manuel Barroso Galindo
Fuentes	• Pablo Neira Ayuso
Dependencias	• [IRQ-0007] Motor • [IRQ-0008] Universo • [OBJ-0012] Métodos Ayudantes
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando *se quiera acceder a la colección de Universos correspondiente al Motor activo.*
Precondición	Hay un Motor activo.

Secuencia normal	**Paso**	**Acción**
	1	El actor Aplicación (ACT-0001) *solicita acceder a la colección de Universos correspondiente al Motor activo,*

	2	El sistema *satisface la petición devolviendo información sobre la colección, y sobre sus universos contenidos.*
Postcondición		
Excepciones	No	
Importancia	quedaría bien	
Urgencia	puede esperar	
Estado	validado	
Estabilidad	alta	
Comentarios	Ninguno	

3.2.2.3.6. Gestión del Motor

UC-0047	**Iniciar Motor**	
Versión	1.0 (01/07/2009)	
Autores	• José Manuel Barroso Galindo	
Fuentes	• Pablo Neira Ayuso	
Dependencias	• [IRQ-0007] Motor • [OBJ-0013] Plantillas	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando *sea necesario preparar las estructuras y recursos de un determinado Motor.*	
Precondición	El Motor no se encontraba inicializado.	
Secuencia normal	**Paso**	**Acción**
	1	El actor Aplicación (ACT-0001) *solicita que se ejecuten las rutinas de iniciación del Motor.*
	2	El sistema *invoca a las rutinas que han sido asignadas para preparar al Motor.*
Postcondición	El Motor está preparado para gestionar Eventos.	
Excepciones	No	
Importancia	vital	
Urgencia	inmediatamente	
Estado	validado	
Estabilidad	alta	

Comentarios	Ninguno

UC-0048	**Terminar Motor**	
Versión	1.0 (01/07/2009)	
Autores	• José Manuel Barroso Galindo	
Fuentes	• Pablo Neira Ayuso	
Dependencias	• [IRQ-0007] Motor • [OBJ-0013] Plantillas	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando *sea necesario liberar los recursos de un determinado Motor*	
Precondición	El Motor se encontraba inicializado.	
Secuencia normal	**Paso**	**Acción**
	1	El actor Aplicación (ACT-0001) *solicita que se ejecuten las rutinas de terminación del Motor dado.*
	2	El sistema *invoca a las rutinas que han sido asignadas para liberar los recursos reservados por el Motor.*
Postcondición	El Motor ahora no es apto para ser procesado.	
Excepciones	No	
Importancia	importante	
Urgencia	inmediatamente	
Estado	validado	
Estabilidad	alta	
Comentarios	Ninguno	

UC-0050	**Establecer Manejador de Eventos**
Versión	1.0 (01/07/2009)
Autores	• José Manuel Barroso Galindo
Fuentes	• Pablo Neira Ayuso
Dependencias	• [IRQ-0006] Evento • [IRQ-0007] Motor • [IRQ-0012] Manejador de Eventos • [OBJ-0006] Ofrecer mecanismos para la manipulación de los eventos de Entrada/Salida

Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando *se quiera asociar un tipo de Evento determinado a un Manejador de Eventos para que éste los gestione.*	
Precondición		
Secuencia normal	**Paso**	**Acción**
	1	El actor Aplicación (ACT-0001) *indica el tipo de Evento y el Manejador de Eventos que vaya a gestionarlo.*
	2	El sistema *asocia el Manejador al tipo de Evento indicado y devuelve el anterior Manejador de Eventos que estaba asociado a dicho tipo.*
Postcondición	Se ha establecido la función indicada como correspondiente al evento indicado.	
Excepciones	**Paso**	**Acción**
	2	Si *el tipo de Evento indicado no es válido o el Manejador de Eventos no es válido,* el sistema *devuelve FRACASO*, a continuación este caso de uso *queda sin efecto*
Importancia	vital	
Urgencia	inmediatamente	
Estado	validado	
Estabilidad	alta	
Comentarios	Ninguno	

UC-0051	**Procesar Eventos**
Versión	1.0 (01/07/2009)
Autores	• José Manuel Barroso Galindo
Fuentes	• Pablo Neira Ayuso
Dependencias	• [IRQ-0006] Evento • [IRQ-0007] Motor • [IRQ-0012] Manejador de Eventos • [OBJ-0005] Soporte de Renderización • [OBJ-0006] Ofrecer mecanismos para la manipulación de los eventos de Entrada/Salida • [OBJ-0010] Soporte de temporización • [OBJ-0013] Plantillas
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando *se desee que el Motor dado sea el Motor activo y gestione los siguientes Eventos entrantes en el sistema.* o durante la realización de

	los siguientes casos de uso: [UC-0002] Procesar Motores	
Precondición	Se ha iniciado la Biblioteca y el Motor.	
Secuencia normal	**Paso**	**Acción**
	1	El actor Aplicación (ACT-0001) *solicita expresamente que se procesen los eventos....*
	2	El sistema *se atiende todos los Eventos entrantes en el sistema, llamando a los manejadores de eventos que les correspondan para gestionarlos. Tras esto, se procesan los siguientes casos de uso.*
	3	Se realiza el caso de uso Fase de Renderización del Motor (UC-0054)
	4	Se realiza el caso de uso Siguiente Etapa de ejecución (UC-0064)
	5	El sistema *vuelve al Paso 1, a no ser que el Motor ahora ya no sea el activo, en cuyo caso devuelve ÉXITO.*
Postcondición		
Excepciones	**Paso**	**Acción**
	-	-
Importancia	vital	
Urgencia	inmediatamente	
Estado	validado	
Estabilidad	alta	
Comentarios	Ninguno	

UC-0054	**Fase de Renderización del Motor**	
Versión	1.0 (01/07/2009)	
Autores	• José Manuel Barroso Galindo	
Fuentes	• Pablo Neira Ayuso	
Dependencias	• [IRQ-0007] Motor • [OBJ-0005] Soporte de Renderización	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando *se desee representar en la pantalla a todos los elementos gestionados por el Motor correspondiente,* o durante la realización de los siguientes casos de uso: [UC-0051] Procesar Eventos	
Precondición	El Motor se ha inizializado.	
Secuencia normal	**Paso**	**Acción**

	1	El actor Aplicación (ACT-0001) *solicita el comienzo de la fase de renderización del motor.*
	2	El sistema *invoca a las rutinas que han sido asignadas para ser procesadas con motivo de representar en pantalla los elementos gráficos que correspondan a la situación del sistema actual.*
Postcondición		
Excepciones	**Paso**	**Acción**
	-	-
Importancia	vital	
Urgencia	inmediatamente	
Estado	validado	
Estabilidad	alta	
Comentarios	Ninguno	

3.2.2.3.7. Gestión de Mensajería

UC-0055	**Mandar Mensaje**	
Versión	1.0 (01/07/2009)	
Autores	• José Manuel Barroso Galindo	
Fuentes	• Pablo Neira Ayuso	
Dependencias	• [OBJ-0012] Métodos Ayudantes	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando *se quiera hacer uso de la mensajería.*	
Precondición	El destinatario debe tener la capacidad de procesar mensajería.	
Secuencia normal	**Paso**	**Acción**
	1	El actor Aplicación (ACT-0001) *crea un mensaje y lo transmite al destinatario deseado.*
	2	El sistema *atiende el mensaje y si tiene capacidad para ello, lo procesa inmediatamente o al final de la actual etapa de ejecución.*
Postcondición		
Excepciones	No	
Importancia	importante	

Urgencia	hay presión
Estado	validado
Estabilidad	alta
Comentarios	Ninguno

UC-0059	**Borrar Mensajes Pendientes**	
Versión	1.0 (01/07/2009)	
Autores	• José Manuel Barroso Galindo	
Fuentes	• Pablo Neira Ayuso	
Dependencias	• [OBJ-0012] Métodos Ayudantes	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando *se desee eliminar todos los mensajes pendientes de un destinatario dado.*	
Precondición		
Secuencia normal	**Paso**	**Acción**
	1	El actor Aplicación (ACT-0001) *solicita eliminar todos los mensajes registrados por el destinatario pero que aun no han sido procesados.*
	2	El sistema *satisface dicha petición.*
Postcondición	Ya no hay ningún mensaje pendiente.	
Excepciones	**Paso**	**Acción**
	-	-
Importancia	quedaría bien	
Urgencia	puede esperar	
Estado	validado	
Estabilidad	alta	
Comentarios	Ninguno	

3.2.2.3.8. Gestión del Ritmo de Ejecución

UC-0064	**Siguiente Etapa de ejecución**
Versión	1.0 (01/07/2009)
Autores	• José Manuel Barroso Galindo

Fuentes	• Pablo Neira Ayuso	
Dependencias	• [OBJ-0010] Soporte de temporización	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando *se desee dar por finalizada la actual etapa de ejecución del motor activo y preparar una nueva,* o durante la realización de los siguientes casos de uso: [UC-0051] Procesar Eventos	
Precondición	Se ha iniciado la Biblioteca.	
Secuencia normal	**Paso**	**Acción**
	1	El actor Aplicación (ACT-0001) *solicita dar por finalizada la actual etapa de ejecución, y pasar a la siguiente.*
	2	Se realiza el caso de uso Actualizar la Pantalla (UC-0018)
	3	Si *se ha especificado algún ritmo temporal de la ejecución,* el sistema *lo cumple.*
	4	El sistema *a continuaciónd devuelve ÉXITO.*
Postcondición	Se ha preparado una nueva etapa de ejecución.	
Excepciones	**Paso**	**Acción**
	2	Si *el caso de uso ha fallado,* el sistema *devuelve FRACASO*, a continuación este caso de uso *queda sin efecto*
	4	Si *no hay ningún Motor activo,* el sistema *considera válido el Motor nulo como caso excepcional*, a continuación este caso de uso *continúa*
Importancia	importante	
Urgencia	inmediatamente	
Estado	validado	
Estabilidad	alta	
Comentarios	Ninguno	

UC-0065	**Acceder al Contador de Etapas de ejecución**
Versión	1.0 (01/07/2009)
Autores	• José Manuel Barroso Galindo
Fuentes	• Pablo Neira Ayuso
Dependencias	• [OBJ-0012] Métodos Ayudantes
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando *se quiera saber cuantas fases de ejecución han pasado para el Motor activo.*

Precondición	Se ha inicializado la biblioteca.	
Secuencia normal	**Paso**	**Acción**
	1	El actor Aplicación (ACT-0001) *solicita conocer cuantas etapas de ejecución han acontecido para el motor activo.*
	2	El sistema *devuelve dicha información.*
Postcondición		
Excepciones	**Paso**	**Acción**
	2	Si *no hay ningún Motor activo,* el sistema *considera válido el Motor nulo como caso excepcional*, a continuación este caso de uso *continúa*
Importancia	quedaría bien	
Urgencia	puede esperar	
Estado	validado	
Estabilidad	alta	
Comentarios	Ninguno	

UC-0066	**Establecer ritmo de ejecución en unidad de tiempo**	
Versión	1.0 (01/07/2009)	
Autores	• José Manuel Barroso Galindo	
Fuentes	• Pablo Neira Ayuso	
Dependencias	• [OBJ-0010] Soporte de temporización	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando *se quiera que haya un intervalo de tiempo mínimo medido en la unidad de tiempo deseada.*	
Precondición		
Secuencia normal	**Paso**	**Acción**
	1	El actor Aplicación (ACT-0001) *indica un intervalo de tiempo en la unidad deseada.*
	2	El sistema *lo aplica y devuelve el anterior valor.*
Postcondición	Se ha cambiado el ritmo temporal de la ejecución.	
Excepciones	No	
Importancia	importante	
Urgencia	hay presión	
Estado	validado	

Estabilidad	alta
Comentarios	Ninguno

3.2.2.3.9. Gestión de la Cámara

UC-0078	**Ubicar Cámara en la Pantallla**	
Versión	1.0 (01/07/2009)	
Autores	• José Manuel Barroso Galindo	
Fuentes	• Pablo Neira Ayuso	
Dependencias	• [IRQ-0011] Cámara • [OBJ-0012] Métodos Ayudantes	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando *se quiera especificar un área de la Pantalla que se corresponda con una cámara.*	
Precondición		
Secuencia normal	**Paso**	**Acción**
	1	El actor Aplicación (ACT-0001) *especifica un área de la pantalla en la cual quiere que se visualice la cámara.*
	2	El sistema *especifica el área de visualicación o sistituye el anterior si lo hubiera por la especificada por el usuario.*
Postcondición	Se ha especificado el área de ocupación en la Pantalla.	
Excepciones	**Paso**	**Acción**
	2	Si *es inválida el área especificada,* el sistema *devuelve FRACASO*, a continuación este caso de uso *continúa*
Importancia	vital	
Urgencia	inmediatamente	
Estado	validado	
Estabilidad	alta	
Comentarios	Ninguno	

UC-0073	**Obtener Información de la Cámara**
Versión	1.0 (01/07/2009)
Autores	• José Manuel Barroso Galindo

Fuentes	• Pablo Neira Ayuso	
Dependencias	• [IRQ-0011] Cámara • [OBJ-0012] Métodos Ayudantes	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando *el usuario quiera información varia como cual es el Actor Objetivo, el Universo Objetivo, cuales son las coordenadas... etc.*	
Precondición		
Secuencia normal	**Paso**	**Acción**
	1	El actor Aplicación (ACT-0001) *solicita información determinada sobre el estado de la cámara y su entorno.*
	2	El sistema *satisface dicha petición devolviendo la información requerida.*
Postcondición		
Excepciones	**Paso**	**Acción**
	2	Si *se ha especificado mal la información requerida,* el sistema *devuelve FRACASO*, a continuación este caso de uso *queda sin efecto*
	2	Si *ha ocurrido algún error mientras se procesaba la información a devolver,* el sistema *devuelve FRACASO*, a continuación este caso de uso *queda sin efecto*
Importancia	importante	
Urgencia	inmediatamente	
Estado	validado	
Estabilidad	alta	
Comentarios	Ninguno	

UC-0088	**Cambiar propiedades de la Cámara**
Versión	1.0 (06/07/2009)
Autores	• José Manuel Barroso Galindo
Fuentes	• Pablo Neira Ayuso
Dependencias	• [IRQ-0007] Motor • [OBJ-0012] Métodos Ayudantes
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso

	de uso cuando *se quieran cambiar uno o varios atributos de la Cámara.*	
Precondición		
Secuencia normal	**Paso**	**Acción**
	1	El actor Aplicación (ACT-0001) *solicita cambiar el estado de la cámara especificando los nuevos valores que ha de adoptar en los atributos que van a cambiar.*
	2	El sistema *ejecuta dicha petición y devuelve ÉXITO.*
Postcondición	Las características de la Cámara han cambiado.	
Excepciones	**Paso**	**Acción**
	2	Si *se ha especificado que un valor no válido para un atributo,* el sistema *devuelve FRACASO*, a continuación este caso de uso *queda sin efecto*
Importancia	importante	
Urgencia	hay presión	
Estado	validado	
Estabilidad	alta	
Comentarios	Ninguno	

UC-0094	**Aplicar efecto sobre Cámara**	
Versión	1.0 (07/07/2009)	
Autores	• José Manuel Barroso Galindo	
Fuentes	• Pablo Neira Ayuso	
Dependencias	• [IRQ-0011] Cámara • [OBJ-0007] Soporte de efectos gráficos adicionales	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando *se quiera alterar la representación gráfica de todo aquello enfocado por la Cámara. Dicho efecto puede consistir en modificar los colores de los elementos representados, aplicar transparencias, zooms y rotaciones, así como traslaciones, espejados o transformaciones geométricas de perspectiva.*	
Precondición	La Cámara está preparada para un proceso de renderización.	
Secuencia normal	**Paso**	**Acción**
	1	El actor Aplicación (ACT-0001) *especifica el efecto gráfico que quiere que sufra la Cámara en la siguiente fase de renderización.*
	2	El sistema *satisface dicha petición, y en la fase de renderización de la Cámara aplicará los efectos gráficos esperados sobre los*

		elementos enfocados.
Postcondición	Los elementos visuales enfocados por la Cámara registran un efecto gráfico para la siguiente fase de renderización.	
Excepciones	**Paso**	**Acción**
	2	Si *el efecto gráfico no es aplicable,* el sistema *devuelve FRACASO*, a continuación este caso de uso *queda sin efecto*
Importancia	quedaría bien	
Urgencia	puede esperar	
Estado	validado	
Estabilidad	alta	
Comentarios	Ninguno	

UC-0079	**Renderizar elementos enfocados por la Cámara**	
Versión	1.0 (01/07/2009)	
Autores	• José Manuel Barroso Galindo	
Fuentes	• Pablo Neira Ayuso	
Dependencias	• [IRQ-0011] Cámara • [OBJ-0005] Soporte de Renderización • [IRQ-0008] Universo • [IRQ-0009] Actor • [IRQ-0010] Figurante	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando *se quieran representar en la pantalla todos los elementos que están dentro del rango del acción de la cámara, ya sean figurantes o actores dentro de un universo.*	
Precondición	Se ha inicializado el Contexto de Vídeo.	
Secuencia normal	**Paso**	**Acción**
	1	El actor Aplicación (ACT-0001) *solicita iniciar la renderización de la cámara.*
	2	El sistema *actualiza el estado de coordenadas de la Cámara, y posteriormente recorre todos los elementos visuales de la Cámara hayados dentro del área de enfoque llamando a su correspondiente fase de renderización. A continuación devuelve ÉXITO.*
Postcondición		
Excepciones	**Paso**	**Acción**
	2	Si *durante este proceso ha ocurrido algún error,* el sistema

		devuelve FRACASO, a continuación este caso de uso *queda sin efecto*
Importancia	vital	
Urgencia	inmediatamente	
Estado	validado	
Estabilidad	alta	
Comentarios	Ninguno	

UC-0070	**Establecer Actor Objetivo**	
Versión	1.0 (01/07/2009)	
Autores	• José Manuel Barroso Galindo	
Fuentes	• Pablo Neira Ayuso	
Dependencias	• [IRQ-0009] Actor • [IRQ-0011] Cámara • [OBJ-0012] Métodos Ayudantes	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando *se quiera que la cámara cambie de objetivo.*	
Precondición		
Secuencia normal	**Paso**	**Acción**
	1	El actor Aplicación (ACT-0001) *transmite una instancia de el Actor que quiera que sea el objetivo de la cámara.*
	2	El sistema *satisface la petición y devuelve ÉXITO.*
Postcondición	El Actor Objetivo de la cámara ahora es el indicado.	
Excepciones	**Paso**	**Acción**
	2	Si *el Actor no se puede sincronizar con la cámara,* el sistema *devuelve FRACASO*, a continuación este caso de uso *queda sin efecto*
Importancia	vital	
Urgencia	inmediatamente	
Estado	validado	
Estabilidad	alta	
Comentarios	Ninguno	

UC-0071	**Preparar Universo Objetivo**
Versión	1.0 (01/07/2009)

Autores	• José Manuel Barroso Galindo	
Fuentes	• Pablo Neira Ayuso	
Dependencias	• [IRQ-0008] Universo • [IRQ-0011] Cámara • [OBJ-0013] Plantillas	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando *sea necesario preparar el universo huésped del actor objetivo de la cámara.*	
Precondición	El universo objetivo no está preparado.	
Secuencia normal	**Paso**	**Acción**
	1	El actor Aplicación (ACT-0001) *solicita preparar el Universo en el que está contenido el actual actor objetivo de esta cámara.*
	2	El sistema *ejecuta las rutinas de inicialización especificadas para cargar los recursos necesarios para que dicho Universo luego pueda ser representado en Pantalla. A continuación devuelve ÉXITO.*
Postcondición	El universo objetivo está preparado.	
Excepciones	**Paso**	**Acción**
	2	Si *ha ocurrido algún error durante la ejecución de dichas rutinas,* el sistema *devuelve FRACASO*, a continuación este caso de uso *queda sin efecto*
Importancia	importante	
Urgencia	hay presión	
Estado	validado	
Estabilidad	alta	
Comentarios	Ninguno	

UC-0072	**Liberar Universo Objetivo**
Versión	1.0 (01/07/2009)
Autores	• José Manuel Barroso Galindo
Fuentes	• Pablo Neira Ayuso
Dependencias	• [IRQ-0008] Universo • [IRQ-0011] Cámara • [OBJ-0013] Plantillas

Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando *ya no sea necesario utilizar la información del universo por la cámara.*	
Precondición	Se había preparado un Universo.	
Secuencia normal	**Paso**	**Acción**
	1	El actor Aplicación (ACT-0001) *desee informar que no va hacer uso de dicho universo por parte de la actual cámara.*
	2	El sistema *ejecuta las rutinas de liberación especificadas para descargar los recursos que fueron necesarios para que dicho Universo estuviese preparado. A continuación devuelve ÉXITO.*
Postcondición	Se han eliminado los recursos propios del Universo y por tanto ya no se considera preparado.	
Excepciones	**Paso**	**Acción**
	2	Si *ha ocurrido algún error durante la ejecución de dichas rutinas,* el sistema *devuelve FRACASO*, a continuación este caso de uso *queda sin efecto*
Importancia	importante	
Urgencia	hay presión	
Estado	validado	
Estabilidad	alta	
Comentarios	Ninguno	

3.2.2.3.10. Gestión del Actor

UC-0084	**Inicializar Actor**	
Versión	1.0 (01/07/2009)	
Autores	• José Manuel Barroso Galindo	
Fuentes	• Pablo Neira Ayuso	
Dependencias	• [IRQ-0009] Actor • [OBJ-0013] Plantillas	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando *sea necesario preparar a un actor antes de ejecutar alguna vez su fase de Movimiento.*	
Precondición		
Secuencia normal	**Paso**	**Acción**

	1	El actor Aplicación (ACT-0001) *solicita ejecutar la rutina de inicialización del Actor.*
	2	El sistema *invoca a las rutinas que han sido asignadas para preparar al actor de cara a una posterior fase de movimiento.*
Postcondición	El actor está preparado para el proceso de Movimiento.	
Excepciones	**Paso**	**Acción**
	2	Si *ha ocurrido algún error durante la ejecución de dichas rutinas,* el sistema *devuelve FRACASO*, a continuación este caso de uso *queda sin efecto*
Importancia	vital	
Urgencia	inmediatamente	
Estado	validado	
Estabilidad	alta	
Comentarios	Ninguno	

UC-0083	**Clonar Actor**	
Versión	1.0 (01/07/2009)	
Autores	• José Manuel Barroso Galindo	
Fuentes	• Pablo Neira Ayuso	
Dependencias	• [IRQ-0009] Actor • [OBJ-0012] Métodos Ayudantes	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando *se necesite crear nuevo actor idéntico a uno dado.*	
Precondición	El actor actual se ha inicializado.	
Secuencia normal	**Paso**	**Acción**
	1	El actor Aplicación (ACT-0001) *solicita la clonación de un Actor.*
	2	El sistema *crea un nuevo clon, invoca a las rutinas que han sido asignadas para preparar dicho clon e inicializarlo, y a continuación lo devuelve.*
Postcondición	Se ha creado un nuevo actor idéntico al anterior.	
Excepciones	**Paso**	**Acción**
	2	Si *ha ocurrido algún error durante la ejecución de dichas rutinas,* el sistema *devuelve FRACASO*, a continuación este caso de uso *queda sin efecto*
Importancia	quedaría bien	

Urgencia	puede esperar
Estado	validado
Estabilidad	alta
Comentarios	Ninguno

UC-0085	**Procesar Movimiento del Actor**	
Versión	1.0 (01/07/2009)	
Autores	• José Manuel Barroso Galindo	
Fuentes	• Pablo Neira Ayuso	
Dependencias	• [IRQ-0009] Actor • [OBJ-0012] Métodos Ayudantes • [OBJ-0013] Plantillas	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando *se quiera actualizar el estado del Actor, incluyendo ahí su información gráfica, sus coordenadas y ejecutar sus tareas periódicas.*	
Precondición	Existe un universo huésped, y el actor ha sido inicializado.	
Secuencia normal	**Paso**	**Acción**
	1	El actor Aplicación (ACT-0001) *solicita ejecutar la rutina de movimiento del Actor.*
	2	El sistema *invoca a las rutinas que han sido asignadas para actualizar el estado (coordenadas, información gráfica, etc..) del Actor.*
Postcondición	El actor ha actualizado sus coordenadas y su información gráfica.	
Excepciones	**Paso**	**Acción**
	2	Si *ha ocurrido algún error durante la ejecución de dichas rutinas,* el sistema *devuelve FRACASO*, a continuación este caso de uso *queda sin efecto*
Importancia	importante	
Urgencia	hay presión	
Estado	validado	
Estabilidad	alta	
Comentarios	Ninguno	

UC-0086	**Obtener Información del Actor**
Versión	1.0 (01/07/2009)

Autores	• José Manuel Barroso Galindo	
Fuentes	• Pablo Neira Ayuso	
Dependencias	• [IRQ-0009] Actor • [OBJ-0012] Métodos Ayudantes	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando *se quiera consultar información del actor como averiguar cual es su universo huésped, sus coordenadas, su información gráfica, etc.*	
Precondición		
Secuencia normal	**Paso**	**Acción**
	1	El actor Aplicación (ACT-0001) *solicita información determinada sobre el estado del Actor.*
	2	El sistema *satisface dicha petición devolviendo la información requerida.*
Postcondición		
Excepciones	**Paso**	**Acción**
	2	Si *se ha especificado mal la información requerida,* el sistema *devuelve FRACASO*, a continuación este caso de uso *queda sin efecto*
	2	Si *ha ocurrido algún error mientras se procesaba la información a devolver,* el sistema *devuelve FRACASO*, a continuación este caso de uso *queda sin efecto*
Importancia	vital	
Urgencia	inmediatamente	
Estado	validado	
Estabilidad	alta	
Comentarios	Ninguno	

UC-0087	**Cambiar Información del Actor**
Versión	1.0 (01/07/2009)
Autores	• José Manuel Barroso Galindo
Fuentes	• Pablo Neira Ayuso
Dependencias	• [IRQ-0009] Actor • [OBJ-0012] Métodos Ayudantes
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso

	de uso cuando *se quieran cambiar uno o varios atributos del Actor.*	
Precondición		
Secuencia normal	**Paso**	**Acción**
	1	El actor Aplicación (ACT-0001) *solicita cambiar el estado del actor especificando los nuevos valores que ha de adoptar en los atributos que van a cambiar.*
	2	El sistema *ejecuta dicha petición y devuelve ÉXITO.*
Postcondición	Las información del Actor ha cambiado.	
Excepciones	**Paso**	**Acción**
	2	Si *se ha especificado que un valor no válido para un atributo,* el sistema *devuelve FRACASO*, a continuación este caso de uso *queda sin efecto*
Importancia	vital	
Urgencia	inmediatamente	
Estado	validado	
Estabilidad	alta	
Comentarios	Ninguno	

UC-0093	**Aplicar efecto sobre Actor**	
Versión	1.0 (07/07/2009)	
Autores	• José Manuel Barroso Galindo	
Fuentes	• Pablo Neira Ayuso	
Dependencias	• [IRQ-0009] Actor • [OBJ-0007] Soporte de efectos gráficos adicionales	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando *se quiera alterar la representación gráfica del Actor. Dicho efecto puede consistir en cambiar los colores de sus sprites asociados, aplicarles transparencias, zooms y rotaciones, así como traslaciones o espejados, o establecer efectos de partículas y trazas de iluminación.*	
Precondición	El Actor ha sido Inicializado.	
Secuencia normal	**Paso**	**Acción**
	1	El actor Aplicación (ACT-0001) *especifica el efecto gráfico que quiere que sufra en la siguiente fase de renderización el Actor.*
	2	El sistema *satisface dicha petición, y en la etapa de renderización del Actor aplicará los efectos gráficos esperados. A continuación devuelve ÉXITO.*

Postcondición	El Actor registra un efecto gráfico para la siguiente fase de renderización.	
Excepciones	**Paso**	**Acción**
	2	Si *el efecto gráfico no es aplicable,* el sistema *devuelve FRACASO*, a continuación este caso de uso *queda sin efecto*
Importancia	quedaría bien	
Urgencia	hay presión	
Estado	validado	
Estabilidad	alta	
Comentarios	Ninguno	

UC-0104	**Renderizar un Actor en coordenadas de la Pantalla**	
Versión	1.0 (20/07/2009)	
Autores	• José Manuel Barroso Galindo	
Fuentes	• Pablo Neira Ayuso	
Dependencias	• [IRQ-0009] Actor • [OBJ-0005] Soporte de Renderización	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando *se quiera concretar en qué cordenadas de la pantalla deberá renderizarse el actor durante la siguiente fase de renderización.*	
Precondición	El Contexto de Vídeo debe estar inicializado.	
Secuencia normal	**Paso**	**Acción**
	1	El actor Aplicación (ACT-0001) *especifica las coordenadas en las cuales desea representar el Actor en Pantalla.*
	2	El sistema *llama a la correspondiente fase de renderización del Actor. A continuación devuelve ÉXITO.*
Postcondición		
Excepciones	**Paso**	**Acción**
	2	Si *durante este proceso ha ocurrido algún error,* el sistema *devuelve FRACASO*, a continuación este caso de uso *queda sin efecto*
Importancia	quedaría bien	
Urgencia	hay presión	
Estado	validado	
Estabilidad	alta	

Comentarios	Ninguno

3.2.2.3.11. Gestión del Universo

UC-0089	Añadir Actor al Universo	
Versión	1.0 (06/07/2009)	
Autores	• José Manuel Barroso Galindo	
Fuentes	• Pablo Neira Ayuso	
Dependencias	• [IRQ-0008] Universo • [IRQ-0009] Actor • [OBJ-0012] Métodos Ayudantes	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando *se desee agregar un Actor al Universo especificado.*	
Precondición	El Actor no está almacenado en ningún Universo.	
Secuencia normal	**Paso**	**Acción**
	1	El actor Aplicación (ACT-0001) *transmite al Universo especificado el Actor que quiere que añadir.*
	2	El sistema *almacena el Actor en la colección de Actores del Universo y por tanto este último pasa a ser el Universo huésped del Actor. A continuación devuelve ÉXITO.*
Postcondición	El actor está almacenado en la colección de Actores del Universo.	
Excepciones	**Paso**	**Acción**
	-	-
Importancia	vital	
Urgencia	inmediatamente	
Estado	validado	
Estabilidad	alta	
Comentarios	Ninguno	

UC-0090	Retirar Actor del Universo
Versión	1.0 (06/07/2009)
Autores	• José Manuel Barroso Galindo
Fuentes	• Pablo Neira Ayuso
Dependencias	• [IRQ-0008] Universo

	• [IRQ-0009] Actor • [OBJ-0012] Métodos Ayudantes	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando *desee retirar el Actor del Universo especificado.*	
Precondición	El actor estaba contenido en la colección de actores del Universo.	
Secuencia normal	**Paso**	**Acción**
	1	El actor Aplicación (ACT-0001) *indica que Actor quiere que se retire del Universo especificado.*
	2	El sistema *borra dicho Actor de la colección de Actores del Universo y por tanto lo deja sin Universo huésped. A continuación devuelve ÉXITO.*
Postcondición	El actor ya no está almacenado en la colección de actores del Universo.	
Excepciones	**Paso**	**Acción**
	2	Si *el actor tiene alguna restricción que evite salir de ese universo,* el sistema *devuelve FRACASO*, a continuación este caso de uso *queda sin efecto* (*La restricción se puede deber a un efecto gráfico o a un bloqueo explícito.*)
Importancia	quedaría bien	
Urgencia	puede esperar	
Estado	validado	
Estabilidad	alta	
Comentarios	Ninguno	

UC-0091	**Acceder a la Colección de Actores del Universo**	
Versión	1.0 (06/07/2009)	
Autores	• José Manuel Barroso Galindo	
Fuentes	• Pablo Neira Ayuso	
Dependencias	• [IRQ-0008] Universo • [IRQ-0009] Actor • [OBJ-0012] Métodos Ayudantes	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando *el usuario requiera recorrer la colección de actores del Universo especificado.*	
Precondición		
Secuencia normal	**Paso**	**Acción**
	1	El actor Aplicación (ACT-0001) *hace una petición para tener*

		acceso a la colección de Actores.
	2	El sistema *satisface dicha petición devolviendo información sobre la colección y sus actores contenidos.*
Postcondición		
Excepciones	**Paso**	**Acción**
	-	-
Importancia	importante	
Urgencia	hay presión	
Estado	validado	
Estabilidad	alta	
Comentarios	Ninguno	

UC-0092	**Migrar de un Universo a otro**	
Versión	1.0 (06/07/2009)	
Autores	• José Manuel Barroso Galindo	
Fuentes	• Pablo Neira Ayuso	
Dependencias	• [IRQ-0008] Universo • [OBJ-0012] Métodos Ayudantes	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando *se requiera cambiar la información del Universo especificado a la de otro Universo compatible (del mismo tipo).*	
Precondición		
Secuencia normal	**Paso**	**Acción**
	1	El actor Aplicación (ACT-0001) *indica la ruta del Universo en el que quiere que se convierta el Universo especificado.*
	2	El sistema *satisface dicha petición devolviendo ÉXITO.*
Postcondición	El Universo se ha transformado.	
Excepciones	**Paso**	**Acción**
	2	Si *el sistema determina que el descriptor de Universo no es compatible con el Universo especificado,* el sistema *devuelve FRACASO*, a continuación este caso de uso *queda sin efecto*
Importancia	quedaría bien	
Urgencia	puede esperar	
Estado	validado	
Estabilidad	alta	

Comentarios	Ninguno

UC-0095	**Aplicar efecto sobre Universo**	
Versión	1.0 (07/07/2009)	
Autores	• José Manuel Barroso Galindo	
Fuentes	• Pablo Neira Ayuso	
Dependencias	• [IRQ-0008] Universo • [IRQ-0009] Actor • [IRQ-0010] Figurante • [OBJ-0007] Soporte de efectos gráficos adicionales	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando *se quiera alterar la representación gráfica de los Figurantes del Universo y/o de sus Actores. Dicho efecto puede consistir en cambiar los colores de los sprites asociados, aplicarles transparencias, zooms y rotaciones, así como traslaciones o espejados.*	
Precondición	El Universo ha sido Inicializado.	
Secuencia normal	**Paso**	**Acción**
	1	El actor Aplicación (ACT-0001) *especifica el efecto gráfico que quiere que sufra en la siguiente fase de renderización los Figurantes o Actores del Universo.*
	2	El sistema *satisface dicha petición, y en la etapa de renderización de la Cámara que enfoque al Universo, aplicará los efectos gráficos especificados. A continuación devuelve ÉXITO.*
Postcondición	El Universo rgistra un efecto gráfico en sus Figurantes y/o Actores.	
Excepciones	**Paso**	**Acción**
	2	Si *el universo no ha sido cargado por ninguna Cámara,* el sistema *devuelve FRACASO*, a continuación este caso de uso *queda sin efecto*
	2	Si *el efecto gráfico no es aplicable,* el sistema *devuelve FRACASO*, a continuación este caso de uso *queda sin efecto*
Importancia	quedaría bien	
Urgencia	hay presión	
Estado	validado	
Estabilidad	alta	
Comentarios	Ninguno	

3.2.3. Requisitos no Funcionales

NFR-0001	Portabilidad
Versión	1.0 (07/07/2009)
Autores	• José Manuel Barroso Galindo
Fuentes	• Pablo Neira Ayuso
Dependencias	Ninguno
Descripción	El sistema deberá *ser facilmente portable a sistemas Windows y GNU/Linux.*
Importancia	importante
Urgencia	inmediatamente
Estado	validado
Estabilidad	alta
Comentarios	Ninguno

NFR-0002	Uso de Software Libre
Versión	1.0 (07/07/2009)
Autores	• José Manuel Barroso Galindo
Fuentes	• Pablo Neira Ayuso
Dependencias	Ninguno
Descripción	El sistema deberá *desarrollarse haciendo uso de bibliotecas de apoyo que sean libres y con licencias poco restrictivas.*
Importancia	vital
Urgencia	inmediatamente
Estado	validado
Estabilidad	alta
Comentarios	Ninguno

NFR-0003	Usabilidad
Versión	1.0 (07/07/2009)
Autores	• José Manuel Barroso Galindo

Fuentes	• Pablo Neira Ayuso
Dependencias	Ninguno
Descripción	El sistema deberá *ofrecer una interfaz de la biblioteca al usuario lo más simple y funcional posible. Nos ayudaremos del paradigma de la programación orientada a objetos para que el usuario se familiarice rápido con el uso de la biblioteca.*
Importancia	importante
Urgencia	inmediatamente
Estado	validado
Estabilidad	alta
Comentarios	Ninguno

NFR-0004	**Seguridad**
Versión	1.0 (07/07/2009)
Autores	• José Manuel Barroso Galindo
Fuentes	• Pablo Neira Ayuso
Dependencias	Ninguno
Descripción	El sistema deberá *evitar en la medida de lo posible que la aplicación usuaria pueda comprometer la estabilidad del programa debido a un mal uso de la biblioteca. Y para ello bloqueará el acceso a determinados elementos de la misma.*
Importancia	importante
Urgencia	hay presión
Estado	validado
Estabilidad	alta
Comentarios	Ninguno

NFR-0005	**Fiabilidad**
Versión	1.0 (07/07/2009)
Autores	• José Manuel Barroso Galindo
Fuentes	• Pablo Neira Ayuso

Dependencias	Ninguno
Descripción	El sistema deberá *deberá garantizar su correcto funcionamiento cuando un usuario esté haciendo uso de la biblioteca. Es decir, debe no bloquear el sistema ni perjudicar a otros procesos ajenos.*
Importancia	vital
Urgencia	inmediatamente
Estado	validado
Estabilidad	alta
Comentarios	Ninguno

NFR-0006	**Eficacia**
Versión	1.0 (07/07/2009)
Autores	• José Manuel Barroso Galindo
Fuentes	• Pablo Neira Ayuso
Dependencias	Ninguno
Descripción	El sistema deberá *usar adecuadamente los recursos de los que dispone, ya sean los de la propia biblioteca, o los que accede a través de otras APIs para desempeñar sus funciones.*
Importancia	importante
Urgencia	inmediatamente
Estado	validado
Estabilidad	alta
Comentarios	Ninguno

NFR-0007	**Estabilidad de la interfaz**
Versión	1.0 (07/07/2009)
Autores	• José Manuel Barroso Galindo
Fuentes	• Pablo Neira Ayuso
Dependencias	Ninguno
Descripción	El sistema deberá *ser lo suficientemente robusto, para que no haya necesidad de cambiar las interfaces actuales de la biblioteca en futuras*

	actualizaciones.
Importancia	quedaría bien
Urgencia	puede esperar
Estado	validado
Estabilidad	alta
Comentarios	Ninguno

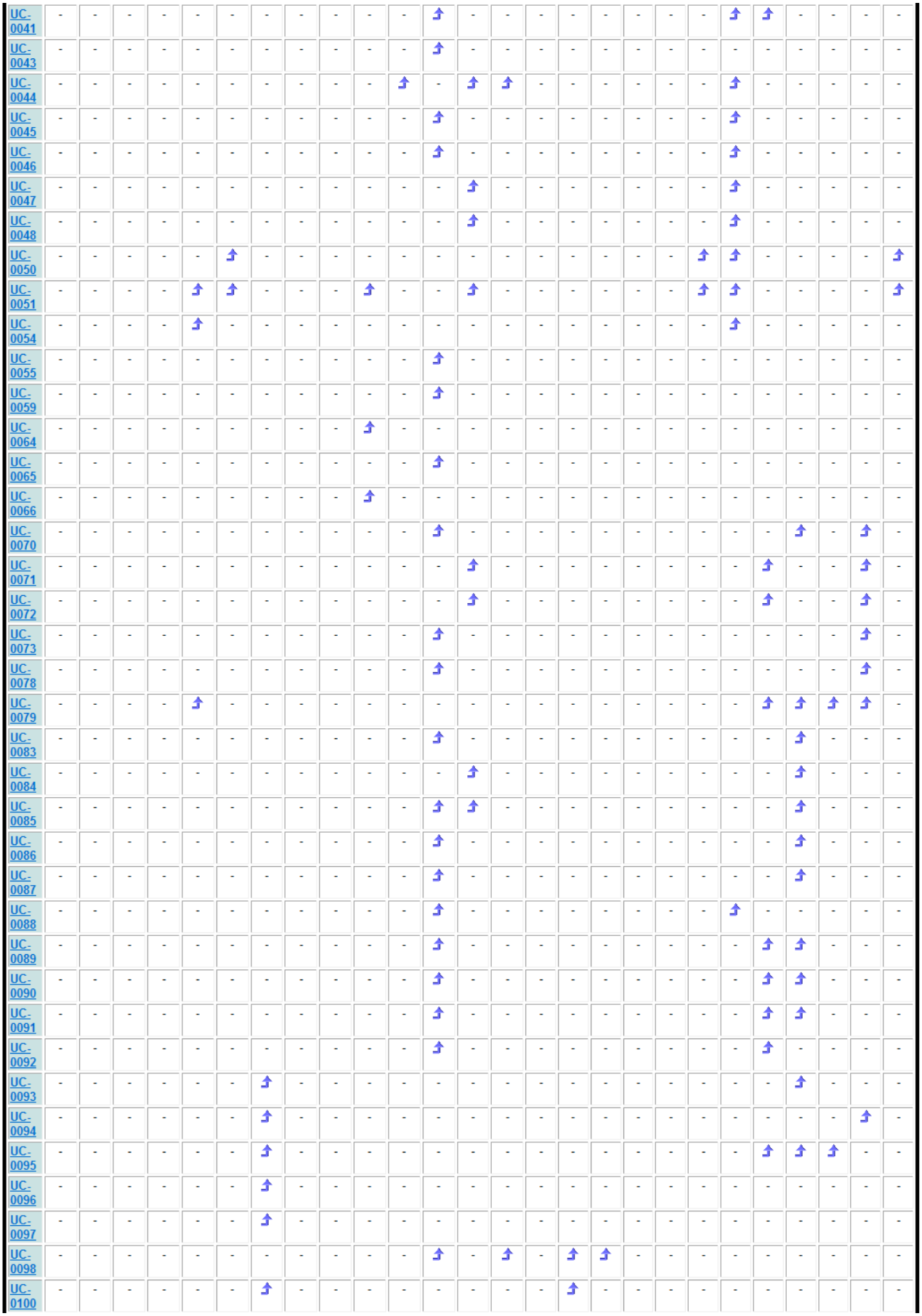
NFR-0008	**Ofrecer los detalles necesarios**
Versión	1.0 (07/07/2009)
Autores	• José Manuel Barroso Galindo
Fuentes	• Pablo Neira Ayuso
Dependencias	Ninguno
Descripción	El sistema deberá *de ofrecer a la aplicación unas interfaces completas, para no sesgar el control del usuario sobre las funcionalidades del sistema.*
Importancia	importante
Urgencia	hay presión
Estado	validado
Estabilidad	alta
Comentarios	Ninguno

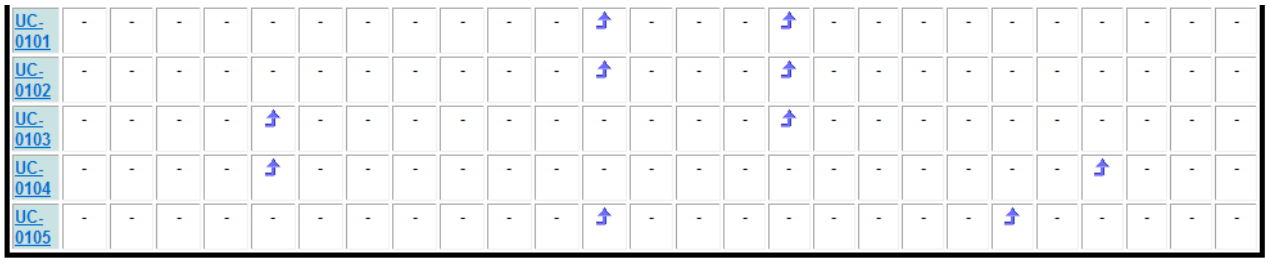
3.3. Matrices de Rastreabilidad

TRM-0001	OBJ-0001	OBJ-0002	OBJ-0003	OBJ-0004	OBJ-0005	OBJ-0006	OBJ-0007	OBJ-0008	OBJ-0009	OBJ-0010	OBJ-0011	OBJ-0012	OBJ-0013	OBJ-0014
IRQ-0001	-	-	↑	-	↑	-	-	-	-	-	-	-	-	↑
IRQ-0002	-	-	-	↑	-	-	-	-	-	-	-	-	-	↑
IRQ-0003	-	-	-	-	-	-	-	-	-	-	-	-	-	↑
IRQ-0004	-	-	-	-	-	-	-	↑	-	-	-	-	-	↑
IRQ-0005	↑	-	-	-	-	-	-	-	-	-	-	↑	-	-
IRQ-0006	-	-	-	-	-	↑	-	-	-	-	-	-	-	↑
IRQ-0007	-	-	-	-	-	↑	-	-	-	↑	-	↑	↑	↑
IRQ-0008	-	-	-	-	-	-	-	-	-	-	-	↑	↑	↑
IRQ-0009	-	-	-	-	-	-	-	-	-	-	-	-	-	↑
IRQ-0010	-	-	-	↑	-	-	-	-	-	-	-	-	-	-
IRQ-0011	-	-	-	-	↑	-	-	-	-	-	-	↑	↑	↑
IRQ-0012	-	-	-	-	-	↑	-	-	-	-	-	-	-	-

Matriz de rastreabilidad: Requisitos de Información

TRM-0002	OBJ-0001	OBJ-0002	OBJ-0003	OBJ-0004	OBJ-0005	OBJ-0006	OBJ-0007	OBJ-0008	OBJ-0009	OBJ-0010	OBJ-0011	OBJ-0012	OBJ-0013	OBJ-0014	IRQ-0001	IRQ-0002	IRQ-0003	IRQ-0004	IRQ-0005	IRQ-0006	IRQ-0007	IRQ-0008	IRQ-0009	IRQ-0010	IRQ-0011	IRQ-0012
UC-0001	↑	-	-	-	-	-	-	-	-	-	-	-	↑	-	-	-	-	-	-	-	-	-	-	-	-	-
UC-0002	-	-	-	-	-	↑	-	-	-	-	-	-	-	-	-	-	-	-	-	-	↑	-	-	-	-	-
UC-0003	-	-	-	-	-	-	-	-	-	-	-	↑	-	-	-	-	↑	-	-	-	-	-	-	-	-	-
UC-0004	-	↑	-	-	-	-	-	-	-	-	-	-	-	↑	-	-	↑	-	-	-	-	-	-	-	-	-
UC-0005	-	-	-	-	-	-	-	-	-	-	-	↑	-	-	-	-	↑	-	-	-	-	-	-	-	-	-
UC-0007	-	-	-	-	-	-	-	-	-	-	-	↑	-	-	-	-	↑	-	-	-	-	-	-	-	-	-
UC-0011	↑	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	↑	-	-	-	-	-	-	-
UC-0012	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	↑	-	-	-	-	-	-	-
UC-0013	↑	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	↑	-	-	-	-	-	-	-
UC-0018	-	-	-	-	↑	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
UC-0019	-	-	-	-	-	-	-	-	-	-	-	↑	-	-	-	-	-	-	-	-	-	-	-	-	-	-
UC-0020	-	-	-	-	-	-	-	-	-	-	-	↑	-	-	-	-	-	-	-	-	-	-	-	-	-	-
UC-0021	-	-	-	-	-	-	-	↑	-	-	-	-	-	-	-	-	-	↑	-	-	-	-	-	-	-	-
UC-0023	-	-	-	-	-	-	-	-	-	-	-	↑	-	-	-	-	-	↑	-	-	-	-	-	-	-	-
UC-0024	-	-	-	-	-	-	-	-	-	-	-	↑	-	-	-	-	-	↑	-	-	-	-	-	-	-	-
UC-0025	-	-	-	-	-	-	-	-	-	-	-	↑	-	-	-	-	-	↑	-	-	-	-	-	-	-	-
UC-0026	-	-	-	-	-	-	-	↑	-	-	-	-	-	-	↑	-	-	-	-	-	-	-	-	-	-	-
UC-0027	-	-	-	-	-	-	-	-	-	-	-	↑	-	-	↑	-	-	-	-	-	-	-	-	-	-	-
UC-0029	-	-	-	-	↑	-	-	-	-	-	-	-	↑	-	-	-	-	-	-	-	-	-	-	-	-	-
UC-0030	-	-	-	-	-	-	-	-	-	-	-	↑	-	-	↑	-	-	-	-	-	-	-	-	-	-	-
UC-0031	-	-	-	-	-	-	-	-	-	-	-	↑	-	-	↑	-	-	-	-	-	-	-	-	-	-	-
UC-0035	-	-	-	-	-	-	-	-	-	-	-	-	↑	↑	-	-	-	-	-	-	-	↑	-	-	-	-
UC-0036	-	-	-	-	-	-	-	-	-	-	-	↑	-	-	-	-	-	-	-	-	-	↑	-	-	-	-
UC-0037	-	-	-	-	-	-	-	-	-	-	-	↑	-	-	-	-	-	-	-	-	-	↑	-	-	-	-


UC-0041
UC-0043
UC-0044
UC-0045
UC-0046
UC-0047
UC-0048
UC-0050
UC-0051
UC-0054
UC-0055
UC-0059
UC-0064
UC-0065
UC-0066
UC-0070
UC-0071
UC-0072
UC-0073
UC-0078
UC-0079
UC-0083
UC-0084
UC-0085
UC-0086
UC-0087
UC-0088
UC-0089
UC-0090
UC-0091
UC-0092
UC-0093
UC-0094
UC-0095
UC-0096
UC-0097
UC-0098
UC-0100



UC-0101	-	-	-	-	-	-	-	-	-	-	-	↑	-	-	-	↑	-	-	-	-	-	-	-	-	-
UC-0102	-	-	-	-	-	-	-	-	-	-	-	↑	-	-	-	↑	-	-	-	-	-	-	-	-	-
UC-0103	-	-	-	-	↑	-	-	-	-	-	-	-	-	-	-	↑	-	-	-	-	-	-	-	-	-
UC-0104	-	-	-	-	↑	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	↑	-	-
UC-0105	-	-	-	-	-	-	-	-	-	-	-	↑	-	-	-	-	-	-	-	-	↑	-	-	-	-

Matriz de rastreabilidad: Requisitos Funcionales

TRM-0003	OBJ-0001	OBJ-0002	OBJ-0003	OBJ-0004	OBJ-0005	OBJ-0006	OBJ-0007	OBJ-0008	OBJ-0009	OBJ-0010	OBJ-0011	OBJ-0012	OBJ-0013	OBJ-0014
NFR-0001	↑	-	-	-	↑	-	-	-	-	-	-	-	-	-
NFR-0002	-	↑	-	-	↑	-	-	-	-	-	-	-	-	-
NFR-0003	-	-	-	-	-	-	-	-	-	-	↑	-	-	↑
NFR-0004	↑	-	-	-	-	↑	↑	-	-	-	-	-	-	-
NFR-0005	-	-	-	-	↑	-	-	-	-	↑	-	-	-	-
NFR-0006	-	↑	-	-	↑	-	-	-	-	-	-	↑	-	-
NFR-0007	-	-	-	-	-	-	-	-	-	-	↑	-	-	↑
NFR-0008	-	-	-	-	-	↑	↑	-	-	↑	↑	-	-	↑

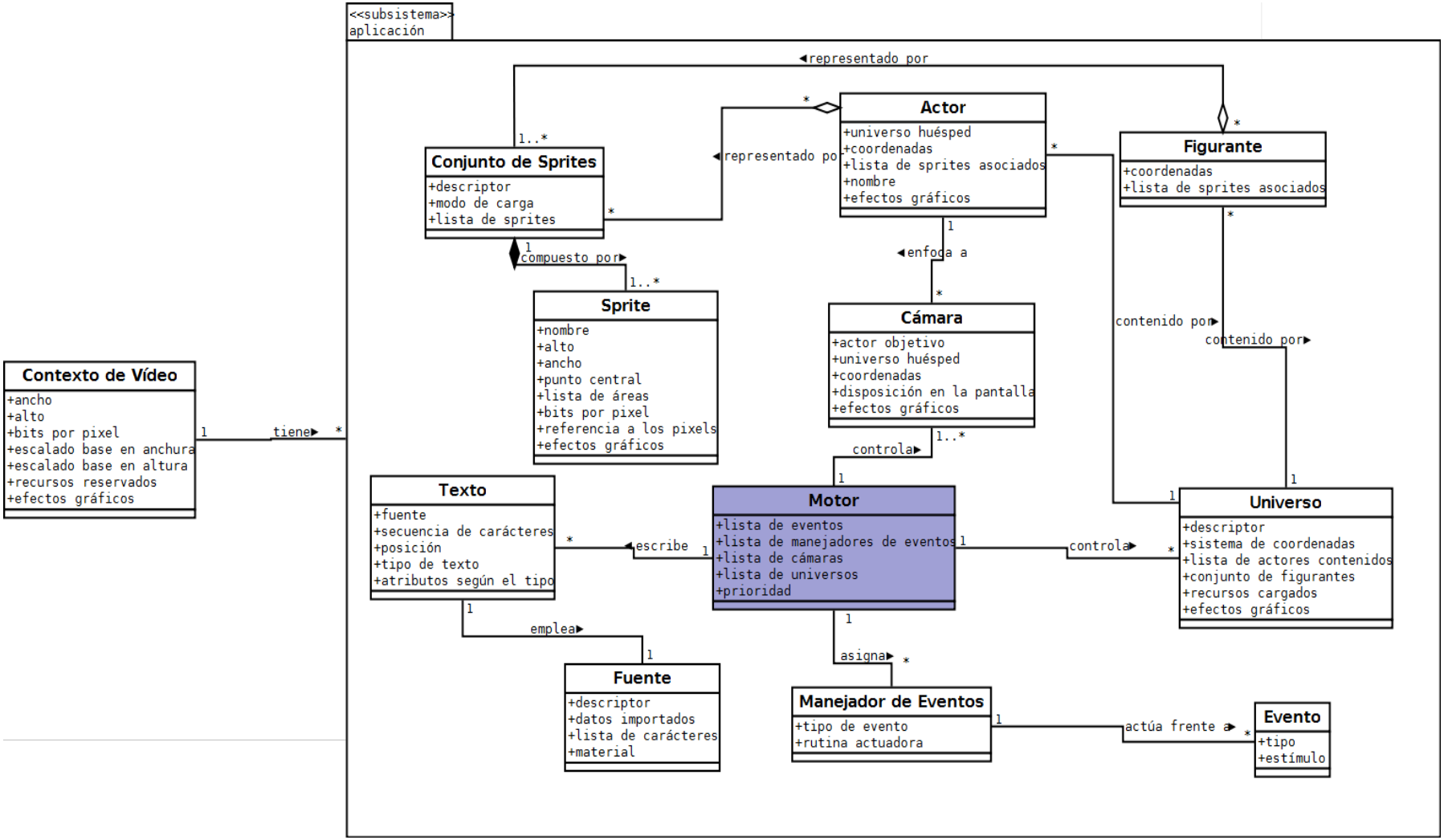
Matriz de rastreabilidad: Requisitos no Funcionales

Capítulo 4 : Análisis de Requisitos

En este capítulo encontraremos :

- Un diagrama que representa el **modelo estático** de la biblioteca.
- Una conjunto de tipos de **objetos**, tipos de **valores** y **asociaciones**.
- Una lista donde aparecen todas las **operaciones del sistema**.

4.1. Modelo Estático



TYP-0001	Contexto de Vídeo		
Versión	1.0 (27/07/2009)		
Autores	• José Manuel Barroso Galindo		
Dependencias	Ninguno		
Descripción	Este tipo de objetos representa *el contexto de vídeo de la aplicación, osea la manipulación directa de la pantalla con la información gráfica que ofrece el sistema.*		
Supertipo	Ninguno		
Subtipos	Ninguno		
Componentes	**Nombre**	**Tipo**	**Multiplicidad**
	-	-	-
Comentarios	Ninguno		

Atributo variable	Contexto de Vídeo:: Ancho
Descripción	Este atributo representa *el ancho en pixeles de la resolución que adopta el contexto de vídeo.*
Tipo	Natural
Comentarios	Ninguno

Atributo variable	Contexto de Vídeo:: Alto
Descripción	Este atributo representa *el alto en píxeles de la resolución que adopta el contexto de vídeo.*
Tipo	Natural
Comentarios	Ninguno

Atributo variable	Contexto de Vídeo:: Bits por pixel
Descripción	Este atributo representa *la profundidad de color que tiene cada pixel del contexto de vídeo.*
Tipo	Natural
Comentarios	Ninguno

Atributo variable	Contexto de Vídeo:: Escalado base en anchura
Descripción	Este atributo representa *el factor de escalado base en ancho de las proyecciones de la pantalla.*
Tipo	Real

Comentarios	Ninguno

Atributo variable	**Contexto de Vídeo:: Escalado base en altura**
Descripción	Este atributo representa *el factor de escalado base en alto de las proyecciones de la pantalla.*
Tipo	Real
Comentarios	Ninguno

Atributo variable	**Contexto de Vídeo:: Recursos Reservados**
Descripción	Este atributo representa *la información sensible del contexto de vídeo que debe ser almacenada para poder desempeñar correctamente sus operaciones.*
Tipo	Set(Integer)
Comentarios	Ninguno

Atributo variable	**Contexto de Vídeo:: Efectos gráficos**
Descripción	Este atributo representa *la información relativa a los efectos gráficos que se están ejecutando actualmente a toda información gráfica que se quiera representar en la pantalla.*
Tipo	Sequence(String)
Comentarios	Ninguno

TYP-0002	**Sprite**		
Versión	1.0 (27/07/2009)		
Autores	• José Manuel Barroso Galindo		
Dependencias	Ninguno		
Descripción	Este tipo de objetos representa *a una imagen individual importada desde un recurso externo.*		
Supertipo	Ninguno		
Subtipos	Ninguno		
Componentes	**Nombre**	**Tipo**	**Multiplicidad**
	-	-	-
Comentarios	Ninguno		

Atributo variable	**Sprite:: nombre**
Descripción	Este atributo representa *un calificativo que tiene el sprite a modo de*

	información consultable y modificable por la aplicación.
Tipo	String
Comentarios	Ninguno

Atributo variable	**Sprite:: alto**
Descripción	Este atributo representa *el valor de alto en pixels del sprite.*
Tipo	Natural
Comentarios	Ninguno

Atributo variable	**Sprite:: ancho**
Descripción	Este atributo representa *el valor de ancho en pixels del sprite.*
Tipo	Natural
Comentarios	Ninguno

Atributo variable	**Sprite:: Punto Central**
Descripción	Este atributo representa *las 2 coordenadas bidimensionales que describen un punto de la superficie del sprite que el sistema identificará como su centro.*
Tipo	Set(Natural)
Comentarios	Ninguno

Atributo variable	**Sprite:: Lista de áreas**
Descripción	Este atributo representa *la definición de diferentes áreas en el sprite que se indicaban en el recurso externo desde el que se ha importado el conjunto de sprites al que pertenece este sprite.*
Tipo	Set(Natural)
Comentarios	Ninguno

Atributo variable	**Sprite:: Recursos Reservados**
Descripción	Este atributo representa *la información gráfica del sprite, consistente en el valor de cada uno de sus pixels, entre otros datos.*
Tipo	Set(Integer)
Comentarios	Ninguno

Atributo variable	**Sprite:: Efectos gráficos**
Descripción	Este atributo representa *la información relativa a los efectos gráficos que se están ejecutando actualmente a toda información gráfica que se*

	quiera representar en la pantalla.
Tipo	Sequence(String)
Comentarios	Ninguno

TYP-0003	**Conjunto de Sprites**		
Versión	1.0 (27/07/2009)		
Autores	• José Manuel Barroso Galindo		
Dependencias	Ninguno		
Descripción	Este tipo de objetos representa *un conjunto de objetos Sprite, los cuales han sido importados desde el mismo recurso externo y por tanto tienen cierta información común especificada en este objeto.*		
Supertipo	Ninguno		
Subtipos	Ninguno		
Componentes	**Nombre**	**Tipo**	**Multiplicidad**
	lista de sprites	Set(Sprite)	1.. *
Comentarios	Ninguno		

Atributo variable	**Conjunto de Sprites:: descriptor**
Descripción	Este atributo representa *la ruta del fichero del cual ha sido importado este conjunto de sprites.*
Tipo	String
Comentarios	Ninguno

Atributo variable	**Conjunto de Sprites:: modo de carga**
Descripción	Este atributo representa *el modo de carga de la información gráfica (cargado en memoria principal, en la de vídeo o en ambas) que contienen todos los sprites del conjunto.*
Tipo	Natural
Comentarios	Ninguno

TYP-0004	**Fuente**
Versión	1.0 (27/07/2009)
Autores	• José Manuel Barroso Galindo
Dependencias	Ninguno
Descripción	Este tipo de objetos representa *una fuente tipográfica cargada desde un*

	recurso externo, y la colección de carácteres en forma de imágenes construida en base a dicha información.		
Supertipo	Ninguno		
Subtipos	Ninguno		
Componentes	**Nombre**	**Tipo**	**Multiplicidad**
	-	-	-
Comentarios	Ninguno		

Atributo variable	**Fuente:: descriptor**
Descripción	Este atributo representa *al fichero externo desde el cual ha sido importada esta fuente.*
Tipo	String
Comentarios	Ninguno

Atributo variable	**Fuente:: datos importados**
Descripción	Este atributo representa *los datos en memoria correspondientes a la tipografía de esta fuente.*
Tipo	Integer
Comentarios	Ninguno

Atributo variable	**Fuente:: lista de carácteres**
Descripción	Este atributo representa *la lista de carácteres en formato imagen generadas a partir de la tipografía de esta fuente.*
Tipo	Set(Integer)
Comentarios	Ninguno

Atributo variable	**Fuente:: tamaño**
Descripción	Este atributo representa *el tamaño de la fuente, que constituye una parte del material de la fuente.*
Tipo	Natural
Comentarios	Ninguno

Atributo variable	**Fuente:: color**
Descripción	Este atributo representa *el color de la fuente, que constituye la otra parte del material de la fuente.*
Tipo	Natural
Comentarios	Ninguno

TYP-0005	Texto		
Versión	1.0 (27/07/2009)		
Autores	• José Manuel Barroso Galindo		
Dependencias	Ninguno		
Descripción	Este tipo de objetos representa *un mensaje concreto que se desea representar en la pantalla, cuya información gráfica viene determinada por la Fuente que tenga asignada y los efectos gráficos.*		
Supertipo	Ninguno		
Subtipos	Ninguno		
Componentes	**Nombre**	**Tipo**	**Multiplicidad**
	-	-	-
Comentarios	Ninguno		

Atributo variable	Texto:: fuente
Descripción	Este atributo representa *la fuente que se está usando para representar este texto.*
Tipo	Real
Comentarios	Ninguno

Atributo variable	Texto:: tipo de texto
Descripción	Este atributo representa *el tipo de representación que hace este texto en pantalla (en un cuadro de texto, o en forma de línea).*
Tipo	Natural
Comentarios	Ninguno

Atributo variable	Texto:: secuencia de carácteres
Descripción	Este atributo representa *la cadena de carácteres que conforma el mensaje del texto.*
Tipo	Sequence(Real)
Comentarios	Ninguno

Atributo variable	Texto:: posición
Descripción	Este atributo representa *las 2 coordenadas que especifican la posición del texto en la pantalla.*
Tipo	Set(Integer)

Comentarios	Ninguno

Atributo variable	**Texto:: atributos del tipo**
Descripción	Este atributo representa *los atributos propios del tipo de este tipo de texto.*
Tipo	String
Comentarios	Ninguno

TYP-0006	**Evento**		
Versión	1.0 (27/07/2009)		
Autores	• José Manuel Barroso Galindo		
Dependencias	Ninguno		
Descripción	Este tipo de objetos representa *el estímulo que el sistema recibe.*		
Supertipo	Ninguno		
Subtipos	Ninguno		
Componentes	**Nombre**	**Tipo**	**Multiplicidad**
	-	-	-
Comentarios	Ninguno		

Atributo variable	**Evento:: tipo**
Descripción	Este atributo representa *el tipo de evento que se trata (movimiento del ratón,pulsación del teclado, etc..)*
Tipo	Natural
Comentarios	Ninguno

Atributo variable	**Evento:: estímulo**
Descripción	Este atributo representa *los datos asociados al estímulo propio de este evento.*
Tipo	Real
Comentarios	Ninguno

TYP-0007	**Manejador de Eventos**
Versión	1.0 (27/07/2009)
Autores	• José Manuel Barroso Galindo

Dependencias	Ninguno		
Descripción	Este tipo de objetos representa *la función que debe ejecutarse para gestionar un evento de un tipo específico.*		
Supertipo	Ninguno		
Subtipos	Ninguno		
Componentes	**Nombre**	**Tipo**	**Multiplicidad**
	-	-	-
Comentarios	Ninguno		

Atributo variable	**Manejador de Eventos:: tipo de evento**
Descripción	Este atributo representa *el tipo de evento que debe recibir este manejador.*
Tipo	Natural
Comentarios	Ninguno

Atributo variable	**Manejador de Eventos:: rutina actuadora**
Descripción	Este atributo representa *la función que se ha de ejecutar recibiendo como parámetro el evento del tipo adecuado.*
Tipo	Real
Comentarios	Ninguno

TYP-0008	**Motor**		
Versión	1.0 (27/07/2009)		
Autores	• José Manuel Barroso Galindo		
Dependencias	Ninguno		
Descripción	Este tipo de objetos representa *el núcleo arquitectónico del sistema, el cual gestionará los eventos entrantes, y almacena y controla los elementos de representación de cara a la fase de renderización de la biblioteca. (planificador)*		
Supertipo	Ninguno		
Subtipos	Ninguno		
Componentes	**Nombre**	**Tipo**	**Multiplicidad**
	-	-	-
Comentarios	Ninguno		

Atributo variable	**Motor:: lista de eventos**

Descripción	Este atributo representa *los elementos entrantes en el sistema listos para ser gestionados por un manejador.*
Tipo	Sequence(Integer)
Comentarios	Ninguno

Atributo variable	**Motor:: lista de manejadores de eventos**
Descripción	Este atributo representa *el conjunto de manejadores de eventos identificados según el tipo de eventos que pueden manejar.*
Tipo	Set(Real)
Comentarios	Ninguno

Atributo variable	**Motor:: lista de universos**
Descripción	Este atributo representa *el conjunto de universos bajo el control de este motor.*
Tipo	Sequence(Integer)
Comentarios	Ninguno

Atributo variable	**Motor:: lista de cámaras**
Descripción	Este atributo representa *el conjunto de cámaras bajo el control de este motor.*
Tipo	Set(Integer)
Comentarios	Ninguno

Atributo variable	**Motor:: prioridad**
Descripción	Este atributo representa *el valor de prioridad que tiene este motor a la hora de ser procesado por la biblioteca frente a otros motores.*
Tipo	Integer
Comentarios	Ninguno

TYP-0009	**Universo**
Versión	1.0 (27/07/2009)
Autores	• José Manuel Barroso Galindo
Dependencias	Ninguno
Descripción	Este tipo de objetos representa *la entidad de representación con coordenadas propias que es capaz de aceptar actores y figurantes, los cuales deben cumplir en todo momento las reglas de coordenadas del*

	universo.		
Supertipo	Ninguno		
Subtipos	Ninguno		
Componentes	**Nombre**	**Tipo**	**Multiplicidad**
	-	-	-
Comentarios	Ninguno		

Atributo variable	**Universo:: lista de actores**
Descripción	Este atributo representa *el conjunto de actores presentes en este universo.*
Tipo	Set(Integer)
Comentarios	Ninguno

Atributo variable	**Universo:: lista de figurantes**
Descripción	Este atributo representa *el conjunto de figurantes presentes en este universo.*
Tipo	Set(Integer)
Comentarios	Ninguno

Atributo variable	**Universo:: descriptor del fichero**
Descripción	Este atributo representa *la ruta del fichero desde el cual ha sido importado el universo.*
Tipo	String
Comentarios	Ninguno

Atributo variable	**Universo:: descriptor adicional numérico**
Descripción	Este atributo representa *el valor numérico que sirve para completar la descripción unívoca de un universo concreto.*
Tipo	Natural
Comentarios	Ninguno

Atributo variable	**Universo:: sistema de coordenadas**
Descripción	Este atributo representa *el sistema de coordenadas espaciales propio del universo, el cual conforma un medio de localización e interacción para los actores y figurantes.*
Tipo	Set(Natural)
Comentarios	Ninguno

Atributo variable	Universo:: Recursos Gráficos Cargados
Descripción	Este atributo representa *la información gráfica o la falta de ésta, que es necesaria para representar un universo.*
Tipo	Real
Comentarios	Ninguno

Atributo variable	Universo:: Efectos gráficos
Descripción	Este atributo representa *la información relativa a los efectos gráficos que se están ejecutando actualmente a toda información gráfica que se quiera representar en la pantalla.*
Tipo	Sequence(String)
Comentarios	Ninguno

TYP-0010	Actor		
Versión	1.0 (27/07/2009)		
Autores	• José Manuel Barroso Galindo		
Dependencias	Ninguno		
Descripción	Este tipo de objetos representa *a un elemento representable que puede interactuar y ser representado siempre que esté dentro de un universo.*		
Supertipo	Ninguno		
Subtipos	Ninguno		
Componentes	**Nombre**	**Tipo**	**Multiplicidad**
	-	-	-
Comentarios	Ninguno		

Atributo variable	Actor:: universo huésped
Descripción	Este atributo representa *el universo al cual pertenece el actor en un momento determinado.*
Tipo	Integer
Comentarios	Ninguno

Atributo variable	Actor:: coordenadas
Descripción	Este atributo representa *las coordenadas espaciales que tiene el actor para su localización en su universo huésped.*
Tipo	Set(Natural)

Comentarios	Ninguno

Atributo variable	**Actor:: lista de sprites asociados**
Descripción	Este atributo representa *los sprites que representan al actor en un momento determinado.*
Tipo	Set(Integer)
Comentarios	Ninguno

Atributo variable	**Actor:: nombre**
Descripción	Este atributo representa *un calificativo que tiene el actor a modo de información consultable y modificable por la aplicación.*
Tipo	String
Comentarios	Ninguno

Atributo variable	**Actor:: Efectos gráficos**
Descripción	Este atributo representa *la información relativa a los efectos gráficos que se están ejecutando actualmente a toda información gráfica que se quiera representar en la pantalla.*
Tipo	Sequence(String)
Comentarios	Ninguno

TYP-0011	**Figurante**		
Versión	1.0 (27/07/2009)		
Autores	• José Manuel Barroso Galindo		
Dependencias	Ninguno		
Descripción	Este tipo de objetos representa *las entidades gráficas contenidas en un universo para otorgar una visualización al mismo pero que no interactuan con su entorno de ningún modo.*		
Supertipo	Ninguno		
Subtipos	Ninguno		
Componentes	**Nombre**	**Tipo**	**Multiplicidad**
	-	-	-
Comentarios	en la implementación se percibe el figurante como un actor, ya que posee metodos comunes a el, pero por consistencia por el actor hemos mantenido el tipo figurante		

Atributo variable	Figurante:: coordenadas
Descripción	Este atributo representa *la localización espacial en el universo en el que están contenidos.*
Tipo	Real
Comentarios	Ninguno

Atributo variable	Figurante:: lista de sprites asociados
Descripción	Este atributo representa *los sprites que representan al figurante en un momento determinado.*
Tipo	Set(Integer)
Comentarios	Ninguno

TYP-0012	Cámara		
Versión	1.0 (27/07/2009)		
Autores	• José Manuel Barroso Galindo		
Dependencias	Ninguno		
Descripción	Este tipo de objetos representa *la herramienta que posee la aplicación para representar la información gráfica de un universo consistente en los actores y figurantes contenidos en él.*		
Supertipo	Ninguno		
Subtipos	Ninguno		
Componentes	**Nombre**	**Tipo**	**Multiplicidad**
	-	-	-
Comentarios	Ninguno		

Atributo variable	Cámara:: actor objetivo
Descripción	Este atributo representa *es el actor que utiliza de referencia la cámara para representar la parte que esté próxima a dicho actor del universo que lo contiene.*
Tipo	Integer
Comentarios	Ninguno

Atributo variable	Cámara:: universo huésped
Descripción	Este atributo representa *al universo huésped del actor objetivo.*
Tipo	Integer
Comentarios	Ninguno

Atributo variable	Cámara:: coordenadas
Descripción	Este atributo representa *son las coordenadas espaciales que ocupa la cámara en el universo huésped,y que se basan en las coordenadas del actor objetivo (normalmente son las mismas, pero hay muchas situaciones en las que son distintas, como cuando un actor se acerca al borde del espacio del universo).*
Tipo	Real
Comentarios	Ninguno

Atributo variable	Cámara:: disposición en la pantalla
Descripción	Este atributo representa *define el área de visualización de la representación de la cámara situándola en una parte arbitraria de la pantalla.*
Tipo	Set(Integer)
Comentarios	Ninguno

Atributo variable	Cámara:: Efectos gráficos
Descripción	Este atributo representa *la información relativa a los efectos gráficos que se están ejecutando actualmente a toda información gráfica que se quiera representar en la pantalla.*
Tipo	Sequence(String)
Comentarios	Ninguno

ASO-0002	emplea(Texto, Fuente)
Versión	1.0 (27/07/2009)
Autores	• José Manuel Barroso Galindo
Dependencias	Ninguno
Descripción	Este tipo de asociación representa el hecho de que *que un Texto necesita de una Fuente para representar su mensaje en pantalla.*
Comentarios	Ninguno

Rol variable	emplea(Texto, Fuente):: emplea
Descripción	Este rol representa *al Texto que almacena el puntero de la Fuente que usa.*
Tipo	Texto
Multiplicidad	1

Comentarios	Ninguno

Rol variable	**emplea(Texto, Fuente):: emplea**
Descripción	Este rol representa *la Fuente que contiene la información gráfica necesaria para que el texto la utilice a la hora de representar su mensaje.*
Tipo	Fuente
Multiplicidad	1
Comentarios	Ninguno

ASO-0003	**asigna(Motor, Manejador de Eventos)**
Versión	1.0 (27/07/2009)
Autores	• José Manuel Barroso Galindo
Dependencias	Ninguno
Descripción	Este tipo de asociación representa el hecho de que *que un Motor se utiliza para asignar el manejo de un tipo de evento determinado al Manejador de Eventos deseado.*
Comentarios	Ninguno

Rol variable	**asigna(Motor, Manejador de Eventos):: asigna**
Descripción	Este rol representa *el Motor que se usa para hacer la asignación*
Tipo	Motor
Multiplicidad	1
Comentarios	Ninguno

Rol variable	**asigna(Motor, Manejador de Eventos):: asigna**
Descripción	Este rol representa *la función que se elije para el tipo de eventos deseado.*
Tipo	Manejador de Eventos
Multiplicidad	1
Comentarios	Ninguno

1.1 Valores

VAL-0001	**Punto**
Versión	1.0 (07/11/2010)
Autores	• José Manuel Barroso Galindo

Dependencias	Ninguno
Descripción	Este tipo valor representa *un punto en un plano bidimensional.*
Definición	Es un tipo de valor que se compone de 2 variables Integer.
Comentarios	Ninguno

Atributo variable	**Punto:: m_x**
Descripción	Este atributo representa el valor numérico de la coordenada x.
Tipo	Integer
Comentarios	Se asocia al eje horizontal.

Atributo variable	**Punto:: m_y**
Descripción	Este atributo representa el valor numérico de la coordenada y.
Tipo	Integer
Comentarios	Se asocia al eje vertical.

VAL-0002	**Coordenada**
Versión	1.0 (07/11/2010)
Autores	• José Manuel Barroso Galindo
Dependencias	Ninguno
Descripción	Este tipo valor representa *un punto en un plano tridimensional.*
Definición	Es un tipo de valor que se compone de 3 variables Integer.
Comentarios	Ninguno

Atributo variable	**Coordenada:: m_x**
Descripción	Este atributo representa el valor numérico de la coordenada x.
Tipo	Integer
Comentarios	Se asocia al eje horizontal.

Atributo variable	**Coordenada:: m_y**
Descripción	Este atributo representa el valor numérico de la coordenada y.
Tipo	Integer
Comentarios	Se asocia al eje vertical.

Atributo variable	Coordenada:: m_z
Descripción	Este atributo representa el valor numérico de la coordenada z.
Tipo	Integer
Comentarios	Se asocia a un valor de profundidad.

VAL-0003	Boolean
Versión	1.0 (07/11/2010)
Autores	• José Manuel Barroso Galindo
Dependencias	Ninguno
Descripción	Este tipo valor representa *un estado binario de algún elemento.*
Definición	Es un tipo de valor que sólo puede tener dos valores, CIERTO o FALSO.
Comentarios	Ninguno

VAL-0004	Efecto Gráfico
Versión	1.0 (07/11/2010)
Autores	• José Manuel Barroso Galindo
Dependencias	Ninguno
Descripción	Este tipo valor representa *un efecto gráfico concreto.*
Definición	Es un valor que se compone de un valor Natural, que actuará como código identificador del efecto, y un conjunto de Reales que determinarán su configuración.
Comentarios	Ninguno

Atributo variable	Efecto Gráfico:: tipo
Descripción	Este atributo representa *con un valor numérico el tipo de efecto gráfico del que se trata.*
Tipo	Natural
Comentarios	Ninguno

Atributo variable	Efecto Gráfico:: Configuración
Descripción	Este atributo representa *los valores que utilizará el efecto gráfico para lograr el efecto deseado.*
Tipo	Sequence(Real)
Comentarios	Ninguno

4.2. Modelado dinámico, funcional y prototipos de interfaz de usuario

4.2.1. Gestión General y de Motores

SOP-0001	**Iniciar la Biblioteca**
Tipo del resultado	Se ha inicializado la biblioteca.
Versión	1.0 (09/11/2010)
Autores	• José Manuel Barroso Galindo
Dependencias	• [UC-0001] Iniciar la Biblioteca
Descripción	La aplicación solicita la inicialización de la biblioteca dando de manera opcional parámetros si así lo requiere
Parámetros	xmlconfig : Boolean -- *Si se pasa como CIERTO, la biblioteca tratará de procesa un archivo XML llamado 'config.xml' que estará en el directorio de recursos de la biblioteca y que podrá ser editado por el usuario. Gracias a él, se pueden automatizar ciertos parámetros de configuración inicial.*
Expresiones de precondición	**pre1**: La biblioteca no ha sido inicializada con anterioridad. **pre2**: Si se han dado parámetros, deben ser válidos.
Expresiones de precondición (OCL)	**pre1**: true **pre2**: true
Expresiones de postcondición	**post1**: La biblioteca ahora está inicializada.
Expresiones de postcondición (OCL)	**post1**: true
Comentarios	Ninguno

Excepción	**Iniciar la Biblioteca::Imposible inicializar**
Condición	El sistema no ha logrado ejecutar las rutinas de inicialización de la biblioteca con éxito.
Condición (OCL)	false
Expresión	La biblioteca registra un Error con un mensaje explicativo y devuelve

	FRACASO.
Expresión OCL	true
Comentarios	Se debe a un mal montaje, o a una mala respuesta de la plataforma.

Excepción	**Iniciar la Biblioteca::Parámetros erróneos.**
Condición	Los parámetros dados, no tienen valores adecuados.
Condición (OCL)	false
Expresión	La biblioteca registra un Error con un mensaje explicativo y devuelve FRACASO.
Expresión OCL	true
Comentarios	Ninguno

Excepción	**Iniciar la Biblioteca::Inicialización ya realizada**
Condición	La biblioteca ya ha sido inicializada con anterioridad.
Condición (OCL)	false
Expresión	La biblioteca registra un Error con un mensaje explicativo y devuelve FRACASO.
Expresión OCL	true
Comentarios	Ninguno

SOP-0002	**Añadir Motor**
Tipo del resultado	Éxito, se ha añadido el motor solicitado a la lista de motores preparados para su ejecución.
Versión	1.0 (09/11/2010)
Autores	• José Manuel Barroso Galindo
Dependencias	• [UC-0044] Añadir Motor
Descripción	La aplicación solicita añadir un motor a la biblioteca,
Parámetros	engine : Motor -- *Puntero a la instancia del motor que se trata de agregar.* priority : Integer -- *Valor de la prioridad que deberá tener el motor.*
Expresiones de precondición	**pre1**: La biblioteca debe haber sido anteriormente inicializada. **pre2**: El puntero debe ser de la clase apropiada.

Expresiones de precondición (OCL)	**pre1**: true **pre2**: true
Expresiones de postcondición	**post1**: El motor ha quedado registrado en el gestor de motores de la biblioteca.
Expresiones de postcondición (OCL)	**post1**: true
Comentarios	Ninguno

Excepción	**Añadir Motor::Biblioteca no inicializada**
Condición	La biblioteca no ha sido inicializada con anterioridad, por tanto se devuelve FRACASO.
Condición (OCL)	false
Expresión	La biblioteca registra un Error con un mensaje explicativo y devuelve FRACASO.
Expresión OCL	true
Comentarios	Ninguno

Excepción	**Añadir Motor::Puntero erróneo**
Condición	El puntero es de un tipo inválido o es nulo, por tanto se devuelve FRACASO.
Condición (OCL)	false
Expresión	La biblioteca registra un Error con un mensaje explicativo y devuelve FRACASO.
Expresión OCL	true
Comentarios	Ninguno

SOP-0003	**Procesar Motores**
Tipo del resultado	Éxito, si se procesan todos los motores registrados hasta que se solicite el final de su ejecución.
Versión	1.0 (09/11/2010)
Autores	• José Manuel Barroso Galindo
Dependencias	• [UC-0044] Añadir Motor

Descripción	La aplicación solicita la ejecución de los motores registrados por la biblioteca.
Parámetros	Ninguno
Expresiones de precondición	**pre1**: La biblioteca debe haber sido anteriormente inicializada.
Expresiones de precondición (OCL)	**pre1**: true
Expresiones de postcondición	**post2**: Si un motor seleccionado por cualquier motivo no es apto para ser procesado, el sistema devuelve FRACASO.
Expresiones de postcondición (OCL)	**post2**: true
Comentarios	Ninguno

Excepción	**Procesar Motores::Biblioteca no inicializada**
Condición	La biblioteca no ha sido inicializada con anterioridad, por tanto se devuelve FRACASO.
Condición (OCL)	false
Expresión	La biblioteca registra un Error con un mensaje explicativo y devuelve FRACASO.
Expresión OCL	true
Comentarios	Ninguno

SOP-0004	**Modificar prioridad de Motor**
Tipo del resultado	Valor anterior de prioridad, que es un Integer mayor o igual que 0.
Versión	1.0 (09/11/2010)
Autores	• José Manuel Barroso Galindo
Dependencias	• [UC-0045] Modificar prioridad de Motor
Descripción	La aplicación solicita para el motor indicado, cambiar el Integer que determina su prioridad de ejecución.
Parámetros	priority : emplea -- *Valor de la prioridad que deberá tener el motor.*
Expresiones de precondición	Ninguno
Expresiones de	true

precondición (OCL)	
Expresiones de postcondición	**post1**: La proiridad del motor ha sido modificada.
Expresiones de postcondición (OCL)	**post1**: true
Comentarios	Ninguno

SOP-0005	**Obtener Motor activo**
Tipo del resultado	La instancia del motor activo.
Versión	1.0 (09/11/2010)
Autores	• José Manuel Barroso Galindo
Dependencias	• [UC-0046] Obtener Motor activo
Descripción	La aplicación solicita el acceso a la instancia del motor activo en el momento de ejecución actual.
Parámetros	Ninguno
Expresiones de precondición	**pre1**: La biblioteca debe haber sido anteriormente inicializada.
Expresiones de precondición (OCL)	**pre1**: true
Expresiones de postcondición	Ninguno
Expresiones de postcondición (OCL)	true
Comentarios	Ninguno

Excepción	**Obtener Motor activo::biblioteca no inicializada**
Condición	La biblioteca no ha sido inicializada con anterioridad, por tanto se devuelve FRACASO.
Condición (OCL)	false
Expresión	La biblioteca registra un Error con un mensaje explicativo y devuelve FRACASO.
Expresión	true

OCL	
Comentarios	Ninguno

SOP-0006	**Retirar Motor**
Tipo del resultado	Éxito, si la instancia del motor indicada ha sido desregistrada.
Versión	1.0 (09/11/2010)
Autores	• José Manuel Barroso Galindo
Dependencias	• [UC-0105] Retirar Motor
Descripción	La aplicación solicita que se desregistre de la biblioteca el motor indicado.
Parámetros	engine : Motor -- *Puntero a la instancia del motor que se trata de retirar.*
Expresiones de precondición	**pre1**: El motor indicado está registrado por la biblioteca.
Expresiones de precondición (OCL)	**pre1**: true
Expresiones de postcondición	**post1**: El motor indicado ya no está registrado por la biblioteca.
Expresiones de postcondición (OCL)	**post1**: true
Comentarios	Una vez retirado, se finaliza si fuera necesario y elimina.

Excepción	**Retirar Motor::Motor no encontrado**
Condición	El motor que se quería retirar no existe en la biblioteca, por tanto se devuelve FRACASO.
Condición (OCL)	false
Expresión	La biblioteca registra un Error con un mensaje explicativo y devuelve FRACASO.
Expresión OCL	true
Comentarios	Ninguno

SOP-0007	**Leer información sobre incidentes anotados por la biblioteca**

Tipo del resultado	Un conjunto de cadenas que describen todos los incidentes anotados por la biblioteca.
Versión	1.0 (09/11/2010)
Autores	• José Manuel Barroso Galindo
Dependencias	• [UC-0043] Leer información sobre incidentes anotados por la Biblioteca
Descripción	La aplicación solicita acceder a la información sobre los incidentes anotados por la biblioteca.
Parámetros	Ninguno
Expresiones de precondición	**pre1**: La biblioteca debe haber sido anteriormente inicializada.
Expresiones de precondición (OCL)	**pre1**: true
Expresiones de postcondición	**post1**: La información consultada ha sido liberada.
Expresiones de postcondición (OCL)	**post1**: true
Comentarios	Ninguno

Excepción	**Leer información sobre incidentes anotados por la biblioteca::Biblioteca no inicializada**
Condición	La biblioteca no ha sido inicializada con anterioridad, por tanto se devuelve FRACASO.
Condición (OCL)	false
Expresión	La biblioteca registra un Error con un mensaje explicativo y devuelve FRACASO.
Expresión OCL	true
Comentarios	Ninguno

4.2.2. Gestión de la Pantalla

SOP-0008	Inicializar Contexto de Vídeo
Tipo del resultado	Éxito, si la inicialización se ha llevado a cabo correctamente.
Versión	1.0 (09/11/2010)
Autores	• José Manuel Barroso Galindo
Dependencias	• [UC-0011] Inicializar Contexto de Vídeo
Descripción	La aplicación solicita la inicialización del contexto de vídeo.
Parámetros	width : Integer -- *Valor de ancho de la resolución a adoptar.* height : Integer -- *Valor de alto de la resolución a adoptar.* bpp : Integer -- *Valor de bit por pixel para la resolución a adoptar.* fullscreen : Boolean -- *Cierto si se desea iniciar el contexto de vídeo en pantalla completa. Falso si se prefiere que sea en modo ventana.* doublebuff : Boolean -- *Cierto para activar el doble búffer.*
Expresiones de precondición	**pre1**: La biblioteca debe haber sido anteriormente inicializada. **pre2**: No está inicializado el contexto de vídeo.
Expresiones de precondición (OCL)	**pre1**: true **pre2**: true
Expresiones de postcondición	**post1**: Se ha creado el contexto de vídeo.
Expresiones de postcondición (OCL)	**post1**: true
Comentarios	El double buffering es una técnica para mejorar el rendimiento de la actualización de la pantalla a costa de un mallor gasto de memoria gráfica.

Excepción	Inicializar Contexto de Vídeo::biblioteca no inicializada
Condición	La biblioteca no ha sido inicializada con anterioridad, por tanto se devuelve FRACASO.
Condición (OCL)	false
Expresión	La biblioteca registra un Error con un mensaje explicativo y devuelve FRACASO.
Expresión OCL	true
Comentarios	Ninguno

Excepción	Inicializar Contexto de Vídeo::contexto de vídeo ya inicializado
Condición	El contexto de vídeo ya ha sido inicializado anteriormente, por tanto se devuelve FRACASO.
Condición (OCL)	false
Expresión	La biblioteca registra un Error con un mensaje explicativo y devuelve FRACASO.
Expresión OCL	true
Comentarios	Ninguno

Excepción	Inicializar Contexto de Vídeo::parámetros incompatibles
Condición	No se ha podido configurar un contexto de vídeo con los parámetros dados y por tanto se intenta crear uno configurado por defecto. A continuación devuelve FRACASO.
Condición (OCL)	false
Expresión	La biblioteca registra un Error con un mensaje explicativo y devuelve FRACASO.
Expresión OCL	true
Comentarios	Ninguno

SOP-0009	Finalizar Contexto de Vídeo
Tipo del resultado	Éxito, si se ha finalizado el contexto de vídeo sin incidencias.
Versión	1.0 (09/11/2010)
Autores	• José Manuel Barroso Galindo
Dependencias	• [UC-0012] Finalizar Contexto de Vídeo
Descripción	La aplicación solicita finalizar el contexto de vídeo.
Parámetros	Ninguno
Expresiones de precondición	**pre1**: El contexto de vídeo debe estar inicializado.
Expresiones de precondición (OCL)	**pre1**: true
Expresiones de postcondición	**post1**: El contexto de vídeo ya no está inicializado

Expresiones de postcondición (OCL)	**post1**: true
Comentarios	Ninguno

Excepción	**Finalizar Contexto de Vídeo::contexto de vídeo no inicializado**
Condición	La pre1 no se cumple, por tanto se devuelve FRACASO.
Condición (OCL)	false
Expresión	La biblioteca registra un Error con un mensaje explicativo y devuelve FRACASO.
Expresión OCL	true
Comentarios	Ninguno

SOP-0010	**Reiniciar Contexto de Vídeo**
Tipo del resultado	Éxito, si se logra reiniciar el contexto de video salvando reestableciendo los recursos gráficos cargados en el anterior contexto de vídeo.
Versión	1.0 (09/11/2010)
Autores	• José Manuel Barroso Galindo
Dependencias	• [UC-0013] Reiniciar Contexto de Vídeo
Descripción	La aplicación solicita reiniciar el contexto de vídeo con nuevos parámetros. La información gráfica guardada hasta el momento debe ser liberada salvando sus descriptores y su estructura. Se finaliza el contexto de vídeo. Se inicia el contexto de vídeo. Se vuelve a cargar la información gráfica a partir de los datos salvados.
Parámetros	width : Integer -- *Valor de ancho de la resolución a adoptar.* height : Integer -- *Valor de alto de la resolución a adoptar.* bpp : Integer -- *Valor de bit por pixel para la resolución a adoptar.* fullscreen : Boolean -- *Cierto si se desea iniciar el contexto de vídeo en pantalla completa. Falso si se prefiere que sea en modo ventana.* doublebuff : Boolean -- *Cierto para activar el doble búffer.*
Expresiones de precondición	**pre1**: El contexto de vídeo debe estar inicializado.

Expresiones de precondición (OCL)	**pre1**: true
Expresiones de postcondición	Ninguno
Expresiones de postcondición (OCL)	true
Comentarios	Ninguno

Excepción	**Reiniciar Contexto de Vídeo::contexto de vídeo no inicializado**
Condición	La pre1 no se cumple, por tanto se devuelve FRACASO.
Condición (OCL)	false
Expresión	La biblioteca registra un Error con un mensaje explicativo y devuelve FRACASO.
Expresión OCL	true
Comentarios	Ninguno

Excepción	**Reiniciar Contexto de Vídeo::parámetros incompatibles**
Condición	No se ha podido configurar un contexto de vídeo con los parámetros dados y por tanto se intenta crear uno configurado por defecto. A continuación devuelve FRACASO.
Condición (OCL)	false
Expresión	La biblioteca registra un Error con un mensaje explicativo y devuelve FRACASO.
Expresión OCL	true
Comentarios	Ninguno

SOP-0011	**Actualizar la Pantalla**
Tipo del resultado	Éxito, si se ha actualizado la pantalla representando todos los elementos gráficos que estaban pendientes de ser renderizados.
Versión	1.0 (09/11/2010)
Autores	• José Manuel Barroso Galindo
Dependencias	• [UC-0018] Actualizar la Pantalla

Descripción	La aplicación o el caso de uso "Siguiente Etapa de ejecución" solicita la actualización de los elementos gráficos llamados a ser renderizados en la pantalla.
Parámetros	Ninguno
Expresiones de precondición	**pre1**: El contexto de vídeo debe estar inicializado.
Expresiones de precondición (OCL)	**pre1**: true
Expresiones de postcondición	**post1**: La pantalla ahora está actualizada.
Expresiones de postcondición (OCL)	**post1**: true
Comentarios	En este momento la biblioteca invoca a OpenGL que llevará a cabo el vuelco a la memoria de vídeo el resultado de todo el proceso de renderización.

Excepción	**Actualizar la Pantalla::contexto de vídeo no inicializado**
Condición	La pre1 no se cumple, por tanto se devuelve FRACASO.
Condición (OCL)	false
Expresión	La biblioteca registra un Error con un mensaje explicativo y devuelve FRACASO.
Expresión OCL	true
Comentarios	Ninguno

SOP-0012	**Borrar la Pantalla**
Tipo del resultado	Éxito, si se ha llevado a cabo el borrado correctamente.
Versión	1.0 (09/11/2010)
Autores	• José Manuel Barroso Galindo
Dependencias	• [UC-0019] Borrar la Pantalla
Descripción	La aplicación solitia borrar la pantalla, por ello cualquier elemento gráfico que haya sido llamado anteriormente a ser renderizado ya no aparecerá en la siguiente actualización.
Parámetros	Ninguno

Expresiones de precondición	**pre1**: El contexto de vídeo debe estar inicializado.
Expresiones de precondición (OCL)	**pre1**: true
Expresiones de postcondición	**post1**: No hay ningún elemento gráfico dispuesto a ser renderizado.
Expresiones de postcondición (OCL)	**post1**: true
Comentarios	Ninguno

Excepción	**Borrar la Pantalla::contexto de vídeo no inicializado**
Condición	La pre1 no se cumple, por tanto se devuelve FRACASO.
Condición (OCL)	false
Expresión	La biblioteca registra un Error con un mensaje explicativo y devuelve FRACASO.
Expresión OCL	true
Comentarios	Ninguno

SOP-0013	**Situar escena 2D en la Pantalla**
Tipo del resultado	Éxito, si el marco de la escena queda situado en la pantalla según los parametros ofrecidos por la aplicación.
Versión	1.0 (09/11/2010)
Autores	• José Manuel Barroso Galindo
Dependencias	• [UC-0020] Situar Escena 2D en la Pantalla
Descripción	La aplicación indica un marco dentro de la Pantalla en el que la información gráfica que se especificará a continuación se va a representar durante la actual fase de renderización.
Parámetros	posx : Real -- *El valor del eje X de la posición superior-izquierda del marco que se está definiendo.* posy : Real -- *El valor del eje Y de la posición superior-izquierda del marco que se está definiendo.* width : Real -- *El valor de ancho que ocupará el marco a lo largo del eje X a partir de posx.* height : Real -- *El valor de alto que ocupará el marco a lo largo del eje Y a partir de posy.*

Expresiones de precondición	**pre1**: El contexto de vídeo debe estar inicializado.
Expresiones de precondición (OCL)	**pre1**: true
Expresiones de postcondición	**post1**: Toda la información gráfica especificada a partir de entonces se hará en base al marco de la escena establecido.
Expresiones de postcondición (OCL)	**post1**: true
Comentarios	Ninguno

Excepción	**Situar escena 2D en la Pantalla::contexto de vídeo no inicializado**
Condición	La pre1 no se cumple, por tanto se devuelve FRACASO.
Condición (OCL)	false
Expresión	La biblioteca registra un Error con un mensaje explicativo y devuelve FRACASO.
Expresión OCL	true
Comentarios	Ninguno

Excepción	**Situar escena 2D en la Pantalla::parámetros inválidos**
Condición	Si los parametros indicados son inválidos, la biblioteca devuelve FRACASO.
Condición (OCL)	false
Expresión	La biblioteca registra un Error con un mensaje explicativo y devuelve FRACASO.
Expresión OCL	true
Comentarios	Ninguno

4.2.3. Gestión de Imágenes

SOP-0014	Cargar Conjunto de Sprites
Tipo del resultado	Integer identificador asociado al Conjunto de Sprites del descriptor dado.
Versión	1.0 (09/11/2010)
Autores	• José Manuel Barroso Galindo
Dependencias	• [UC-0004] Cargar Conjunto de Sprites
Descripción	La aplicación solicita cargar un Conjunto de Sprites indicando su descriptor.
Parámetros	descriptor : String -- *El descriptor del conjunto de sprites.* mode : Natural -- *El modo del Conjunto de Spriteset. Según el modo, se requerirá más o menos memoria.*
Expresiones de precondición	**pre1**: El contexto de vídeo debe estar inicializado.
Expresiones de precondición (OCL)	**pre1**: true
Expresiones de postcondición	**post1**: El Conjunto de Sprites correspondiente está almacenado.
Expresiones de postcondición (OCL)	**post1**: true
Comentarios	Valores posibles para 'mode' : ONLY_TEXTURE : El Conjunto de Sprites guarda su información gráfica en la memoria de vídeo. ONLY_SDL_SURFACE : El Conjunto de Sprites guarda su información en la memoria principal (entonces el conjunto no es apto para ser renderizado). WITH_SDL_SURFACE : Combinación de ambas opciones anteriores.

Excepción	Cargar Conjunto de Sprites::contexto de vídeo no inicializado
Condición	La pre1 no se cumple, por tanto se devuelve FRACASO.
Condición (OCL)	false
Expresión	La biblioteca registra un Error con un mensaje explicativo y devuelve FRACASO.

Expresión OCL	true
Comentarios	Ninguno

Excepción	**Cargar Conjunto de Sprites::carga fallida**
Condición	Si se falla al intentar procesar el descriptor, se devuelve FRACASO.
Condición (OCL)	false
Expresión	La biblioteca registra un Error con un mensaje explicativo y devuelve FRACASO.
Expresión OCL	true
Comentarios	Ninguno

SOP-0015	**Buscar Conjunto de Sprites**
Tipo del resultado	Integer identificador del Conjunto de Sprites especificado.
Versión	1.0 (09/11/2010)
Autores	• José Manuel Barroso Galindo
Dependencias	• [UC-0003] Buscar Conjunto de Sprites
Descripción	La aplicación solicita el identificador de un Conjunto de Sprites indicando su descriptor o su instancia.
Parámetros	name : String -- *Una cadena que define el Conjunto de Sprites.* object : Conjunto de Sprites -- *Puntero a la instancia del Conjunto de Sprites.*
Expresiones de precondición	**pre1**: El Conjunto de Sprites buscado está cargado.
Expresiones de precondición (OCL)	**pre1**: true
Expresiones de postcondición	Ninguno
Expresiones de postcondición (OCL)	true
Comentarios	Ninguno

Excepción	**Buscar Conjunto de Sprites::no encontrado**

Condición	Si el conjunto de sprites no ha sido encontrado, se devuelve FRACASO
Condición (OCL)	false
Expresión	La biblioteca registra un Error con un mensaje explicativo y devuelve FRACASO.
Expresión OCL	true
Comentarios	Ninguno

SOP-0016	**Eliminar Conjunto de Sprites**
Tipo del resultado	Éxito si el conjunto de sprites especificado ha sido correctamente eliminado.
Versión	1.0 (09/11/2010)
Autores	• José Manuel Barroso Galindo
Dependencias	• [UC-0005] Eliminar Conjunto de Sprites
Descripción	La aplicación solicita eliminar un Conjunto de Sprites indicando su identificador.
Parámetros	n : Integer -- *El identificador asociado al Conjunto de Sprites.*
Expresiones de precondición	**pre1**: El Conjunto de Sprites buscado está cargado.
Expresiones de precondición (OCL)	**pre1**: true
Expresiones de postcondición	**post1**: Se ha registrado el cese de uso de dicho Conjunto de Sprites y posiblemente haya sido eliminado.
Expresiones de postcondición (OCL)	**post1**: true
Comentarios	Ninguno

Excepción	**Eliminar Conjunto de Sprites::no encontrado**
Condición	Si no localiza el Conjunto de Sprites requerido, la biblioteca devuelve FRACASO.
Condición (OCL)	false
Expresión	La biblioteca registra un Error con un mensaje explicativo y devuelve FRACASO.
Expresión	true

OCL	
Comentarios	Ninguno

SOP-0017	**Obtener instancia de un Conjunto de Sprites**
Tipo del resultado	Puntero a la instancia del Conjunto de Sprites.
Versión	1.0 (09/11/2010)
Autores	• José Manuel Barroso Galindo
Dependencias	• [UC-0007] Obtener instancia de un Conjunto de Sprites
Descripción	La aplicación solicita la instancia de un Conjunto de Sprites mediante su identificador.
Parámetros	n : Integer -- *El identificador asociado al Conjunto de Sprites.*
Expresiones de precondición	**pre1**: El Conjunto de Sprites está cargado.
Expresiones de precondición (OCL)	**pre1**: true
Expresiones de postcondición	Ninguno
Expresiones de postcondición (OCL)	true
Comentarios	Ninguno

Excepción	**Obtener instancia de un Conjunto de Sprites::no encontrado**
Condición	Si la biblioteca no encuentra instancia asociada al identificador dado, devuelve FRACASO.
Condición (OCL)	false
Expresión	La biblioteca registra un Error con un mensaje explicativo y devuelve FRACASO.
Expresión OCL	true
Comentarios	Ninguno

SOP-0018	**Obtener Sprite**
Tipo del	Puntero a la instancia del Sprite.

resultado	
Versión	1.0 (09/11/2010)
Autores	• José Manuel Barroso Galindo
Dependencias	• [UC-0098] Obtener Sprite de un Conjunto de Sprites
Descripción	La aplicación solicita a través de la instancia obtenida de un Conjunto de Sprites, solicita un Sprite.
Parámetros	spriteset : Integer -- *El identificador asociado al Conjunto de Sprites.* sprite : Integer -- *El identificador asociado al Sprite.*
Expresiones de precondición	**pre1**: El Sprite está contenido en el Conjunto de Sprites.
Expresiones de precondición (OCL)	**pre1**: true
Expresiones de postcondición	Ninguno
Expresiones de postcondición (OCL)	true
Comentarios	Ninguno

Excepción	**Obtener Sprite::no encontrado**
Condición	El identificador no está asociado a ningún Sprite del Conjunto de Sprites indicado, o el Conjunto de Sprites indicado no ha sido encontrado. Devuelve FRACASO.
Condición (OCL)	false
Expresión	La biblioteca registra un Error con un mensaje explicativo y devuelve FRACASO.
Expresión OCL	true
Comentarios	Ninguno

SOP-0019	**Obtener información de un Sprite**
Tipo del resultado	Entero o un conjunto de tipos Punto, dependiendo de la petición que se haya hecho.
Versión	1.0 (09/11/2010)
Autores	• José Manuel Barroso Galindo

Dependencias	• [UC-0101] Obtener información de un Sprite
Descripción	La aplicación solicita de un Sprite indicando qué tipo de información necesita.
Parámetros	spriteset : Integer -- *El identificador asociado al Conjunto de Sprites.* sprite : Integer -- *El identificador asociado al Sprite.*
Expresiones de precondición	**pre1**: El Sprite está contenido en el Conjunto de Sprites.
Expresiones de precondición (OCL)	**pre1**: true
Expresiones de postcondición	Ninguno
Expresiones de postcondición (OCL)	true
Comentarios	El tipo de información requerida se especifica eligiendo entre los diferentes tipos de métodos a invocar, y aportando si fuera necesario parámetros adicionales para ajustarse a ellos.

Excepción	**Obtener información de un Sprite::no encontrado**
Condición	El identificador no está asociado a ningún Sprite del Conjunto de Sprites indicado, o el Conjunto de Sprites indicado no ha sido encontrado. Devuelve FRACASO.
Condición (OCL)	false
Expresión	La biblioteca registra un Error con un mensaje explicativo y devuelve FRACASO.
Expresión OCL	true
Comentarios	Ninguno

Excepción	**Obtener información de un Sprite::petición de información no válida**
Condición	No se ha pedido una información posible. Devuelve FRACASO.
Condición (OCL)	false
Expresión	La biblioteca registra un Error con un mensaje explicativo y devuelve FRACASO.
Expresión OCL	true

Comentarios	Ninguno

SOP-0020	**Cambiar información de un Sprite**
Tipo del resultado	Éxito, si la información se ha cambiado según lo especificado.
Versión	1.0 (10/11/2010)
Autores	• José Manuel Barroso Galindo
Dependencias	• [UC-0102] Cambiar información de un Sprite
Descripción	La aplicación solicita cambiar la información de un sprite aportando el valor Punto o Integer correspondiente de acuerdo a la naturaleza del cambio solicitado.
Parámetros	spriteset : Integer -- *El identificador asociado al Conjunto de Sprites.* sprite : Integer -- *El identificador asociado al Sprite.*
Expresiones de precondición	**pre1**: El Sprite está contenido en el Conjunto de Sprites.
Expresiones de precondición (OCL)	**pre1**: true
Expresiones de postcondición	**post1**: La información queda cambiada con el nuevo dato.
Expresiones de postcondición (OCL)	**post1**: true
Comentarios	El tipo de información a modificar se especifica eligiendo entre los diferentes tipos de métodos a invocar, y aportando si el parámetro del tipo correspondiente.

Excepción	**Cambiar información de un Sprite::no encontrado**
Condición	El identificador no está asociado a ningún Sprite del Conjunto de Sprites indicado, o el Conjunto de Sprites indicado no ha sido encontrado. Devuelve FRACASO.
Condición (OCL)	false
Expresión	La biblioteca registra un Error con un mensaje explicativo y devuelve FRACASO.
Expresión OCL	true
Comentarios	Ninguno

Excepción	Cambiar información de un Sprite::cambio no válido
Condición	El nuevo dato no puede registrarse correctamente. Devuelve FRACASO.
Condición (OCL)	false
Expresión	La biblioteca registra un Error con un mensaje explicativo y devuelve FRACASO.
Expresión OCL	true
Comentarios	Ninguno

SOP-0021	Aplicar Efecto sobre Sprite
Tipo del resultado	Éxito, si el efecto se ha registrado para el Sprite.
Versión	1.0 (10/11/2010)
Autores	• José Manuel Barroso Galindo
Dependencias	• [UC-0100] Aplicar efecto sobre Sprite
Descripción	La aplicación obtiene un Sprite de un Conjunto de Sprites y a continuación solicita aplicar un efecto en el Sprite indicado de cara a la siguiente fase de renderización. Para ello indicará el tipo y configuración del efecto.
Parámetros	spriteset : Integer -- *El identificador asociado al Conjunto de Sprites.* sprite : Integer -- *El identificador asociado al Sprite.* efecto : Efecto Gráfico -- *El efecto al que será sometido el Sprite.*
Expresiones de precondición	**pre1**: El contexto de vídeo debe estar inicializado.
Expresiones de precondición (OCL)	**pre1**: true
Expresiones de postcondición	**post1**: El efecto ha quedado registrado para el Sprite.
Expresiones de postcondición (OCL)	**post1**: true
Comentarios	Ninguno

Excepción	Aplicar Efecto sobre Sprite::contexto de vídeo no inicializado
Condición	La pre1 no se cumple, por tanto se devuelve FRACASO.
Condición	false

(OCL)	
Expresión	La biblioteca registra un Error con un mensaje explicativo y devuelve FRACASO.
Expresión OCL	true
Comentarios	Ninguno

Excepción	**Aplicar Efecto sobre Sprite::efecto no procedente**
Condición	Si el efecto gráfico no es aplicable, devuelve FRACASO
Condición (OCL)	false
Expresión	La biblioteca registra un Error con un mensaje explicativo y devuelve FRACASO.
Expresión OCL	true
Comentarios	Ninguno

SOP-0022	**Renderizar Sprite en coordenadas de la Pantalla**
Tipo del resultado	Éxito, si el Sprite ha sido registrado para ser renderizado en la siguiente fase de renderización.
Versión	1.0 (10/11/2010)
Autores	• José Manuel Barroso Galindo
Dependencias	• [UC-0103] Renderizar un Sprite en coordenadas de la Pantalla
Descripción	La aplicación solicita que el Sprite dado sea renderizado en la siguiente fase de renderización en las coordenadas dadas del marco actual del contexto gráfico.
Parámetros	ptDst : Punto -- *Punto destino donde habrá de renderizarse el elemento.* flags : Integer -- *El valor de espejado que puede tomar el elemento.*
Expresiones de precondición	**pre1**: El contexto de vídeo debe estar inicializado.
Expresiones de precondición (OCL)	**pre1**: true
Expresiones de postcondición	Ninguno
Expresiones de postcondición (OCL)	true

Comentarios	Valores posibles para 'flags' : 0 -> No espejado. 1-> Espejado horizontal. 2-> Espejado vertical. 4-> Espejado horizontal y vertical.

Excepción	**Renderizar Sprite en coordenadas de la Pantalla::fallo en renderización**
Condición	Si ha habido algún error inesperado, devuelve FRACASO.
Condición (OCL)	false
Expresión	La biblioteca registra un Error con un mensaje explicativo y devuelve FRACASO.
Expresión OCL	true
Comentarios	Ninguno

Excepción	**Renderizar Sprite en coordenadas de la Pantalla::contexto de vídeo no inicializado**
Condición	La pre1 no se cumple, por tanto se devuelve FRACASO.
Condición (OCL)	false
Expresión	La biblioteca registra un Error con un mensaje explicativo y devuelve FRACASO.
Expresión OCL	true
Comentarios	Ninguno

4.2.4. Gestión de Textos

SOP-0023	Establecer Material predeterminado de Fuente
Tipo del resultado	Integer del valor anterior de material.
Versión	1.0 (10/11/2010)
Autores	• José Manuel Barroso Galindo
Dependencias	• [UC-0024] Cargar Fuente
Descripción	La aplicación solicita especificar un material para las próximas Fuentes a cargar. Con material englobamos características gráficas tales como tamaño, color, etc...
Parámetros	Ninguno
Expresiones de precondición	Ninguno
Expresiones de precondición (OCL)	true
Expresiones de postcondición	**post1**: El valor de material pasa a ser el especificado.
Expresiones de postcondición (OCL)	**post1**: true
Comentarios	El tipo de material se especifica eligiendo entre los diferentes tipos de métodos a invocar, y aportando si el parámetro del tipo correspondiente.

Excepción	Establecer Material predeterminado de Fuente::material no válido
Condición	Si el material especificado no es válido, devuelve FRACASO.
Condición (OCL)	false
Expresión	La biblioteca registra un Error con un mensaje explicativo y devuelve FRACASO.
Expresión OCL	true
Comentarios	Ninguno

SOP-0024	Cargar Fuente
Tipo del resultado	Integer identificador de la fuente correspondiente con el descriptor dado y el material establecido.

Versión	1.0 (10/11/2010)
Autores	• José Manuel Barroso Galindo
Dependencias	• [UC-0024] Cargar Fuente
Descripción	La aplicación solicita cargar una Fuente indicando su descriptor.
Parámetros	fuente : String -- *El descriptor de la fuente.*
Expresiones de precondición	**pre1**: El contexto de vídeo debe estar inicializado.
Expresiones de precondición (OCL)	**pre1**: true
Expresiones de postcondición	**post1**: La fuente ha quedado almacenada.
Expresiones de postcondición (OCL)	**post1**: true
Comentarios	Ninguno

Excepción	**Cargar Fuente::descriptor no válido**
Condición	Si no se ha podido referenciar una fuente con dicho nombre porque no es válido, la biblioteca devuelve FRACASO.
Condición (OCL)	false
Expresión	La biblioteca registra un Error con un mensaje explicativo y devuelve FRACASO.
Expresión OCL	true
Comentarios	Ninguno

SOP-0025	**Buscar Fuente**
Tipo del resultado	Integer identificador de la fuente solicitada.
Versión	1.0 (10/11/2010)
Autores	• José Manuel Barroso Galindo
Dependencias	• [UC-0023] Buscar Fuente
Descripción	La aplicación solicita el identificador de una fuente a través de su descriptor, el identificador de un texto que la use, u otros atributes que

	tengan una correspondencia única con la fuente.
Parámetros	fuente : String -- *El descriptor de la fuente a buscar.* idtext : Integer -- *El identificador de un texto que usa la fuente buscada.* fnt : Fuente -- *El puntero a la estructura propia de la fuente.*
Expresiones de precondición	**pre1**: La fuente está almacenada.
Expresiones de precondición (OCL)	**pre1**: true
Expresiones de postcondición	Ninguno
Expresiones de postcondición (OCL)	true
Comentarios	Ninguno

Excepción	**Buscar Fuente::fuente no encontrada**
Condición	Si la fuente con el identificador dado no ha sido encontrada, devuelve FRACASO.
Condición (OCL)	false
Expresión	La biblioteca registra un Error con un mensaje explicativo y devuelve FRACASO.
Expresión OCL	true
Comentarios	Ninguno

SOP-0026	**Eliminar Fuente**
Tipo del resultado	Éxito, si la biblioteca ha gestionado el cese de uso de la fuente correctamente.
Versión	1.0 (10/11/2010)
Autores	• José Manuel Barroso Galindo
Dependencias	• [UC-0025] Eliminar Fuente
Descripción	La aplicación indica que cesará el uso de la fuente asociada al identificador dado.
Parámetros	fuente : Integer -- *El identificador de la fuente.*
Expresiones de precondición	**pre1**: La fuente está almacenada.

Expresiones de precondición (OCL)	**pre1**: true
Expresiones de postcondición	**post1**: Se ha registrado el cese de uso de dicha Fuente y posiblemente haya sido eliminada.
Expresiones de postcondición (OCL)	**post1**: true
Comentarios	Ninguno

Excepción	**Eliminar Fuente::fuente no encontrada**
Condición	Si la fuente con el identificador dado no ha sido encontrada, devuelve FRACASO.
Condición (OCL)	false
Expresión	La biblioteca registra un Error con un mensaje explicativo y devuelve FRACASO.
Expresión OCL	true
Comentarios	Ninguno

SOP-0027	**Cargar Texto**
Tipo del resultado	Identificador del texto solicitado.
Versión	1.0 (10/11/2010)
Autores	• José Manuel Barroso Galindo
Dependencias	• [UC-0026] Cargar Texto
Descripción	La aplicación solicita crear un texto indicando su contenido y la fuente a usar através del identificador de fuente.
Parámetros	fuente : Integer -- *El identificador de la fuente.* x : Integer -- *El valor del eje X donde deberá comenzar a posicionarse el texto.* y : Integer -- *El valor del eje Y donde deberá comenzar a posicionarse el texto.* text : String -- *La cadena de texto que se quiere representar.*
Expresiones de precondición	**pre1**: Hay alguna fuente cargada.
Expresiones de precondición	**pre1**: true

(OCL)	
Expresiones de postcondición	**post1**: Se ha creado un nuevo texto.
Expresiones de postcondición (OCL)	**post1**: true
Comentarios	Además, se pueden añadir un número ilimitado de parámetros para representar el valor de variables como con la función de c 'printf'.

Excepción	**Cargar Texto::fuente no encontrada**
Condición	Si la fuente con el identificador dado no ha sido encontrada, devuelve FRACASO.
Condición (OCL)	false
Expresión	La biblioteca registra un Error con un mensaje explicativo y devuelve FRACASO.
Expresión OCL	true
Comentarios	Ninguno

Excepción	**Cargar Texto::cadena conflictiva**
Condición	Si la cadena de texto no contiene ningún carácter, sólo contiene carácteres sin representación gráfica o tiene algún carácter conflictivo, la biblioteca devuelve FRACASO.
Condición (OCL)	false
Expresión	La biblioteca registra un Error con un mensaje explicativo y devuelve FRACASO.
Expresión OCL	true
Comentarios	Ninguno

SOP-0028	**Eliminar Texto**
Tipo del resultado	Éxito, si el/los texto/s ha sido eliminado.
Versión	1.0 (10/11/2010)
Autores	• José Manuel Barroso Galindo
Dependencias	• [UC-0027] Eliminar Texto

Descripción	La aplicación solicita eliminar un texto asociado al identificador dado.
Parámetros	text : Integer -- *El identificador del texto.* nextframe : Boolean -- *Un booleano que especifica si está a cierto, que el texto se borrará al comienzo de la próxima etapa de ejecución. Si está a falso, el texto se borrará instantáneamente.*
Expresiones de precondición	**pre1**: El texto con el identificador dado existe.
Expresiones de precondición (OCL)	**pre1**: true
Expresiones de postcondición	**post1**: El texto con el identificador dado ha sido eliminado.
Expresiones de postcondición (OCL)	**post1**: true
Comentarios	Ninguno

Excepción	**Eliminar Texto::texto no encontrado**
Condición	El identificador dado no estaba asociado a ningún texto, por tanto devuelve FRACASO,
Condición (OCL)	false
Expresión	La biblioteca registra un Error con un mensaje explicativo y devuelve FRACASO.
Expresión OCL	true
Comentarios	Ninguno

SOP-0029	**Establecer Criterio de Visualización de Textos**
Tipo del resultado	Éxito, si la petición ha sido llevada a cabo con normalidad.
Versión	1.0 (11/11/2010)
Autores	• José Manuel Barroso Galindo
Dependencias	• [UC-0030] Establecer criterio de visualización de Textos
Descripción	La aplicación solicita que los textos especificados mediante su identificador sólo sean visibles en una determinada zona de la pantalla : La aplicación establece un área de la pantalla y el criterio de visualización deseado.

	La biblioteca almacena el área especificada junto al criterio y los selecciona para su uso a continuación. La aplicación indica los identificadores de texto sobre los que tendrá efecto el criterio de visualización. La biblioteca mostrará los textos con las restricciones visuales requeridas en la siguiente fase de renderización del texto
Parámetros	posx : Real -- *El valor del eje X de la posición superior-izquierda del área que se está definiendo.* posy : Real -- *El valor del eje Y de la posición superior-izquierda del área que se está definiendo.* width : Real -- *El valor de ancho que ocupará el área a lo largo del eje X a partir de posx.* height : Real -- *El valor de alto que ocupará el área a lo largo del eje Y a partir de posy.* texto : Integer -- *El identificador del texto.*
Expresiones de precondición	**pre1**: Hay textos registrados en la biblioteca
Expresiones de precondición (OCL)	**pre1**: true
Expresiones de postcondición	**post1**: Se han asociado los Textos indicados a la nueva Área, que ahora está almacenada.
Expresiones de postcondición (OCL)	**post1**: true
Comentarios	Ninguno

Excepción	**Establecer Criterio de Visualización de Textos::area inválida**
Condición	El area definida no es procesable, por tanto devuelve FRACASO,
Condición (OCL)	false
Expresión	La biblioteca registra un Error con un mensaje explicativo y devuelve FRACASO.
Expresión OCL	true
Comentarios	Ninguno

Excepción	**Establecer Criterio de Visualización de Textos::texto no encontrado**
Condición	El identificador dado no estaba asociado a ningún texto, por tanto

	devuelve FRACASO,
Condición (OCL)	false
Expresión	La biblioteca registra un Error con un mensaje explicativo y devuelve FRACASO.
Expresión OCL	true
Comentarios	Ninguno

SOP-0030	**Cambiar Atributos asociados al Texto**
Tipo del resultado	Éxito, si los cambios se han llevado a cabo.
Versión	1.0 (11/11/2010)
Autores	• José Manuel Barroso Galindo
Dependencias	• [UC-0031] Cambiar Atributos asociados al Texto
Descripción	La aplicación solicita realizar un cambio de algún atributo en el texto con el identificador dado.
Parámetros	texto : Integer -- *El identificador del texto.*
Expresiones de precondición	**pre1**: Existe el texto asociado a tal identificador.
Expresiones de precondición (OCL)	**pre1**: true
Expresiones de postcondición	**post1**: Los atributos del texto han sido cambiados correctamente.
Expresiones de postcondición (OCL)	**post1**: true
Comentarios	El tipo de información a modificar se especifica eligiendo entre los diferentes tipos de métodos a invocar, y aportando si el parámetro del tipo correspondiente.

Excepción	**Cambiar Atributos asociados al Texto::texto no encontrado**
Condición	El identificador dado no estaba asociado a ningún texto, por tanto devuelve FRACASO,
Condición (OCL)	false
Expresión	La biblioteca registra un Error con un mensaje explicativo y devuelve

	FRACASO.
Expresión OCL	true
Comentarios	Ninguno

Excepción	**Cambiar Atributos asociados al Texto::atributo inválido**
Condición	El nuevo valor dado para el atributo especifcado es incompatible, por tanto devuelve FRACASO.
Condición (OCL)	false
Expresión	La biblioteca registra un Error con un mensaje explicativo y devuelve FRACASO.
Expresión OCL	true
Comentarios	Ninguno

SOP-0031	**Fase de Renderización de Textos**
Tipo del resultado	Éxito, si la fase de renderización ha sido llevad a cabo con normalidad.
Versión	1.0 (11/11/2010)
Autores	• José Manuel Barroso Galindo
Dependencias	• [UC-0029] Fase de Renderización de Textos
Descripción	La aplicación solicita que los textos preparados para ello, sean renderizados.
Parámetros	Ninguno
Expresiones de precondición	**pre1**: El contexto de vídeo debe estar inicializado.
Expresiones de precondición (OCL)	**pre1**: true
Expresiones de postcondición	Ninguno
Expresiones de postcondición (OCL)	true
Comentarios	Si hay algún texto que no ha sido invocado durante el procesamiento de ningún Motor, el sistema los dibuja del mismo modo, pero quedando por encima de los carácteres anteriores.

Excepción	Fase de Renderización de Textos::contexto de vídeo no inicializado
Condición	La pre1 no se cumple, por tanto se devuelve FRACASO.
Condición (OCL)	false
Expresión	La biblioteca registra un Error con un mensaje explicativo y devuelve FRACASO.
Expresión OCL	true
Comentarios	Ninguno

SOP-0032	Aplicar Efecto sobre Texto
Tipo del resultado	Éxito, si el efecto ha quedado registrado para el texto indicado.
Versión	1.0 (11/11/2010)
Autores	• José Manuel Barroso Galindo
Dependencias	• [UC-0096] Aplicar efecto sobre Texto
Descripción	La aplicación solicita aplicar un efecto determinado a el texto dado de cara a la siguiente fase de renderización de textos.
Parámetros	texto : Integer -- *El identificador del texto.* efecto : Efecto Gráfico -- *El efecto al que será sometido el texto.*
Expresiones de precondición	**pre1**: Existe un texto asociado al identificador dado.
Expresiones de precondición (OCL)	**pre1**: true
Expresiones de postcondición	**post1**: Ha quedado registrado un efecto para el texto indicado.
Expresiones de postcondición (OCL)	**post1**: true
Comentarios	Ninguno

Excepción	Aplicar Efecto sobre Texto::texto no encontrado
Condición	El identificador dado no estaba asociado a ningún texto, por tanto devuelve FRACASO,
Condición (OCL)	false

Expresión	La biblioteca registra un Error con un mensaje explicativo y devuelve FRACASO.
Expresión OCL	true
Comentarios	Ninguno

Excepción	**Aplicar Efecto sobre Texto::efecto no procedente**
Condición	Si el efecto gráfico no es aplicable, devuelve FRACASO
Condición (OCL)	false
Expresión	La biblioteca registra un Error con un mensaje explicativo y devuelve FRACASO.
Expresión OCL	true
Comentarios	Ninguno

4.2.5. Gestión del Multiverso

SOP-0033	**Cargar Universo**
Tipo del resultado	Instancia del Universo.
Versión	1.0 (11/11/2010)
Autores	• José Manuel Barroso Galindo
Dependencias	• [UC-0035] Cargar Universo
Descripción	La aplicación solicita la creación de un Universo y su almacenaje en el Multiverso.
Parámetros	uni : Universo -- *El puntero de la instancia del Universo que desea registrarse.* slot : Integer -- *Si es deseable que varios universos con el mismo descriptor se carguen simultáneamente, se debe especificar un valor máximo de repetición con este parámetro.*
Expresiones de precondición	**pre1**: Hay algún motor activo
Expresiones de precondición (OCL)	**pre1**: true

Expresiones de postcondición	**post1**: El Universo correspondiente está almacenado en la colección de Universos correspondiente al Motor activo
Expresiones de postcondición (OCL)	**post1**: true
Comentarios	El Multiverso es el gestor de Universos.

Excepción	**Cargar Universo::instancia de universo inválida**
Condición	Si la instancia del universo que se ha pasado es nula o errónea, la biblioteca devuelve FRACASO
Condición (OCL)	false
Expresión	La biblioteca registra un Error con un mensaje explicativo y devuelve FRACASO.
Expresión OCL	true
Comentarios	Ninguno

Excepción	**Cargar Universo::descriptor fallido**
Condición	Si el descriptor no se procesa correctamente, se devuelve FRACASO.
Condición (OCL)	false
Expresión	La biblioteca registra un Error con un mensaje explicativo y devuelve FRACASO.
Expresión OCL	true
Comentarios	Ninguno

SOP-0034	**Buscar Universo**
Tipo del resultado	Puntero a la instancia del Unviverso con el descriptor dado.
Versión	1.0 (11/11/2010)
Autores	• José Manuel Barroso Galindo
Dependencias	• [UC-0036] Buscar Universo
Descripción	La aplicación solicita la instancia del Universo que corresponda a un descriptor dado.
Parámetros	uniName : String -- *El descriptor del Universo,*

Expresiones de precondición	**pre1**: El universo está contenido en la colección de universos del Multiverso asociada al Motor Activo.
Expresiones de precondición (OCL)	**pre1**: true
Expresiones de postcondición	Ninguno
Expresiones de postcondición (OCL)	true
Comentarios	Ninguno

Excepción	**Buscar Universo::universo no encontrado**
Condición	Si no encuentra ningún universo como resultado de la búsqueda, la biblioteca devuelve FRACASO
Condición (OCL)	false
Expresión	La biblioteca registra un Error con un mensaje explicativo y devuelve FRACASO.
Expresión OCL	true
Comentarios	Ninguno

SOP-0035	**Eliminar Universo**
Tipo del resultado	Éxito, si el Universo se ha eliminado correctamente.
Versión	1.0 (11/11/2010)
Autores	• José Manuel Barroso Galindo
Dependencias	• [UC-0037] Eliminar Universo
Descripción	La aplicación solicita la eliminar un Universo aportando su instancia.
Parámetros	uniKilled : Universo -- *El puntero de la instancia del Universo que desea eliminarse.*
Expresiones de precondición	**pre1**: El universo está correctamente almacenado en el Multiverso.
Expresiones de precondición (OCL)	**pre1**: true
Expresiones de	**post1**: El universo indicado ha sido eliminado.

postcondición	
Expresiones de postcondición (OCL)	**post1**: true
Comentarios	Ninguno

Excepción	**Eliminar Universo::universo no encontrado**
Condición	Si no encuentra ningún universo como resultado de la búsqueda, la biblioteca devuelve FRACASO
Condición (OCL)	false
Expresión	La biblioteca registra un Error con un mensaje explicativo y devuelve FRACASO.
Expresión OCL	true
Comentarios	Ninguno

SOP-0036	**Acceder a la Colección de Universos del Motor Activo**
Tipo del resultado	Puntero, acceso mediante iteradores a la colección de Universos del motor activo.
Versión	1.0 (11/11/2010)
Autores	• José Manuel Barroso Galindo
Dependencias	• [UC-0041] Acceder a la Colección de Universos del Motor activo
Descripción	La aplicación solicita acceder a la Colección de Universos del motor activo.
Parámetros	Ninguno
Expresiones de precondición	**pre1**: Hay un motor activo.
Expresiones de precondición (OCL)	**pre1**: true
Expresiones de postcondición	Ninguno
Expresiones de postcondición (OCL)	true
Comentarios	Ninguno

4.2.6. Gestión del Motor

SOP-0037	Iniciar Motor
Tipo del resultado	Éxito, si el motor se ha iniciado correctamente.
Versión	1.0 (11/11/2010)
Autores	• José Manuel Barroso Galindo
Dependencias	• [UC-0047] Iniciar Motor
Descripción	La aplicación solicita que el motor dado se inicie, para ello se invocaran a las rutinas que tiene asociadas a su fase de inicialización.
Parámetros	Ninguno
Expresiones de precondición	**pre1**: El motor no estaba inicializado.
Expresiones de precondición (OCL)	**pre1**: true
Expresiones de postcondición	**post1**: El motor ahora está inicializado
Expresiones de postcondición (OCL)	**post1**: true
Comentarios	Ninguno

Excepción	Iniciar Motor::inicialización fallida
Condición	Ha ocurrido algún suceso inesperado en las rutinas de inicialización y por tando se devuelve FRACASO.
Condición (OCL)	false
Expresión	La biblioteca registra un Error con un mensaje explicativo y devuelve FRACASO.
Expresión OCL	true
Comentarios	Ninguno

Excepción	Iniciar Motor::previamente inicializado
Condición	El motor ya se encontraba previamente inicializado, por tanto devuelve FRACASO.

Condición (OCL)	false
Expresión	La biblioteca registra un Error con un mensaje explicativo y devuelve FRACASO.
Expresión OCL	true
Comentarios	Ninguno

SOP-0038	**Terminar Motor**
Tipo del resultado	Éxito, si el motor se ha finalizado correctamente.
Versión	1.0 (11/11/2010)
Autores	• José Manuel Barroso Galindo
Dependencias	• [UC-0048] Terminar Motor
Descripción	La aplicación solicita que el motor dado se termine, para ello se invocaran a las rutinas que tiene asociadas a su fase de finalización.
Parámetros	Ninguno
Expresiones de precondición	**pre1**: El motor estaba inicializado.
Expresiones de precondición (OCL)	**pre1**: true
Expresiones de postcondición	**post1**: El motor ya no está inicializado.
Expresiones de postcondición (OCL)	**post1**: true
Comentarios	Ninguno

Excepción	**Terminar Motor::finalización fallida**
Condición	Ha ocurrido algún suceso inesperado en las rutinas de finalización y por tando se devuelve FRACASO.
Condición (OCL)	false
Expresión	La biblioteca registra un Error con un mensaje explicativo y devuelve FRACASO.
Expresión OCL	true

Comentarios	Ninguno

Excepción	**Terminar Motor::no inicializado**
Condición	El motor no se encontraba previamente inicializado, por tanto devuelve FRACASO.
Condición (OCL)	false
Expresión	La biblioteca registra un Error con un mensaje explicativo y devuelve FRACASO.
Expresión OCL	true
Comentarios	Ninguno

SOP-0039	**Establecer Manejador de Eventos**
Tipo del resultado	Puntero a función del tipo Manejador de Eventos asociado anteriormente al tipo de Evento que se haya especificado. Si no hay, se devuelve un valor nulo.
Versión	1.0 (12/11/2010)
Autores	• José Manuel Barroso Galindo
Dependencias	• [UC-0050] Establecer Manejador de Eventos
Descripción	La aplicación solicita que a un tipo concreto de Eventos se le asocie el Manejador de Eventos dado.
Parámetros	type : Evento -- *El código de evento según SDL de los eventos que deberán ser tratados por el Manejador de Eventos dado.* eventHandler : Manejador de Eventos -- *El puntero a la función manejadora de eventos.*
Expresiones de precondición	Ninguno
Expresiones de precondición (OCL)	true
Expresiones de postcondición	**post1**: Se ha establecido un Manejador de Eventos.
Expresiones de postcondición (OCL)	**post1**: true
Comentarios	Ninguno

Excepción	Establecer Manejador de Eventos::manejador no válido
Condición	Si el puntero no es a una función de tipo Manejador de Eventos o el tipo de Evento no es válido, se devuelve FRACASO.
Condición (OCL)	false
Expresión	La biblioteca registra un Error con un mensaje explicativo y devuelve FRACASO.
Expresión OCL	true
Comentarios	Ninguno

SOP-0040	Procesar Eventos
Tipo del resultado	Éxito, si se han procesado los eventos pendientes con normalidad.
Versión	1.0 (12/11/2010)
Autores	• José Manuel Barroso Galindo
Dependencias	• [UC-0051] Procesar Eventos
Descripción	La aplicación solicita que se traten los eventos que han entrado en el sistema.
Parámetros	Ninguno
Expresiones de precondición	**pre1**: La biblioteca ha sido inicializada. **pre2**: El motor estaba inicializado. **pre3**: El motor activo es éste, o no hay actualmente.
Expresiones de precondición (OCL)	**pre1**: true **pre2**: true **pre3**: true
Expresiones de postcondición	Ninguno
Expresiones de postcondición (OCL)	true
Comentarios	Ninguno

Excepción	Procesar Eventos::biblioteca no inicializada
Condición	Si la biblioteca no está inicializada, devuelve FRACASO.
Condición (OCL)	false
Expresión	La biblioteca registra un Error con un mensaje explicativo y devuelve

	FRACASO.
Expresión OCL	true
Comentarios	Ninguno

Excepción	**Procesar Eventos::motor no inicializado**
Condición	Si la biblioteca no está inicializada, devuelve FRACASO.
Condición (OCL)	false
Expresión	La biblioteca registra un Error con un mensaje explicativo y devuelve FRACASO.
Expresión OCL	true
Comentarios	Ninguno

Excepción	**Procesar Eventos::otro motor activo**
Condición	Si la biblioteca comunica que hay otro motor activo, se devuelve FRACASO.
Condición (OCL)	false
Expresión	La biblioteca registra un Error con un mensaje explicativo y devuelve FRACASO.
Expresión OCL	true
Comentarios	Ninguno

SOP-0041	**Fase de Renderización del Motor**
Tipo del resultado	Éxito, si se han ejecutado con normalidad la fase de renderización del motor
Versión	1.0 (12/11/2010)
Autores	• José Manuel Barroso Galindo
Dependencias	• [UC-0054] Fase de Renderización del Motor
Descripción	La aplicación solicita que se ejecute la fase de renderización del motor
Parámetros	Ninguno
Expresiones de precondición	**pre1**: El motor ha sido inicializado.
Expresiones de	**pre1**: true

precondición (OCL)	
Expresiones de postcondición	Ninguno
Expresiones de postcondición (OCL)	true
Comentarios	Ninguno

Excepción	**Fase de Renderización del Motor::motor no inicializado**
Condición	Si la biblioteca no está inicializada, devuelve FRACASO.
Condición (OCL)	false
Expresión	La biblioteca registra un Error con un mensaje explicativo y devuelve FRACASO.
Expresión OCL	true
Comentarios	Ninguno

4.2.7. Gestión de Mensajería

SOP-0042	**Mandar Mensaje**
Tipo del resultado	Éxito, si el mensaje se ha mandado con normalidad.
Versión	1.0 (12/11/2010)
Autores	• José Manuel Barroso Galindo
Dependencias	• [UC-0055] Mandar Mensaje
Descripción	La aplicación solicita que se atienda un mensaje instantáneamente o en un futuro.
Parámetros	MsgID : Integer -- *El tipo de mensaje que se está mandando.* Parm : Integer -- *Parámetro a gestionar por el mensaje. Puede ser un conjunto de parámetros.*
Expresiones de precondición	**pre1**: El destinatario debe heredar de una clase Mensajera.
Expresiones de precondición (OCL)	**pre1**: true

Expresiones de postcondición	Ninguno
Expresiones de postcondición (OCL)	true
Comentarios	Ninguno

Excepción	**Mandar Mensaje::clase no mensajera**
Condición	El destinatario no hereda de una clase mensajera.
Condición (OCL)	false
Expresión	La biblioteca registra un Error con un mensaje explicativo y devuelve FRACASO.
Expresión OCL	true
Comentarios	Ninguno

SOP-0043	**Borrar Mensajes Pendientes**
Tipo del resultado	Éxito, si los monsajes han sido correctamente eliminados.
Versión	1.0 (12/11/2010)
Autores	• José Manuel Barroso Galindo
Dependencias	• [UC-0055] Mandar Mensaje
Descripción	La aplicación solicita que no se atiendan los mensajes pendientes.
Parámetros	Ninguno
Expresiones de precondición	Ninguno
Expresiones de precondición (OCL)	true
Expresiones de postcondición	**post1**: Ya no hay ningún mensaje pendiente.
Expresiones de postcondición (OCL)	**post1**: true
Comentarios	Ninguno

4.2.8. Gestión del Ritmo de Ejecución

SOP-0044	Siguiente Etapa de Ejecución
Tipo del resultado	Éxito, si no hay incidentes.
Versión	1.0 (12/11/2010)
Autores	• José Manuel Barroso Galindo
Dependencias	• [UC-0064] Siguiente Etapa de ejecución
Descripción	La aplicación solicita que se de por finalizada la actual etapa de ejecución para iniciar una nueva. La biblioteca llama a la operación de actualizar la pantalla y espera al requisito temporal que se le haya especificado.
Parámetros	Ninguno
Expresiones de precondición	**pre1**: La biblioteca ha sido inicializada.
Expresiones de precondición (OCL)	**pre1**: true
Expresiones de postcondición	**post1**: Se ha preparado una nueva etapa de ejecución.
Expresiones de postcondición (OCL)	**post1**: true
Comentarios	Ninguno

Excepción	Siguiente Etapa de Ejecución::fallo de operación
Condición	La operación de actualizar la pantalla devolvió fracaso, por tanto se devuelve FRACASO.
Condición (OCL)	false
Expresión	La biblioteca registra un Error con un mensaje explicativo y devuelve FRACASO.
Expresión OCL	true
Comentarios	Ninguno

Excepción	Siguiente Etapa de Ejecución::biblioteca no inicializada
Condición	Si la biblioteca no se ha inicializado, devuelve FRACASO.
Condición (OCL)	false
Expresión	La biblioteca registra un Error con un mensaje explicativo y devuelve FRACASO.
Expresión OCL	true
Comentarios	Ninguno

SOP-0045	Acceder al Contador de Etapas de Ejecución
Tipo del resultado	Integer con el valor de la etapa de ejecución actual para el motor activo.
Versión	1.0 (12/11/2010)
Autores	• José Manuel Barroso Galindo
Dependencias	• [UC-0065] Acceder al Contador de Etapas de ejecución
Descripción	La aplicación solicita conocer cuantas etapas de ejecución han pasado para el motor activo.
Parámetros	Ninguno
Expresiones de precondición	**pre1**: La biblioteca ha sido inicializada.
Expresiones de precondición (OCL)	**pre1**: true
Expresiones de postcondición	Ninguno
Expresiones de postcondición (OCL)	true
Comentarios	Ninguno

Excepción	Acceder al Contador de Etapas de Ejecución::biblioteca no inicializada
Condición	Si la biblioteca no se ha inicializado, devuelve FRACASO.
Condición (OCL)	false
Expresión	La biblioteca registra un Error con un mensaje explicativo y devuelve FRACASO.

Expresión OCL	true
Comentarios	Ninguno

SOP-0046	**Establecer ritmo de ejecución en unidad de tiempo**
Tipo del resultado	Integer, con el valor anterior de ritmo de ejecución.
Versión	1.0 (12/11/2010)
Autores	• José Manuel Barroso Galindo
Dependencias	• [UC-0066] Establecer ritmo de ejecución en unidad de tiempo
Descripción	La aplicación solicita definir un ritmo de ejecución específicado en milisegundos o frames por segundo.
Parámetros	fpsNew : Natural -- *El valor de FPS que se desea adoptar.* msNew : Natural -- *El valor de milisegundos que se desea adoptar.*
Expresiones de precondición	Ninguno
Expresiones de precondición (OCL)	true
Expresiones de postcondición	**post1**: Se ha cambiado el ritmo temporal de la ejecución.
Expresiones de postcondición (OCL)	**post1**: true
Comentarios	Se dice FPS o Frames Por Segundo a la frecuencia de actualizaciones de la pantalla en 1 segundo.

4.2.9. Gestión de la Cámara

SOP-0047	**Ubicar Cámara en Pantalla**
Tipo del resultado	Éxito, si la cámara ha quedado configurada correctamente.
Versión	1.0 (12/11/2010)
Autores	• José Manuel Barroso Galindo
Dependencias	• [UC-0078] Ubicar Cámara en la Pantallla
Descripción	La aplicación especifica un área de la pantalla donde debe mostrarse el

	resultado de la renderización de la cámara.
Parámetros	posx : Real -- *El valor del eje X de la posición superior-izquierda del área que se está definiendo.* posy : Real -- *El valor del eje Y de la posición superior-izquierda del área que se está definiendo.* width : Real -- *El valor de ancho que ocupará el área a lo largo del eje X a partir de posx.* height : Real -- *El valor de alto que ocupará el área a lo largo del eje Y a partir de posy.*
Expresiones de precondición	Ninguno
Expresiones de precondición (OCL)	true
Expresiones de postcondición	**post1**: Se ha especificado un área para la Cámara.
Expresiones de postcondición (OCL)	**post1**: true
Comentarios	Ninguno

SOP-0048	**Obtener Información de la Cámara**
Tipo del resultado	Valor del tipo correspondiente, con la información consultada.
Versión	1.0 (13/11/2010)
Autores	• José Manuel Barroso Galindo
Dependencias	• [UC-0073] Obtener Información de la Cámara
Descripción	La aplicación solicita conocer algún dato de la configuración de la cámara.
Parámetros	Ninguno
Expresiones de precondición	Ninguno
Expresiones de precondición (OCL)	true
Expresiones de postcondición	Ninguno
Expresiones de postcondición	true

(OCL)	
Comentarios	El tipo de información requerida se especifica eligiendo entre los diferentes tipos de métodos a invocar, y aportando si fuera necesario parámetros adicionales para ajustarse a ellos.

Excepción	**Obtener Información de la Cámara::información errónea**
Condición	Si la información que se ha pedido no se puede ofrecer, se devuelve FRACASO.
Condición (OCL)	false
Expresión	La biblioteca devuelve FRACASO.
Expresión OCL	true
Comentarios	Ninguno

SOP-0049	**Cambiar propiedades de la Cámara**
Tipo del resultado	Éxito, si el dato se ha cambiado correctamente.
Versión	1.0 (13/11/2010)
Autores	• José Manuel Barroso Galindo
Dependencias	• [UC-0088] Cambiar propiedades de la Cámara
Descripción	La aplicación indica que dato quiere cambiar y aporta un nuevo valor del tipo correspondiente.
Parámetros	Ninguno
Expresiones de precondición	Ninguno
Expresiones de precondición (OCL)	true
Expresiones de postcondición	**post1**: Las características de la cámara han cambiado.
Expresiones de postcondición (OCL)	**post1**: true
Comentarios	El tipo de información a modificar se especifica eligiendo entre los diferentes tipos de métodos a invocar, y aportando si el parámetro del tipo correspondiente.

Excepción	Cambiar propiedades de la Cámara::valor no válido
Condición	Si se ha especificado que un valor no válido para un atributo, la biblioteca devuelve FRACASO.
Condición (OCL)	false
Expresión	La biblioteca devuelve FRACASO.
Expresión OCL	true
Comentarios	Ninguno

SOP-0050	Aplicar Efectos sobre Cámara
Tipo del resultado	Éxito, si se ha registrado el efecto con normalidad.
Versión	1.0 (13/11/2010)
Autores	• José Manuel Barroso Galindo
Dependencias	• [UC-0094] Aplicar efecto sobre Cámara
Descripción	La aplicación solicita aplicar un efecto sobre la cámara de cara a la siguiente fase de renderización.
Parámetros	efecto : Efecto Gráfico -- *El efecto al que será sometido el Sprite.*
Expresiones de precondición	**pre1**: El contexto de vídeo debe estar inicializado.
Expresiones de precondición (OCL)	**pre1**: true
Expresiones de postcondición	**post1**: El efecto ha quedado registrado para la Cámara.
Expresiones de postcondición (OCL)	**post1**: true
Comentarios	Ninguno

Excepción	Aplicar Efectos sobre Cámara::contexto de vídeo no inicializado
Condición	La pre1 no se cumple.
Condición (OCL)	false
Expresión	La biblioteca devuelve FRACASO.
Expresión	true

OCL	
Comentarios	Ninguno

Excepción	**Aplicar Efectos sobre Cámara::efecto gráfico no procedente**
Condición	Si el efecto gráfico no es aplicable.
Condición (OCL)	false
Expresión	La biblioteca devuelve FRACASO.
Expresión OCL	true
Comentarios	Ninguno

SOP-0051	**Renderizar elementos enfocados por la Cámara**
Tipo del resultado	Éxito, si se ha realizado la fase de renderización con normalidad.
Versión	1.0 (13/11/2010)
Autores	• José Manuel Barroso Galindo
Dependencias	• [UC-0094] Aplicar efecto sobre Cámara
Descripción	La aplicación solicita que se ejecuten las rutinas correspondientes a la fase de renderización de la cámara.
Parámetros	Ninguno
Expresiones de precondición	**pre1**: El contexto de vídeo debe estar inicializado.
Expresiones de precondición (OCL)	**pre1**: true
Expresiones de postcondición	Ninguno
Expresiones de postcondición (OCL)	true
Comentarios	Ninguno

Excepción	**Renderizar elementos enfocados por la Cámara::contexto de vídeo no inicializado**
Condición	La pre1 no se cumple.
Condición	false

(OCL)	
Expresión	La biblioteca devuelve FRACASO.
Expresión OCL	true
Comentarios	Ninguno

Excepción	**Renderizar elementos enfocados por la Cámara::fallo en renderización**
Condición	Si ha habido algún error inesperado.
Condición (OCL)	false
Expresión	La biblioteca devuelve FRACASO.
Expresión OCL	true
Comentarios	Ninguno

SOP-0052	**Establecer Actor Objetivo**
Tipo del resultado	Éxito, si se ha establecido el actor dado como nuevo actor objetivo.
Versión	1.0 (13/11/2010)
Autores	• José Manuel Barroso Galindo
Dependencias	• [UC-0070] Establecer Actor Objetivo
Descripción	La aplicación solicita que el actor objetivo a partir de ahora sea el acto rdado.
Parámetros	newTarget : Actor -- *El nuevo actor objetivo.*
Expresiones de precondición	Ninguno
Expresiones de precondición (OCL)	true
Expresiones de postcondición	**post1**: El Actor Objetivo de la cámara ahora es el indicado.
Expresiones de postcondición (OCL)	**post1**: true
Comentarios	Ninguno

Excepción	Establecer Actor Objetivo::incompatibilidad
Condición	Si el actor no se puede sincronizar con la cámara.
Condición (OCL)	false
Expresión	La biblioteca devuelve FRACASO.
Expresión OCL	true
Comentarios	Ninguno

SOP-0053	Preparar Universo Objetivo
Tipo del resultado	Éxito, si el universo húesped del actor objetivo ahora está preparado.
Versión	1.0 (13/11/2010)
Autores	• José Manuel Barroso Galindo
Dependencias	• [UC-0071] Preparar Universo Objetivo
Descripción	La aplicación solicita que el universo huesped del actor objetivo cargue sus recursos necesarios para poder ser renderizado por una cámara.
Parámetros	Ninguno
Expresiones de precondición	**pre1**: El universo objetivo no está preparado.
Expresiones de precondición (OCL)	**pre1**: true
Expresiones de postcondición	**post1**: El universo objetivo ahora está preparado.
Expresiones de postcondición (OCL)	**post1**: true
Comentarios	Ninguno

Excepción	Preparar Universo Objetivo::universo ya preparado
Condición	El universo ya estaba preparado.
Condición (OCL)	false
Expresión	La biblioteca devuelve FRACASO.
Expresión OCL	true

Comentarios	Ninguno

SOP-0054	**Liberar Universo Objetivo**
Tipo del resultado	Éxito, si los recursos del Universo huesped del actor objetivo se han liberado sin problemas.
Versión	1.0 (13/11/2010)
Autores	• José Manuel Barroso Galindo
Dependencias	• [UC-0072] Liberar Universo Objetivo
Descripción	La aplicación solicita que el universo huesped del actor objetivo libere sus recursos gráficos ya que no van a ser utilizados en las siguientes fases de renderización de la cámara.
Parámetros	Ninguno
Expresiones de precondición	**pre1**: El universo objetivo está preparado.
Expresiones de precondición (OCL)	**pre1**: true
Expresiones de postcondición	**post1**: El universo objetivo ya no está preparado.
Expresiones de postcondición (OCL)	**post1**: true
Comentarios	Ninguno

Excepción	**Liberar Universo Objetivo::universo no preparado**
Condición	El universo no estaba preparado.
Condición (OCL)	false
Expresión	La biblioteca devuelve FRACASO.
Expresión OCL	true
Comentarios	Ninguno

4.2.10. Gestión del Actor

SOP-0055	Inicializar Actor
Tipo del resultado	Éxito, si las rutinas de inicialización se han ejecutado con normalidad.
Versión	1.0 (13/11/2010)
Autores	• José Manuel Barroso Galindo
Dependencias	• [UC-0084] Inicializar Actor
Descripción	La aplicación solicita preparar el actor, para poder ejecutar posteriormente su fase de movimiento.
Parámetros	Ninguno
Expresiones de precondición	**pre1**: El actor no estaba inicializado.
Expresiones de precondición (OCL)	**pre1**: true
Expresiones de postcondición	**post1**: El actor ahora está inicializado.
Expresiones de postcondición (OCL)	**post1**: true
Comentarios	Ninguno

Excepción	Inicializar Actor::actor ya inicializado
Condición	Si el actor se encontraba inicializado.
Condición (OCL)	false
Expresión	La biblioteca devuelve FRACASO.
Expresión OCL	true
Comentarios	Ninguno

SOP-0056	Procesar Movimiento del Actor
Tipo del resultado	Éxito, si la fase de movimiento del actor se ha ejecutado correctamente.
Versión	1.0 (13/11/2010)
Autores	• José Manuel Barroso Galindo

Dependencias	• [UC-0085] Procesar Movimiento del Actor
Descripción	La aplicación solicita que se ejecute la fase de movimiento del actor dado.
Parámetros	Ninguno
Expresiones de precondición	**pre1**: El actor está inicializado.
Expresiones de precondición (OCL)	**pre1**: true
Expresiones de postcondición	Ninguno
Expresiones de postcondición (OCL)	true
Comentarios	Ninguno

Excepción	**Procesar Movimiento del Actor::actor no inicializado**
Condición	Como el actor no ha sido inicializado.
Condición (OCL)	false
Expresión	La biblioteca devuelve FRACASO.
Expresión OCL	true
Comentarios	Ninguno

SOP-0057	**Obtener Información del Actor**
Tipo del resultado	Valor del tipo correspondiente al atributo que se ha consultado.
Versión	1.0 (13/11/2010)
Autores	• José Manuel Barroso Galindo
Dependencias	• [UC-0086] Obtener Información del Actor
Descripción	La aplicación solicita información sobre algún atributo del actor.
Parámetros	Ninguno
Expresiones de precondición	Ninguno

Expresiones de precondición (OCL)	true
Expresiones de postcondición	Ninguno
Expresiones de postcondición (OCL)	true
Comentarios	El tipo de información requerida se especifica eligiendo entre los diferentes tipos de métodos a invocar, y aportando si fuera necesario parámetros adicionales para ajustarse a ellos.

Excepción	**Obtener Información del Actor::información errónea**
Condición	Si la información que se ha pedido no se puede ofrecer.
Condición (OCL)	false
Expresión	La biblioteca devuelve FRACASO.
Expresión OCL	true
Comentarios	Ninguno

SOP-0058	**Cambiar Información del Actor**
Tipo del resultado	Éxito, si se ha modificado la información pertinente con el nuevo valor dado.
Versión	1.0 (13/11/2010)
Autores	• José Manuel Barroso Galindo
Dependencias	• [UC-0087] Cambiar Información del Actor
Descripción	La aplicación solicita cambiar la información sobre algún atributo del actor aportando el nuevo valor del tipo correspondiente.
Parámetros	Ninguno
Expresiones de precondición	Ninguno
Expresiones de precondición (OCL)	true
Expresiones de postcondición	**post1**: Las características de la cámara han cambiado.
Expresiones de	**post1**: true

postcondición (OCL)	
Comentarios	El tipo de información a modificar se especifica eligiendo entre los diferentes tipos de métodos a invocar, y aportando si el parámetro del tipo correspondiente.

Excepción	**Cambiar Información del Actor::valor no válido**
Condición	Si se ha especificado que un valor no válido para un atributo.
Condición (OCL)	false
Expresión	La biblioteca devuelve FRACASO.
Expresión OCL	true
Comentarios	Ninguno

4.2.11. Gestión del Universo

SOP-0061	**Añadir Actor al Universo**
Tipo del resultado	Éxito, si se añadido al Actor correctamente.
Versión	1.0 (13/11/2010)
Autores	• José Manuel Barroso Galindo
Dependencias	• [UC-0089] Añadir Actor al Universo
Descripción	La aplicación solicita añadir un Actor al Universo dado.
Parámetros	act : Actor -- *El actor que se desea añadir al Universo.*
Expresiones de precondición	**pre1**: El Actor no está almacenado en ningún Universo.
Expresiones de precondición (OCL)	**pre1**: true
Expresiones de postcondición	**post1**: El actor está almacenado en la colección de Actores del Universo.
Expresiones de postcondición (OCL)	**post1**: true
Comentarios	Ninguno

Excepción	Añadir Actor al Universo::actor intruso
Condición	El actor indica que pertenece a otro universo, y aparece en su colección.
Condición (OCL)	false
Expresión	No se incluye y la biblioteca devuelve FRACASO.
Expresión OCL	true
Comentarios	Ninguno

SOP-0062	Retirar Actor del Universo
Tipo del resultado	Éxito, si el Acto restaba en el Universo y se ha retirado correctamente.
Versión	1.0 (13/11/2010)
Autores	• José Manuel Barroso Galindo
Dependencias	• [UC-0090] Retirar Actor del Universo
Descripción	La aplicación solicita retirar un Actor al Universo dado.
Parámetros	act : Actor -- *El actor que se desea retirar del Universo.*
Expresiones de precondición	**pre1**: El actor estaba contenido en la colección de actores del Universo.
Expresiones de precondición (OCL)	**pre1**: true
Expresiones de postcondición	**post1**: El actor ya no está almacenado en la colección de actores del Universo.
Expresiones de postcondición (OCL)	**post1**: true
Comentarios	Ninguno

Excepción	Retirar Actor del Universo::actor no perteneciente
Condición	Se ha tratado de retirar un actor que no pertenece al conjunto de actores el Universo.
Condición (OCL)	false
Expresión	La biblioteca devuelve FRACASO.
Expresión OCL	true

Comentarios	Ninguno

SOP-0063	**Acceder a la Colección de Actores del Universo**
Tipo del resultado	Puntero, acceso mediante iteradores a la colección de Actores del Universo.
Versión	1.0 (13/11/2010)
Autores	• José Manuel Barroso Galindo
Dependencias	• [UC-0091] Acceder a la Colección de Actores del Universo
Descripción	La aplicación solicita consultar la colección de actores del Universo dado.
Parámetros	Ninguno
Expresiones de precondición	Ninguno
Expresiones de precondición (OCL)	true
Expresiones de postcondición	Ninguno
Expresiones de postcondición (OCL)	true
Comentarios	Ninguno

SOP-0064	**Migrar de un Universo a otro**
Tipo del resultado	Éxito, si la migración se ha llevado a cabo sin problemas.
Versión	1.0 (13/11/2010)
Autores	• José Manuel Barroso Galindo
Dependencias	• [UC-0092] Migrar de un Universo a otro
Descripción	La aplicación solicita cambiar la configuración del Universo dado para sustituirla por la de otro de un Universo compatible (del mismo tipo), cuyo descriptor también se da.
Parámetros	slot : Integer -- *Si es deseable que varios universos con el mismo descriptor se carguen simultáneamente, se debe especificar un valor máximo de repetición con este parámetro.* name : String -- *El descriptor del nuevo Universo al que se va a transformar.*
Expresiones de	Ninguno

precondición	
Expresiones de precondición (OCL)	true
Expresiones de postcondición	**post1**: El Universo se ha transformado.
Expresiones de postcondición (OCL)	**post1**: true
Comentarios	Ninguno

Excepción	**Migrar de un Universo a otro::incompatibilidad**
Condición	Si el descriptor de Universo no es compatible con el Universo que se trata de transformar.
Condición (OCL)	false
Expresión	La biblioteca devuelve FRACASO.
Expresión OCL	true
Comentarios	Ninguno

Capítulo 5 : Diseño y Arquitectura

En este capítulo encontraremos :

– Una **introducción** que trata sobre la evolución de las bibliotecas y su importancia en el mundo de la informática.

– Una descripción de la **biblioteca SDL**, la cual es una parte clave del diseño del sistema.

– Otra descripción, esta vez de la **API OpenGL**, que se utiliza exhaustivamente para las tareas de renderización de la biblioteca.

– **XML**, el metalenguaje que facilita la definición de estructuras de datos, y **TinyXML**, un parser minimalista.

– El funcionamiento básico del patrón arquitectónico que usa la biblioteca : **Programación dirigida por Eventos**.

– Un mecanismo de **gestión de mensajes** para establecer jerarquías de comunicación entre diferentes elementos de la biblioteca.

– Un ejemplo de patrón de **clases de instanciación única**, que es adecuado para aquellas clases que tienen que cumplir esta restricción.

5.1 Introducción

Durante la historia del diseño de Software, ha habido una tendencia desde tiempos muy tempranos a usar bibliotecas en el desarrollo de programas. Al principio, los primeros programas informáticos controlaban directamente los registros, espacios de memoria, operaciones e interrupciones de la máquina. Pero con este panorama, desarrollar un sistema complejo era una tarea realmente tediosa. Es por ello, que las grandes compañías tardaron poco en crear compilaciones de rutinas que reutilizaban sistemáticamente a la hora de crear nuevo Software.

El primer paso en esta dirección fue motivado por la idoneidad de separar las definiciones de datos de la implementación del programa. El concepto COMPOOL (Communication Pool) fue popularizado en 1959 con este fin. Inspirado en el clásico problema de la computación "divide et impera", procuraba la separación entre problemas para aislarlos en otros más reducidos y simples, y también abordaba la necesidad de ocultación de información. Poco más tarde, la llegada del lenguaje de programación FORTRAN otorgaba un mecanismo que facilitaba la tarea de crear bibliotecas, ofreciendo subrutinas por primera vez. Con el tiempo, el desarrollo y mantenimiento de bibliotecas se convierte en una parte clave de la industria y de la investigación, dentro de la informática. Son muchas las ventajas que otorga la utilización de bibliotecas desde la perspectiva de la Ingeniería del Software. Como sabemos, el uso de buenas bibliotecas, normalmente da una abstracción más humana al diseñador, le garantiza fiabilidad, le ahorra tiempo, y por tanto, dinero.

Tan importantes han sido algunas bibliotecas para la industria, que han llegado a convertirse en estándares. Esto ha posibilitado la búsqueda de imponer estándares por parte de algunas empresas, a veces con un fin de lucro. Por ejemplo, algunas importantes bibliotecas privadas obligan a un pago de licencia, a la inclusión de determinadas dependencias o al uso de herramientas y plataformas específicas. Además la maquinaria de algunas empresas para promocionar sus bibliotecas a veces ha logrado que se estandaricen bibliotecas no lo suficientemente robustas desde el punto de vista de la ingeniería del software, ocasionando múltiples problemas a desarrolladores y usuarios. Por ello, también ha sido creciente el interés en el desarrollo de bibliotecas de manera colaborativa y con licencias poco restrictivas. Principalmente por una filosofía de estandarización eficaz del Software.

Una biblioteca que busca popularidad también suele tener pretensiones multiplataforma y en desarrollo colaborativo éste suele ser uno de los objetivos principales. Es por ello, que en esta biblioteca que se pretende que sea portable, usaremos varias bibliotecas libres con la aspiración de disfrutar de las mismas ventajas. En ese caso, quizás podría tener más sentido usar directamente las bibliotecas de las que hace uso este proyecto, pues ahorraríamos dependencias, pero con esta fórmula no se cumpliría un objetivo clave de esta biblioteca : la abstracción. Tanto SDL como OpenGL, las 2 librerías que se usan principalmente en el desarrollo, pecan en bastantes puntos de tener un nivel muy bajo de abstracción, y de ahí surge la necesidad de crear una capa de interfaces superiores, aunque sea a costa de tener varias dependencias. Normalmente esto significa una pérdida de capacidad a la hora de controlar el sistema con libertad, y también de eficacia, pero también nos centraremos en reducir dicho impacto lo máximo posible.

Imagen esquemática con la relación entre la aplicación, la biblioteca, SDL, OpenGL y el sistema operativo.

Para ello, esta biblioteca se desarrollará en C++, un lenguaje universal, que posibilita un gran rendimiento de la CPU en la mayoría de los compiladores de numerosas plataformas, y cuenta con una buena convivencia entre niveles bajos y altos de abstracción. Además se permitirá el uso directo de funciones SDL y OpenGL para no penalizar la libertad del usuario de la biblioteca, pues lamentablemente es excesivo pretender adaptar toda la funcionalidad de ambas a nuestras interfaces en un proyecto de este tipo.

5.2 Tecnologías

5.2.1 **La biblioteca SDL**

Figura 5.1 : Logo de SDL

SDL (**S**imple **D**irectMedia **L**ayer) es una biblioteca multiplataforma, libre y abierta (licencia LGPL), escrita en C con el objetivo de dar interfaces muy simples que se encarguen de representar gráficos por pantalla, reproducir sonidos, y tratar algunos eventos de E/S como los provenientes del teclado, ratón y cualquier periférico similar. Está adaptada a numerosos lenguajes de programación como C++ (de manera nativa), Perl, Python y Pascal, también a un gran número de plataformas : GNU/Linux, Windows, Mac OS, Mac OS X, Windows CE, BeOS, FreeBSD, NetBSD, Open BSD, BSD/OS, Solaris, IRIX, QNX, PSP (Playstation Portable), Google Android, etc...

Los inconvenientes de la versión actual 1.2.X pasan por la incapacidad de utilizar aceleración 3D en su tratamiento gráfico, quedando por tanto toda la carga gráfica en la CPU, La simplicidad de sus funciones puede ser también otro inconveniente desde nuestra perspectiva. En este caso se usará SDL para el tratamiento de eventos de E/S, y también para crear el contexto de vídeo que más tarde se manipulará mediante OpenGL. La recolección de eventos, es una tarea que internamente la hace SDL en un hilo paralelo, por tanto cualquier aplicación que use esta biblioteca está implícitamente usando varios hilos de ejecución.

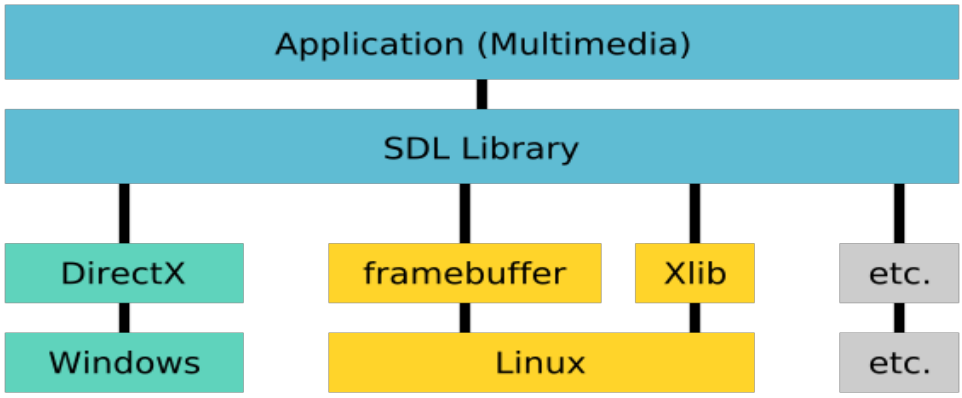


Figura 5.2 : SDL actúa como una capa
capaz de funcionar en gran número de plataformas

Cabe destacar que esta biblioteca no está apoyada en su desarrollo por ninguna empresa, aunque ha sido usada en multitud de ocasiones para elaborar software comercial, en especial videojuegos. Su siguiente versión 1.3 está llamada a acabar con ciertos inconvenientes de la biblioteca como el de la aceleración gráfica, y además estará respaldada por una empresa llamada Galaxy Gameworks.

5.2.2 **La API de OpenGL**

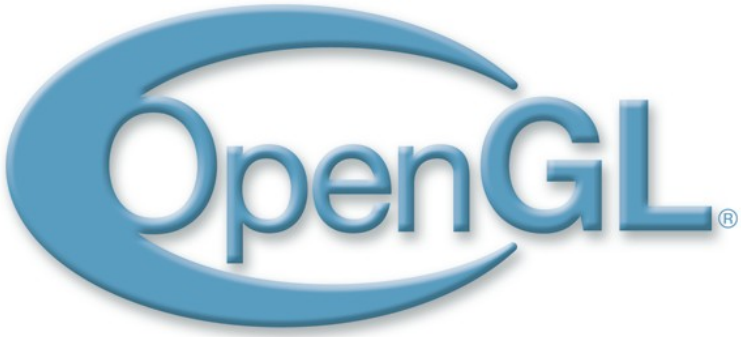


Figura 5.3 : Logo de OpenGL

OpenGL no es realmente una biblioteca en sí misma, si no una especificación estándar de una interfaz (API) multiplataforma y multilenguaje secundada por muchas de las empresas líderes de la industria, véase AMD, Apple, Blizzard, Intel Graphics Controllers y nVIDIA, entre otras. Su objetivo es el de ofrecer una interfaz para tratar gráficos 2D y 3D, cuya implementación se lleva a cabo por los fabricantes de hardware. Hay implementaciones eficientes para GNU/Linux, Mac OS, Windows, Mac OS, otras tantas plataformas Unix, e incluso para el sistema Playstation 3. También hay implementaciones en software como la biblioteca de código abierto Mesa 3D, que

permiten usar programas de OpenGL aunque no se cuente con aceleración por hardware.

La programación en OpenGL se basa en primitivas tales como puntos, líneas y polígonos que son manipuladas por funciones de muy bajo nivel. Todo ello es transformado por el pipeline gráfico conocido como la Máquina de Estados de OpenGL, hasta dar lugar a el buffer de píxeles que será representado en la pantalla. La lógica de las interfaces a usar siguen las directrices de la computación gráfica a bajo nivel, donde cada operación se suele identificar por un producto de matrices de un mismo tipo (de proyección, de vista, de color, etc..)

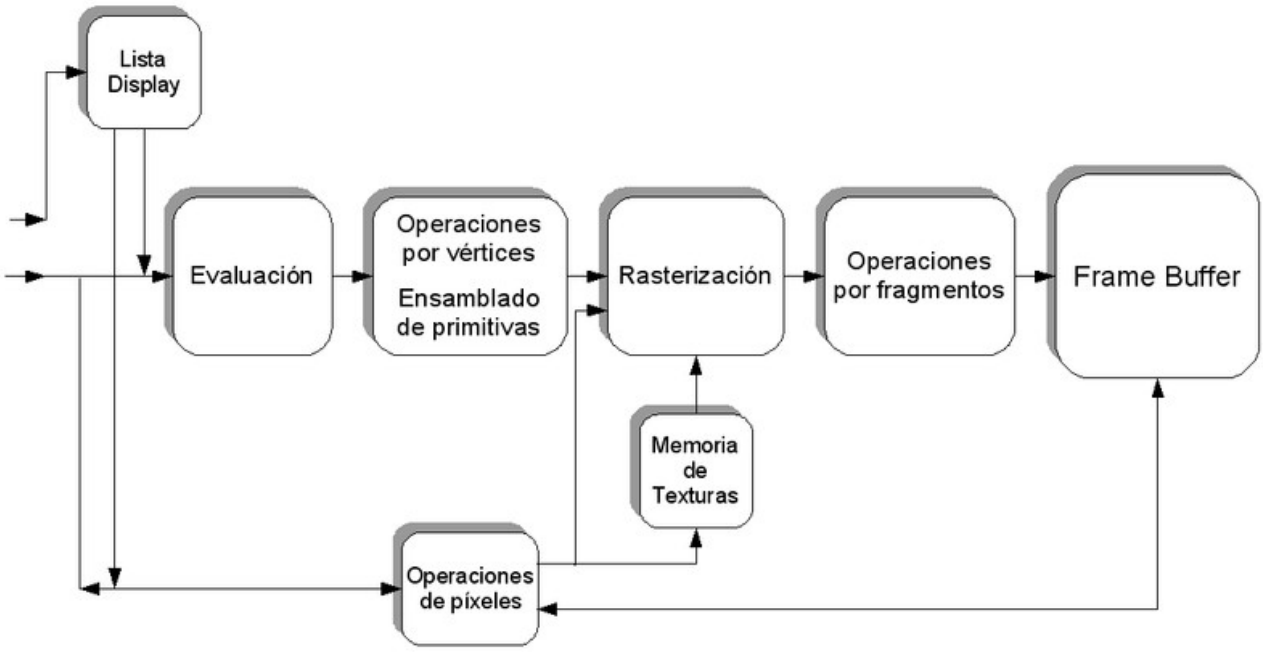


Figura 5.4 : Máquina de Estados de OpenGL simplificada

OpenGL ha evolucionado mucho desde que fué primeramente desarrollada por Silicon Graphics, y continúa haciéndolo en la actualidad para adaptarse a las nuevas técnicas y tecnologías gráficas. La última versión estable es la 4.1, sin embargo, esta biblioteca ha sido desarrollada haciendo uso de las interfaces que se publicaron en la versión 2.0 o anterior. La compatibilidad entre tal diferencia de versiones es perfecta, al menos si nos referimos al subconjunto de interfaces usadas en este proyecto.

5.2.3 **XML**

Este metalenguaje, E**x**tensible **M**arkup **L**enguaje o XML se ha definido como un estándar a la hora de definir series complejas de estructuras de datos. Gracias a su desarrollo en la Web bajo la tutela del W3C, XML es una tecnología muy popular y práctica, por tanto se le ha querido dar soporte en esta biblioteca.

Las aplicaciones desarrolladas por la biblioteca cumplen el estereotipo de necesitar una ingente cantidad de estructuras de datos externas, pues normalmente

acompañan a la información gráfica. Para este propósito, XML es una tecnología muy apropiada, pues dispone de las siguiente ventajas :

- Los archivos XML son muy fáciles de entender y se pueden modificar desde un sin fin de aplicaciones.
- XML es un estándar de la web, y por tanto es muy conocido por la inmensa mayoría de profesionales de la informática.
- En caso de no conocerse, existe mucha documentación sobre este metalenguaje.

Por contra, el punto débil de XML radica en su costoso procesamiento o parseo[14]. La W3C establece dos mecanismos de parseo del XML, SAX y DOM. El modelo DOM consiste en cargar la totalidad del archivo XML en la memoria, lo cual sólo es una gran carga cuando el documento es muy grande, mientras que el SAX accede secuencialmente mediante eventos. El problema, es que SAX no permite el mismo control que DOM del archivo cargado, no permitiendo modificaciones de los nodos ya analizados, por tanto, desde la perspectiva de esta biblioteca DOM es mejor opción. Por tanto se ha integrado una biblioteca nativa en C++ capaz de parsear XML siguiendo el modelo DOM. Se trata de TinyXML.

TinyXML es un parser de modelo DOM muy simple y minimalista, ocupando así poco espacio en memoria. Cuenta con capacidad para manejar la mayoría de los archivos XML y es un software de licencia libre y abierta.

14 **Parsear** es el proceso de analizar una secuencia de símbolos a fin de determinar una estructura gramatical con respecto a una gramática formal dada

5.3 Arquitectura

El modelo arquitectónico que seguiremos tendrá como piedra angular el tratamiento de los eventos recogidos por la librería SDL. La biblioteca obtendrá los eventos mediante la función **SDL_PollEvent** y los distribuirá a los manejadores de eventos correspondientes, siempre que estén definidos por el usuario. Este tipo de modelo arquitectónico se conoce como programación dirigida por eventos (Event-driven Model en Inglés).

```
  while (CLibrary::getActualEngine() == this) {

     if(SDL_PollEvent(&event)==1) {

           if(event.type==SDL_QUIT) {
             CLibrary::getLibrary()->SendMessage(CLibrary::MSGID_Exit);
             break;
           }

           if (eventHandlerRegister.find(event.type)!=
                                              eventHandlerRegister.end())

  ((void (*)(SDL_Event*)) eventHandlerRegister[event.type])(&event);

           } else {

                 onIdle();

                 drawFrame();

                 Chrono.nextFrame();
           }
     }
  }
```

Cuadro 5.1 : Bucle principal de la clase motora.

5.3.1 **Programación dirigida por Eventos**

En general, en este modelo arquitectónico el flujo del programa está determinado por los eventos entrantes en el sistema. Por tanto, el bucle principal del programa está dividido en 2 partes claramente diferenciables. La detección o selección de Eventos y el tratamiento mediante Manejadores de Eventos.

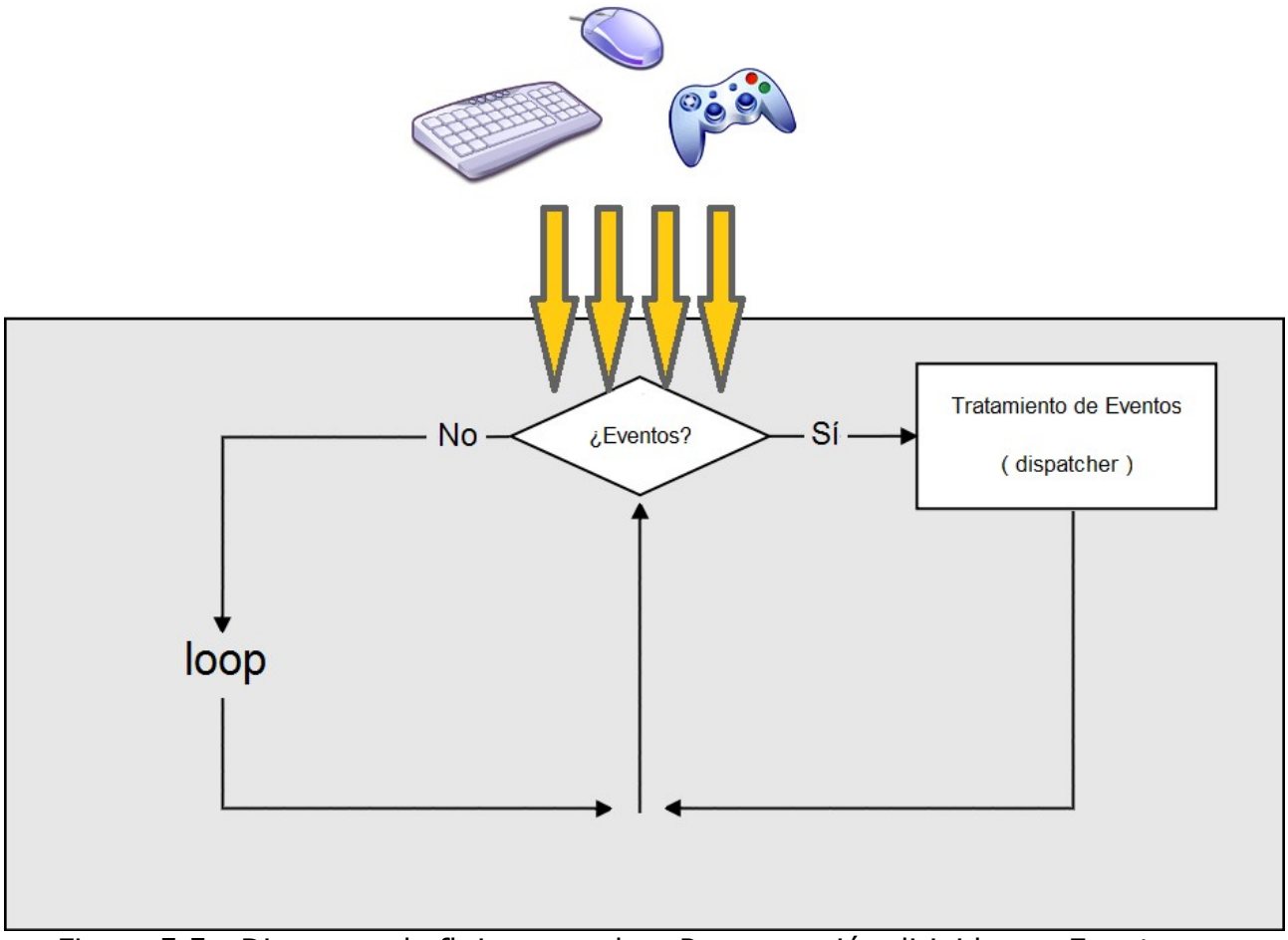


Figura 5.5 : Diagrama de flujo general en Programación dirigida por Eventos.

Este gráfico explica de manera conceptual el flujo de esta arquitectura. Que como se puede apreciar es muy coherente con el código del cuadro 5.1 En ese caso, tenemos en amarillo la parte correspondiente a la detección de eventos, y en rojo la correspondiente al tratamiento mediante Manejadores de Eventos (esta línea se explicará con más profundidad en el capítulo de implementación). El código en azul es el correspondiente a 'loop', que suele ser la parte de espera y/o actualización del sistema (en este caso, ambas).

Este tipo de arquitectura otorga una alta abstracción al diseñador, que tratará cada tipo de Evento de forma individual conforme se vayan registrando. Es por ello, que se suele usar en interfaces gráficas, entornos de programación asistida por diálogos y aplicaciones de tiempo real entre otros. Ante la ausencia de eventos, lo habitual es permanecer a la espera, o invocar rutinas de procesamiento que actualicen el estado del sistema, como ocurre en el desarrollo de nuestra biblioteca.

También es destacable lo fácil que puede ser lograr una gran eficiencia de los recursos del sistema, ya que cada Manejador de Eventos puede ser lanzado tras el detector de Eventos en muy pocos ciclos de reloj.

5.3.2 **Gestión de Mensajería**

Otra funcionalidad que tiene esta biblioteca es la posibilidad de usar mensajes en contraposición a los eventos definidos por el usuario. El concepto es el de organizar jerárquicamente todos los objetos de trascendencia en ejecución y facilitar la comunicación directa entre ellos mediante protocolos definidos por el usuario que se comunican a través de una simple interfaz.

De esta manera, cada clase debería heredar de una clase manejadoras de mensajes, y por tanto ser capaz de procesar mensajes entrantes y salientes tal y como lo estime oportuno el usuario. Por defecto este tipo de comunicación será usado también en el seno de la aplicación para realizar funciones donde el tiempo de respuesta no sea crítico.

5.3.3 **Patrón de Clases de instanciación Única**

Se trata de un mecanismo que logra forzar que para determinadas clases sólo sea posible crear un objeto. La idea es muy simple :

```
       … ejemplo con constructor …

Clase::Clase() {

   if (Clase::instancia_unica == NULL)
     Clase::instancia_unica = this;
   } else {
     error();
   }
}
```

Cuadro 5.2 : Ejemplo de instanciación única

En la biblioteca se plantea también el uso de mecanismos similares fuera de constructores, ya que al no tener estos la posibilidad de devolver un valor, es más difícil notificar si se ha tratado incumplir la restricción de unicidad.

Capítulo 6 : Implementación

En este capítulo encontraremos :

– Una descripción del **entorno tecnológico** utilizado para desarrollar la documentación, las pruebas y las 15000 líneas de código de las que consta la implementación.

– Una serie de indicaciones sobre la **compilación** del código de la biblioteca.

– Un recopilatorio de los **tipos de datos** que aparecen en las interfaces de la biblioteca.

– Todas las **interfaces de la biblioteca**, junto a descripciones de su funcionamiento general, y su lógica interna.

6.1 Entorno y Material Utilizado

Para realizar esta biblioteca, se han utilizado los siguientes recursos tecnológicos :

Elemento	Tipo	Detalle
Hardware	PC	· Intel Core 2 Duo E6320 (4M L2 Cache, 1.86 Ghz) · 2GB DDR2 (667 Mhz) · ATI Radeon X550 (128 MB)
Software	Lenguaje de Programación	· C++
	Entorno de Programación	· Visual C++ 2008/2010 · VIM
	Compiladores	· Compilador de VC++ 2010 · g++
	Dependencias	· OpenGL 2.0 o superior · LibSDL 1.2.14 · STL · TinyXML
	Sistema de Control de Versiones	· Subversion 1.5.5
	Sistema de Gestión y Control de Proyectos	· Trac 0.11.1
	Sistemas Operativos	· Windows 7 64 bits · Ubuntu 10.04 LTS 32 bits
Documentación	Procesador de Textos	· Go Open Office 3.2.1
	Gestor de Requisitos	· REM 1.2.2
	Elaboración de Diagramas	· DIA 0.97.1
	Retoque Gráfico	· GIMP 2.6
	Edición XML	· Notepad++ 5.8.4

6.2 Despliegue del entorno de compilación :

El desarrollo de esta biblioteca se ha realizado en 2 sistemas operativos simultáneamente para garantizar que se cumplía el objetivo de portabilidad que se perseguía desde el inicio.

El código de la biblioteca se compone de archivos .h y .cpp . Pero a parte, deberá contarse con el código de TinyXML, también distribuida en el CD y que debería situarse en el directorio 'tinyxml'. SDL_rotozoom.cpp y .h que es un pequeño código reutilizado para eventualmente redimensionar estructuras la estructura de SDL SDL_Surface, deberá aparecer en el directorio principal. Y también deberá incluirse la carpeta 'resources' con su contenido, ya que se trata de los recursos gráficos y varios archivos en formato XML.

6.2.1 **Windows 7**

En Windows 7 se deberá enlazarse manualmente con las cabezeras y módulos de la SDL. Para ello simplemente se añade el directorio 'include' y el directorio 'lib' de la carpeta de la librería. Una vez hecho esto sólo quedará indicar los .lib a usar :

sdlmain.lib, sdl.lib, sdl_image.lib, sdl_ttf.lib, opengl32.lib, glu32.lib

Para la distribución del ejecutable, deberá meterse en el mismo directorio todos lo archivos .dll disponibles.

6.2.2 **Ubuntu 10.04**

En sistemas linux hay que instalar los paquetes de desarrollo de SDL. Estos paquetes son libsdl1.2-dev, libsdl-ttf2.0-dev y libsdl-image1.2-dev . Con esto, simplemente habría que usar el Makefile contenido en el directorio principal, usando la sentencia 'make all'.

Para la ejecución hay que tener instalados los siguientes paquetes : libsdl1.2debian-all (éste puede variar según la distribución), libsdl-image1.2 y libsdl-ttf2.0-0 .

Hay que tener en cuenta, tanto para Linux, como para Windows, que una vez realiada la compilación, para la ejecución OpenGL debe poder ejecutarse. Esto se logra instalando los controladores de la tarjeta gráfica. El problema es que sobre todo en Linux, hay cierto hardware gráfico que no controladores compatibles con OpenGL, o estos son muy deficientes.

6.3 Tipos de datos usados en las interfaces de la biblioteca

Primeramente se ha de analizar que tipo de datos entran en juego a la hora de usar la biblioteca. Como veremos, algunos son propios del lenguaje, otros son definidos por la propia biblioteca y por último encontramos tipos de datos provenientes definidos por la biblioteca SDL.

6.3.1. Tipos básicos de C++

La librería usa los tipos básicos de C++ en la mayoría de los casos, tanto para los parámetros de sus funciones, como para sus valores de retorno. Como en ciertos de ellos, puede haber ambigüedades según el compilador que se use, es conveniente especificar los tipos básicos que se usan por la biblioteca. En la siguiente tabla veremos los tipos que se deben conocer antes de usar la biblioteca.

Tipo	bits	Rango de Valores
unsigned char / Uint8[15]	8	[0 , 255]
char	8	[-128 , 127]
short	16	[-32.768 , 32.767]
unsigned short / Uint16	16	[0 , 65.535]
Unsigned / Uint32	32	[0 , 4.294.967.295]
int	32	[-2.147.483.648 , 2.147.483.647]
unsigned long	32	[0 , 4.294.967.295]
long	32	[-2.147.483.648 , 2.147.483.647]
float	32	[±1.18e-38 , ±3.40e38] precisión : 24 bits
double	64	[± 2.23e-308 , ±1.79e308] precisión : 53 bits

15 **Uint8**, **Uint16** y **Uint32** son equivalencias definidas por SDL de los tipos que acompañan en la misma celda. Se usarán dichas definiciones por ser más cortas en las interfaces de la biblioteca.

bool	1	Valor : {true, false}

Cómo también importamos de serie la *Standard Template Library* (STL), también contamos con ciertos contenedores de gran utilidad para el manejo de datos. Uno de ellos es **std::string**, que dado a que posibilita manipular cadenas de texto de una manera bastante cómoda, es usado también como parámetro y valor de retorno de algunas funciones. No obstante, el tipo **const char***, que es la forma nativa de las cadenas de texto en C++ también se puede usar en ocasiones que pueda llegar a ser más idóneo.

6.3.2 Tipos de datos definidos por la biblioteca.

La definición de tipos de datos específicos se hace irremediablemente necesaria para el desarrollo de esta biblioteca. Además el uso exhaustivo de C++ permite diseñar la biblioteca desde la óptica de la programación orientada a objetos, y por tanto, cada módulo de los que consta el diseño, es fácilmente definible mediante una o varias **clases** de manera bastante sistemática. Algunas de ellas, actúan como tipos de datos importantes que aparecen en varias de las interfaces de la biblioteca.

Por otro lado, se aprovechan las plantillas de contenedores de STL, para tratar conjuntos de datos y se definen clases parametrizadas a partir de ellas. Según conviene, las plantillas usadas son **std::list** y **std::vector** en las interfaces de la biblioteca y **std::map** se se usa frecuentemente en la lógica interna de ésta. Con **std::list** definimos listas doblemente enlazadas del tipo pertinente, mientras que con **std::vector** definimos arrays dinámicos. Gracias a **std::map** se definen funciones, también conocidas como mapas, es decir, una forma de contenedor donde se sigue el patrón "llave → valor" a la hora de almacenar y acceder a los elementos. Para todas las definiciones que se formulan a partir de estas 3 plantillas, también contamos con sus correspondientes iteradores, que se identifican de la forma **std::tipoPlantilla**<**tipoParametrización**>**::iterator** y que el algunos casos son el único recurso que facilita la biblioteca para acceder a determinadas colecciones.

Otros tipos de datos usados por la biblioteca son los **enum**. En C++, los **enum** o **enumeraciones** son un tipo especial de variables cuya propiedad radica en que tienen por rango de valores un conjunto de constantes enteras denominadas *constantes de enumeración*. La biblioteca define para diferentes funciones, algunas enumeraciones para así forzar al usuario a que introduzca valores que dichas funciones esperan. Dichas enumeraciones serán explicadas conforme vayan siendo usadas por las interfaces que se van a definir más adelante.

A parte, también se definen varias constantes de precompilación. Las relacionadas con el tratamiento de errores son **EXITO**, **FRACASO**, y **NOERROR**, mientras que **MSGPARM** es una definición del tipo de dato usado por la mensajería que realmente se traduce como un **void***, o lo que es lo mismo, un puntero de tipo genérico. En la tabla de la siguiente página vemos éstos y los anteriores tipos de datos definidos por la biblioteca :

Familias de datos	Denominación	Comentarios	
Clases de Tipos de datos básicos	CColor	Definida por la biblioteca.	
	CPoint	Definida por la biblioteca.	
	CRectangle	Definida por la biblioteca.	
	CCoordinate	Definida por la biblioteca.	
Clases de la biblioteca	CMessageHandler	Definida pero modificable por el usuario.	
	CEngine	Clase redefinida por el usuario.	
	CSpriteset	Definida por la biblioteca.	
	CSprite	Definida por la biblioteca.	
	CUniverse	Clase redefinida por el usuario.	
	CActor	Clase redefinida por el usuario.	
	CCamera	Clase redefinida por el usuario.	
Definiciones de Colecciones	ActorCollection	Tipo : std::list<CActor*>	
	UniverseCollection	Tipo : std::list<CUniverse*>	
	RectArea	Tipo : std::vector<CRectangle*>	
Enumeraciones	TypeText	Constantes de enumeración : TT_LINE TT_BOX	
	TypeError	Constantes de enumeración : TE_standard TE_fileExists TE_controlViolation TE_SDL_NOMSG TE_SDL_MSG TE_OPENGL_NOMSG TE_OPENGL_MSG	
	TypeRendeProjection	Constantes de enumeración : TRP_ORTHO TRP_PERSPECTIVE	
	TypeColorTBox	Constantes de enumeración : TCTB_ALL TCTB_TEXT TCTB_BOX	
Definiciones de constantes de precompilación	EXITO	Tipo : int	Valor : 0
	FRACASO	Tipo : int	Valor : -1
	NOERROR	Tipo : const char*	Valor : "|-| No error"
	ONLY_TEXTURE	Tipo : int	Valor : 0x01
	ONLY_SDL_SURFACE	Tipo : int	Valor : 0x02
	WITH_SDL_SURFACE	Tipo : int	Valor : 0x04
	ALL_TEXT	Tipo : int	Valor : -1
Otras definiciones	MSGPARM	Tipo : void*	

6.3.3 **SDL_Event, un tipo de dato definido por la biblioteca SDL**

SDL como ya sabemos, recolecta los eventos entrantes al sistema para ponerlos a disposición de la biblioteca. El formato de los eventos que SDL devuelve una vez llamada la función de petición de eventos, es una estructura llamada SDL_Event. Por tanto, el reconocimiento de los eventos se deberá llevar a cabo analizando esta estructura. Y dicho análisis, se debe realizar en el Manejador de Eventos correspondiente, previa identificación del tipo de evento por la clase motor que se esté ejecutando.

Sin embargo el Manejador de Eventos es realmente una función definida por el usuario, preparada para recibir un puntero a SDL_Event como único argumento. Así, el trabajo con esta estructura, es una tarea a la que debe estar preparado aquel que quiera hacer uso del soporte de eventos que ofrece la biblioteca. SDL_Event, como vemos en el siguiente cuadro es realmente un tipo **union** de varias estructuras.

```
typedef union{
  Uint8 type;
  SDL_ActiveEvent active;
  SDL_KeyboardEvent key;
  SDL_MouseMotionEvent motion;
  SDL_MouseButtonEvent button;
  SDL_JoyAxisEvent jaxis;
  SDL_JoyBallEvent jball;
  SDL_JoyHatEvent jhat;
  SDL_JoyButtonEvent jbutton;
  SDL_ResizeEvent resize;
  SDL_ExposeEvent expose;
  SDL_QuitEvent quit;
  SDL_UserEvent user;
  SDL_SysWMEvent syswm;
} SDL_Event;
```

El elemento 'type' actúa como elemento discriminador, es decir, según su valor podemos saber qué estructura está realmente activa de todas las que aparecen en el cuadro. Sus valores posibles, disponen de un alias fácilmente identificable con las estructuras de la unión y son los siguientes :

SDL_ACTIVEEVENT	SDL_JOYAXISMOTION	SDL_SYSWMEVENT
SDL_KEYDOWN	SDL_JOYBALLMOTION	SDL_VIDEORESIZE
SDL_KEYUP	SDL_JOYHATMOTION	SDL_VIDEOEXPOSE
SDL_MOUSEMOTION	SDL_JOYBUTTONDOWN	SDL_USEREVENT
SDL_MOUSEBUTTONDOWN	SDL_JOYBUTTONUP	SDL_NUMEVENTS
SDL_MOUSEBUTTONUP	SDL_QUIT	

Dichos valores son los que deberá aportar el usuario a la clase CEngine que crea conveniente, cuando le desee añadir un nuevo Manejador de Eventos. A continuación vemos en el siguiente cuadro un ejemplo de uso :

```
    … se ha inicializado la biblioteca …

    CEngineExample* e = new CEngineExample();
    e->setEventHandler(SDLK_KEYDOWN,CEngineExample::manejaTeclado);

    CLibrary::addEngine(engine);
    CLibrary::processEngines();

… en la clase CEngineExample se define la función manejaTeclado …

  static void CEngineExample manejaTeclado(SDL_Event* event) {

    if (event->key.keysym.sym == SDLK_SPACE)
         CEngineExample::instance->pulsadaTeclaEspaciadora();
  }
```

A través del método **setEventHandler** (en amarillo) se especifica el tipo de evento y la función que lo ha de tratar. Dicha función se define de manera estática dentro de la propia clase por comodidad, aunque no tiene porque ser así.

Dentro de la función se ve en azul un ejemplo de una navegación por la estructura **SDL_Event** gracias a la cual se detecta si la tecla pulsada es la barra espaciadora o no. Por tanto, cada vez que entra una pulsación de dicha tecla en el sistema, se ejecuta el método **pulsadaTeclaEspaciadora**, donde se debería realizar el procesamiento que se crea oportuno.

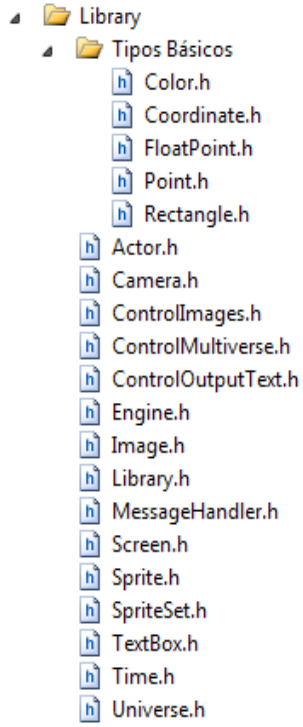
Hay que tener en cuenta que, como los Manejadores de Eventos son funciones globales o métodos estáticos públicos (que viene a ser lo mismo), se ha de tener en cuenta que no se puede acceder a la variable miembro de ningún objeto. Por ello, en esta ocasión se accede a una variable miembro estática de una clase, en la cual se presupone que se ha almacenado una instancia cuyos métodos por tanto, sí podríamos invocar al estar contenidos en la misma clase que **manejaTeclado**.

Todos estos métodos y funciones que aparecen en el código, incluido **setEventHandler**, se describen detalladamente a continuación, en la siguiente sección de este capítulo. Se recomienda que se relea este ejemplo una vez que se comprendan todas las funciones que aparecen en este código.

6.4 Interfaces de la Biblioteca

Se dispone de un numeroso conjunto de interfaces públicas al alcance del usuario para invocar las diversas funciones de esta biblioteca. Están organizadas de manera que cada familia de funciones con propósitos similares se ven englobadas dentro de la misma clase. Siguiendo así el paradigma de POO[16] que enfoca esta biblioteca, algunas de estas interfaces, son en realidad métodos de clases instanciables, mientras otras son funciones de clases estáticas. También se ha optado por definir objetos globales de instancia única, para invocar de manera cómoda sus métodos como si de funciones globales se tratasen.

A continuación se recorren todas las interfaces de la biblioteca, describiendo su procedimiento correcto de uso, y explicando con detenimiento la lógica interna de aquellas funciones que son claves en el funcionamiento del sistema. Cada cabecera corresponde con una única clase, de modo que se describirá el contenido público de los archivos .h componentes de la biblioteca, según podemos ver a continuación :



Una consideración común a todas las interfaces es el uso de las definiciones **EXITO** y **FRACASO**. Por norma general, una interfaz que tiene un retorno del tipo **int**, devuelve **EXITO** si ha cumplido su propósito correctamente. En caso contrario devuelve **FRACASO**, y registra un error. En caso de que una interfaz retorne punteros, el puntero nulo **NULL**, asume el rol de **FRACASO**. Los errores anotados por la biblioteca pueden ser consultados por el usuario mediante las funciones establecidas para dicho fin que veremos en la sección '*Gestión de la Biblioteca – Library.h'.*

16 POO : Programación Orientada a Objetos

6.4.1 Clase Gestora de Mensajería - MessageHandler.h

Se trata de CMessageHandler, una clase que aporta métodos para gestionar el recibimiento y enviado de mensajes, a aparte de dar un mecanismo que garantiza la unicidad de los identificadores asociados a los tipos de mensajes. Está pensada para ser la clase base o **superclase** de otras clases esenciales para la biblioteca.

De CMessageHandler derivan las siguientes clases propias de la biblioteca : CLibrary, CEngine, CUniverse, CCamera y CActor. El motivo es que éstas clases son las entidades estructurales que dirigen el rumbo de la ejecución en el sistema, y por tanto son quienes deben responder a las peticiones demandadas por los mensajes.

Esta clase es instanciable, y todas sus interfaces públicas son métodos exceptuando la función asignadoras de identificadores de mensajes que será la primera que comentemos.

Uint32 CMessageHandler::getNextMSGID(**bool** instant);

Cuando se define un mensaje, es necesario que la ID asignada provenga de esta función, ya que de otro modo el funcionamiento de las funciones **SendMessage** y **readMessages** sería errático. Existen 2 clases de mensajes, los que se atienden instantáneamente y los que esperan a la llamada de **readMessages**. En esta función debemos especificar la clase del nuevo mensaje a definir, y para ello disponemos del parámetro **instant**, que por defecto vale **true**.

int SendMessage(**Uint32** MsgID,**MSGPARM** Parm1,**MSGPARM** Parm2);

Este método es el encargado de enviar mensajes. Se debe invocar en la instancia de CMessageHandler que se tome como destinatario. El parámetro **MsgID** debe ser un identificador válido de mensaje, y **Parm1** y ***Parm2*** son parámetros adicionales opcionales que por defecto valen **NULL**.

El esquema de este método es bastante simple. Si MsgID es par, se llama al método **onMessage** para un procesamiento inmediato, en cambio si es impar se almacena el mensaje en un búfer de mensajes.

void readMessages();

El gestor de mensajes de la biblioteca procesa los mensajes entrantes de 2 formas. Una instantánea y otra bajo petición explícita. **readMessages** es el método que tiene por objetivo formular dicha petición explícita y así procesar todos los mensajes pendientes acumulados. Internamente se limita a llamar a **pendingMessage** con cada mensaje registrado, tras lo cual limpia el búfer de mensajes.

```
CMessageHandler* getParent();
void setParent(CMessageHandler* pmhNewParent);
bool HasParent();
```

En la propia clase CMessageHandler definimos 2 mensajes que se usan por defecto en toda clase derivada de CMessageHandler. Estos 2 mensajes ofrecen un mecanismo para establecer una jerarquía de objetos, gracias a la cual la comunicación entre clases se facilita.

La definición de ambos mensajes es la siguiente :

```
Uint32 CMessageHandler::MSGID_addChild =  getNextMSGID(true);
Uint32 CMessageHandler::MSGID_RemoveChild =  getNextMSGID(true);
```

Y para su uso se requieren introducir los 2 parámetros adicionales, donde el primero debe ser una instancia de la clase Padre, y el segundo una instancia de la clase Hija.

El mecanismo es accionado mediante el método **setParent** como vemos a continuación :

```
void CMessageHandler::setParent(CMessageHandler* pmhNewParent) {

      if(HasParent())
            SendMessage(MSGID_RemoveChild,
                        (MSGPARM)getParent(),(MSGPARM)this);

      m_pmhParent=pmhNewParent;


      if(HasParent())
            SendMessage(MSGID_addChild,
                        (MSGPARM)getParent(),(MSGPARM)this);
}
```

Dicho método se invoca por defecto en el constructor '`setParent(pmhParent)`' y en el destructor '`setParent(NULL)`' de cada clase derivada de CmessageHandler.

Los siguientes métodos de CMessageHandler no son públicos. Son métodos protegidos de la clase (es decir, el acceso desde otras clases no está permitido), no obstante, admiten ser redefinidos en las clases derivadas por el usuario de cara a gestionar cada tipo de mensaje definido anteriormente por él mismo.

```
void pendingMessage(Uint32 MsgID,MSGPARM Parm1,MSGPARM Parm2);
int onMessage(Uint32 MsgID,MSGPARM Parm1,MSGPARM Parm2);
```

Son los métodos de procesamiento de los mensajes. **pendingMessage** es invocado por **readMessages**, mientras que **onMessage** es invocado directamente por **SendMessage** cuando llega un mensaje instantáneo. Por defecto,

pendingMessage no tiene ningún tratamiento de mensajes, por tanto es necesario redefinirlo cada vez que se quieran usar mensajes no instantáneos en la biblioteca. En cambio, **onMessage** cuenta con una definición que responde al tratamiento de mensajes del tipo **CMessageHandler::MSGID_addChild** y **CMessageHandler::MSGID_RemoveChild**.

```
int CMessageHandler::onMessage(Uint32 MsgID,
                               MSGPARM Parm1,MSGPARM Parm2) {
   if(MsgID==MSGID_addChild) {
     if(this==(CMessageHandler*)Parm1) {
          OnaddChild((CMessageHandler*)Parm2);
          return EXITO;
     } else {
          return FRACASO;
     }

   } else if(MsgID==MSGID_RemoveChild) {
     if(this==(CMessageHandler*)Parm1) {
          onRemoveChild((CMessageHandler*)Parm2);
          return EXITO;
     } else {

          return FRACASO;
     }

   } else  {
     CLibrary::Error("Mensaje de tipo no registrado",TE_standard);
     return FRACASO;
   }

}
```

La redefinición de ambos métodos deberían seguir un modelo consistente con la arquitectura de la biblioteca. Por ello, el usuario debería consultar las plantillas ofrecidas para tal propósito, que están presentes en el Anexo A del de este documento.

void OnaddChild(**CMessageHandler*** pmhChild);
void onRemoveChild(**CMessageHandler*** pmhChild);

Ambos métodos, como se puede apreciar en el cuadro anterior, son invocados por **onMessage**. Por defecto, no contienen ninguna instrucción. Simplemente son útiles ante una eventual redefinición por el usuario.

6.4.2 Gestión de la Biblioteca – Library.h

CLibrary es la clase principal de la Biblioteca. Contiene funciones para la iniciación y terminación de la biblioteca, la gestión de motores y la gestión de errores principalmente. Hereda de **CMessageHandler**, y por tanto es una clase que se puede

instanciar. No obstante, como sigue el patrón de instancia única, todas las interfaces genuinas de esta clase son funciones estáticas.

Como clase derivada de **CMessageHandler**, es capaz de tratar mensajería, y en algunos casos es indispensable su uso para realizar determinadas funcionalidades. No se le asigna ninguna instancia padre, pero es idóneo que otras clases derivadas de **CMessageHandler** tengan a **CLibrary** como padre, sobre todo los objetos de la clase **CEngine**.

int CLibrary::startLibrary(**bool** xmlconfig);
int CLibrary::startLibrary(**int** width , **int** height , **int** bpp , **bool** fullscreen, **bool** doublebuff) ;

La inicialización de la biblioteca se lleva a cabo por estos métodos. En la primera versión, podemos optar por que se realice una inicialización en la cual simplemente se inicializan los componentes de SDL. Sin embargo, con el parámetro **xmlconfig** a **true** se parsea el documento '*config.xml*' en busca de una inicialización del contexto de vídeo predefinida. Por defecto dicho parámetro se pone a **false**.

En la segunda versión, el usuario directamente establece los parámetros para una inicialización del contexto de vídeo. Es por ello que la naturaleza de dichos parámetros será comentada en la sección de '*Gestión de la Pantalla – Screen.h*'.

Por tanto, la única posibilidad de lanzar la biblioteca sin inicializar la pantalla es mediante la forma '`CLibrary::startLibrary();`'.

int CLibrary::addEngine(**CEngine*** engine,**int** priority);

Esta función se utiliza para registrar un **CEngine** por la biblioteca. Internamente simplemente se agrega el **CEngine** a una colección, y se modifica el atributo privado **priority** del mismo.

int CLbrary::processEngines();

La ejecución formal de la biblioteca se realiza mediante la función presente. processEngines consta de un bucle en el que se selecciona un CEngine de acuerdo a su atributo priority, y lo ejecuta. Una vez se haya especificado el fin del procesado de la colección de CEngine (ya sea porque se han retirado los CEngine, o porque el usuario lo ha notificado por mensajería), la función finaliza.

La lógica interna de esta función es una parte vital de la biblioteca. El proceso completo se muestra a continuación :

```
int CLibrary::processEngines() {

   if(!s_pTheLibrary) {
     Error("Biblioteca no inicializada");
     return FRACASO;
   }

   setActualEngine(NULL);

   // Selección del CEngine a ejecutar entre el Conjunto de CEngine.

   getLibrary()->engineIn.sort(CLibrary::orderEngine);

   for (list<CEngine*>::iterator it = getLibrary()->engineIn.begin(),
                                 jt = getLibrary()->engineIn.end();
         it != jt && getActualEngine()==NULL;       ++it) {

     if (!(*it)->done) {
          (*it)->done = true;
          setActualEngine(*it);
     }
   }

   if (getActualEngine() == NULL) {

     for (list<CEngine*>::iterator it = getLibrary()->engineIn.begin(),
                                   jt = getLibrary()->engineIn.end();
                                     it != jt;       ++it)
          (*it)->done = false;

     setActualEngine(getLibrary()->engineIn.front());
     getLibrary()->engineIn.front()->done=true;
   }

   // Ejecución del CEngine seleccionado

   while (getActualEngine()) {

     if (!getActualEngine()->isInitialized())
          if (!getActualEngine()->onInit()) {
               Error("Could not initialize new interface!");
               return(-1);
          }

     getActualEngine()->loop();

     getLibrary()->readMessages();
   }

   return EXITO;
}
```

Consta de 2 partes. En la primera (amarillo) simplemente se hace la selección del **CEngine** a ejecutar según el atributo **priority**. El criterio es el siguiente : se ordena el conjunto de **CEngine** según su priority en orden ascendente mediante la instrucción '`getLibrary()->engineIn.sort(CLibrary::orderEngine);`'. Después, en el primer bucle **for** se descartan aquellos **CEngine** cuyo atributo **done** valga **true**. Esto es así para que no se seleccione siempre el mismo **CEngine** en el caso de que no se varíe el atributo **priority** y se lance varias veces la función **processEngines**. El segundo bucle **for** se realiza en el caso de que todos los atributos **done** valgan **true**. Se ponen a **false**, para inicializar una nueva vuelta.

En la segunda parte (azul), la compuesta por el bucle **while**, se ejecuta el CEngine seleccionado, inicializándolo si fuera necesario y finalmente se revisa la mensajería que pudiera haber llegado a la biblioteca. Esta acción de revisar la mensajería es la que determina qué **CEngine** constará como seleccionado en la siguiente iteración del bucle **while**. Por consiguiente, para comprender la ejecución real del bucle, es necesario comentar la gestión que lleva a cabo **CLibrary** de la mensajería.

Se definen 6 tipos de mensajes, todos de procesado no instantáneo, a continuación se describe su funcionalidad y se explican los tratamientos más trascendentes para el funcionamiento de **processEngines** :

· **CLibrary::MSGID_ChangeEngine** se encarga de cambiar el **CEngine** actual por el siguiente siguiendo el mismo criterio de selección visto en los dos primeros bucles **for**. Por tanto, el tratamiento interno de este mensaje como vemos a continuación es bastante similar.

```
if (MsgID==MSGID_ChangeEngine) {

   setActualEngine(NULL);
   getLibrary()->engineIn.sort(CLibrary::orderEngine);

   for (list<CEngine*>::iterator it = getLibrary()->engineIn.begin(),
                                 jt = getLibrary()->engineIn.end();
         it != jt && getActualEngine()==NULL;            ++it) {
     if (!(*it)->done) {
                (*it)->done = true;
                setActualEngine(*it);
     }
   }
   if (getActualEngine() == NULL) {
     for (list<CEngine*>::iterator it = getLibrary()->engineIn.begin(),
                jt = getLibrary()->engineIn.end();   it != jt;  ++it)
                (*it)->done = false;
     setActualEngine(getLibrary()->engineIn.front());
     getLibrary()->engineIn.front()->done=true;
   }
}
```

· **CLibrary::MSGID_Exit** se utiliza para dar por finalizado el bucle **while** terminando así la función **processEngines**. Su tratamiento pasa simplemente por asignar **NULL** al **CEngine** actual o activo.

· **CLibrary::MSGID_Restart** es una opción para restablecer el estado inicial de todos los **CEngine** registrados, ejecutando el que tenga la menor priodidad.

· **CLibrary::MSGID_ReloadEngine** realiza una tarea similar a la anteriormente vista. En este caso, se restablece el estado inicial del **CEngine** actual y se sigue ejecutando.

· **CLibrary::MSGID_RunEngine** tiene por objetivo ejecutar un **CEngine** concreto, aunque ni siquiera haya sido registrado por la biblioteca. Para ello **Parm1** deberá ser el **CEngine** que se quiera ejecutar. Tiene el siguiente tratamiento :

```
} else if (MsgID==MSGID_RunEngine) {

   CEngine* engine = (CEngine*)Parm1;
   if (dynamic_cast<CEngine*>(engine)) {

     bool find = false;

   for (list<CEngine*>::iterator it = getLibrary()->engineIn.begin(),
                      jt = getLibrary()->engineIn.end();it!=jt;++it)
     find = find || engine == *it;

   for (list<CEngine*>::iterator it = getLibrary()->engineOut.begin(),
                      jt = getLibrary()->engineOut.end();it!=jt;++it)
     find = find || engine == *it;

   if (!engine) {
     engineOut.push_back(engine);
   }

   setActualEngine(engine);

}
```

En él, se comprueba que **Parm1** es realmente una instancia de la clase **CEngine**, y se busca entre la colección de **CEngine**s registrados. En el caso de que no pertenezca a ella, se introduce en una colección distinta, y se lanza.

· **CLibrary::MSGID_KillEngine** por contra, indica un **CEngine** mediante **Parm1**, el cual deberá ser finalizado, eliminado de las colecciones a las que pertenezca, y borrado de la memoria. Es decir, retira definitivamente el **CEngine** del sistema.

CLibrary* CLibrary::getLibrary();
CEngine* CLibrary::getActualEngine();

Estas funciones simplemente devuelven la instancia de la biblioteca en el primer caso para por ejemplo, mandar un mensaje, y la instancia del **CEngine** actual o activo en el segundo.

string CLibrary::readLastError();
string CLibrary::popError();

void CLibrary::Error (**const char***,**TypeError** e);
void CLibrary::Error (**std::string**,**TypeError** e);
void CLibrary::Error (**char***,**TypeError** e);

Finalmente, éstas funciones son las encargadas de gestionar los errores de la biblioteca. Con las 2 primeras leemos los errores ya registrados. La diferencia radica en que **readLastError** devuelve un **string** con el último error registrado por la biblioteca, mientras que **popError** también lo devuelve y además desregistra dicho error. En caso de que no haya errores, ambos devuelven **NOERROR**.

```
string CLibrary::readLastError()
{
   if (errors.empty())
     return NOERROR;
   else
     return errors.back();
}
```

```
string CLibrary::popError() {
   if (errors.empty())
     return NOERROR;
   else {
     string ret = errors.back();
     errors.pop_back();
     return ret;
   }
}
```

La función Error en sus múltiples formas realiza la misma tarea que consiste en registrar un error. La biblioteca utiliza este mecanismo para registrar los errores que ésta detecta, pero con esta función también brinda al usuario la oportunidad de utilizar la gestión de errores de la biblioteca para tratar los errores de su aplicación.

Las constantes de enumeración de tipo **TypeError** se muestran en el cuadro de tipos definidos por la biblioteca. En general la especificación de errores según dichas constantes no es más que una característica secundaria que otorga más orden a la gestión de errores. Sin embargo, con **TE_OPENGL_MSG** y **TE_SDL_MSG** recopilamos también información del error proveniente de *OpenGL* y *SDL* respectivamente.

6.4.3 **Gestión del Motor – Engine.h**

CEngine es la clase que dirige en mayor parte el flujo de ejecución de la biblioteca. Su cometido es el de recibir Eventos, desplazarlos hacia el Manejador de Eventos correspondiente si estuviera definido, y establecer una lógica que se encargue de actualizar y llamar a la renderización del sistema, que tiene la consideración de bucle principal.

También deriva de **CMessageHandler**, pero no define ningún tipo de mensaje, por tanto toda su funcionalidad está plasmada en sus interfaces. Sin embargo, es una clase de la que deben heredar clases definidas por el usuario, ya que en la redefinición de sus métodos deben ubicarse las rutinas de la lógica más primarias de la aplicación.

Como clase instanciable, sus interfaces adquieren la forma de métodos de la instancia a la que se refiera el usuario. También cuenta con un constructor que debe recoger como parámetro una instancia padre **(NULL** por defecto), y un destructor donde deberá tratarse de la liberación de los recursos reservados.

Los métodos públicos son los siguientes :

bool isInitialized();

Es una simple llamada que devolverá **false** cuando el objeto no ha sido inicializado, o tras su finalización, y **true** una vez ejecutado el método **onInit**.

int drawFrame();

Dicho método por definición no contiene ninguna instrucción. Depende de una redefinición por el usuario para su funcionamiento, el cual debería situar en este método todas las rutinas de llamadas a la renderización del estado actual del motor.

const void* setEventHandler(**Uint8** type,void (eventHandler)(SDL_Event*));

Una parte clave de la lógica del motor reside en esta función. Gracias a ella, se establece un Manejador de Eventos asociado a un tipo de Eventos entre los definidos por SDL (ver cuadro de tipos de eventos de SDL_Event). El segundo parámetro, que corresponde con el Manejador de Eventos es realmente un puntero a una función del tipo **void func(SDL_Event*)**, y por tanto, el usuario deberá definir una función de estas características con la visibilidad adecuada para poder asignarla como Manejadora de Eventos. Para ello, lo idóneo es que se defina como un método estático de la clase motor que se esté definiendo.

La lógica interna de **setEventHandler** es la que aparece en el siguiente cuadro :

```
const void* CEngine::setEventHandler(Uint8 type,
                                     void (eventHandler)(SDL_Event*)) {

   const void* ret;

   if (this->eventHandlerRegister.find(type) ==
                                        this->eventHandlerRegister.end())
     ret = NULL;
   else
     ret = this->eventHandlerRegister[type];

   this->eventHandlerRegister[type] = (const void*) eventHandler;

   return ret;
}
```

Consiste en añadir el Manejador de Eventos a una colección de tipo mapa (osea, par llave → valor), rescatando previamente el anterior puntero asignado al tipo de Evento correspondiente si lo hubiera, y devolviéndolo al usuario como retorno.

Como no es posible definir una colección de objetos a funciones, se ha usado una conversión mediante casting del puntero a función en **const void***, siendo este tipo de dato también el tipo de valor de retorno.

Otra serie de métodos son protegidos. Y en este caso, como **CLibrary** se define como una clase amiga (**friend class**) de **CEngine**, dichos métodos protegidos son invocados por **CLibrary** en el contexto del procesamiento de **CEngines** que anteriormente hemos comentado. Todos estos métodos son redefinibles por el usuario, siendo esta una parte crítica del funcionamiento del sistema.

int loop();

El método **loop**, es el bucle principal del motor. Él se encarga de recoger los eventos entrantes en el sistema y de lanzarlos al Manejador de Eventos correspondiente en cada iteración, además de actualizar el sistema cuando se mantiene a la espera de nuevos eventos.

Es ejecutado durante la función **CLibrary::processEngines** como una parte fundamental de la lógica que ya comentamos. Por tanto, una vez analicemos loop podremos terminar de comprender el funcionamiento de **CLibrary::processEngines.** En el siguiente cuadro vemos la lógica interna del método **loop** :

```
int CEngine::loop() {

     … comprobación de precondiciones ...

   CLibrary::setActualEngine(this);

   SDL_Event event;

   while (SDL_PollEvent(&event)==1);

   while (CLibrary::getActualEngine() == this) {

     if(SDL_PollEvent(&event)==1) {

           if(event.type==SDL_QUIT) {
             CLibrary::getLibrary()->SendMessage(CLibrary::MSGID_Exit);
             break;
           }

//si hay manejador de eventos registrado para este evento, lo ejecuta

           if (eventHandlerRegister.find(event.type)!=
                                          eventHandlerRegister.end())

  ((void (*)(SDL_Event*)) eventHandlerRegister[event.type])(&event);

           } else {

                onIdle();

                drawFrame();

                Chrono.nextFrame();
           }
     }

   return EXITO;

}
```

Este procedimiento ya ha sido parcialmente analizado en el apartado de arquitectura de la biblioteca, pues resulta ser un bucle principal típico de una arquitectura de programación orientada a eventos.

Deteniéndonos en otros detalles, observamos como el motor ejecutado se declara ante **CLibrary** como motor actual o activo en la primera instrucción, lo cual es clave para el correcto funcionamiento de **CLibrary::processEngines**. Tras dicha instrucción, un primer bucle **while** se encarga de liberar todos los eventos registrados por la biblioteca que no se han producido durante la ejecución del motor presente.

A continuación, viene el bucle **while** principal que a su vez se divide en 2 partes, la del procesamiento de eventos y la de espera y actualización. En la primera, como

ya se ha visto anteriormente, se recoge el evento de la función **SDL_PollEvent** y se lanza el Manejador de Eventos apropiado, ignorando el evento en caso de que no hubiera Manejador de Eventos disponible para el tipo de evento entrante. La segunda parte, ocurre si **SDL_PollEvent** no ha devuelgo ningún evento, y por tanto se lanzan las rutinas de actualización y espera. Estas son **onIdle**, encargada de la actualización de la lógica del motor, **drawFrame**, que se ocupa de llamar a la renderización de los elementos gráficos y **Chrono.nextFrame**, que como ya veremos en "*Gestión de la Temporización - CTime*" se ocupa del control de etapas de tiempo.

La redefinición del método **loop** por el usuario no es necesaria, y en el caso de que se haga, debería respetar la plantilla del Anexo A, ya que garantiza un correcto funcionamiento. Ante una posible redefinición, las rutinas idóneas son aquellas que restablezcan el control del motor, si existiera algún mecanismo de control definido por el usuario que así lo requiriera.

void deselect();

Es un mecanismo que usa **CLibrary** para retirar el control de la ejecución al motor actual. La definición por defecto simplemente cambia el valor del motor actual a un puntero **NULL**. En una eventual redefinición por el usuario, aquí debería tratarse la pérdida del control de la ejecución por parte del motor, si para la lógica definida por el usuario fuera necesario.

int onInit();

Este método se encarga de la inicialización del motor. Por defecto simplemente cambia un booleano para que el método **isInitialized** devuelva **true**. Sin embargo, es importante redefinir este método ya que en él deberían de situarse las reservas de recursos necesarios por el motor.

int onExit();

Al contrario que el método anterior, **onExit** finaliza el motor. La implementación por defecto también consiste en cambiar únicamente un booleano, en este caso a **false** de cara al método **isInitialized**. En la redefinición de este método debería tratarse la liberación de los recursos reservados en **onInit** o en otras posibles circunstancias.

int onIdle();

onIdle se ejecuta en la fase de espera y actualización, justo antes de **drawFrame**. Al igual que este último, **onIdle** tampoco cuenta con ninguna instrucción por defecto. Por tanto, debería ser redefinido por el usuario para actualizar el estado del motor antes de llamar a las rutinas de renderización de los elementos gráficos. Es de esperar que en este método se sitúe la parte más pesada del procesamiento definido por el usuario.

6.4.4 Gestión de la Pantalla – Screen.h

En la clase **CScreen** se trata la inicialización y modificación del Contexto de Vídeo y la renderización del sistema. Está compuesta exclusivamente por funciones estáticas, por tanto no hay instanciación posible de objetos del tipo **CScreen**.

int CScreen::start (**int** width , **int** height , **int** bpp , **bool** fullscreen, **bool** doublebuff) ;
int CScreen::start (**int** width , **int** height , **int** bpp , **float** scalex, **float** scaley,**bool** fullscreen, **bool** doublebuff) ;

CScreen::start es la función utilizada para la inicialización del Contexto de Vídeo. Es por tanto una función de obligatorio uso por parte del usuario, antes de que quiera tratar cualquier tipo de información gráfica. No obstante, cómo existen versiones de **CLibrary::startLibrary** que llaman internamente a esta función, el usuario puede no necesitar invocarla.

En la primera versión, los parámetros indican la resolución adoptada por la pantalla. Es decir el ancho, el alto, el valor de bits por pixel y si se desea entrar en modo pantalla completa o no. El valor especificado por el usuario debería estar acorde con la capacidad de la máquina huésped de la aplicación para que no haya errores por parte de la SDL.

En la segunda nos encontramos con los mismos parámetros, más dos que indican valores de escalada horizontal y vertical, por si el usuario estima oportuno que haya una deformación visual de tamaño en todos los elementos gráficos tratados por la biblioteca.

doublebuff es un parámetro que aparece en ambas versiones y por defecto vale **true**. Cuando adquiere ese valor, la biblioteca activa el doble búfer para el Contexto de Video creado. El double buffering es una técnica para mejorar el rendimiento de la actualización de la pantalla a costa de un mayor gasto de memoria gráfica.

int CScreen::render();

render es una de las funciones más importantes de esta biblioteca. Se encarga de procesar con la ayuda de *OpenGL* toda elemento gráfico almacenado durante la etapa de ejecución actual, y representar el estado resultante en la pantalla. Una vez realizado dicho proceso, limpia el sistema de recursos gráficos y se dispone a recopilar los nuevos recursos gráficos provenientes de las otros módulos de la biblioteca para la siguiente etapa de ejecución.

En su implementación se usan intensivamente primitivas de *OpenGL* de manera secuencial, conformando un procesamiento en cadena que logra un tiempo más compacto entre la primera y la última llamada a una función gráfica comprendida en una misma etapa de ejecución. Esto posibilita un mejor rendimiento del hardware gráfico. A continuación se muestra dicho procedimiento :

```
int CScreen::render ( ) {

	if (!m_SDL_Surface) {
		CLibrary::Error("Video context not inicialized");
		return false;
	}


	rendering = true;

	beginRenderMode(RENDER_TEXTURE_STANDARD);

	for (list<SToRender*>::iterator it = graphicMaterial.begin(),
			jt = graphicMaterial.end(); it != jt ; ++it) {

		SToRender* em =(*it);

		procRenders[em->type](em->pointer);

		delete *it;
	}

	endRenderMode(RENDER_TEXTURE_STANDARD);

	graphicMaterial.clear();

	rendering = false;

	SDL_GL_SwapBuffers();

	glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
	glLoadIdentity();

	deleteResources();

	return EXITO;
}
```

El proceso de renderizado como tal ocurre entre las funciones **beginRenderMode** y **endRenderMode** (sección amarilla). En ese espacio vemos que se recorre un bucle **for** donde se van recuperando recursos gráficos **SToRender**, y mediante un array de punteros a funciones llamado **procRenders**, se procesan de manera individual y según el tipo de recurso gráfico.

A continuación del bucle **for** se limpia la lista de materiales gráficos, que anteriormente han sido borrados de la memoria, y se ejecuta la instrucción '*SDL_GL_SwapBuffers();*', que es la encargada de representar el resultado de la renderización, en ese momento almacenado en el búfer correspondiente, en la pantalla.

La siguiente instrucción **glClear** se dedica a limpiar los búferes de *OpenGL* usados. Y **deleteResources** es un procedimiento que elimina los objetos de la clase **CSpriteset**, **CSprite**, y **CImage** que se han dejado de usar durante la etapa de ejecución anterior.

El atributo **type** de la estructura **SToRender** es un tipo enumerado definido de la siguiente manera :

```
enum TypeResource {   TR_CANVAS, TR_FLOATCANVAS, TR_ROTATION,
     TR_TRANSLATION, TR_LOCATION,    TR_PUSHMATRIX,
     TR_POPMATRIX,   TR_SCALATION,   TR_COLOR   };
```

Por tanto, a cada constante de enumeración, le corresponde una función de procesamiento gráfico, de las que eran invocadas durante el transcurso del bucle **for**. Es en ellas donde se realiza el proceso de renderización como tal. A continuación se exponen sus implementaciones :

Para **TR_CANVAS** y **TR_FLOATCANVAS**:

```
void CScreen::procRendCanvas(void* pointer) {

   SRenderCanvas* n = (SRenderCanvas*) pointer;

   SCanvas m_pSurface = n->canvas;
   Uint8 flags = n->flags;
   CPoint ptDst = n->ptDst;

   delete n;

   if (m_pSurface.h != 0 || m_pSurface.w !=0 ) {

        glBindTexture(GL_TEXTURE_2D, m_pSurface.tex);

        float relW = (float)m_pSurface.w2/(float)m_pSurface.w;
        float relH = (float)m_pSurface.h2/(float)m_pSurface.h;

        glScalef((1.0/m_ScaleX ),(1.0/m_ScaleY ),0.0);
```

```
            glBegin(GL_QUADS);
                  if (flags == 0) {
                        glTexCoord2f(0.0f, relH);
                        glVertex2f(0, m_pSurface.h2);
                        glTexCoord2f(relW, relH);
                        glVertex2f(m_pSurface.w2, m_pSurface.h2);
                        glTexCoord2f(relW, 0.0f);
                        glVertex2f(m_pSurface.w2, 0);
                        glTexCoord2f(0.0f, 0.0f);
                        glVertex2f(0,0);
                  } else if (flags == 1) {

                        glTexCoord2f(relW, relH);
                        glVertex2f(0, m_pSurface.h2);
                        glTexCoord2f(0.0f, relH);
                        glVertex2f(m_pSurface.w2, m_pSurface.h2);
                        glTexCoord2f(0.0f, 0.0f);

                        glVertex2f(m_pSurface.w2, 0);
                        glTexCoord2f(relW, 0.0f);
                        glVertex2f(0,0);
                  } else if (flags==2) {
                        glTexCoord2f(0.0f, 0.0f);
                        glVertex2f(0, m_pSurface.h2);
                        glTexCoord2f(relW, 0.0f);
                        glVertex2f(m_pSurface.w2, m_pSurface.h2);
                        glTexCoord2f(relW, relH);
                        glVertex2f(m_pSurface.w2, 0);
                        glTexCoord2f(0.0f, relH);
                        glVertex2f(0,0);
                  } else  {
                        glTexCoord2f(relW, 0.0f);
                        glVertex2f(0, m_pSurface.h2);
                        glTexCoord2f(0.0f, 0.0f);
                        glVertex2f(m_pSurface.w2, m_pSurface.h2);
                        glTexCoord2f(0.0f, relH);

                        glVertex2f(m_pSurface.w2, 0);
                        glTexCoord2f(relW, relH);
                        glVertex2f(0,0);
                  }
            glEnd();

      }

   }
```

En dicho procesamiento se muestra como tratar las texturas para que sean representadas mediante OpenGL. Internamente cada textura se representa como la cara superior de un cubo (**GL_QUADS**) que se adapta a las dimensiones de la imagen procesada. La cadena de **if else** cubre dicha renderización para distintos valores del parámetro **flags**, es decir, para distintos valores de espejado (nulo, horizonta, vertical, o ambos).

Los siguientes procesamientos son los correspondientes a los efectos gráficos, y se aplican a las imágenes que se renderizan en futuras procesados de **TR_CANVAS** hasta que el efecto se revoque mediante modificaciones de la matriz de *OpenGL*.

Para **TR_ROTATION** :

```
void CScreen::procRendRotation(void* pointer) {

     SRenderRotation* n = (SRenderRotation*) pointer;

     GLfloat angle = n->angle;
     GLfloat x = n->x;
     GLfloat y = n->y;
     GLfloat z = n->z;

     delete n;

     glRotatef(angle,x,y,z);

   }
```

Se llama a la primitiva **glRotatef**, que como se aprecia, trabaja con el tipo de dato **Glfloat**, un alias del tipo básico **float** de C++. El parámetro **angle** indica la intensidad del giro en grados, mientras los demás parámetros sirve para especificar el vector unitario sobre el cual se aplica dicho giro. Para un plano bidimensional en perspectiva ortogonal el vector unitario adecuado sería **x=0**, **y=0**, **z=1**, por ejemplo.

Para **TR_TRANSLATION** :

```
void CScreen::procRendTranslation(void* pointer) {

   SRenderTranscalation* n = (SRenderTranscalation*) pointer;

   GLfloat x = n->x;
   GLfloat y = n->y;
   GLfloat z = n->z;

   delete n;

   glTranslatef(x,y,z);
}
```

La primitiva **glTranslatef** simplemente se encarga de desplazar una eventual imagen a renderizar tanta distancia como se especifica en los parámetros **x**, **y**, **z**.

Para **TR_LOCATION** :

```
void CScreen::procRendLocation(void* pointer) {

   SRenderLocation* n = (SRenderLocation*)pointer;

   float posx = n->posx;
   float posy = n->posy;
   float width = n->width;
   float height = n->height;
   float zoom = n->zoom;

   delete n;

  glViewport(posx*m_ScaleX,posy*m_ScaleY,width*m_ScaleX,height*m_ScaleY);
   glMatrixMode( GL_PROJECTION );
   glLoadIdentity();

   if (trp == TRP_ORTHO) {
      glOrtho( 0.0, width*m_ScaleX, height*m_ScaleY, 0.0, 0.0, 1.0 );

   } else {
      gluPerspective(90.0f,width/height,1.0,m_maxZ);
      glTranslatef(-width,height,-240.0);
      glRotatef(180.0,1.0,0.0,0.0);
   }
}
```

Mediante este proceso, se especifica una porción de la pantalla concreta mediante **glViewport** , se inicializa la matriz de proyección con **glMatrixMode** y **glLoadIdentity**, anulando por tanto todo efecto definido por anterioridad a las próximas renderizaciones de texturas y se define una perspectiva. La perspectiva a definir se elige de acuerdo a la configuración de **CScreen** en ese momento, y puede tratarse de una proyección ortogonal (**glOrtho**) o una perspectiva de corte realista (**gluPerspective**).

Antes de renderizar texturas, siempre debe haberse procesado algún recurso del tipo **TR_LOCATION** para que finalmente puedan aparecer por pantalla.

Para **TR_PUSHMATRIX** y **TR_POPMATRIX** :

```
void Cscreen::procRendPush(...) {
   glPushMatrix();
}
```

```
void Cscreen::procRendPop(...) {
     glPopMatrix();
}
```

Estos procesos sirven para salvar (**push**) y recuperar (**pop**) el estado actual de la matriz de *OpenGL*. El mecanismo que se emplea es el de una pila, por tanto se pueden hacer sucesivas modificaciones de la matriz intercalando **push**, para más tarde ir recuperando los estados anteriores mediante **pop**.

Para **TR_SCALATION** :

```
   void CScreen::procRendScalation(void* pointer) {

      SRenderTranscalation* n = (SRenderTranscalation*)
pointer;

      GLfloat x = n->x;
      GLfloat y = n->y;
      GLfloat z = n->z;

      delete n;

      glScalef(x ,y ,z);

   }
```

De manera idéntica a lo definido en **TR_TRANSLATION**, se invoca la primitiva **glScalef** con los factores de escalamiento en horizontalidad, verticalidad y profundidad.

Para **TR_COLOR** :

```
   void CScreen::procRendColor(void* pointer) {

      SRenderColor* n = (SRenderColor*) pointer;

      GLfloat red = n->red;
      GLfloat green = n->green;
      GLfloat blue = n->blue;
      GLfloat alpha = n->alpha;

      delete n;

      glColor4f(red,green,blue,alpha);

   }
```

Con este proceso aplicamos una máscara de deformación del color en las próximas texturas especificando la deformación en cada componente de color con la primitiva **glColor4f**. Los valores de los parámetros deben estar comprendidos entre 0.0 y 1.0,

siendo 1.0 el valor asociado a la intensidad máxima de color, y 0.0 a la intensidad mínima.

Con **TR_COLOR** se ha terminado de profundizar en los procesamientos internos de renderización. A continuación volvemos con las interfaces públicas, comentando primeramente aquellas que crean los recursos gráficos que acabamos de analizar.

int CScreen::locateRenderScene (**float** posx, **float** posy, **float** width, **float** height, **float** zoom) ;

En **locateRenderScene** definimos un recurso del tipo **TR_LOCATION**. Como se ha comentado anteriormente, es necesario que haya un procesamiento de este tipo antes de renderizar cualquier otro recurso gráfico. Los parámetros definen el recurso de la siguiente manera :

```
int CScreen::locateRenderScene(float posx, float posy, float width,
                                float height,float zoom) {

   SRenderLocation* n = new SRenderLocation;
   n->posx = posx;
   n->posy = posy;
   n->width = width;
   n->height = height;
   n->zoom = zoom;

   SToRender* em = new SToRender;

   em->type = TR_LOCATION;
   em->pointer = (void*) n;

   graphicMaterial.push_back(em);

   return EXITO;
}
```

Los parámetros por defecto de **locateRenderScene** se ajustan a la resolución del Contexto de Vídeo actual, siendo el parámetro **zoom** un factor de escalada adicional.

int CScreen::rotate(**float** angle, **float** x=0.0, **float** y=0.0, **float** z=1.0);
int CScreen::translate(**float** x, **float** y, **float** z);
int CScreen::scale(**float** x, **float** y, **float** z);
int CScreen::color(**float** red, **float** green, **float** blue, **float** alpha);
int CScreen::color(CColor* col,**float** alpha=1.0);
int CScreen::pushMatrix();
int Cscreen::popMatrix();

De la misma manera se generan los recursos de los otros tipos, creando las estructuras de datos correspondientes y rellenándolas con los parámetros ofrecidos por el usuario.

Lo único a destacar por tanto de estas interfaces, es la versión de **CScreen::color** que usa el tipo de dato **CColor**. Esta manera de manipular colores puede ser más comoda pora el usuario, pues cada componente de color se especifica en 8 bits, que traducen a **float** de la siguiente manera :

```
int CScreen::color(CColor* col, float alpha) {

   return color(((float)col->getR())/255.0,
         ((float)col->getG())/255.0,
         ((float)col->getB())/255.0,
         alpha);
}
```

Por tanto, se pierde algo de precisión.

int CScreen::projectionMode(TypeRendeProjection trp, **float** zMax);

Configura el valor de proyección por defecto de CScreen. Sólo está la opción de elegir entre **TRP_ORTHO** y **TRP_PERSPECTIVE**. No obstante, también sirve para especificar el valor de profundidad máximo en caso que se elija **TRP_PERSPECTIVE**. Por defecto, el valor máximo es **400.0**, y por regla general a mayor **zMax**, menor rendimiento si hay valores de profundidad muy heterogéneos, y a menor **zMax**, mayor rendimient a costa de poder recortar elementos gráficos que han excedido el límite de profundidad.

int CScreen::clear ( ) ;

Esta función realiza la misma tarea que se lleva a cabo en **render**, al terminar el bucle **for**. Por tanto, se utiliza para eliminar la información de renderización almacenada hasta el momento durante la etapa de ejecución actual.

int Cscreen::quit();

Finaliza el Contexto de Vídeo. Antes de volver a procesar información gráfica el usuario debería volver a inicializar el Contexto de Vídeo mediante la función **start**.

int CScreen::changeScreen(**int** width , **int** height , **int** bpp ,
float scalex, **float** scaley, **bool** fullscreen) ;
int CScreen::ToggleFullscreen ();
int CScreen::setDoublebuffer (**bool** doublebuff);

Las funciones presentes modifican el Contexto de Vídeo actual, y para ello deben reiniciarlo. En concreto, **changeScreen** actúa del mismo modo que **start**, **ToggleFullScreen** se limita a alternar entre modo pantalla completa y modo ventana y **setDobulebuffer** activa o desactiva el doble búfer en el Contexto de Vídeo.

El proceso de **changeScreen** es el siguiente :

```
int CScreen::changeScreen(int width, int height, int bpp, float
                      scalex, float scaley, bool fullscreen) {
   if (!m_SDL_Surface) {
     CLibrary::Error("Video context not inicialized");
     return FRACASO;
   }

   clear();

   GraphicResources info;

   saveResources(info);

   quit();

   if ( start (width,height,bpp,scalex,scaley,fullscreen,m_Doublebuff)
                == FRACASO)
     return FRACASO;

   reloadResources(info);

   return EXITO;

}
```

Se hace uso de la función **quit** y **start**, pero antes debe hacer 2 procesos para salvar los recursos gráficos actuales y restaurarlos una vez creado el nuevo Contexto de Vídeo. Este se lleva a cabo en **saveResources** y **reloadResources**.

float CScreen::getScaleX();
float CScreen::getScaleY();
Uint8 CScreen::getBpp();
int CScreen::getWidth();
int CScreen::getHeight();
bool CScreen::isFullscreen();

Este conjunto de funciones responde únicamente a un acceso a los atributos privados correspondientes de CScreen.

float Cscreen::getR(),getG(),getB(),getA();

Mediante este otro conjunto, accedemos a los valores de la última modificación de color registrada por la biblioteca.

6.4.5 Gestión de Imágenes – ControlImages.h,CSpriteset.h,CSprite.h

Con esta clase, se gestionan todos los recursos gráficos importados de archivos externos. Se trata por tanto, de una clase contenedora de recursos gráficos almacenados en unidades denominadas **CSpritesets**.

CControlImages es una clase instanciable pero que sigue el patrón de instanciación única. Se ha optado por declarar un objeto global de esta clase llamado **CImg** que será el modo de interactuar con este módulo de la biblioteca. Así, cada interfaz de las definidas a continuación, es un método a llamar del objeto **CImg**.

int CImg.add(const char* name,**Uint8** mode);

Carga un recurso externo de un fichero descrito por el parámetro **name**. El parámetro mode indica como debe almacenarse dicho fichero en la memoria. Las alternativas para **mode** son **ONLY_TEXTURE** indicando que la se debe guardar la textura en la memoria de vídeo, **ONLY_SDL_SURFACE**, que por el contrario indica que debe hacerse en la memoria principal y **WITH_SDL_SURFACE** que lo salva en ambos espacios de memoria. Es importante tener en cuenta antes de especificar **mode**, que para renderizar una imagen debe haberse cargado con **ONLY_TEXTURE** o **WITH_SDL_SURFACE**. Por defecto, **mode** adqueire el valor de **ONLY_TEXTURE**.

Como vemos a continuación, **CImg.add** lleva el control de cuantas veces se ha cargado el mismo fichero, y ahorra la duplicación de objetos idénticos.

```
int CControlImages::add(const char* name,Uint8 mode) {
   CSpriteset* sptset;

   int ret=search(name);
   if (ret <0) {
      if (!lastIndexAdded.empty()) {
            ret=lastIndexAdded.top();
            lastIndexAdded.pop();
      } else
            for (int i=0 ; ret < 0 ; i++)
                  if (set.find(i)==set.end())
                        ret = i;

      sptset=new CSpriteset(name,mode);
      set[ret]=sptset;
   } else {
      sptset=set[ret];
   }
   ++count[sptset];
   return ret;
}
```

CImg, en este y los demás métodos, almacena una cuenta de los índices asociados a cada **CSpriteset** y registra cuantas ocurrencias del mismo objeto se han

solicitado en cada instante. De este modo, se crean nuevos objetos sólo cuando es necesario.

Por otro lado, al llamar al constructor de **CSpriteset**, se invoca un proceso de carga del fichero externo llamado **loadChipset** que se describe a continuación :

```
void CSpriteset::loadChipset(string& c,Uint8 mode,string* cPrev) {

     … inicialización de variables …

     … análisis del descriptor del archivo …

     … recopilación de la estructura xml almacenada en el archivo .grd
       asociado al descriptor …

     chipset=IMG_Load(resource(s_aux.c_str()).c_str());
     if (!chipset) { CLibrary::Error(s_aux,TE_fileExists); }

     … redimensionamiento de chipset en caso de que sea necesario …

     … fragmentación de chipset en imágenes más pequeñas de acuerdo a la
           descripción rescatada del .grd …

     m_pImage=CImage::toSCanvas(fragmento_chipset,mode,sp_scale);

     … creación de estructura de áreas para el Sprite, y del cpoint …

     m_pSprite=new CSprite(m_pImage,new CPoint(globalCP));

     … introducción de las áreas anteriormente creadas en el Sprite …

     m_pSprite->setName(iinfo.name);

     add(m_pSprite); //Añadimos el sprite a este objeto cspriteset.

     … finalización ...

     SDL_FreeSurface(chipset);
     chipset=NULL;

}
```

En este proceso, que es bastante más largo de lo que se ha expuesto, se trata de parsear un archivo **.grd**. Un archivo **.grd** es un archivo XML que define la estructura de un **CSpriteset**. Debe llevar el mismo nombre que la imagen a importar, y debe construirse como se indica en el Anexo B del documento presente.

La importación del archivo gráfico como tal, llega con **IMG_Load**, función de un modulo auxiliar de *SDL* que soporta gran cantidad de formatos gráficos como *BMP, GIF, JPEG, LBM, PCX, PNG, PNM, TGA, TIFF, XCF, XPM.*

Tras ello, se procede a la conversión de cada fragmento del chipset original, en una estructura de datos **SCanvas**, que es con la que trabaja internamente la biblioteca. Esto se realiza en la función **CImage::toSCanvas** como se describe en el próximo cuadro.

Por último, se crea el objeto **CSprite** a partir de la estructura **SCanvas**, se le da su nombre correspondiente, y se añade al **CSpriteset**. Después sólo queda liberar los recursos del chipset mediante **SDL_FreeSurface**.

```
SCanvas Cimage::toSCanvas(...) {

     … precondiciones …

     SCanvas pSurface;

     … inicialización de variables …

     if (mode == ONLY_TEXTURE ) {
           int saved_flags;
           int  saved_alpha;

           image = SDL_CreateRGBSurface(SDL_SWSURFACE, pSurface.w,
                     pSurface.h, surface->format->BitsPerPixel,
                     0x000000ff, 0x0000ff00, 0x00ff0000, 0xff000000);

           … detección de errores …

           … formateo de image …

           glGenTextures(1, &pSurface.tex );
           glBindTexture( GL_TEXTURE_2D,pSurface.tex );

glTexParameteri( GL_TEXTURE_2D,GL_TEXTURE_MIN_FILTER,GL_LINEAR);
glTexParameteri( GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER,GL_LINEAR );
glTexParameteri( GL_TEXTURE_2D, GL_TEXTURE_WRAP_S,GL_CLAMP_TO_EDGE);
glTexParameteri( GL_TEXTURE_2D, GL_TEXTURE_WRAP_T,GL_CLAMP_TO_EDGE);

           glTexImage2D( GL_TEXTURE_2D, 0,GL_RGBA, pSurface.w,
           pSurface.h, 0, GL_RGBA, GL_UNSIGNED_BYTE, image->pixels );

           glPixelStorei(GL_UNPACK_ROW_LENGTH, 0);

           … liberación de estructura SDL_Surface …
     }

   … tratamiento para otros valores de 'mode' …
      return pSurface;
   }
```

En esta función se genera la textura interpretable por *OpenGL*, dicho acto ocurre en la función **glTexImage2D** y como se ve, los valores de los pixeles que componen

la imagen se toman de la estructura **SDL_Surface** llamada **image** definida y formateada anteriormente.

int CImg.remove(**Uint32** n);

El usuario debe utilizar esta función para notificar el cese de uso de un **CSpriteset** asociado al identificador **n**. Cuando el contador de ocurrencias para el identificador **n**, baja a 0, automáticamente se prodece a eliminar el **CSpriteset** a través de la pila de eliminación que dispone **CScreen**, y que se ejecutará al final de la actual etapa de ejecución.

```
int CControlImages::remove(Uint32 n) {
   if (set.find(n)!=set.end()) {
      CSpriteset* sptset = set[n];
      int c=--count[sptset];
      if (c < 1)  {
            if (c==0) {
                  CScreen::spritesetToDelete.push_back(sptset);
                  set.erase(set.find(n));
                  count.erase(count.find(sptset));
                  lastIndexAdded.push(n);
            } else {
                  CLibrary::Error("Cantidad de Spriteset violada.");
                  return FRACASO;
                  }
            }

            return EXITO;
      }

      CLibrary::Error("No existe el Spriteset.");
      return FRACASO;

   }
```

CSpriteset* CImg.get(**Uint32** n);

CImg.get devuelve la instancia asociada al identificador **n** dado, siempre que dicho identificador sea válido, en caso contrario devuelve **NULL**.

int CImg.size();

Con esta función simplemente se devuelve la cantidad de **CSpriteset** registrados por **CImg**.

int CImg.search(const char* name);
int CImg.search(CSpriteset* object);

Con **search** se busca el identificador de un **CSpriteset** en la colección que se corresponda con los parámetros que otorga el usuario.

void CImg.clear();

Elimina todos los **CSpriteset** registrados por CImg hasta el momento a través de **CScreen**.

int CImg.getCount(**Uint32** n);

Esta función devuelve el número de ocurrencias registradas para un identificador dado como parámetro **n**.

La gestión de imágenes no es llevada a cabo únicamente por la clase CControlImages, sino también recae en las clases **CSpriteset** y **CSprite**. A continuación, se describen métodos de estas clases instanciables que no obstante sólo pueden ser creadas por el objeto **CImg**, de la manera vista anteriormente en su método **add**.

Los siguientes métodos corresponden a la clase **CSpriteset** :

bool has (CSprite* pspt) ;

Devuelve **true** si el puntero pspt pertenece a la colección de **CSprite** registrados en la instancia de **CSpriteset** que se ha usado para invocar dicho método. **false** en caso contrario.

int search (CSprite* pspt) ;

De manera similar al método anterior, éste devuelve el identificador que un **CSprite pspt** ocupa en la colección de **Csprites**.

CSprite* get (**int** n) ;

Este método funciona de manera inversa al anterior. A partir de un identificador **n**, se devuelve la instancia del **CSprite** correspondiente.

int size () ;

Devuelve el número de **CSprites** contenidos en la colección del **CSpriteset**.

string getName();
Uint8 getMode();

Devuelven los atributos **name** y **mode** respectivamente del **CSpriteset**.

Los siguientes métodos corresponden a la clase **CSprite** :

void setName(**const char*** newName);
string getName();

Metodos que establece y devuelve respectivamente el atributo **name** de el **CSprite** que se está tratando.

vector<RectArea*>& getAllAreas () ;

Devuelve la colección del conjuntos tipo RectArea, de manera que puede ser recorrida según los mecanismos que crea conveniente el usuario.

RectArea* getArea(**int** n);

El usuario obtiene un conjunto de tipo **RectArea** siempre y cuando el parámetro **n** tenga asociado un conjunto de este tipo. En caso contrario devuelve **NULL**.

CRectangle* getRect(**int** n,**int** m);

Con este método, el usuario recoge el **CRectangle** número **m** perteneciente a el conjunto **RectArea** número **n**. En caso de ausencia de uno o ambos tipos, se devuelve **NULL**.

CPoint* getCenter();

El usuario recibe el punto central del **CSprite**.

void replaceCenter(**CPoint*** c);

El usuario sustituye el punto central del **CSprite** por un nuevo **CPoint** creado por él definido por **c**. El punto central anterior es eliminado.

int addArea(**RectArea*** area);

El usuario añade un nuevo **RectArea** creado por él, y obtiene su identificador asociado.

int addRect(**int** area,**CRectangle*** rect);

El usuario añade un nuevo **CRectangle** creado por él a el conjunto **RectArea** asociado al identificador **area**.

void replaceArea(**int** n,**RectArea*** area);
void replaceRect(**int** area,**int** n,**CRectangle*** rect);

De manera similar a **replaceCenter** ambos métodos sirven para sustituir un conjunto anterior de **RectArea** o un **CRectangle** por otros del mismo tipo creados por el usuario. Tras ello, los antiguos valores serán eliminados.

int size();

Este método devuelve la cantidad total de **RectArea** almacenada en el **CSprite**.

Uint32 getPixel (**int** x , **int** y) ;

Este método funciona sólo para **CSprites** pertenecientes a un **CSpriteset** cargado con el parámetro **mode** con valor **WITH_SDL_SURFACE** o **ONLY_SDL_SURFACE**.

Consiste en devolver al usuario un valor asociado al pixel definido por el parámetro **x** e **y**, de acuerdo con la siguiente lógica :

```
Uint32 CImage::getPixel ( int x , int y ) {
   SDL_Surface* surface= m_pSurface.sdl_surf;
   if (surface && surface->w > x && surface->h > y ) {
     Uint32 color = 0 ;
     int position = y * surface->pitch +
                        surface->format->BytesPerPixel * x ;
     char* buffer = ( char* ) surface->pixels ;
     buffer += position ;
     memcpy ( &color , buffer , surface->format->BytesPerPixel ) ;
     return ( color ) ;
   } else {
     return 0;
   }
}
```

Se aprovecha la estructura **SDL_Surface**, para extraer de ella la información correspondiente a un pixel situado en las coordenadas **(x,y)** del búfer de pixels. El valor **0** corresponde a un pixel negro sin componente alpha.

int getWidth ( ) ;
int getHeight ( ) ;

Estos métodos devuelven las dimensiones ancho y alto respectivamente de el **CSprite** correspondiente.

void put (CPoint& ptDst , **Uint8** flags=0) ;

Este método es el dedicado a renderizar el **CSprite** que se está tratando. Su funcionamiento consiste en recuperar los efectos gráficos almacenados por el **CSprite** hasta ahora, transmitirlos a **CScreen**. Luego, transmitir un recurso gráfico del tipo **TR_CANVAS** a **CScreen** con la posición que debe tomar el **CSprite** (dicha posición es la suma del punto central del **CSprite** y **ptDst**). y por último, deshacer los efectos gráficos invocados para el **CSprite** a tratar, para no afectar la etapa de renderización de otros **CSprite**.

```
void CImage::put ( CPoint& ptDst, Uint8 flags) {

   … creación de recurso tipo TR_PUSHMATRIX y envío a CScreen …

      if (m_pSurface.sp_scale != 1.0) {
           … creación de recurso tipo TR_SCALATION y envio a CScreen …
      }

      //TRANSLATE

   … creación de recurso tipo TR_TRANSLATION con los parámetros
      provenientes de ptDst y envío a CScreen …

      // USER DEFINED EFFECTS IN

      for (list<SToRender*>::iterator iri = initRenderList.begin(),
                     ire = initRenderList.end();iri!=ire;++iri) {

           SToRender* irp = *iri;
           procRenders[irp->type](irp->pointer);
           delete irp;
      }

      initRenderList.clear();

      em = new SToRender;

      em->type = TR_CANVAS;
      em->pointer = (void*) new SRenderCanvas(m_pSurface,ptDst,flags);
      CScreen::graphicMaterial.push_back(em);

      for (list<SToRender*>::iterator eri = endRenderList.begin(),
                     ere = endRenderList.end();eri!=ere;++eri) {

           SToRender* erp = *eri;
           procRenders[erp->type](erp->pointer);
           delete erp;
      }
      endRenderList.clear();

      … creación de recurso tipo TR_POPMATRIX y envío a CScreen …
   }
```

La peculiaridad de este procesamiento radica en que sólo pretende afectar a la textura cuyo recurso gráfico se declara en la parte central del código. Por ello, todo el proceso está enmarcado entre un **TR_PUSHMATRIX** y un **TR_POPMATRIX**, para así no afectar al estado anterior de la matriz.

Los recursos **TR_SCALATION** y **TR_TRANSLATION** enviados al principio, sitúan la futura renderización de la textura en la posición que indican sus atributos actuales. Por otro lado, en el primer bucle **for** (amarillo) se recuperan recursos gráficos previamente almacenados por el **CSprite** para activarlos de cara a su renderización. Mientras que en el segundo bucle **for** (naranja) se recuperan los valores originales de componentes de color, pues las componentes de color no se modifican con **TR_PUSH/POPMATRIX**.

Las funciones ejecutadas en ambos bucles **for** (azul) son mucho más sencillas que las del mismo nombre que se invocaban en **CScreen::render**. Se limitan a ejecutar las funciones de efectos gráficos de la clase **CScreen** vistas anteriormente con los parámetros almacenados, y a eliminar a continuación dicha estructura.

```
int rotate(float angle, float x=0.0, float y=0.0, float z=1.0);
int translate(float x, float y, float z);
int scale(float x, float y, float z);
int color(float red, float green, float blue, float alpha);
int color(CColor* col,float alpha=1.0);
```

Estos son los métodos públicos para aplicar efectos al **CSprite**. Actúan de una manera idéntica a las funciones de efectos gráficos de la clase **CScreen**. Es decir, crean un recurso gráfico que almacene los parámetros ofrecidos por el usuario, y a continuación los almacena en este caso en el búfer de recursos gráficos que se recorre en el primer **for** del método **put**.

Los métodos de transformaciones de color tienen la peculiaridad de que también deben agregar un recurso gráfico al búfer de recursos gráficos del segundo **for** del método **put** para restablecer los valores originales de las componentes de color, como ya se explico anteriormente.

6.4.6 Gestión de Textos – ControlOutputText.h

CControlOutputText es la clase con la que se gestionan los textos mostrados por pantalla. Como **CControlImages**, se trata de una clase instanciable que sigue un patrón de instanciación única y que declara un objeto global. Dicho objeto global se llama **Write** y a través de sus métodos se trata toda la funcionalidad referente a los textos que posee esta biblioteca.

```
int Write.setfontSize(int newSize);
```

A través de este método, el usuario establece un tamaño de fuente por defecto. Es decir, se almacena el valor **newSize** en una variable miembro que es utilizada por otros métodos.

int Write.searchFont(const char* name,**int** withSize);
int Write.searchFont(const char* name);
int Write.searchFont(TTF_Font* fnt);
int Write.searchFont(**int** idtext);

Este conjunto de métodos busca obtener el identifiacor de una fuente a partir de diferentes tipos de datos.

- El primer método busca el identificador a partir del descriptor de la fuente **name** y su tamaño **withSize**.
- El segundo realiza la misma tarea, tomando por tamaño el valor por defecto en ese instante.
- El tercero realiza la búsqueda a partir de un puntero a recurso de fuente.
- Mientras el cuarto busca el identificador apropiado a través del identificador de un texto que use la fuente buscada.

int Write.loadFont(const char* fuente,**int** withSize);
int Write.loadFont(const char* fuente);

Estos métodos se encargan de la carga d e una fuente a través de su descriptor. Si no se especifica un tamaño, se tomará el tamaño por defecto en ese momento.

```
int CControlOutputText::loadFont(const char* fuente) {

 … inicialización de variables y búsqueda de una fuente con dicho id …

   if (ret < 0) { // Si la búsqueda no ha dado con la fuente.

     if (!lastIndexFontAdded.empty()) {
          ret=lastIndexFontAdded.back();
          lastIndexFontAdded.pop_back();
     } else {
          for (int i=0 ; ret < 0 ; i++) {
               if (Fonts.find(i)==Fonts.end()) {
                    ret = i;
                    break;
               }
          }
     }

     font_ttf = Fonts[ret] = new SFont;
     font_ttf->fuente =
     TTF_OpenFont(resource((s+".ttf").c_str()).c_str(),fontSize *
                                          Cscreen::getScaleX());
```

```
    if (font_ttf->fuente==NULL) {
          Clibrary::Error("Fallo al cargar",TE_fileExists);
          delete font_ttf;
          lastIndexFontAdded.push_back(ret);
          return FRACASO;
    }
    font_ttf->cadena=s;
    font_ttf->size=fontSize;
  } else {
    font_ttf=Fonts[ret];
  }

  ++countFonts[font_ttf];

  return ret;

}
```

De un modo similar a lo que ocurre con el método **CImg.add** , en este método se carga una única fuente por cada par descriptor-tamaño, contando las ocurrencias en caso de que se dichos pares se repitan en llamadas sucesivas al método.

La función encargada de generar la fuente en sí misma, es **TTF_OpenFont** y está preparada para soportar cualquier fichero de formato **.ttf**.

int Write.unloadFont(const char* fuente,**int** withSize);
int Write.unloadFont(const char* fuente);
int Write.unloadFont(**int** fuente);

Estos 3 métodos, como es de esperar realizan la función contraria a **Write.loadFont**. De la misma manera que **CImg.remove** retiraba **CSpritestes** de su colección, **Write.unloadFont** sólo libera los recursos de una fuente cuando su ocurrencia llega al valor **0**.

int Write.line(**int** fuente,**int** x,**int** y,const char* text,...);

Con dicho método, el usuario puede imprimir líneas de texto, además de una manera similar a la función **printf** de C. El primer parámetro sirve para indicar el identificador de la fuente a usar, los dos siguientes para establecer las coordenadas en pantalla donde dará inicio la línea de texto. Y a partir de ahí, nos encontramos con el mismo tratamiento de **printf**, donde una llamada del tipo :
'`Write.line(fuente,x,y,"ID fuente : %d, X: %d, Y: %",fuente,x,y);`' sería totalmente válida.

Dicho funcionamiento es posible gracias a la siguiente implementación :

```
int CControlOutputText::line(int fuente, int x,int y, const char*
text,...) {
   int ret = éxito;

   … precondiciones …

   if (Fonts.find(fuente) != Fonts.end()) {
      va_list lista;
      char buffer [1024];

      va_start (lista, text);
      vsprintf (buffer, text, lista);

      va_end (lista);

      … búsqueda de un identificador para el nuevo texto …
   }

   … creación de la estructura de la línea de texto …

   return ret;
}
```

Con las funciones de C, **va_start vsprintf** y **va_end** usadas tal y como aparece en el código, podemos copiar el tratamiento de **printf** a la perfección. Esto da una oportunidad de representar variables por pantallas muy cómoda para la mayoría de usuarios acostumbrados a C.

El valor de retorno de esta función es un identificador asociado al texto, al que habrá que remitirse cuando se quiera borrarlo o aplicarle un efecto.

Otro detalle importante a la hora de crear un texto, es que nada más ser creado, se asocia al **CEngine** actual. Así, sólo será posible la manipulación de dicho texto mientras el **CEngine** actual sea el mismo que aquel **CEngine** que estaba en ejecución en el momento de haber sido creado.

int Write.inBox(const char* file, **int** index);

Un tipo más complejo de texto se crea mediante este método. Gracias a el, se puede invocar un diálogo de texto, que se lee a partir de archivos XML creados para tal propósito. El formato de dichos archivos se puede consultar en el Anexo B del documento presente.

Del mismo modo, **Write.inBox** retorna un identificador que se trata del mismo modo que el identificador asociado a una línea de texto. Y también se asocia al **CEngine** actual.

int Write.erase(**int** text,**bool** nextframe);

Para borrar textos el usuario dispone de este método. Basta con indicar el identificador del texto a borrar, para que se elimine por completo y no vuelva a ser renderizado.

Además, **Write.erase** posee otras 2 funcionalidades. Cuando el parámetro **text** vale **ALL_TEXT**, en lugar de eliminar un único texto, la biblioteca elimina todos los asociados al **CEngine** actual. Y si **nextframe** vale **true**, el texto asociado al identificador **text** esperará para ser eliminado tras haber sido renderizado en lugar de eliminarse intantáneamente.

Se debe tener en cuenta que sólo es posible eliminar un texto si el **CEngine** actual es el mismo que el **CEngine** que era el actual cuando el texto indicado se creó.

int Write.locateRenderScene (**float** posx, **float** posy, **float** width, **float** height, **float** zoom) ;

Este método se define para que durante la etapa de renderización de textos, se llame a **CScreen::locateRenderScene** con los parámetros dados antes de comenzar a procesar la información gráfica asociada a los textos. Con ello se define la porción de la pantalla en la que los textos podrán ser representados en la pantalla.

Por defecto, define un marco de representación igual a la resolución del Contexto de Vídeo.

int Write.color(**int** text,**float** red, **float** green, **float** blue, **float** alpha, TypeColorTBox boxflags, **bool** persistent);
int Write.color(**int** text,CColor* col, **float** alpha,TypeColorTBox boxflags, **bool** persistent);

Estos métodos actúan de la misma manéra que sus homólogos en **CImg** y **CScreen**. La diferencia radica en que como por definición, todo texto creado por la biblioteca es blanco, éste es el único mecanismo para alterar el color de los textos. Y ya que el color blanco es transparente de cara la modificación del color, el color definido en éste método es exactamente el color que tomará el texto, a diferencia de lo que ocurre con las texturas.

Otra peculiaridad es que existen dos parámetros adicionales. El primero, **boxflags** es para especificar si se desea que la modificación de color se aplique al cuadro de diálogo entero, a las letras, o sólo al fondo. El segundo, **persistent**, es para indicar cuando vale **true** que se desea que la modificación del color actualmente definida siga presente en futuras etapas de renderización.

int Write.render();

El método render de **Write**, simplemente representa el estado actual de la gestión de textos. Eso quiere decir, que llama a la renderización de todos los textos asociados al **CEngine** actual para su tratamiento en **CScreen**.

Para ello, recorre las líneas de texto registradas y asociadas al **CEngine** actual, y accede a sus caracteres 1 por 1. Dichos caracteres son del tipo **CImage**, la clase base de **CSprite**, y por lo tanto cuenta con los mismos métodos de renderización. De este modo se invoca el método **color** si hubiera un efecto definido, y el método **put** de cada carácter.

Para el tratamiento de los diálogos de texto, se realiza un procedimiento similar, contando además con otra **CImage** que se trata la correspondiente al fondo del cuadro de diálogo.

Por último, se eliminan los textos asociados al **CEngine** actual que hayan sido programados para eliminarse tras la fase de renderización con la ayuda del método **Write.erase**. O en el caso de los diálogos de texto, mediante el XML de definición.

void Write.clear();

Cuando se llama a este método, la biblioteca se dispone a eliminar toda la información gráfica reservada en el contexto del objeto **Write**, restableciendo su estado al del comienzo de la ejecución del sistema.

6.4.7 Gestión del Ritmo de Ejecución – Time.h

CTime es la otra clase que se define como una clase de única instancia declarada como objeto global. Dicho objeto global se llama **Chrono**, y a través de sus métodos se accede a toda la funcionalidad destinada al control del ritmo de ejecución de la biblioteca.

int Chrono.setInterval(**int** msNew, **bool** all);
int Chrono.setFPS(**int** fpsNew, **bool** all);

Mediante estos 2 métodos, el usuario es capaz de establecer un ritmo de ejecución para su aplicación medido en unidades de tiempo. El primer parámetro indica la unidad de tiempo. Internamente, dichas unidades siempre se acaban traduciendo a milisegundos. El segundo, indica si el nuevo ritmo de tiempo deberá aplicarse a todos los **CEngine** de la biblioteca cuando **all** valga **true**, o sólo al **CEngine** actual, cuando valga **false**. Por defecto está a **false**.

```
int CTime::setInterval(int msNew, bool all) {

   if (!all &&
   admin != CLibrary::getActualEngine()) {
      admin = CLibrary::getActualEngine();
      actTime = & fc[admin];
   }
   int aux=0;

   if (all) {
      CTime::all = true;
      aux = allMsInterval;
      allMsInterval = msNew;
   } else {
      CTime::all = false;
      aux = actTime->msInterval;
      actTime->msInterval = msNew;
   }
   return aux;
}

int CTime::setFPS(int fpsNew, bool all) {

   int aux = setInterval(1000 / fpsNew, all);

   if (aux == FRACASO)
      return FRACASO;

   return (1000 / aux);
}
```

bool Chrono.isTimeForAll();

Devuelve **false**, si actualmente el control del tiempo se está haciendo a cada **CEngine** de forma individual y **true** en caso contrario.

int Chrono.nextFrame();

nextFrame es el método que se invoca para dar por finalizada la etapa de ejecución actual y comenzar una nueva. El usuario puede invocar dicho método siempre lo desee, pero hay que tener en cuenta que la biblioteca ya lo invoca durante la fase de espera y actualización del método **loop** de la clase **CEngine**.

Realiza la espera hasta que se cumpla el ritmo de tiempo establecido y además internamente llama a la función **CScreen::render**, es decir, actualiza la pantalla.

Su funcionamiento interno se describe en el siguiente cuadro :

```
int  CTime::nextFrame() {

   int ret = EXITO;

   if (!CLibrary::getLibrary()) {
     CLibrary::Error("Library not inicialized");
     return FRACASO;
   }

   if (admin != CLibrary::getActualEngine()) {
     admin = CLibrary::getActualEngine();
     actTime = & fc[admin];
   }

   if ( CScreen::render() == FRACASO )
     return FRACASO;

   if (all) {

     while ((actTime->msLast + allMsInterval) > SDL_GetTicks()) {
          SDL_Delay(1);
     }
     actTime->msLast = SDL_GetTicks();
     actTime->frameCount++;

   } else {

     while ((actTime->msLast + actTime->msInterval) > SDL_GetTicks()){
          SDL_Delay(1);
     }
     actTime->msLast = SDL_GetTicks();
     actTime->frameCount++;

   }
   return ret;
}
```

Es decir, una vez cargada la estructura propia de gestión del tiempo asociada al CEngine actual, se ejecuta la función **CScreen::render**. Y tras eso, se continua con un bucle **while** hasta que el ritmo de tiempo establecido se ha cumplido. Después, se aumenta el contador de frames.

Para esperar en el interior del bucle **while** se utiliza la instrucción '`SDL_Delay(1);`', la cual provoca la conmutación por otro proceso por parte del planificador del sistema operativo.

int Chrono.getTick();

Devuelve el valor de la cuenta de etapas asociada al CEngine actual.

6.4.8 Gestión del Multiverso – ControlMultiverse.h

Con **CControlMultiverse** hemos llegado a la última clase de instanciación única que se declara como un objeto global. Dicho objeto se llama **CMultiverse**, y a través de sus métodos gestionamos el contenedor de objetos **CUniverse** que facilita esta biblioteca.

En esta clase se asocian las colecciones de **CUniverses** a los **CEngine**, de modo que implícitamente cada vez que utilicemos un método de **CMultiverse**, estaremos tratando la colección de **CUniverses** asociada al **CEngine** actual.

CUniverse* add(**CUniverse*** uni,**Uint8** slot=0);

Funciona de manera similar a los otros contenedores anteriormente vistos. Pero esta vez, debido a que los objetos **CUniverse** son definidos por el propio usuario de la biblioteca, es necesario que se pase la instancia misma para que pueda ser registrado. Esto delega cierta responsabilidad en el usuario, pues se confía en que siempre que éste cree un nuevo **CUniverse**, lo agregue a **CMultiverse** antes de usarlo.

El parámetro **slot** permite que varios **CUniverse** con el mismo descriptor puedan existir al mismo tiempo, siempre que tengan un valor de **slot** diferente.

CUniverse* universeNamed(**string** uniName,**Uint8** slot=0);
CUniverse* universeNamed(**const char*** uniName,**Uint8** slot=0);

universeNamed es un método de búsqueda. Simplemente obtiene el CUniverse cuyo par **(descriptor,slot)** coincida con los parámetros de entrada.

void erase(**CUniverse*** uniKilled);

Cuando el usuario desee eliminar un **CUniverse**, deberá pasar la instancia de dicho **CUniverse** al método **erase.** De este modo, la biblioteca lanzará el destructor correspondiente y retirará el puntero correspondiente de la colección.

void clear();

clear elimina todos los **CUniverse** asociados al **CEngine** actual.

int size();

devuelve el número total de **CUniverse** pertenecientes a la colección asociada al **CEngine** actual.

```
UniverseCollection::iterator begin();
UniverseCollection::iterator end();
```

Con estos métodos, disponemos de un mecanismo para recorrer la colección de **CUniverse** asociada al **CEngine** actual mediante iteradores.

6.4.9 Gestión del Universo – Universe.h

CUniverse es una clase instanciable redefinible por el usuario. No se trata de una clase imprescindible para el correcto funcionamiento de la biblioteca, sin embargo, es un una buena práctica hacer uso de esta clase siempre que se pueda.

Para aplicaciones más elaboradas, disponer de una abstracción la lógica espacial mediante una clase derivada de **CUniverse** se vuelve altamente recomendable para mantener una correcta estructuración del código.

```
CUniverseTemplate(string name);
~CuniverseTemplate();
```

En el constructor por defecto simplemente se inicializan las variables miembros de la clase, como son **name**, **numCameras**, **loaded** o **slot**. Mientras que en el destructor se recorre la colección de CActor eliminándolos.

```
void load();
bool isLoaded();
void unload();
int changeUniverse(string name, Uint8 slot);
int incActor(CActor* act);
int decActor(CActor* act);
void incCameras();
void decCameras();
```

Estos métodos están mayormente vacíos. Sus cometidos son ser redefinidos por el usuario para introducir las funcionalides que él necesite. Para la redefinición se ofrece se debería consultar el Anexo A, donde se contienen algunas plantillas sobre las que basarse.

```
bool operator == (CUniverse& uni);
bool equal (CUniverse* uni);
```

Compara la variable miembro **name** de el **CUniverse** que se trata, con la variable miembro **name** de **uni**, y en el caso de coincidir devuelve **true**. En caso contrario **false**.

6.4.10 **Gestión de la Cámara – Camera.h**

CCamera es otra clase instanciable redefinible por el usuario. Su funcionalidad es la de facilitar la tarea de renderización, por tanto su uso es altamente recomendable.

```
CUniverse* getUniverse();
bool isOpened();
CActor* Target();
CRectangle* getArea();
```

Estos métodos devuelven los valores de las variables miembro correspondientes. **isOpened** vale **true** cuando la variable miembro **uni** no vale **NULL**.

```
int& CX();
int& CY();
```

Indican las coordenadas de la CCamera en el contexto que se haya definido. También se pueden modificar dichas coordenadas directamente ya que devuelven una referencia.

```
int setTarget(CActor* newTarget);
```

Establece un nuevo **CActor** como target para la **CCamera** que se está tratando.

```
int loadUniverse();
int unloadUniverse();
int resyncUniverse();
```

Métodos redefinibiles por el usuario que por defecto no incluyen ninguna instrucción. Su uso es conveniente para inicializar y finalizar el posible CUniverse sobre el que se pudiera estar actuando. Consultar la plantilla del Anexo A para más información al respecto.

```
int render();
```

Este es el método que se ha de invocar para iniciar la renderización de todos los elementos enfocados por **CCamera**. No es redefinible por el usuario, pero internamente llama a refresh, que sí lo es. En el siguiente cuadro se explica su implementación :

```
int CCamera::render() {

   rendering = true;

   for (list<SToRender*>::iterator iri = initRenderList.begin(),
        ire = initRenderList.end();iri!=ire;++iri) {

        SToRender* irp = *iri;

        procRenders[irp->type](irp->pointer);

        delete irp;

   }

   initRenderList.clear();

   int ret = refresh();

   for (list<SToRender*>::iterator eri = endRenderList.begin(),
        ere = endRenderList.end();eri!=ere;++eri) {

        SToRender* erp = *eri;

        procRenders[erp->type](erp->pointer);

        delete erp;

   }

   endRenderList.clear();

   rendering = false;

   return ret;
}
```

Este proceso de renderización es similar al que encontramos en el método **put** de **CSprite** . Consta de 3 partes : el primer bucle **for** (amarillo), la llamada a **refresh** (verde), y el segundo bucle **for** (naranja).

- En el primer bucle **for** se invocan las funciones que tratan los recursos gráficos registrados de una manera idéntica a la que veíamos en el método **put** de **CSprite**.
- La llamada a **refresh** delega la parte más importante de la renderización a un método redefinible por el usuario que inicialmente está vacío. Por tanto, es vital que el usuario lo redefina llamando a la renderización de los elementos enfocados por la cámara de la manera que crea conveniente. Como siempre, en el Anexo A vemos más información en la plantilla de **CCamera**.
- El segundo bucle **for**, como sucede en el método put anteriormente citado, se destina para restablecer las transformaciones de color que se hayan podido introducir como efectos en el primer bucle **for**.

Otra consideración a tener en cuenta es el booleano **rendering**. Una vez que **rendering** esté a true la llamada a los métodos de efectos gráficos propios de **CCamera** quedarán inutilizadas, por tanto, se ha de llamar a dichos métodos antes de usar el método **render** o después.

```
int refresh();
```

Como se ha comentado anteriormente, esta función es llamada por **render**, y debe ser redefinida por el usuario para que **CCamera** renderice correctamente.

```
int rotate(float angle, float x=0.0, float y=0.0, float z=1.0);
int translate(float x, float y, float z);
int scale(float x, float y, float z);
int color(float red, float green, float blue, float alpha);
int color(CColor* col,float alpha=1.0);
```

Estas son los métodos de llamada a efectos gráficos. Su planteamiento es idéntico al que vimos en CSprite, salvando el detalle referente al booleano **rendering** : Cuando **rendering** vale **true**, es inútil invocar estos métodos, por tanto hay que llamarlos mientras no esté ejecutándose el método **render**.

```
int locateRenderScene (float posx, float posy, float width, float height,float zoom);
int locateRenderScene ( CRectangle* areaSc, float zoom);
```

Este método en ambas variantes, sitúa un recurso gráfico de tipo **TR_LOCATION** en el inicio de la lista del primer for del método render, para así llamar a la función **CScreen::locateRenderScene**, en primer lugar durante la ejecución de dicho método. Además, también introduce un elemento **TR_PUSHMATRIX** a continuación de **TR_LOCATION** y otro **TR_POPMATRIX** al principio de la lista del segundo bucle for del método render (ejecutándose por tanto en último lugar).

De esta manera se salva y restablece la matriz una al principio y al final de las modificaciones realizadas por **CCamera**.

```
CCamera(CActor* target,CRectangle* area,CMessageHandler* pmhParent);
~CCamera();
```

Como de costumbre, el constructor y destructor están pensados para la inicialización de variables y recursos y la liberación de éstas. Pero en el caso del constructor además de define según el parámetro **area** la variable miembro también llamada **area**, que será la que por defecto dará los parámetros a **locateRenderScene** a no ser que el usuario especifique unos concretos.

int reubicate(CRectangle* nArea);

Con este método el usuario puede redefinir la variable miembro **area** de **CCamera**.

6.5.11 Gestión del Actor – Actor.h

```
CActorTemplate(CMessageHandler* pmhParent);
~CActorTemplate();

CSprite* getSprite();
int move();
string getCreature();
int setUniverse(CUniverse* m);
CActor* clone();
```

Esta clase está pensada para ser totalmente redefinida por el usuario. Su correcta implementación debe ajustarse a la plantilla que aparece en el Anexo A del documento presente.

Capítulo 7 : Pruebas del Sistema

En este capítulo encontraremos :

- Una breve **introducción**.
- Una serie de pruebas, clasificadas en pruebas de **estrés**, de **funcionalidad** y de **errores**.

7.1 Introducción

Someter a pruebas a esta biblioteca es una tarea relativamente sencilla. El procedimiento a usar será el de llamar a ciertas interfaces públicas y detectar si se han producido errores, o en última instancia si se causa la inestabilidad del sistema.

Ante una implementación de este tamaño (15000 líneas), es importante procurar que haya facilidad para depurar el código. Para ello, primero se ha tenido muy presente durante la implementación, que en la medida de lo posible, hay que usar instrucciones que ante un uso ilícito cuenten como error en tiempo de compilación.

Otra medida para facilitar la depuración ha sido la inclusión de instrucciones adicionales activadas por definiciones de precompilación.

7.2 Pruebas

Las pruebas se clasificarán en 3 tipos, de estrés, de funcionalidad y de errores. En adelante se trata cada caso con detenimiento.

7.2.1. **Pruebas de Estrés** :

En este tipo de pruebas se ejecutan repetidamente diversas funciones de la biblioteca con el objetivo de comprobar una buena respuesta de éstas en el tiempo.

- Como primera prueba de estrés, se creará un bucle dedicado a crear y acto seguido destruir objetos tipo CSpriteset a través de CImg.add y CImg.remove .

- A continuación se hará el mismo procedimiento pero para cargar fuentes, con Write.loadFont y Write.unloadFont.

- Tras esto, se creará otro bucle donde se creen y elimen textos instantáneamente.

- Otra prueba de estrés, consistirá en renderizar un mismo CSprite* muchas veces en una misma etapa de ejecución.

Una vez llegado a este punto, se realizarán pruebas dentro de un contexto más complejo. Se ha desarrollado un conjunto de clases que utilizando esta biblioteca lanzan una demo de lo que podría ser un juego. Dicho juego posibilita hacer migraciones de CUniverse, y representar por pantalla mapas conformados por piezas de imágenes individuales llamadas tiles[17].

17 Un **tile** es un fragmento de imagen que actúa como pieza de estructural básica para formar fondos complejos. El tileado es una técnica común en videojuegos para ahorrar en memoria y en trabajo artístico.

- En este contexto, la primera prueba consistirá en hacer un gran número de migraciones seguidas.

- La última prueba, consistirá en ejecutar un CUniverse que contiene muchos actores, aproximadamente 3000, que se mueven individualmente.

El código correspondiente a las 4 primeras pruebas está presente en la función **funcEstres** que se encuentra en el archivo **main.cpp**.

Figura 7.1 : Muestra de la prueba de estrés de actores (3000 en pantalla).

Las 2 últimas pruebas anteriormente descritas, son los tests que se añaden a la biblioteca mediante **addEngine** con la variable **priority** más baja, por tanto se ejecutan los primeros. Siendo ejecutado antes el test de universos, y después el de actores.

7.2.2. **Pruebas de Funcionalidad** :

Consisten en explorar las funcionalidades de la biblioteca en busca de comportamientos no esperados.

En concreto, se va a experimentar principalmente con las interfaces referentes a los efectos gráficos. Se programarán 5 tests :

- En el primero se prueba la convivencia de 4 cámaras que tienen criterios de seguimientos a su actor objetivo diferentes.
- En el segundo se ha implementado un pequeño juego de 2 jugadores donde el objetivo es que un jugador elimine más enemigos que el otro. Aquí se pone a prueba la interacción de los actores entre ellos y el universo. A cada enemigo se le aplica el efecto gráfico de transparencia (transformación de **color** únicamente en la componente **alpha**) durante cada etapa de ejecución.
- En el tercero se experimenta con otra resolución, con los eventos entrantes al ratón, con la importación de gráficos sin .grd asociados y con rotaciones y traslaciones.
- En el cuarto se usan transformaciones de color en diferentes cámaras.
- En el quinto se usan rotaciones. Debido a estas rotaciones, se puede observar como se monta el fondo dinámicamente a través de los tiles.

Figura 7.2 : izquierda, **transformacion** de color, derecha, **rotacion** sobre vectores X e Y.

Además, los cuadros de diálogo son otra funcionalidad que constantemente se prueba en esta biblioteca.

7.2.3. **Pruebas de reconocimiento de Errores** :

Para este tipo de pruebas, se utilizan las funciones de la biblioteca de manera diferente a la especificada con la intención de intentar provocar fallos.

- Tratando de engañar a los contenedores : Podemos tratar de llamar a CImg.get o Write.line pasando identificadores que sabemos que no están asignados. La consecuencia es simplemente un valor devuelto de retorno como NULL en el primer caso y FRACASO en el segundo, más un error que puede ser recogido mediante las funciones para dicho propósito de CLibrary.

```
Int s = Cimg.size() + 4;
CSpriteset* set;
if ( (set = Cimg.get(s)) == NULL )
  printf("\n%s\n",CLibrary::popError().c_str());

Cimg.get(s)->get(0); // ERROR
```

Este cuadro también revela el peligro que entrañan algunas funciones de esta biblioteca. Cada interfaz que devuelve un puntero, es posible que devuelva **NULL**, por tanto la práctica de no directamente llamar un método de un puntero recién devuelto puede causar un cuelgue de la aplicación.

Anexo A : Plantillas de la Biblioteca

A.1 Introducción

Con la intención de facilitar el uso de esta biblioteca para el Usuario, se han elaborado una serie de plantillas de las clases con métodos redefinibles. El propósito es garantizar una correcta implementación de los métodos y ahorrar el tiempo del usuario que no tiene porqué conocer los detalles más íntimos de la biblioteca.

A.2 Plantilla de clase derivada de CEngine

En esta clase además, se incluye la plantilla de 2 métodos correspondientes a **CMessageHandler**, que encontraremos en último lugar.

CEngineTemplate.h :

```
#ifndef __ENGINE_TEMPLATE__
#define __ENGINE_TEMPLATE__

#include "Engine.h"

    class CEngineTemplate : public CEngine {
    public:

/* Constructor del Motor. */
       CEngineTemplate(CMessageHandler* pmhParent=NULL);

/* Destructor, ahí deberemos liberar todos los recursos. */
       virtual ~CEngineTemplate();

/* Opcional, pero invocado por la Librería. Debe devolver 'true' para que la librería
pueda tomar el motor como "Motor Activo" ( CLibrary::getActualEngine(); )
 En este método se pueden reservar recursos necesarios para la ejecución de lmotor.*/
       int onInit();

/* Opcional, pero invocado por la Librería. En este método se pueden liberar recursos que
eran necesarios para la ejecución de lmotor. */
       int onExit();

/* Función necesaria para el procesamiento de Eventos por parte de este Motor. */
       int loop();

/* Rutina principal. Aquí se debe incluir la implementación principal del motor para
actualizar su estado. */
       int onIdle();

/* Opcional. Pensado para resolver conflictos ante inesperadas conmutaciones entre
diferentes Motores. */
       virtual void deselect();

/* Rutina de renderización. Aquí deberían invocarse todas las rutina de renderización de
los elementos controlados por el Motor. */
       int drawFrame();

/*A continuación, se definen 2 métodos hereados de la clase CMessageHandler.

 Dichos métodos también están presentes en CActor, CUniverse y CCamera.

Pueden ser redefinidos para gestionar mensajería en el objeto actual.*/
```

```
/* Este método gestiona los mensajes en el instante que se llama a "SendMessage", método
definido en CMessageHandler. */
        int onMessage(Uint32 MsgID,MSGPARM Parm1,MSGPARM Parm2);

/* Este método gestiona los mensajes no instantáneos. Se leen los mensajes tras llamar al
método "readMessages" definido en CMessageHandler. */
        void pendingMessage(Uint32 MsgID,MSGPARM Parm1,MSGPARM Parm2);
    };


#endif
```

CEngineTemplate.cpp :

```
#include "EngineTemplate.h"

    CEngineTemplate::CEngineTemplate(CMessageHandler* pmhParent) : CEngine(pmhParent) {

       // TODO
    }

    CEngineTemplate::~CEngineTemplate() {

       // TODO
    }


    int CEngineTemplate::drawFrame() {

       // TODO : Fase de renderización del Motor.

       // Ejemplo de renderización simple suponiendo que disponemos de un Conjunto de
Cámaras

       /*for (int i=0;i<camaras.size();i++) {

              camaras[i]->locateRenderScene();
              camaras[i]->render();

       }

       Write.refresh();*/

       return EXITO;
    }

    int  CEngineTemplate::onInit() {

       // TODO : Inicialización de recursos no cargados necesarios en 'onIdle'

       return EXITO;

    }

    //idle. Main loop.
    int CEngineTemplate::onIdle(){

       // TODO : Lógica principal del Motor.

       return EXITO;

    }
```

```
int CEngineTemplate::onExit() {

   // TODO : Liberación de recursos no necesarios mientras no haya ejecución de
'onIdle'.

   return EXITO;

}

void CEngineTemplate::deselect() {
   CEngine::deselect();

   // TODO : Siempre se debe llamar previamente al método de la clase base.

}

int CEngineTemplate::loop() {

   // TODO : Siempre se debe llamar posteriormente al método de la clase base.

   return CEngine::loop();
}


int CEngineTemplate::onMessage(Uint32 MsgID,MSGPARM Parm1,MSGPARM Parm2) {

   // TODO : Gestión de mensajes

   return EXITO;

}


void CEngineTemplate::pendingMessage(Uint32 MsgID,MSGPARM Parm1,MSGPARM Parm2) {

   // TODO : Gestión de mensajes

}
```

A.3 Plantilla de clase derivada de CCamera

CameraTemplate.h :

```
#ifndef __CAMERA_TEMPLATE__
#define __CAMERA_TEMPLATE__

#include "Camera.h"

class CCameraTemplate : public CCamera {
private:

   // TODO : funciones y variables privadas

public:

   // TODO : variables y funciones públicas adicionales

   /* Constructor, se recomienda que se establezcan los siguientes parámetros.
    Pueden ser necesarios para inicializar la clase base. */
   CCameraTemplate(CActor* target,CRectangle* area,CMessageHandler* pmhParent=NULL);
```

```
       // Destructor, ahí deberemos liberar los recursos
       virtual ~CCameraTemplate();

       /* Opcional. Este método está pensado para usarse cada vez que se enfoca un
Universo por primera vez. */
       int loadUniverse();

       /* Opcional. Este método está pensado para usarse cuando la cámara deja de usar un
Universo. */
       int unloadUniverse();

       /* Opcional. Este método está pensado para recargar los recursos gráficos
provenientes del Universo si fuera necesario. */
       int resyncUniverse();
       /* La implementación por defecto de los 3 métodos anteriores simplemente devuelve
ÉXITO.

          Opcional. Fija el objetivo de la cámara. La implementación por defecto está
comentada
        junto al resto de la implementación de la clase. */
       int setTarget(CActor* newTarget);

       /* Método que se llama desde el método de la clase base "render". Se debe encargar
de actualizar los elementos gráficos.*/
       int refresh();

    };


#endif
```

CameraTemplate.cpp :

```
#include "CameraTemplate.h"
    #include "Library.h"

    // Library.h es necesario para el manejo de errores.

    CCameraTemplate::CCameraTemplate(CActor* target, CRectangle* area,CMessageHandler*
pmhParent) :
    // Se ha de llamar a la clase Base para una correcta inicialización
    CCamera(target,area,pmhParent) {

       // TODO

    }

    CCameraTemplate::~CCameraTemplate() {

       // TODO

    }

    int CCameraTemplate::loadUniverse() {

       if (uni!=NULL) {

       // TODO

       }

       return EXITO;
```

```
    }

    int CCameraTemplate::unloadUniverse() {

       if (uni!=NULL) {

       // TODO

       }

       return EXITO;

    }

    int CCameraTemplate::resyncUniverse() {

       /*           IMPLEMENTACIÓN POR DEFECTO
       /*
       if (unloadUniverse() == EXITO)
              return loadUniverse();
       else
              return FRACASO;
       */

       // TODO

       return EXITO;
    }

    int CCameraTemplate::setTarget(CActor* newTarget) {

       /*           IMPLEMENTACIÓN POR DEFECTO */

       /*if (newTarget == this->target) {
              CLibrary::Error("Actor objetivo ya establecido");
              return FRACASO;
       }

       if (newTarget->getUniverse()!= this->target->getUniverse()) {
              this->target=newTarget;
              resyncUniverse();
       } else {
              this->target=newTarget;
       }

       CX()=CY()=-1000; // Truco usado para forzar una recalibración de las coordenadas
de la cámara en su primer uso.
       */

       // TODO

       return EXITO;
    }

    int CCameraTemplate::refresh() {

       // TODO : Aquí debería ir las rutinas que invocan el proceso de renderización de
cada elemento dentro del contexto de la cámara.

       return EXITO;
    }
```

A.4 Plantilla de clase derivada de CUniverse

CUniverseTemplate.h :

```
#ifndef __UNIVERSE_TEMPLATE__
    #define __UNIVERSE_TEMPLATE__

    #include "Universe.h"

    class CUniverseTemplate : public CUniverse {
    private:

       // TODO : funciones y variables privadas

    public:

       // TODO : variables y funciones públicas adicionales

       // Constructor. La cadena name es el descriptor del Universo. Para la gestión de
Universos, debería usarse el objeto global CMultiverse.
       // Cuando se crea un Universo, debería añadirse su instancia inmediatamente a
CMultiverse mediante CMultiverse.add(CUniverse*)
       CUniverseTemplate(string name);

       // Destructor, ahí deberemos liberar los recursos. Si queremos que sólo pueda
invocarse a través de CMultiverse,
       // se recomienda mover el destructor a la sección 'private', y seguidamente añadir
la línea "friend class CControlMultiverse;".
       virtual ~CUniverseTemplate();

       // Opcional. Método dedicado para la inicialización de los recursos del universo.
       void load();

       // Opcional. Método dedicado para la finalización de los recursos del universo.
       void unload();

       // Opcional. Método que sustituya las características de este Universo por las de
otro compatible a través de un nuevo descriptor.
       int changeUniverse(string name, Uint8 slot);

       // Opcional. Método para notificación de la entrada de un CActor.
       int incActor(CActor* act);

       // Opcional. Método para notificación de la salida de un CActor.
       int decActor(CActor* act);

       // Opcional. Método para notificación de la entrada de una CCamera.
       void incCameras();

       // Opcional. Método para notificación de la salida de una CCamera.
       void decCameras();

    };

    #endif
```

CUniverse.Template.cpp :

```
#include "UniverseTemplate.h"

    CUniverseTemplate::CUniverseTemplate(string name) : CUniverse(name) {
```

```
	//TODO

}

CUniverseTemplate::~CUniverseTemplate() {

	if (isLoaded())
		unload();

}


void CUniverseTemplate::load() {

	// TODO

	loaded=true;
}

void CUniverseTemplate::unload() {

	// TODO

	loaded=false;

}

int CUniverseTemplate::changeUniverse(string name,Uint8 slot) {

	// TODO

	return EXITO;

}

int CUniverseTemplate::incActor(CActor* act) {

	/*		IMPLEMENTACIÓN POR DEFECTO	*/

	/*if (!act) {
		CLibrary::Error("Puntero a CActor nulo");
		return FRACASO;
	}

	CUniverse* u = act->getUniverse();

	if (u)
		for (ActorCollection::iterator it = u->actorBegin(), jt = u->actorEnd();it!
=jt;++it)
			if (act == *it) {
				CLibrary::Error("Actor actualmente incluido en otro Universe");
				return FRACASO;
			}*/

	// TODO

	return EXITO;
}

int CUniverseTemplate::decActor(CActor* act) {

	/*		IMPLEMENTACIÓN POR DEFECTO	*/

	/*if (!act) {
```

```
            CLibrary::Error("Puntero a CActor nulo");
            return FRACASO;
      }

      for (ActorCollection::iterator it = actorBegin(), jt = actorEnd();it!=jt;++it)
            if (act == *it) {
                  act->setUniverse(NULL);
                  actor.remove(act);
                  return EXITO;
            }

      CLibrary::Error("Actor actualmente no incluido en este Universe");
      return FRACASO;*/

      // TODO

      return EXITO;
   }

   void CUniverseTemplate::incCameras() {

      // POR DEFECTO : numCameras++;

      // TODO

   }

   void CUniverseTemplate::decCameras() {

      /*          IMPLEMENTACIÓN POR DEFECTO       */

      /*numCameras--;
      if (numCameras==0)
            if (isLoaded())
                  unload();*/
   }
```

A.5 Plantilla de clase derivada de CActor

CActorTemplate.h :

```
#ifndef __CACTOR_TEMPLATE__
    #define __CACTOR_TEMPLATE__

    #include "Actor.h"


    class CActorTemplate : public CActor
    {
    private:

       // TODO : funciones y variables privadas

    public:

       // TODO : variables y funciones públicas adicionales

       // Constructor con clase menajera como parámetro de entrada
       CActorTemplate(CMessageHandler* pmhParent=NULL);

       // Destructor, ahí deberemos liberar los recursos
```

```
	~CActorTemplate();

	// Devuelve un CSprite. Definición opcional. Ya tiene la siguiente
implementación :
	//
	//	CSprite* CActor::getSprite() {
	//		return CImg.get(file)->get(graph);
	//	}
	//
	// Sólo definir si se desea hacer una operación diferente
	CSprite* getSprite();

	// Método pensado para ser invocado en CEngine::onIdle.
	// Deberá actualizar el estado del CActor cuando sea necesario.
	int move();

	// Opcional. Debería devolver una cadena que identificara de algún modo al actor.
	// Su implementación por defecto devuelve la cadena "criature" que debemos
introducir
	// en la implementación del Constructor (ver en la parte de implementación).
	string getCreature();

	// Opcional. Su implementación por defecto es una simple asignación a la variable
miembro CUniverse* inUniverse.
	int setUniverse(CUniverse* m);

	// Opcional. Por si se quiere implementar una función que facilite la
inicialización de muchos actores dle mismo tipo.
	// Si no se le da una implementación y el método es invocado, devuelve NULL y
registra un error.
	virtual CActor* clone();

};



#endif
```

CActorTemplate.cpp :

```
#include "ActorTemplate.h"

CActorTemplate::CActorTemplate(CMessageHandler* pmhParent) :
// Se ha de llamar a la clase Base para una correcta inicialización
CActor("valor creature",pmhParent) {

	// TODO

}
CActorTemplate::~CActorTemplate() {

	// TODO

}

int CActorTemplate::move() {

	// TODO

	return EXITO;
}
```

```
string CActorTemplate::getCreature() {

   // TODO

   return string("cadena personalizable");

   // CActor::getCreature() accede al valor que hemos configurado en el Constructor.
}

int CActorTemplate::setUniverse(CUniverse* m) {

   // TODO

   return EXITO;

   // CActor::setUniverse(m) realiza la asignación : inUniverse=m;
}

CSprite* CActorTemplate::getSprite() {

   CSprite* spt;

   // TODO

   return spt;
}

CActor* CActorTemplate::clone() {
   CActor* act;

   // TODO

   return act;
}
```

Anexo B : Estructuras XML

B.1 GRD : El formato para importar gráficos.

Como se ha explicado a lo largo de la documentación, la biblioteca está preparada para importar información adicional de las imágenes que no tenga que ver directamente con información gráfica.

El medio para realizar esta funcionalidad, es mediante **GRD**. Se trata de un archivo **XML**, que especifica como se debe importar y almacenar la imagen que comparte el mismo descriptor que el **GRD** (salvando la extensión, obviamente). Mediante este archivo, se puede configurar un **CSpriteset** de múltiples maneras, ya que cada archivo que se importa, se convierte directamente en un objeto de tipo **CSpriteset**.

Según la lógica de la biblioteca, es de esperar que cada **CSpriteset** tenga varios **Csprites**, pero eso no puede ser posible si no hay un **GRD** que haya especificado tal procesamiento. Es por eso, que cuando se intenta abrir una imagen que no tiene sociado ningún **GRD**, se crea un **CSpriteset** con un único **CSprite**.

A continuación vemos un ejemplo de un fichero **GRD** que genera un **CSpriteset** de tiles.

```
<Spriteset sprites='8' cellwidth='32' cellheight='32' simple='true' sp-scale='2.0'>
</Spriteset>
```

Con esta estructura, simplemente se ha indicado que cada CSprite debe tener por dimensiones 32 de ancho, y 32 de alto. Y que se pueden sacar 8 sprites del fichero asociado. Esto provocará que el procesamiento cumpla tal requisito y fragmente la imagen según la dimensión especificada.

Otro **GRD** un poco más complejo define la animación del **CActor** que se suele usar como objetivo de las cámaras en las Pruebas :

```
<Spriteset  sprites='21'  cellwidth='32'  cellheight='32'  sp-
scale='2.0'>
     <globalareas>
          <area id='0' relative='true'>
         <rectangle x1='-6' x2='6' y1='-6' y2='6'/>
          </area>
          <area id='1' relative='true'>
               <rectangle x1='-6' x2='6' y1='-20' y2='6'/>
          </area>
     </globalareas>
     <img id='0' name='down0' width='17' height='31'>
          <cpoint x='7' y='27'/>
     </img>
     <img id='1' width='18' height='30'>
          <cpoint x='8' y='27'/>
     </img>
     <img id='2' name='down2' width='19' height='30'>
          <cpoint x='8' y='27'/>
     </img>
     <img id='3' name='down3' width='20' height='29'>
          <cpoint x='9' y='27'/>
     </img>
     <img id='4' name='down4' width='20' height='29'>
          <cpoint x='9' y='27'/>
     </img>
     <img id='5' name='down5' width='19' height='30'>
          <cpoint x='9' y='27'/>
     </img>

     … hasta 20 …

</Spriteset>
```

En este caso, se debe definir detalladamente cada fragmento de la imagen, pues las dimensiones son diferentes en cada uno de ellos. Además se definen puntos centrales, que se usan por la biblioteca para calcular los espejados. El resto de información son las áreas, y éstas no son procesadas por la biblioteca en ningún momento. Simplemente sirve como mecanismo para definir una estructura interna que el usuario quiera asociar al gráfico por diversas razones.

Las áreas de este caso en concreto, se usan para calcular las durezas de los **CSprite** y por tanto, las colisiones. Pero tal procedimiento es totalmente ajeno a la lógica de la biblioteca.

B.2 El archivo config.xml :

Como se ha visto en el Capítulo de Implementación, este archivo se inspecciona si se elige inicializar la biblioteca a través del método **startLibrary** con el parámetro **xmlconfig** a **true**. Su estructura es muy simple, simplemente consta de los siguientes campos imprescindibles :

```
<System>
  <mode width="640" height="480" bpp="32"/>
  <fullscreen option="no"/>
</System>
```

En ellos, como se puede deducir, se especifica cómo se desearía iniciar la pantalla automáticamente. La ventaja que ofrece este mecanismo, es que no hace falta volver a compilar el código cada vez que se quiera tocar la resolución de la pantalla.

Las demás etiquetas que aparecen en la versión **config.xml** del directoro **resources** no son formalmente miembros de **config.xml**. Es decir, la biblioteca los ignora, aunque quedan a disposición de la lógica de los test. Como en el caso que se ha implementado como prueba, que utiliza los campos adicionales definidos en **config.xml** para sus propósitos.

B.3 Los Diálogos de Texto :

Por último, vamos a analizar el **XML** que usa el cuadro de diálogo de prueba que aparece en los tests :

```
<TextList>
    <text id="0" msg="Bienvenido al Test de diálogos!" x="20" y="20" w="120" next="1"/>
    <text id="1" msg="Esto es un diálogo de texto, al que le aplico una transparencia." x="20" y="20" w="120" next="2"/>
    <text id="2" msg="Estos diálogos están definidos en el archivo 'texts0.xml' de la carpeta 'resources'" x="20" y="20" w="120" next="3"/>
    <text id="3" msg="Cuando termina un diálogo, comienza el siguiente según el parámetro 'next'" x="20" y="20" w="120" next="4"/>
    <text id="4" msg="Es muy cómodo tratar los textos gracias a XML." x="20" y="20" w="140" next="5"/>
    <text id="5" msg="FIN." x="40" y="40" w="50" next="0"/>
    <text id="6" msg="Este texto no sale porque no lo llamo." x="150" y="80" w="160" next="7"/>
    <text id="7" msg="Pero bastaría llamar a la función Write.inBox pasándole los parámetros apropiados'." x="150" y="180" w="160" next="8"/>
    <text id="8" msg="Pruebalo!" x="150" y="80" w="160" next="6"/>
</TextList>
```

Como vemos, la definición de cada apareición de un cuadro de diálogo es muy simple. Tenemos el parámetro **id**, que se necesitará para invocar al método **Write.inBox** , y luego en **msg** va la cadena de texto, **x**, e **y** indican la posición donde empieza la cadena, mientras que **w** delimita la longitud límite.

El último parámetro es **next**, que sirve para enlazar un diálogo con otro, formando una cadena de diálogos como se puede apreciar en los tests de prueba.

La modificación de este archivo durante la ejecución produce resultados sin tener que reiniciar la aplicación, ya que en este caso, cada vez que surge un nuevo diálogo de texto se accede a este fichero porque la cadena es infinita.