Universidad de las Ciencias Informáticas

Facultad 5

Entornos Virtuales

Título: Biblioteca de Efectos de Transición de Pantallas para las Aplicaciones de Realidad Virtual.

Trabajo de Diploma para optar por el Título de Ingeniería en Ciencias Informáticas.

Autor(es): Reinier Jesús Garcia Abrahantes.

Dariel Leyva Cabrera.

Tutor(es): Ing. Yirka Céspedes Boch.
Ing. Yoander Cabrera Díaz.

Co-tutor: Ing. Frank Puig Placeres.

Ciudad de La Habana, Julio 2008

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los _____ días del mes de _____ del año _____.

Autores:

Dariel Leyva Cabrera Reinier Jesús Garcia Abrahantes

_____ _____

Tutores:

Yirka Céspedes Boch Yoander Cabrera Díaz

_____ _____

DATOS DE CONTACTO

Nombre y Apellidos: Yirka Céspedes Boch.
Edad: 24
Ciudadanía: cubano.
Categoría instructor recién graduado de Ingeniero en Ciencias Informáticas. Actualmente cursa los Diplomados de Realidad Virtual, Teoría y Técnicas de Dirección.
E-mail: ycespedes@uci.cu

Nombre y Apellidos: Yoander Cabrera Díaz.
Edad: 25 años.
Ciudadanía: cubano.
Categoría instructor recién graduado. Graduado de Ingeniero en Ciencias Informáticas. Actualmente cursa el Diplomado de Realidad Virtual, líder del proyecto Juegos de Consola en la Universidad de las Ciencias Informáticas.
E-mail: ycabrerad@uci.cu

Nombre y Apellidos: Frank Puig Placeres.
Edad: 26
Ciudadanía: cubano.
Graduado de la UCI, con diez años de experiencia en el tema de la Gráfica Computacional, Coautor de los libros ShaderX5, Game Programming Gems 5, Game Programming Gems 6 y AI Wisdoms 4.
E-mail: fpuig@uci.cu

AGRADECIMIENTOS

A mi familia por su apoyo y confianza en mí, en especial a mis padres por la formación que me han dado.

A mi hermana Katia por darme apoyo en todo momento.

A mi primo y hermano mayor Victor por su constante preocupación y apoyo, a su esposa Dianelis, a Ela, a mi tío Oslirio, a mis hermanitos pequeños Victor y Yanet, por estar junto a mí en todo este tiempo, a demás familiares que han contribuido en la realización de este trabajo.

A mis tutores Yirka y Yoander, por su apoyo e incondicionalidad, por siempre estar cuando los he necesitado.

A mis amigos Mario, Fabienny, Juan y otros del grupo por compartir sus ideas, sugerencias y contribuir en la realización de esta difícil tarea.

A Frank Puig, por sus sugerencias muy oportunas.

A todos los compañeros de la universidad, por su preocupación y apoyo.

A nuestro Comandante en Jefe por ser el autor de esta gran obra maestra que es la UCI, gracias a él y a la Revolución he podido estudiar esta carrera.

Reinier Jesús Garcia Abrahantes.

Agradezco a todas aquellas personas que de una forma u otra contribuyeron en el desarrollo de este trabajo.

En especial:

A Yirka y Yoander, por haber sido parte de esta solución.

A Frank Puig por sus buenas ideas y sugerencias.

A Matos y Yisel, por su ayuda incondicional.

A todos los amigos y hermanos de la Universidad, del grupo, piquete y apto, por su preocupación y constante apoyo.

A todos ustedes, gracias.

Dariel Leyva Cabrera.

DEDICATORIA

De Reinier :

A todas las personas especiales para mí, principalmente a mis padres Angela y Jesús, por ser mi principal fuente de inspiración, por ser éste su gran sueño se ha cumplido, por guiarme por el buen camino, a mi hermana por su amor e incondicionalidad, por traer al mundo un niño tan lindo como lo es Orlandito.

A Ela, quien me ha acogido como un hijo cuando más lo he necesitado, a mi primo y hermano mayor Victor y su esposa Dianelis por su apoyo y consejos, a sus hijos Victor (Muñeco) y Yanet y mi sobrina, de los que me corresponde ser su guía y ejemplo en sus estudios para que continúen y lleguen a ser profesionales y me dediquen sus tesis, a mi tío Oslirio por sus sabios consejos, a Ustedes en especial por estar y apoyarme en todos los momentos buenos y malos de mi vida, a mi familia en general.

A todos mis profesores desde los inicios en la primaria hasta la universidad, por formar parte de mi formación.

A mis tutores Yirka y Yoander por su gran ayuda en la realización de este trabajo para que tenga la calidad que se requiere.

A todos mis amigos.

A la Revolución, al Comandante en Jefe Fidel y a la Universidad de las Ciencias Informáticas.

Reinier Jesús Garcia Abrahantes.

De Dariel:

A mis padres Martha y Rafael por su apoyo, cariño, fuerza y faro de guía en todo momento, ya todos tenemos un sueño cumplido, mi formación es premio de su dedicación, el árbol sigue creciendo bajo sus sombras.

A mis hermanos Roberto, Carlos y Yeni por siempre estar junto a mí y ser fuente de inspiración en cada avanzar del camino, que sirva de ejemplo para mi hermano Carlos, a mi sobrino Dayron (el pequeño gigante), mi prima Sheila y Samantha.

A mis abuelas Margot, Mercedes y Lucila, tíos(as) Ubaldo, Ileana, primos y demás familiares que también han sido parte de este sueño, y su confianza fue estímulo del resultado final.

A mi tío Albertiñi y familia por ser código abierto a todo consejo y ayuda, este anhelo es parte de ustedes también.

A mis tías o segundas madres en este mundo (Mercedita, Mercedes, Meme, Lilia, Cucha) por apoyarme estos 5 años y estar al lado mío en todo momento, su dedicación, confianza, y cariño han sido cauce en el logro de este sueño, gracias por estar en los momentos que siempre necesité.

A mi primo Emilio más que primo hermano, por su apoyo incondicional, y por ser amigo en todo momento.

A Yuliesky (el negro) eres parte de este logro por tu preocupación y apoyo, pronto veremos cumplir tu sueño.

A Teresa y familia por su confianza en todo momento.

A todos los amigos que son muchos sin dejar a nadie afuera, que siempre han estado al tanto de mi desempeño en los estudios y que siempre me han dado ánimo para seguir adelante y muchos consejos.

A mis compañeros de grupo.

Al piquete de friendlys, la hermandad sigue en lo adelante.

Dariel Leyva Cabrera.

"El conocimiento en la mayoría de los que lo cultivan, es una especie de moneda, que se estima en mucho, pero que solo contribuye a nuestro bienestar en la medida en que se comunica…"

Rousseau

RESUMEN

Mediante el estudio de una importante rama de la computación denominada "Realidad Virtual" (RV), se ha tratado de apoyar el proceso de formación del individuo, ya que esta estimula de manera considerable el proceso de aprendizaje por su capacidad de proveer entornos virtuales creíbles.

La RV se ha expandido considerablemente a diferentes sectores en estos últimos años, puede encontrarse en aplicaciones militares, la medicina, entretenimiento, ofreciendo servicios de manera interactiva y segura. Esto ha provocado la evolución del hardware, y del software usado en sistemas de visualización, simulación y efectos de transiciones que se pueden realizar con imágenes o plano para dar realismo.

Con la introducción de las transiciones en el área de los efectos el usuario queda sumergido en un espacio virtual en el que se le hace perceptible, por medio de técnicas de procesamiento de imágenes, las transiciones que se pueden realizar; dando lugar a una mejor visualización y engranaje a una secuencia de planos bidimensionales, así como el diseño de una rica interfaz gráfica.

Para dar un mayor dinamismo e interactividad entre las interfaces de las aplicaciones de RV de la facultad 5 (F5), se ha hecho un estudio de los efectos de transiciones y tecnologías con el fin de lograr obtener una biblioteca que contenga la implementación de cada una de las técnicas de transición.

Palabras claves:

Biblioteca, Efectos, Transiciones, Pantalla, Realidad Virtual.

ÍNDICE

INTRODUCCIÓN

El desarrollo acelerado actual en cuanto al avance de las tecnologías del mundo virtual trae consigo que las apariencias gráficas se encuentren sujetas constantemente a nuevos cambios en dependencia de la evolución de la ciencia y las necesidades que la sociedad impone en los momentos en que se vive.

Las transiciones son efectos visuales que cambian la apariencia gráfica de una aplicación, permiten esa interactividad, cambiar de un estado de vista a otro, no es más que la forma en la cual una imagen cambia a otra en una animación o una presentación de una multimedia o en las interfaces de los videos-juegos, la definición de las transiciones dependen de las clases de efectos que se utilicen.

Tienen la designación de trabajar específicamente con cambios que ocurran simultáneamente en las interfaces, un cambio en la misma, cambia significativamente múltiples componentes que estén dentro de las interfaces al mismo tiempo.

Los efectos de transición tienen muchas formas de ser utilizados en la actualidad ya sea para alterar la forma de una página o cambiar los elementos de la misma a otros elementos, este es el caso de un sitio Web, otra de las áreas donde se aplica es en las herramientas de multimedia Flash, donde se utilizan los efectos de transición para darle animación a una escena o imagen, también tiene vínculo en los videos, con el empleo del Adobe Premier, donde se edita y se reemplaza cada escena con cada uno de estos efectos, obteniendo una especie de animación en cada escena.

Los efectos de transición no se quedan alejados de las Aplicaciones de Realidad Virtual (ARV), sus vínculos están muy estrechados, su empleo permite jugar con efectos sorprendentes en cada interfaz, permite gran dinamismo en la apariencia de las aplicaciones y en las confecciones de menú.

En el mundo se han desarrollado múltiples proyectos de Realidad Virtual (RV) donde se usan los efectos de transición de pantalla por los beneficios que estos brindan. En Cuba, la Universidad de las Ciencias Informáticas (UCI), institución educacional dedicada a la producción de software, tiene entre sus perfiles de desarrollo la RV, donde se crean video-juegos y simuladores.

En la actualidad las ARV que se desarrollan en la universidad no cuentan con una biblioteca que permita facilitar la elaboración de menú que contengan efectos de transición de pantallas.

De lo anteriormente planteado, se desprende como interrogante el siguiente ***problema científico*** a resolver:

¿Cómo desarrollar una biblioteca de transición de pantalla que permita la elaboración de menú con efectos similares a las transiciones de diapositivas del Power Point para las ARV de la F5?

Para contribuir a solucionar esta interrogante se plantea como ***objeto de estudio:*** la utilización de diferentes técnicas de transiciones de pantallas en los Sistemas de Realidad Virtual (SRV).

Como ***objetivo general*****:** se propone desarrollar una biblioteca de transición de pantallas para dar mayor dinamismo en las interfaces de las ARV desarrolladas en la F5.

El ***Campo de acción*** de la investigación está definido como las bibliotecas de transición de pantallas para los SRV.

Como ***ideas a defender, se sugiere que*****:** las técnicas seleccionadas para conformar la librería de transición de pantallas de los videos-juegos, contribuirá a aumentar el dinamismo en las apariencias gráficas de las interfaces en las ARV y consigo minimizará el trabajo de los desarrolladores de video-juegos.

Tareas Investigativas

- Estudiar las técnicas de transición de pantallas existentes y aplicadas en la actualidad.
- Seleccionar las técnicas de transición de pantallas, cuyas características respondan a las necesidades de las ARV de la facultad 5.
- Modelar computacionalmente los algoritmos de las diferentes transiciones seleccionadas.
- Determinar las tecnologías y lenguajes de programación a utilizar en el desarrollo de la investigación.
- Elaborar un diagrama de clases, lo suficientemente extensible, que permita la inclusión de nuevas técnicas de transición de pantallas.
- Diseñar la arquitectura de la biblioteca de transición de pantallas.
- Implementar la arquitectura de dicha biblioteca.

- Desarrollar un demo para mostrar la aplicabilidad de las diferentes transiciones de pantallas en las ARV de la F5.

El trabajo ha sido estructurado de la siguiente manera:

Capítulo 1: Fundamentación Teórica, refleja algunos conceptos importantes relacionados con los efectos de transición así como los diferentes tipos de efectos utilizados actualmente, explicándose de manera científica en que consisten los mismo. Además se justifican las tecnologías, metodología de desarrollo, lenguaje de visualización UML y de programación ha utilizar, se argumenta sobre una de las técnicas que permiten algunos de los efectos que se abordan.

Capítulo 2: Descripción de la solución propuesta, se propone la solución para resolver el problema científico identificado. También se crea el modelo del dominio, se definen los requerimientos funcionales y no funcionales con que debe contar la biblioteca y se especifican los casos de uso del sistema y sus descripciones en formato expandido.

Capítulo 3: ***Diseño e Implementación***, se presenta la relación entre las clases de la biblioteca de transición de pantallas (STE), agrupadas en un diagrama para un mejor entendimiento. Además se presentan los diagramas de secuencia y componentes donde se muestran las interacciones entre los .h y .cpp, los estándares de codificación por los que se regirá la implementación de la biblioteca.

Finalmente se proporciona un apéndice compuesto por un glosario de abreviaturas, y un glosario de términos, para apoyar la comprensión del lenguaje técnico utilizado en la investigación y en el desarrollo del trabajo.

1 Fundamentación Teórica

1.1 Introducción

El cambio de un escenario en una ARV requiere usar una transición con el objetivo de darle una visión dinámica a la aplicación, dicha transición debe ser realmente rápida en cuestiones de tiempo para no interponer la transición en la vista del escenario. La forma más frecuente en que ocurra la transición es cuando el usuario interrumpe ya sea para salir de la aplicación o cuando se prepara para entrar en ella o hacerle algunos cambios a las opciones que ella puede presentar.

En este capítulo se reflejarán algunos conceptos importantes relacionados con los efectos de transición, así como los diferentes tipos de ellos utilizados actualmente, explicándose de manera científica en que consisten los mismos. Además se abordarán sobre las diferentes tecnologías que se emplearán en el desarrollo del trabajo.

1.2 Transición

Las transiciones son tan antiguas como el cine mudo, pero siguen siendo uno de los principales recursos en la edición de películas, y el trabajo en el entorno informático, manejarlas adecuadamente es todo un arte. Su inicio fue respuesta de limitaciones técnicas que existían a la hora de manejar el paso de algún elemento para otro, el cine fue uno de los ámbitos que dieron origen a la formulación de buscar e implementar una técnica que resolviera los defectos de la edición que existían, hacer transparente e impactante visualmente el montaje conjuntamente con el medio informático, requería para sus marcos de desarrollo implementar efectos que se acoplaran a las necesidades que requerían el trabajo web y ARV de interactuar con los elementos internos de cada uno de los planos en que se desarrolla y lograr una dinámica interactiva con cada uno de los elementos de las páginas web, o las interfaces de las ARV.

Todo fue un marco para que surgiera el despertar tecnológico y científico para desarrollar herramientas, software que agrupen las técnicas de transiciones para lograr el paso de planos, visualizar una secuencia de imágenes que logre generar la sensación de movimiento, y crear videos, películas, presentaciones, multimedia, que jueguen con estos efectos y le den una organización a cada elemento que se desee presentar. También aparecieron los códigos de diferentes lenguajes que englobaban funcionalidades para lograr efectos de transiciones aplicables a todo el entorno que enmarca la creación de un sitio web, manejar el paso de una página a otra, el trabajo con imágenes y letras, Java Script es uno de esos lenguajes que permite darle esa interactividad a cada uno de los elementos que se quieran dar entrada en el desarrollo web. Para las ARV todo se centra en una marco gráfico para el trabajo con el menú y el paso de sus interfaces, ilustrando con el uso de las transiciones una dinámica espectacular en cada interfaz y menú, se apoyan de los beneficios de funcionalidades que aportan las API OpenGL y DirectX para lograr suavizar cada paso de elementos.

La transición es un efecto especial que se usa para indicar la aparición de una diapositiva durante una presentación, permiten variaciones de cambio ya sea en una imagen, sonido o video, por eso los términos para cada una de sus aplicaciones coinciden en sus objetivos. Por ejemplo, puede aparecer gradualmente desde un fondo negro o disolver una diapositiva hasta la aparición de otra. **[Wikipedia "Transiciones", 2007]**

Al acabar una secuencia o clip y empezar otro, el comportamiento por defecto es acabar inmediatamente el primero y dar paso al segundo. Las transiciones son la forma de pasar de uno a otro con diferentes variaciones visuales o sonoras. **[Manual de Cinelerra CV, 2007]**

Es un efecto animado que facilita, o enfatiza, el paso de un clip al siguiente, proporciona un puente entre dos clips de pantalla completa (o entre un clip y la oscuridad si la transición sólo tiene un clip contiguo, como en el comienzo de la película), enlazando dos sobreimpresiones, o una sobreimpresión y una transparencia. **[Wikipedia "Transiciones", 2007]**

Existen varios tipos de transición (persianas, cuadros bicolores, barrido, cubrir, cortar, empujar, disolver, entre otras.), muchos de los cuales disponen de variedades para las diferentes direcciones posibles (desde arriba, izquierda, abajo y derecha). **[Wikipedia "Transiciones", 2007]**

1.2.1 Transición entre imágenes

Permite el cambio de una imagen a otra, añadiendo un efecto visual que se reproduce entre dos fotos para que la transición de imágenes sea más suave, más atractiva o más interesante. **[Configurar un pase de diapositivas, 2007]**

1.2.2 Transición entre pantallas

Permite especificar cómo se va a desarrollar el cambio de una pantalla a otra y suavizar las sucesiones. Mediante esta se puede utilizar el mismo efecto en todas las transiciones o aplicar diferentes modos de unas a otras **[Pérez, 2007]**. Es preciso organizar el paso de una pantalla a la siguiente de una forma variada, pero dentro de una lógica que le de unidad a toda la presentación. La animación dentro de la pantalla permite establecer el orden y la forma en la que van a ir apareciendo los objetos que la constituyen. **[Bravo Ramos, 2002]**

1.3 Efectos de Transición

Las transiciones son unos de los detalles que hace comparar una aplicación de buena realización, son pequeños puntos de detalles que ayudan hacer los SRV más divertido e interesantes, existen diferentes transiciones, las cuales son:

Transición Persianas

Esta transición se atribuye su nombre por la forma en que sucede, cerrar una persiana. Consiste en dibujar una imagen con franjas que se mueven en una misma dirección como persianas que se cierran, puede ser de dos formas, el cerrado vertical u horizontal **[Svarcas, 2004]**. La idea es dividir la imagen en rectángulos de igual tamaño e ir dibujando al mismo tiempo pero solamente con la franja de cada rectángulo por tiempo y así simula el efecto de cerrar persiana.

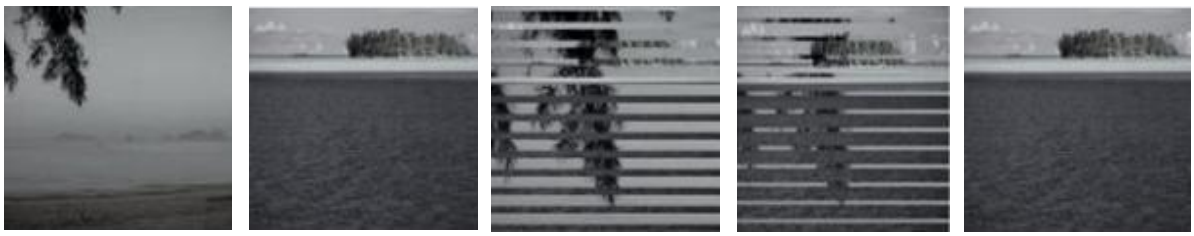

Fig. 1 Efecto de Transición Persiana.

Transición de Empuje

Como bien dice su nombre, consiste en un empuje es decir una imagen se mueve de un lado para otro, apareciendo otra imagen, como si estuvieran empujando la imagen **[Svarcas, 2004]**. En una transición de empuje, la nueva escena se empuja encima de la anterior, usualmente puede ser de izquierda a derecha o de derecha a izquierda. Lo básico está en la dirección del empuje, existiendo ochos direcciones desde las cuales se pueden elegir para poder realizar la transición, un empuje puede empezar en cualquier borde o esquina y avanzar hasta su opuesta **[McCuskey, 2002]**. Se pueden empujar las dos imágenes para hacer la transición. Una vez más la única constante usada es el tiempo. En cambio, de acortar las imágenes contra el lado derecha o izquierdo de la ventana, guardamos una imagen tocando el borde izquierdo mientras que la otra está tocando siempre el borde derecho. Como el tiempo cambia, las imágenes se comprimen y se estiran en la ventana **[Wolfgang, 2003]**.

Fig. 2 Efecto de Transición Empuje hacia arriba.

Existen diferentes transiciones de empuje:

- Empuje hacia arriba, empuja la imagen desde la posición horizontal desde arriba apareciendo la otra imagen.

- Empuje hacia abajo, empuja la imagen desde abajo en la posición horizontal apareciendo la otra imagen.
- Empuje por la división de los lados, no es más que dividir la imagen y empujar hacia los lados apareciendo la otra imagen.
- Empuje por la división de los topes es casi lo mismo que el empuje de división de los lados pero haciéndolo desde arriba.
- Empuje Diagonal, este efecto consiste en empujar la imagen de forma diagonal, apareciendo la otra imagen.

Transición Estirar

En una transición de estiramiento una nueva escena alarga sus dimensiones por los ejes *x* e *y* dando lugar a que la misma ocupe el lugar de la otra escena, posee múltiples formas de representarse:

- Estirar desde el centro.
- Estirar desde abajo de izquierda a derecha.
- Estirar desde abajo de derecha a izquierda.
- Estirar desde arriba de izquierda a derecha.
- Estirar desde arriba de derecha a izquierda.

Fig. 3 Efecto de Transición Estirar.

Transición Girar

Una de las variantes para su realización es hacer variar el ángulo de giro e ir disminuyendo las dimensiones de la escena del fondo para que a medida que gire vaya ocupando el lugar de la escena del frente, otras podrían ser:

- Girar y encoger.
- Girar desde el centro.

Fig. 4 Efecto de Transición Girar desde el Centro.

Transición de División

Esta transición tiene variantes para realizarse, hay dos que consisten en dibujar la imagen desde los lados(o los topes) hacia el centro como puertas deslizantes que se cierran y las otras dos hacen lo opuesto, se dibujan desde el centro hacia lo extremos **[Svarcas, 2004]**. La transición de división no es más que una familia de la transición de empuje que se realiza en dos direcciones contrarias simultáneamente como si dos parejas de puertas se abrieran, se usan dos cuadrantes, se ponen en los lados opuestos y después se reúnen sus bordes **[McCuskey, 2002]**.

Fig. 5 Efecto de Transición División por los lados.

Transición de Barrido

La transición consiste en dividir la imagen en rectángulos de igual tamaño para luego ir moviendo la mitad de estos rectángulos desde la derecha hacia el centro, y la otra mitad desde la izquierda hacia el centro de manera que al encontrarse se intercalen para formar la imagen completa **[Svarcas, 2004]**.

Es un efecto que se produce en la fase de registro de la imagen y que consiste en un giro rapidísimo de la cámara que produce un efecto visual semejante, da paso de un elemento que ocupa toda la pantalla, tan deprisa que no da tiempo a ver de qué se trata. El barrido se utiliza para pasar de un espacio a otro de forma instantánea **[Valverde Berrocoso, 2004]**.

Fig. 6 Efecto de Transición Barrido.

Transición de Rueda

Consiste en dibujar una imagen en forma de rueda en sentido de las agujas del reloj y aparezca otra imagen de fondo. Existen variantes del efecto Rueda de acuerdo a los ejes que se van a dibujar, tal que pueden ser de 2 ejes, 3 ejes, 4 ejes y 8 ejes. La imagen se dividirá en varias partes triangular hasta llenar la imagen completa, especie de torta es la división de la imagen de la forma diagonal **[Svarcas, 2004]**.

Fig. 7 Efecto de Transición Rueda con 3 ejes.

Transición con Círculo

Consiste en dibujar una imagen en forma de círculo, el cual se cierra o se abre, haciendo variar el diámetro del círculo circunscrito que sería el plano del fondo, de esa forma aparece otra imagen **[Svarcas, 2004]**.

Fig. 8 Efecto de Transición Círculo Abre.

Transición Aparecer

Consiste en aparecer una imagen que está detrás de otra de forma transparente, para tomar esa imagen que aparece como fondo **[Svarcas, 2004]**. Desaparece una imagen mientras entra otra ocupando su lugar, el uso de este efecto en la computadora puede ser complicado de hacer pero fácil a la hora de trabajarlo con cuadrantes para poner la imagen que va ser aplicada en esta transición **[McCuskey, 2002]**.

Fig. 9 Efecto de Transición Aparecer.

Transición Simétrica

Dibuja la imagen en forma de dos ruedas que parten del mismo lugar, una dibuja una mitad de la imagen en sentido de las agujas del reloj y la otra en sentido opuesto hasta completar toda la imagen **[Svarcas, 2004]**.

Fig. 10 Efecto de Transición Simétrico en sentido del reloj.

Transición Rodar

El objetivo principal de este efecto es que la imagen del frente inicialmente se encuentra fuera del centro de visión, para una vez que comience el efecto el ángulo de giro progresivo irá aumentando hasta que la imagen de ½ vuelta y tome la posición de la otra imagen llegando a su fin la transición. Esta transición tiene 4 variantes de realizarse:

- Rodar derecha arriba hacia abajo.
- Rodar derecha abajo hacia arriba.
- Rodar izquierda de abajo hacia arriba.
- Rodar izquierda de arriba hacia abajo.

.

Fig. 11 Efecto de Transición Rodar.

1.4 Técnica de Transparencia. Alpha Blending

Alpha Blending es una técnica muy utilizada para lograr las transiciones entre imágenes, es un factor esencial a la hora de desarrollar una transición entre escenarios, permitiendo obtener una gran variedad de efectos. Es el proceso de combinar dos objetos en pantalla teniendo en cuenta los valores alfa que designa su grado de transferencia, posibilitando que el color del píxel del objeto mostrado en segundo plano se vea influenciado por el grado de transparencia del píxel correspondiente al situado en primer plano. Permite crear objetos transparentes de forma que cualquier objeto situado detrás de él sea visible respetando su opacidad **[Efectos gráficos, 2007]**.

Esta técnica se compone por dos elementos: Alpha componente principalmente utilizado para especificar el nivel de transparencia de un píxel y el Blending permite mezclar los píxeles que se están dibujando con otros que fueron dibujados previamente. Se asume que se reserva 8 bits para el valor alfa de cada píxel, donde el intervalo válido para el componente alfa sería [0,255] la cual corresponde a una opacidad de [0%,100%], se obtiene la transparencias con valores blanco (255), el píxel sería completamente opaco y la semi-transparencias con valores grises. La idea del Blending es combinar los píxeles, un primer píxel que se está escribiendo llamado píxel de origen (source píxel) con otro píxel de destino (destination píxel).Se deben dibujar los objetos que no utilizan Blending primero. Luego se deben ordenar los objetos que usan Blending por su distancia a la cámara; esto se puede lograr más eficientemente si los objetos están en el espacio de vista (ya transformados para la pantalla), así que se puede simplemente ordenar el componente ***z.*** Finalmente, se deben dibujar los objetos que utilizan Blending de atrás hacia delante **[Grupo de Desarrollo en Allegro, 2004]**.

La siguiente es la fórmula usada para combinar dos valores de píxeles:

PixelSalida = PixelOrigen x FactorCombinaciónOrigen + PixelDestino x FactorCombinaciónDestino

Todas las variables anteriores son vectores de color 4D (*r, g, b, a*) y el símbolo **x** denota una multiplicación de componentes, donde *r, g, b* son los colores básicos (rojo, verde, azul).

Tabla 1 Descripción de las variables de la fórmula.

Variables	**Significado**
Pixel Salida	El pixel resultante
Pixel Origen	El pixel actual que se está dibujando, y que será combinado con el que ya existe en el back buffer.
Factor Combinación Origen	Un valor en el intervalo [0,1] que especifica cuanto del píxel de origen será usado en la combinación.
Pixel Destino	El píxel existente en el back buffer, dibujado con anterioridad.
Factor Combinación Destino	Un valor en el intervalo [0,1], que especifica cuanto del píxel de destino será usado en la combinación.

Los factores de origen y destino dejan modificar los valores originales de los píxeles en una variedad de formas, permitiendo conseguir diferentes efectos.

1.5 Biblioteca

En Informática, una biblioteca es un conjunto de procedimientos y funciones (subprogramas) agrupadas en un archivo con el fin de que puedan aprovecharlas otros programas, el nombre de biblioteca tiene el mismo significado que el término librería, son trozos de códigos que contienen una

funcionalidad pre construida que puede ser utilizada por un ejecutable, los mismos suelen ser unidos por enlaces.

De forma general el término librería se utiliza para hacer referencia a un conjunto de módulos de objetos donde todos van a estar agrupados en un solo fichero, el cual puede tener una serie de extensiones (.bpl, .lib, .dll). Estos ficheros te permiten tratar una serie de colecciones de módulos en una sola unidad, es muy conveniente a la hora de trabajar con grandes aplicaciones.

Existen dos tipos de Librerías:

- **estáticas** o de **enlace estático** llamada también librerías de objetos

Consiste en unir durante el enlace el código objeto de las librerías con el código del programa, generando así el ejecutable. El programa ejecutable crece notablemente de tamaño respecto de los archivos objetos, ya que incorpora el código de todas las funciones de las librerías.

Son una colección de ficheros (objetos compilados), los ficheros son agrupados en un solo fichero de extensión **.lib**, **.a**, junto con uno o varios ficheros de cabecera (generalmente **.h**). Con las librerías estáticas que contengan las funciones que se usen más frecuentemente no será necesario escribir el código cada vez. La ventaja de este tipo de enlace es que hace que un programa no dependa de ninguna librería (puesto que las enlazó al compilar), haciendo más fácil su distribución.

- **compartidas** o de **enlace dinámico**

Las librerías dinámicas, **DLL**, también conocidas como de enlace dinámico son aquellas en las cuales una librería de código es enlazada cuando un determinado programa se ejecuta (en oposición a un enlace estático, que se produce en tiempo de compilación). La ventaja de este tipo de enlace es que el programa es más liviano, reduce el tamaño del archivo ejecutable, permite la compartición de librerías entre diferentes aplicaciones y evita la duplicación de código (por ejemplo, cuando dos programas enlazan con la misma librería, se necesita sólo una copia).

El enlace dinámico produce ejecuciones más lentas, ya que cada vez que se use una función de librería dinámica es necesario buscar el archivo en el que se encuentra y ejecutar su código. Además, pueden producirse errores de enlace durante la ejecución del programa **[Moreno Vozmediano, 2008]**.

Estos tipos de librerías son las utilizadas por el Sistema Operativo Windows, el mismo contiene una serie de librerías del tipo **DLL** donde se encapsulan el conjunto de objetos que controlan todo el funcionamiento del sistema.

1.6 Fundamentación de las tecnologías y metodología de desarrollo

1.6.1 Metodología de desarrollo

La metodología Rational Unified Process (RUP), que en su traducción al español es El Proceso Unificado, es un marco de trabajo genérico que puede especializarse para gran variedad de sistemas **[JACOBSON, BOOCH, & Rumbaugh, 1999]**.

RUP es el resultado final de tres décadas de desarrollo y uso práctico. Esta es una de las causas que conlleva a que sea la metodología que mejor se ajusta a las necesidades que existen actualmente en el desarrollo de software, pues propone un Modelo iterativo e incremental, centrado en la arquitectura y guiado por casos de uso muy acorde con la naturaleza cambiante de los requisitos en muchos proyectos. Es una metodología que está totalmente respaldada por una excelente herramienta CASE: Rational Rose. Utiliza el paradigma de orientación a objetos para su descripción.

RUP asegura que las expectativas de todas las partes sean sincronizadas y consistentes, esto es asegurado a través de evaluaciones periódicas durante el ciclo de vida del proyecto. Provee un enfoque disciplinado para asignar tareas y responsabilidades del sistema. Reduce en gran medida el riesgo de representar la construcción de sistemas complejos, porque evoluciona de forma incremental partiendo de subsistemas pequeños en los que ya se tiene confianza **[Teoría RUP, 2006]**.

RUP es una guía de cómo usar Lenguaje de Modelado Unificado (UML) de la forma más efectiva, siendo este la base del Modelamiento visual de RUP.

Lenguaje de Modelado

UML es un lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de software **[Teoría RUP, 2006]**. Actualmente UML ofrece un estándar para describir un “plano” del sistema incluyendo aspectos conceptuales como procesos del negocio,

funciones del sistema, y aspectos concretos como expresiones del lenguaje de programación, esquemas de bases de datos y componentes de software utilizados.

Permite representar múltiples vistas de un sistema, usando una variedad de diagramas como: diagramas de casos de uso, diagramas de clase, diagramas de estado, diagramas de secuencia y diagramas de colaboración.

1.6.2 Biblioteca Gráfica OpenGL

OpenGL es la biblioteca gráfica 3D por excelencia, puede utilizarse en Linux, Unix, Mac OS, y Windows. Desarrollada originalmente por Silicon Graphics Incorporated (SGI), es una biblioteca de gráficos que ofrece al programador una Interfaz de Programación de Aplicaciones (API) que ha crecido a la par del hardware. Su nombre viene del inglés **O**pen **G**raphics **L**ibrary, cuya traducción es biblioteca de gráficos abierta.

Es poderosa, con rendimiento a bajo nivel y una biblioteca de software de modelamiento, disponible en la mayoría de las plataformas, con un amplio soporte de hardware (HW). Es diseñado para ser usado en cualquier aplicación gráfica, desde juegos y simuladores hasta modelaciones CAD (Computer Aided Design) **[JACOBSON, BOOCH, & Rumbaugh, 2000]**.

Esta biblioteca gráfica es una especificación estándar que define una API multilenguaje y multiplataforma para escribir aplicaciones que produzcan gráficos 2D y 3D, orientada inicialmente a los video-juegos, pero actualmente multipropósito **[ASTLE & HAWKING, 2001].** OpenGL tiene dos propósitos principales:

- Ocultar la complejidad de la interfaz con las diferentes tarjetas gráficas, presentando al programador una API única y uniforme.
- Ocultar las diferentes capacidades de las diversas plataformas hardware, requiriendo que todas las implementaciones soporten el conjunto completo de características de OpenGL (utilizando emulación software si fuese necesario).

La operación básica de OpenGL es aceptar primitivas tales como puntos, líneas y polígonos, y convertirlas en píxeles. Este proceso es realizado por una *pipeline* gráfica conocida como la Máquina

de estados de OpenGL. La mayor parte de los comandos de OpenGL emiten primitivas a la pipeline gráfica o configuran cómo la pipeline procesa dichas primitivas.

OpenGL ofrece su propio grupo de bibliotecas conocidas como OpenGL Utility Toolkit (GLUT, herramientas y Utilidades de OpenGL) con soportes disponibles en la mayoría de las plataformas y funcionalidades básicas en el área de trabajo con ventanas. GLUT mantiene la independencia de las plataformas, es decir, se puede mover fácilmente una aplicación basada en GLUT de Windows hacia Unix con unos pocos cambios.

Ha influido en el desarrollo de las tarjetas gráficas, promocionando un nivel básico de funcionalidad que actualmente es común en el hardware comercial; algunas de esas contribuciones son:

- Primitivas básicas de puntos, líneas y polígonos rasterizados.
- Una pipeline de transformación e iluminación.
- Z-Buffer.
- Mapeado de texturas.
- Alpha Blending.

1.6.3 Lenguaje de Programación Visual C++. Net

Proporciona a los programadores un lenguaje orientado a objetos de probada eficacia para generar aplicaciones de alto rendimiento gracias a plantillas avanzadas o programación genérica (*templates*), acceso a plataformas de bajo nivel y un compilador que optimiza las compilaciones, ofrece funcionalidad para generar componentes sólidos.

Además posee una serie de propiedades difíciles de encontrar en otros lenguajes de alto nivel:

- Posibilidad de redefinir los operadores (sobrecarga de operadores)
- Identificación de tipos en tiempo de ejecución (RTTI).

C++ trata de un lenguaje de programación basado en el lenguaje C, estandarizado (ISO/IEC 14882:1998), ampliamente difundido, y con una biblioteca estándar que lo ha convertido en un lenguaje universal, de propósito general, y ampliamente utilizado tanto en el ámbito profesional como en el educativo.
C++ es la evolución de C adaptada a la programación orientada a objetos. Tiene algunas cuestiones más pulidas como un control más estricto en el manejo de tipos de datos, y otras características que ayudan a la programación libre de errores **[HIEBERT & CHARLEY, 2006]**.

Es un lenguaje que abarca tres paradigmas de la programación: la programación estructurada, la programación genérica y la programación orientada a objetos. C++ está considerado como un lenguaje potente, debido a que permite trabajar tanto a alto como a bajo nivel **[Analisis de Lenguajes de Programación., 2008]**.

Es uno de los lenguajes de sistemas más conocido del mundo, altamente compatible, soporta las bibliotecas gráficas Glide, OpenGL, DirectX y G3D que son las utilizadas por excelencia en la industria de los videos-juegos **[Guerrero Tala, 2006]**.

1.6.4 Herramientas

Rational Rose Enterprise Edition 2003

Es una herramienta de desarrollo basada en modelos que se integra con las bases de datos y los IDE (Integrated Development Environment, entorno de desarrollo integrado) de las principales plataformas. Soporta UML para trabajar en diseños de aplicaciones, con capacidad de representar la integración de los datos y los requerimientos de aplicación a través de diseños lógicos y físicos **[Características de Visualización C++.Net, 2007]**.

Rational Rose Enterprise es un entorno de modelado que permite generar código a partir de modelos Ada, ANSI C++, C++, CORBA, Java/J2EE, Visual C++ y Visual Basic. Todos los productos de Rational Rose, ofrecen un lenguaje de modelado común que agiliza la creación del software.

Microsoft Visual Studio 2005

Microsoft Visual Studio es un entorno de desarrollo integrado (IDE) para sistemas Windows. Soporta varios lenguajes de programación tales como Visual C++, Visual C#, Visual J#, ASP.NET y Visual Basic .NET, aunque actualmente se han desarrollado las extensiones necesarias para muchos otros.

Visual Studio se ha concebido y probado para ser sistemáticamente confiable, seguro, interoperable y compatible, está diseñado para garantizar la compatibilidad con versiones anteriores siempre que sea posible.

Facilita la validación del diseño con respecto a un entorno operativo de destino, y reduce el riesgo de problemas en la implementación **[Características de Software, 2006]**.

1.7 Consideraciones Finales

- En el desarrollo del trabajo se ha descrito el concepto de transición de imágenes y pantallas.
- Han sido descritos los diferentes tipos de transiciones más utilizados.
- Se analizaron las diferentes características de la técnica de transparencia, AlphaBlending.
- Se dio cumplimiento a la fundamentación de las diferentes tecnologías ha utilizar y la metodología.
- Considerándose como biblioteca gráfica ha utilizar OpenGL, como metodología de desarrollo RUP, para la visualización detallada UML, y como lenguaje de programación se propone a C++.

2 Descripción de la Solución Propuesta

2.1 Introducción

Se proponen soluciones técnicas para el funcionamiento de los efectos de transición y soluciones específicas para lograr su integración a la biblioteca, así como para lograr una manipulación y ejecución de los efectos. Además se comienza a tener una visión del producto a realizar y se dan los primeros pasos en su concepción práctica, basándose en las necesidades y dificultades.

2.2 Descripción de la solución

Dada la necesidad de obtener una biblioteca que maneje los diferentes efectos de transición de pantallas y que ésta pueda ser de uso en diferentes plataformas, se hará uso de la biblioteca gráfica OpenGL teniendo en cuenta las posibilidades y ventajas que brinda.

La biblioteca STE usará como estructura de datos básica el *Vector* de la *STL (Standard Template Library)* de *C++,* por las facilidades que brinda dicha estructura. El uso de esta lista de efectos permite una cómoda manipulación y acceso, aprovechando también la gestión dinámica de la memoria, optimizando de esta forma el trabajo con biblioteca.

El lenguaje de programación que se empleará es el C++ debido a su alto nivel, el cual incorpora una variedad de características que facilitan una programación elegante y modular, además el código escrito en C++ estándar permite mantener su portabilidad hacia otras plataformas.

Además utilizaremos la metodología de desarrollo RUP que se ajusta a las necesidades que presenta la biblioteca STE, reduce las construcciones complejas evolucionando de forma incremental partiendo de subsistema pequeños, a la vez hace uso de UML para la visualización de diferentes diagramas en

cada una de las fases de construcción, este lenguaje de modelado permite definir, detallar y documentar los artefactos del producto final.

2.2.1 Descripción algorítmica de los Efectos de Transición seleccionados

Cada transición posee un proceso de realización en el cual implica el trabajo con textura, pintar primitiva y ejecutar el efecto de acuerdo a la velocidad con que se quiere realizar, donde cada uno se realiza de diferentes formas.

Transición Aparecer

Este efecto se realiza entre dos imágenes que son cargadas como texturas, y asignadas a dos planos, cada uno con forma de cuadrado, la textura del plano de fondo tendrá la función *glBlendFunc (GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA),* para activar los valores que va contener la función AlphaBlending para mezclar los pixeles de origen y destino, y lograr la técnica de transparencia.

Posteriormente se va graduando la opacidad (el valor de *z* es alpha) de la textura que se quiere que aparezca, en este caso, la textura del plano de fondo. La opacidad se calcula:

```
s= (timeII-timeI)*velocity
opacity +=s
Si opacity mayor 1.0
Entonces:
opacity=1.0
Sino
Si opacity menor 0.01
Entonces:
 opacity igual a 0.01
 d igual a 0.01
RedibujaVentana
```

donde:

timeI: tiempo inicial del efecto de transición.

timeII: tiempo final del efecto de transición.

velocity: velocidad con que ocurre el efecto de transición.

s: desplazamiento con que se va lograr la transición.

opacity: valor de la opacidad.

RedibujaVentana: función de OpenGL que redibuja la ventana actual.

Este valor de opacidad se le pasa a la función *glColor4f (1.0, 1.0, 1.0, opacity)* del cuadrado de fondo y se logra el efecto. Ejemplo relacional de las funciones para el cuadrado de fondo:

```
glEnable (GL_BLEND);
glBlendFunc (GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);
glBegin(GL_QUADS);
glNormal3f (0.0f, 0.0f, 1.0f);
glColor4f (1.0, 1.0, 1.0, opacity);
glVertex3f (-vertexX,-vertexY, 0.0);
glVertex3f (vertexX,-vertexY, 0.0);
glVertex3f (vertexX, vertexY, 0.0);
glVertex3f (-vertexX, vertexY, 0.0);
glEnd();
```

Transición Giro desde el Centro

Este efecto de transición se realiza entre dos imágenes, asignadas a dos planos, se inicializan las dimensiones del fondo con valores menores que el frente, para que de forma gradual ir escalando los vértices del plano de fondo hasta que llegue a sobreponerse al plano del frente con igual valor de los vértices, a la vez se va rotando el plano del fondo con respecto al eje *z.* Para lograr el efecto de giro, los valores de los vértices, ángulo de rotación, y eje *z* se calculan de esta forma:

```
s= (timeII-timeI)*velocity
angle+= (360/s)
increment+=s
z=increment
vertexX= increment
vertexY= increment
Si increment mayor sizeQuad
Entonces:
vertexX igual a sizeQuad
vertexY igual a sizeQuad
angle igual a cero
RedibujaVentana
```

donde:

s: desplazamiento con que se va lograr la transición.

timeI: tiempo inicial del efecto de transición.

timeII: tiempo final del efecto de transición.

angle: ángulo de giro

increment: variable incremental del desplazamiento.

sizeQuad: dimensión que tiene el cuadrado.

velocity: velocidad con que ocurre el efecto de transición.

z: eje z de rotación.

vertexX: vértice *x* del cuadrado fondo.

vertexY: vértice *y* del cuadrado fondo.

RedibujaVentana: función de OpenGL que redibuja la ventana actual.

Los valores de los vértices se calculan y se le pasa a la función *glVertex3f (vertexX, vertexY, 0.0)* a la hora de construir el plano del fondo.

Ejemplo:

```
glBegin (GL_QUADS);
glNormal3f( 0.0f, 0.0f, 1.0f);
glVertex3f(-vertexX,-vertexY,0.0);
glVertex3f (vertexX,-vertexY,0.0);
glVertex3f (vertexX, vertexY,0.0);
glVertex3f (-vertexX, vertexY,0.0);
glEnd();
```

Y los valores del eje *z* y ángulo de rotación que se le pasa a la función de *glRotatef* (función que permite rotar respecto a un eje y con un ángulo determinado) sería *glRotatef (angle, 0.0, 0.0, z)*, se le indica esta función al plano de fondo antes de construirlo, de esa forma el plano texturizado del fondo se interpone al plano del frente rotando y aumentando sus vértices. Ejemplo relacional de las dos funciones:

```
glRotatef (angle, 0.0, 0.0, z);
glBegin(GL_QUADS);
     glNormal3f (0.0f, 0.0f, 1.0f);
     glVertex3f (-vertexX,-vertexY,0.0);
     glVertex3f (vertexX,-vertexY, 0.0);
     glVertex3f (vertexX, vertexY, 0.0);
     glVertex3f(-vertexX, vertexY, 0.0);
     glEnd();
```

Transición Girar y Encoger

El sentido de giro y encoger se realiza entre dos planos, a cada uno le corresponde una textura, se define el plano frente siendo el objetivo clave de aplicarle una rotación respecto a un ángulo de rotación y el eje *z*, a la vez se disminuye los vértices para que de paso a la aparición del plano que está de fondo. Los valores para lograr la rotación y el encogimiento se calculan como se muestra a continuación:

```
s= (timeII-timeI)*velocity
angle+= (360/s)
increment+=s
z=increment
vertexX= sizeQuad - increment
vertexY= sizeQuad – increment
Si increment mayor sizeQuad
Entonces:
vertexX igual a cero
vertexY igual a cero
angle igual a cero
RedibujaVentana
```

donde:

s: desplazamiento con que se va lograr la transición.

timeI: tiempo inicial del efecto de transición.

timeII: tiempo final del efecto de transición.

angle: ángulo de giro

increment: variable incremental del desplazamiento.

sizeQuad: dimensión que tiene el cuadrado.

velocity: velocidad con que ocurre el efecto de transición.

z: eje *z* de rotación.

vertexX: vértice *x* del cuadrado fondo.

vertexY: vértice *y* del cuadrado fondo.

RedibujaVentana: función de OpenGL que redibuja la ventana actual.

Una vez calculado los valores del eje *z* y el ángulo de rotación se le pasa a la función de rotación, actualizando a la misma vez los vértices del plano del frente, logrando de esa forma el efecto de girar y encoger a la vez.

Transición Estirar desde el Centro

Este efecto de transición se realiza entre dos imágenes, asignadas a dos planos que cada uno tiene la forma de cuadrado, se construye primero el plano del frente y posteriormente se va aumentando las dimensiones de los vértices del plano de fondo en forma de estiramiento, los vértices se van estirando hasta que escale completo la textura del plano de frente y se quede el plano de fondo como frente. Para eso se calcula las dimensiones del plano de fondo como se muestra a continuación:

```
s= (timeII-timeI)*velocity
stretch+=s
vertexX= stretch
vertexY= stretch
Si stretch mayor sizeQuad
Entonces:
stretch igual a sizeQuad
RedibujaVentana
```

donde:

s: desplazamiento con que se va lograr la transición.

timeI: tiempo inicial del efecto de transición.

timeII: tiempo final del efecto de transición.

stretch: variable estiramiento que se incrementa con el valor del desplazamiento.

sizeQuad: dimensión que tiene el cuadrado.

velocity: velocidad con que ocurre el efecto de transición.

vertexX: vértice *x* del cuadrado fondo.

vertexY: vértice *y* del cuadrado fondo.

RedibujaVentana: función de OpenGL que redibuja la ventana actual.

Una vez calculado los valores de los vértices del plano de fondo se construye el plano y se carga su textura, logrando que sus vértices aparezcan en forma de estiramiento al actualizarle cada uno de los valores que va ir tomando, hasta lograr interponerse ante el plano texturizado del frente.

Transición Estirar desde la esquina abajo a la izquierda

Se realiza entre dos imágenes, asignadas a dos planos, se construye primero el plano del frente, posteriormente el plano del fondo con una dimensión mucho más pequeña que el plano de frente, con el objetivo de que el plano del fondo vaya estirando sus dimensiones, a la vez que se traslada por los ejes *x* e *y* para poder cubrir el plano texturizado del frente y se quede el plano que viene trasladándose como fondo, para obtener los valores de las dimensiones del plano de fondo se tiene:

```
s= (timeII-timeI)*velocity
stretchDleft+=s
traslDLX=traslDLX – stretchDleft
traslDLY=traslDLY - stretchDleft
vertexX= vertexX + stretchDleft
vertexY= vertexY + stretchDleft
Si traslDLX menor igual 0 y vertexX mayor igual
sizeQuad
Entonces:
traslDLX igual a cero
traslDLY igual a cero
vertexX igual a sizeQuad
vertexY igual a sizeQuad
RedibujaVentana
```

donde:

s: desplazamiento con que se va lograr la transición.

timeI: tiempo inicial del efecto de transición.

timeII: tiempo final del efecto de transición.

stretchDleft : variable estiramiento de la esquina de abajo a la izquierda que se incrementa con el valor del desplazamiento.

sizeQuad: dimensión que tiene el cuadrado.

velocity: velocidad con que ocurre el efecto de transición.

vertexX: vértice x del cuadrado fondo.

vertexY: vértice y del cuadrado fondo.

traslDLX: eje de traslación *x.*

traslDLY: eje de traslación *y*.

RedibujaVentana: función de OpenGL que redibuja la ventana actual.

Una vez que se obtiene los valores de traslación y las dimensiones se va repintando la escena del plano del fondo, moviéndose gradualmente y estirándose hasta que logre interponerse al plano texturizado del frente, los valores de la traslación tienen que ser asignado a la función de la traslación como negativos para poder posicionar el plano en la posición de la transición en este caso estiramiento desde abajo a la izquierda.

Transición Estirar desde la esquina abajo a la derecha

Este efecto de transición se realiza entre dos imágenes, asignadas a dos planos, se construye primero el plano del frente, posteriormente el plano del fondo con una dimensión mucho más pequeña que el plano de frente, con el objetivo de que el plano del fondo vaya estirando sus dimensiones y trasladándose desde la derecha a la izquierda por los bordes del plano de frente hasta interponerse y quedar como fondo, los pasos para obtener los valores de la traslación y dimensión del plano de fondo son:

```
s= (timeII-timeI)*velocity
stretchDright+=1/s
traslDRX=traslDRX – stretchDright
traslDRY=traslDRY - stretchDright
vertexX= vertexX + stretchDright
vertexY= vertexY + stretchDright
Si traslDRX menor igual 0 y vertexX mayor igual sizeQuad
Entonces:
traslDRX igual a cero
traslDRY igual a cero
vertexX igual a sizeQuad
vertexY igual a sizeQuad
RedibujaVentana
```

donde:

s: desplazamiento con que se va lograr la transición.

timeI: tiempo inicial del efecto de transición.

timeII: tiempo final del efecto de transición.

stretchDright: variable estiramiento de la esquina de abajo a la derecha que se incrementa con el valor del desplazamiento.

sizeQuad: dimensión que tiene el cuadrado.

velocity: velocidad con que ocurre el efecto de transición.

vertexX: vértice x del cuadrado fondo.

vertexY: vértice y del cuadrado fondo.

traslDRX: eje de traslación *x*.

traslDRY: eje de traslación *y*.

RedibujaVentana: función de OpenGL que redibuja la ventana actual.

La obtención de estos valores van a tener la misma idea que el caso anterior de transición solo cambia la filosofía de pintar el plano, cambia el signo del vértice *x* (en este caso sería negativo) y el valor del eje *y* de traslación también, para poder posicionar el plano de fondo a la derecha y trasladar uniformemente el plano hasta que logre interponerse al otro plano que está en el frente.

Transición Estirar desde la esquina arriba a la izquierda

La transición se realiza entre dos imágenes, y asignadas a dos planos que cada uno tiene las dimensiones un cuadrado, se construye primero el plano del frente, posteriormente el plano del fondo con una dimensión mucho más pequeña que el plano de frente, con el objetivo de que el plano de fondo vaya estirando sus dimensiones y trasladándose desde arriba de izquierda a derecha.

Los pasos para obtener las dimensiones y los valores de traslación del plano de fondo son los mismos pasos del efectos de transición de la familia de estiramiento de abajo, el cambio radica en la posición donde debe estar el plano para que comience su transición desde arriba para eso solo se cambia el sentido del valor de traslación del eje *x* se escala como negativo para poder lograr mover el plano desde ese sentido hasta que se interponga al plano del frente.

Transición Estirar desde la esquina arriba a la derecha

Se efectúa entre dos imágenes, asignadas a dos planos, se construye primero el plano del frente, posteriormente el plano del fondo con una dimensión mucho más pequeña que el plano de frente, con el objetivo de que el plano de fondo vaya estirando sus dimensiones y trasladándose desde arriba de derecha a izquierda. Los pasos para obtener las dimensiones y los valores de traslación del plano de fondo es el mismo caso de los efectos de transición de la familia de estiramiento de abajo, el cambio radica en la posición donde debe estar el plano para que comience su transición desde arriba, para eso solo se cambia el sentido del valor de traslación del eje *x* y *y,* se escala como positivo al igual que los vértices del plano, para poder lograr mover el plano desde ese sentido hasta que se interponga al plano de frente.

Transición Deslizar desde la Derecha

La transición tiene lugar entre dos imágenes, asignadas a dos planos, se construye primero el plano del fondo y se traslada a otra posición que no sea el centro de visión, posteriormente se construye el plano del frente en la posición del centro de visión, la idea de esta transición es: ir deslizando o empujando el plano de fondo desde la derecha hasta que se interponga al plano de frente quedando el plano del fondo como frente, para eso se debe ir trasladando por el eje *x positivo*, el plano de fondo utilizando la función *glTranslatef(eje x,eje y,eje z),* para calcular los valores del eje *x* se tiene:

```
s= (timeII-timeI)*velocity
Si moveRX mayor que cero
Entonces:
moveRX=moveRX-s
Sino
Si moveRX menor que cero
Entonces:
moveRX=0.0
RedibujaVentana
```

donde:

s: desplazamiento con que se va lograr la transición.

timeI: tiempo inicial del efecto de transición.

timeII: tiempo final del efecto de transición.

velocity: velocidad con que ocurre el efecto de transición.

moveRX: variable que identifica el eje *x* de traslación del plano.

RedibujaVentana: función de OpenGL que redibuja la ventana actual.

Como resultado de este cálculo se va obteniendo los valores de traslación en el eje *x* positivo del plano de fondo, hasta que este se interponga al plano de frente que es cuando *movRX* sea cero, indica que el plano de fondo se encuentra en la posición del centro de visión.

Transición Deslizamiento desde la Izquierda

Este efecto de transición tiene las mismas características que el efecto tratado anterior (efecto de transición Deslizamiento desde la derecha), su variación se encuentra en la forma que se va efectuar la traslación del plano de fondo, el valor de traslación es por el eje *x* y negativo, indicando el sentido en que se va a realizar el empuje o traslación, a la izquierda hasta que llegue a interponerse al plano del frente.

El cálculo de los valores de traslación del eje *x* es el mismo que en el caso anterior, termina el efecto de transición cuando el valor del eje *x* (*movRX*) sea cero y de esa forma el plano de fondo pasa a ser el frente de visión.

La función de traslación quedaría de esta forma:

```
glTranslatef(-movRX,ejey,ejez)
glBegin(GL_QUADS);
glNormal3f (0.0f, 0.0f, 1.0f);
glVertex3f (-vertexX,-vertexY, 0.0);
glVertex3f (vertexX,-vertexY, 0.0);
glVertex3f (vertexX, vertexY, 0.0);
glVertex3f (-vertexX, vertexY, 0.0);
glEnd();
```

Transición Deslizamiento desde Arriba

Se realiza entre dos planos en forma de cuadrado, cada uno carga una imagen como textura, inicialmente se construye el plano del fondo, y se traslada en una posición no visible al centro de visión, en este caso con respecto al eje *y*, y después se construye el plano del frente visible al centro de visión. El deslizamiento se realiza en otro sentido respecto al eje *y* positivo desde arriba hacia

abajo, de esa forma va apareciendo el plano de fondo empujando en ese sentido hasta que se interponga al plano de frente, la función de acción del empuje es la traslación *glTranslatef(eje x,movUY,eje z).*

De la siguiente manera se calcula el sentido de empuje:

```
s= (timeII-timeI)*velocity
Si movUY mayor que cero
Entonces:
moveUY= moveUY- s
Sino
Si movUY menor que cero
Entonces:
moveUY=0.0
RedibujaVentana
```

donde:

s: desplazamiento con que se va lograr la transición.

timeI: tiempo inicial del efecto de transición.

timeII: tiempo final del efecto de transición.

velocity: velocidad con que ocurre el efecto de transición.

moveUY: variable que identifica el eje *y* de traslación.

RedibujaVentana: función de OpenGL que redibuja la ventana actual.

Como resultado de este cálculo se van obteniendo los valores de traslación en el eje *y* positivo del plano de fondo hasta que se interponga al plano de frente que es cuando *movUY* sea cero, indica que el plano de fondo se encuentra en la posición del centro de visión.

Transición Deslizamiento desde Abajo

Tiene las mismas características que el efecto de transición Deslizamiento desde arriba, su variación está en la forma en que se va efectuar el empuje o la traslación, por el eje *y* negativo indicando el sentido de la traslación desde abajo hacia arriba hasta que se interponga el plano de fondo al plano de frente.

El cálculo de los valores de traslación del eje *y* negativo es el mismo que en el caso anterior, el efecto de transición termina cuando el valor del eje *y* (*movUY*) sea cero y de esa forma el plano de fondo pasa a ser el frente de visión. La función de traslación quedaría la traslación *glTranslatef(eje x,-movUY,eje z).*

2.3 Modelo del Dominio

El diagrama que a continuación se muestra representa el modelo de dominio de la biblioteca STE. Este modelo ayuda a los programadores a la comprensión y análisis de la relación conceptual a diseñar.

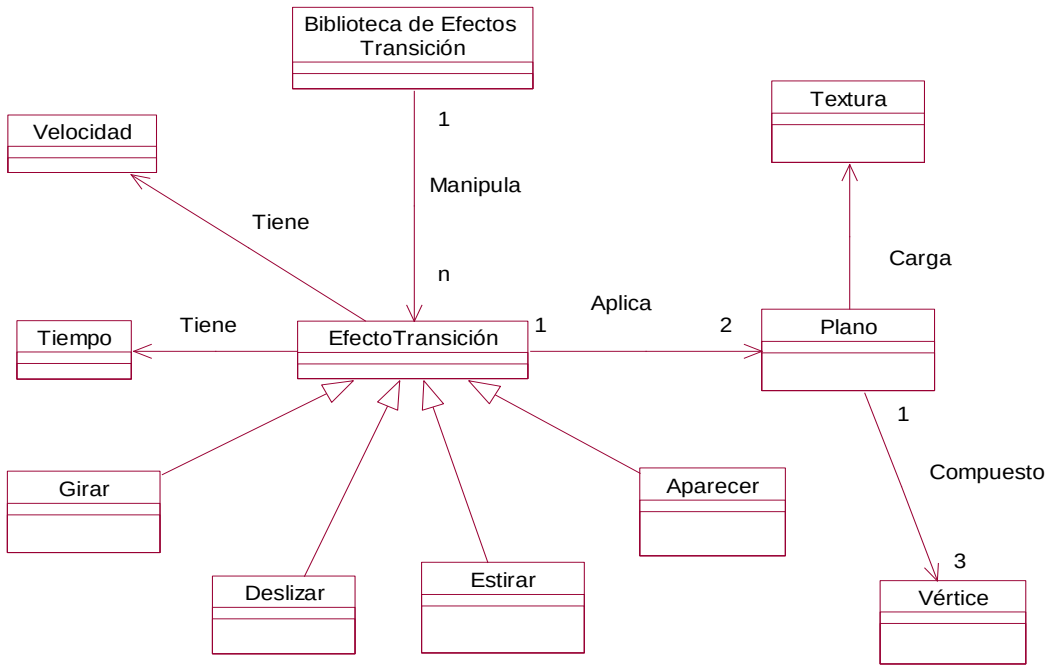

Fig. 12 Modelo de Dominio.

2.3.1 Descripción del dominio

A continuación se presentan un conjunto de conceptos que conforman la terminología del modelo de dominio, con el propósito de facilitar un mayor entendimiento de los términos manejados en el diagrama anterior.

Biblioteca STE: Controlador o manipulador de los efectos de transición y sus propiedades.

Velocidad: Rapidez de cambio del efecto que se efectúe y valor determinante para calcular conjuntamente con el tiempo el desplazamiento del efecto de transición.

Tiempo: Tiempo del sistema que se va a tener en cuenta para calcular el desplazamiento que requiere el Efecto de Transición para poder realizarse.

Plano: Figura geométrica de visibilidad de la escena a la que se le carga una textura para poder aplicar el efecto de transición.

Textura: Imagen de la superficie del plano, que es cargada a cada plano que se crea.

Girar: Efecto de transición que rota un plano determinado y a la vez amplia sus dimensiones hasta poder interponerse al plano que se encuentra como frente de visión.

Deslizar: Efecto de transición que va deslizando una plano determinado encima de otro plano hasta interponerse como frente de visión actual.

Estirar: Efecto de transición que aumenta los vértices del plano determinado hasta interponerse al plano que está como centro de visión.

Aparecer: Efecto de transición que permite la visibilidad de un plano determinado.

Vértices: Valores que se van a tener en cuenta para construir el plano en el eje de coordenada, dimensiones del plano.

2.4 Reglas del negocio

- Los efectos de transición se realizarán mediante dos planos cada uno con su textura.
- Para cargar la textura debe encontrarse dentro de la carpeta de la biblioteca STE.
- El desplazamiento de cada efecto de transición no necesariamente debe ser el mismo puede variar.

2.5 Captura de Requisitos

Los requisitos constituyen capacidades o condiciones que el sistema debe cumplir, facilitando el entendimiento entre usuarios y clientes del módulo a diseñar.

2.5.1 Requisitos Funcionales

Los siguientes requisitos establecen las funcionalidades e instrucciones que la biblioteca STE debe cumplir en su implementación.

1.- Crear Transición.

2. - Eliminar Transición.

3. - Aplicar Transición Aparecer entre planos texturizados.

4. - Aplicar Transición Deslizar entre planos texturizados en todos los sentidos.

4.1- Deslizar desde Arriba.

4.2- Deslizar desde Abajo.

4.3- Deslizar desde Izquierda.

4.4- Deslizar desde Derecha.

5. - Aplicar Transición Girar entre planos texturizados

5.1- Girar y ampliar desde el centro.

5.2- Girar y encoger desde el centro.

6. - Aplicar Transición Estirar entre planos texturizados.

6.1- Estirar desde el centro.

6.2- Estirar desde abajo de izquierda a derecha.

6.3- Estirar desde bajo de derecha a izquierda.

6.4- Estirar desde arriba de izquierda a derecha.

6.5- Estirar desde arriba de derecha a izquierda.

2.5.2 Requisitos no Funcionales

1.- Requerimientos de Software.

- El sistema debe funcionar sobre sistemas operativos Windows 2000, XP o versiones superiores.

2.- Requerimientos de Hardware.

- Memoria RAM 256 MB o superior, 10 GB de disco duro como mínimo, procesador Intel P4 a 2.0 GHz, compatibilidad con tarjetas gráficas de la familia NVIDIA y ATI.

3.- Restricciones en la Implementación y el Diseño.

- **Estándar requerido:** se regirá por el estándar definido en el trabajo para su realización.
- **Lenguaje de programación:** será implementado bajo el lenguaje C++, bajo la filosofía de Programación Orientada a Objetos.
- **Herramientas de desarrollo:** se desarrollará en la plataforma Visual Studio .Net.
- **Bibliotecas de clases:** se hará uso de la biblioteca gráfica OpenGL.

4.- Requerimientos de Usabilidad.

- Para los niveles apropiados de usabilidad se requiere de un personal calificado en el lenguaje de programación C++. Todo será implementado con terminologías del idioma inglés.

5.- Soporte

- **Compatibilidad:** compatible con la plataforma Windows, pero debe estar preparado para que con rápidas modificaciones pueda migrar para Linux.
- **Extensibilidad:** posibilidad de añadir funcionalidades adicionales o modificar la funcionalidad existente sin impactar la funcionalidad actual.

2.6 Modelo de Casos de Uso del Sistema

En esta sección se reconocen los posibles actores del sistema a desarrollar y se conciben los casos de uso del sistema. Además, se seleccionan los casos de uso correspondientes al primer ciclo de desarrollo para hacerles sus especificaciones textuales en formato expandido.

2.6.1 Actores del sistema

En este caso particular el sistema que va a hacer uso de la biblioteca STE, constituye el actor del sistema, específicamente será nombrado (AplicaciónRV) este nombre es muy abarcador pero representa a las ARV que hagan uso de las funcionalidades de la biblioteca.

Tabla 2 Actor del Sistema.

Actores	**Justificación.**
AplicaciónRV	Es quien se beneficia con las funcionalidades que brinda la biblioteca, haciendo uso de las funciones implementadas en la misma.

2.6.2 Diagrama de Casos de Uso

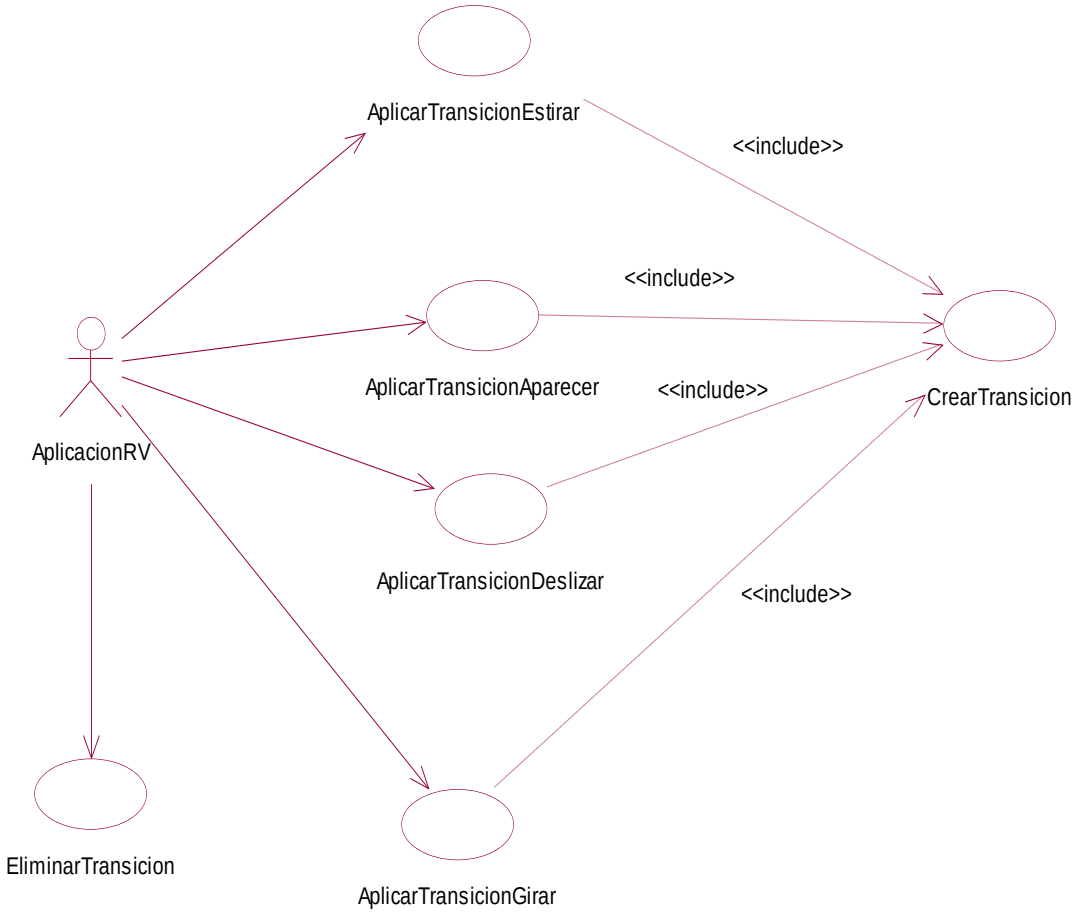

Fig. 13 Diagrama de Casos de Uso del Sistema.

2.6.3 Especificación de los casos de uso en formato expandido.

Tabla 3 CU Expandidos, Aplicar Transición Aparecer.

Caso de Uso		
CU-3	Aplicar Transición Aparecer	
Propósito	Permitir la aparición del plano del fondo texturizado de forma gradual ocupando el lugar del plano frente texturizado.	
Actores: AplicaciónRV.		
Resumen: Se inicia cuando la AplicaciónRV solicita realizar la transición Aparecer, posteriormente se llama al "CU Crear Transición", el cual define los valores de la transición y almacena dicha información. Una vez creada la transición, se utiliza su información para pintar dos planos por separados el plano del frente y el fondo, se le calcula el grado de opacidad del plano de fondo texturizado y se le aplica dicha opacidad hasta que aparezca el plano de fondo, terminando así el CU Aplicar Transición Aparecer.		
Referencias	R3	
Curso Normal de Eventos		
Acción del Actor		**Respuesta del sistema**
1._Solicita aplicar transición Aparecer.		2._Se obtienen los datos de la transición (invoca al **"CU Crear Transición"**). 3._Se calcula la opacidad. 4._Se pintan los planos texturizados por separado (frente y fondo).

	5._Se le aplica los parámetros de la función blending a cada plano. 6._Se actualiza el valor de opacidad al plano de fondo. 7._Efectúa la transición Aparecer.
Precondiciones	Debe corresponderse los parámetros a la hora de crear la transición Aparecer.
Poscondiciones	Transición Aparecer lista para efectuarse.
Prioridad	Crítico.

Tabla 4 CU Expandidos, Aplicar Transición Deslizar.

Caso de Uso	
CU-4	Aplicar Transición Deslizar
Propósito	Deslizar un plano texturizado encima de otro plano que se encuentra como el frente de visión.
Actores: AplicaciónRV.	
Resumen: El CU se inicia cuando la AplicaciónRV desea llevar a cabo la transición Deslizar, luego se llama al "CU Crear Transición", el cual define los valores de la transición y almacena dicha información. Una vez definida la información se utiliza para efectuar la transición, se procede a variar la posición de traslación con respecto al eje *X* e *Y* teniendo en cuenta la velocidad y el tiempo con que será realizado el efecto, el CU culmina cuando el plano de fondo texturizado se desliza encima del plano del frente.	
Referencias	R4.1, R4.2, R4.3, R4.4

Curso Normal de Eventos	
Acción del Actor	**Respuesta del sistema**
Sección: Deslizar desde la Izquierda.	
1._Se solicita realizar la transición Deslizar desde la izquierda.	2._Se obtienen los datos de la transición que se va efectuar (invoca al **"CU Crear Transición"**). 3._Se varía el valor del parámetro que corresponde al eje *X* por donde se trasladará el plano del fondo con respecto a la combinación del tiempo y la velocidad en este caso el resultado es el desplazamiento. 4._Se pintan los planos texturizados por separado (frente y fondo). 5._Se aplica la función de traslación al plano de fondo con su respectiva variable de cambio con respecto al eje *X* con signo negativo. 6._Efectúa la transición Deslizar desde la izquierda.
Sección: Deslizar desde la Derecha.	
1._Se solicita aplicar la transición Deslizar desde la Derecha.	2._Se obtienen los datos de la transición que se va efectuar (invoca al **"CU Crear Transición"**). 3._Se varía el valor del parámetro que corresponde al eje *X* por donde se trasladará el plano del fondo con respecto a la combinación del tiempo y la velocidad en este caso el resultado es el desplazamiento.

	4._Se pintan los planos texturizados por separado (frente y fondo). 5._Se aplica la función de traslación al plano del fondo con su respectiva variable de cambio con respecto al eje *X* con signo positivo. 6._Efectúa la transición Deslizar desde la derecha.
Sección: Deslizar desde Arriba.	
1._Se solicita realizar la transición Deslizar desde arriba.	2._Se obtienen los datos de la transición que se va efectuar (invoca al **"CU Crear Transición"**). 3._Se varía el valor del parámetro del eje *Y* por donde se trasladará el plano del fondo con respecto a la combinación del tiempo y la velocidad. 4._Se pintan los planos texturizados por separado (frente y fondo). 5._Se aplica la función de traslación al plano de fondo con su respectiva variable de cambio con respecto al eje *Y* con signo positivo. 6._Efectúa la transición Deslizar desde arriba
Sección: Deslizar desde Abajo.	
1._Se solicita aplicar la transición Deslizar desde abajo.	2._Se obtienen los datos de la transición que se va efectuar (invoca al **"CU Crear Transición"**). 3._Se varía el valor del parámetro del eje *Y* por donde se trasladará el plano del fondo con

	respecto a la combinación del tiempo y la velocidad en este caso el resultado es el desplazamiento. 4._ Se pintan los planos texturizados por separado (frente y fondo). 5._Se aplica la función de traslación al plano del fondo con su respectiva variable de cambio con respecto al eje *Y* signo negativo. 6._Efectúa la transición Deslizar desde abajo
Precondiciones	Debe corresponderse los parámetros a la hora de crear la transición Deslizar.
Poscondiciones	Transición Deslizar listo para efectuarse.
Prioridad	Crítico

Tabla 5 CU Expandidos, Aplicar Transición Girar.

Caso de Uso	
CU-5	Aplicar Transición Girar.
Propósito	Permitir que el plano del fondo texturizado aparezca girando e incrementando las dimensiones para sobreponerse al plano de frente texturizado.
Actores: AplicaciónRV.	
Resumen: El CU se inicia cuando la aplicación desea llevar a cabo la transición Girar, posteriormente se llama al "CU Crear Transición", el cual define los valores de la transición y almacena dicha información. Una vez definido los valores de la transición entonces se procede a variar los parámetros ángulo, coordenada *Z* y vértices del plano que se requiere que aparezca, todos estos valores varían con respecto al desplazamiento, pasándole los valores requerido a la	

función de rotación. El CU termina cuando el plano de frente sale de la escena de visión para que aparezca el plano de fondo.	
Referencias	R5.1, R5.2
Curso Normal de Eventos.	
Acción del Actor	**Respuesta del sistema.**
Sección: Girar y ampliar desde el centro.	
1._Se solicita realizar la transición Girar y ampliar desde el centro.	2._Se obtienen los datos de la transición que se va efectuar (invoca al **"CU Crear Transición"**). 3._Se varía el valor del ángulo de giro, dimensiones del plano de fondo y el eje *Z* con respecto a la velocidad y el tiempo. 4._Se pintan los planos texturizados por separado (frente y fondo). 5._Se aplica la función de rotación al plano de fondo con su respectiva variable de cambio con respecto al eje *Z* en sentido antihorario y el ángulo de rotación. 6._Efectúa la transición Girar y ampliar desde el centro.
Sección: Girar y encoger desde el centro.	
1._ Se solicita realizar la transición Girar y encoger desde el centro.	2._Se obtienen los datos de la transición que se va efectuar (invoca al **"CU Crear Transición"**). 3._Varía el valor del ángulo de giro,

	dimensiones del plano de frente y el eje Z con respecto a la velocidad y el tiempo que da como resultado el desplazamiento. 4._ Se pintan los planos texturizados por separado (frente y fondo). 5._ Se aplica la función de rotación al plano de frente con su respectiva variable de cambio con respecto al eje *Z* en sentido antihorario y el ángulo de rotación. 6._Efectúa la transición Girar y encoger.
Precondiciones	Debe corresponderse los parámetros a la hora de crear la transición Girar
Poscondiciones	Transición Girar en sus variantes listo para efectuarse.
Prioridad	Crítico.

Tabla 6 CU Expandidos, Aplicar Transición Estirar.

Caso de Uso	
CU-6	Aplicar Transición Estirar.
Propósito	Realizar un estiramiento al plano de fondo hasta que logre interponerse al plano de frente, siendo el plano de fondo el centro de visión final.
Actores: AplicaciónRV.	
Resumen: El CU comienza cuando la AplicaciónRV solicita realizar la transición Estirar, luego se llama al "CU Crear Transición", el cual define los valores de la transición y almacena dicha información. Una vez definidos los valores entonces se procede a variar las coordenadas del plano de fondo, y los ejes de traslación *X* e *Y* con respecto al desplazamiento, pasándole estos parámetros a la función de traslación para que el plano de fondo aparezca estirando sus coordenadas y trasladándose hasta sobreponer el plano de frente, terminando así el CU.	
Referencias	R6.1, R6.2, R6.3, R6.4, R6.5
Curso Normal de Eventos.	
Acción del Actor	**Respuesta del sistema.**
Sección: Estirar desde el centro.	
1._Se solicita realizar la transición Estirar desde el centro.	2._Se obtienen los datos de la transición que se va efectuar (invoca al **"CU Crear Transición"**). 3._Varían los valores de los parámetros de coordenadas del plano de fondo con respecto a la velocidad y el tiempo. 4._Se pintan los planos texturizados por separado (frente y fondo).

	5._Efectúa la transición Estirar desde el centro.
Sección: Estirar desde abajo de izquierda a derecha.	
1._Se solicita realizar la transición Estirar desde abajo de izquierda a derecha.	2._Se obtienen los datos de la transición que se va efectuar (invoca al **"CU Crear Transición"**). 3._Varían los valores de los parámetros de coordenadas del plano de fondo, los ejes *X* e *Y* de traslación con respecto a la velocidad y el tiempo. 4._Se pintan los planos texturizados por separado (frente y fondo). 5._ Se aplica la función de traslación al plano de fondo con su respectiva variable de cambio con respecto al eje *X* e *Y* con valores negativos. 6._Efectúa la transición Estirar desde abajo, de izquierda a derecha.
Sección: Estirar desde abajo de derecha a izquierda.	
1._Se solicita realizar la transición Estirar desde abajo de derecha a izquierda.	2._Se obtienen los datos de la transición que se va efectuar (invoca al **"CU Crear Transición"**). 3._Varían los valores de los parámetros de coordenadas del plano de fondo, los ejes *X* e *Y* de traslación con respecto a la velocidad y el tiempo.

	4._Se pintan los planos texturizados por separado (frente y fondo). 5._Se aplica la función de traslación al plano de fondo con su respectiva variable de cambio con respecto al eje *X* con valor positivo y al eje *Y* con valor negativo. 6._Efectúa la transición Estirar desde abajo, de derecha a izquierda.
Sección: Estirar desde arriba de izquierda a derecha.	
1._Se solicita realizar la transición Estirar desde arriba, de izquierda a derecha.	2._Se obtienen los datos de la transición que se va efectuar (invoca al **"CU Crear Transición"**). 3._Varían los valores de los parámetros de coordenadas del plano de fondo, los ejes *X* e *Y* de traslación con respecto a la velocidad y el tiempo. 4._Se pintan los planos texturizados por separado (frente y fondo). 5._Se aplica la función de traslación al plano de fondo con su respectiva variable de cambio con respecto al eje *X* con valor negativo y el eje *Y* positivo. 6._Efectúa la transición Estirar desde arriba, de izquierda a derecha
Sección: Estirar desde arriba de derecha a izquierda.	
1._Se solicita realizar la transición Estirar desde	2._Se obtienen los datos de la transición

arriba, de derecha a izquierda.	que se va efectuar (invoca al **"CU Crear Transición"**). 3._Varían los valores de los parámetros de coordenadas del plano de fondo, los ejes *X* e *Y* de traslación con respecto a la velocidad y el tiempo. 4._Se pintan los planos texturizados por separado (frente y fondo). 5._Se aplica la función de traslación al plano de fondo texturizado con su respectiva variable de cambio con respecto al eje *X* e *Y* con valor positivo. 6._Efectúa la transición Estirar desde arriba, de derecha a izquierda.
Precondiciones	Debe corresponderse los parámetros a la hora de crear la transición Estirar.
Poscondiciones	Transición Estirar listo para efectuarse.
Prioridad	Crítico.

Tabla 7 CU Expandidos, Eliminar Transición.

Caso de Uso	
CU-2	Eliminar Transición.
Propósito	Eliminar la transición que ya fue aplicada para darle viabilidad al funcionamiento de otra transición.
Actores: AplicaciónRV.	
Resumen: El CU comienza cuando la AplicaciónRV solicita eliminar la transición que está ejecutándose para indicar el funcionamiento de otra. El CU termina cuando la estructura de datos esté vacía y lista para agregar otra transición.	
Referencias	R2.
Curso Normal de Eventos.	
Acción del Actor	**Respuesta del sistema.**
1._ Selecciona la transición que desea eliminar	2._Busca la transición que desea eliminar. 3._Elimina dicha transición de la estructura de datos con toda su información.
Precondiciones	Debe encontrarse en ejecución.
Poscondiciones	Listo para almacenar y ejecutar otra transición.
Prioridad	Crítico.

Tabla 8 CU Expandidos, Crear Transición.

Caso de Uso	
CU-1	Crear Transición.
Propósito	Construir el efecto de transición con sus parámetros correspondientes, dependiendo del tipo de transición que se desee crear y guardarlo en una estructura de datos.
Actores: *AplicaciónRV*	
Resumen: Se inicia cuando unos de los CU que lo puede iniciar solicitan definir la transición que le corresponde. Entonces procede a crear la transición con los parámetros que lo definen y lo guarda en una estructura de datos. El CU finaliza cuando la información de dicha transición esté almacenada.	
Referencias	R1.
Curso Normal de Eventos.	
Acción del Actor	**Respuesta del sistema.**
1._ Solicita crear una transición pasándole los parámetros correspondientes a este.	2._Define la transición con sus parámetros. 3._Se guarda dicha transición en una estructura de datos con toda su información.
Precondiciones	Debe encontrarse la estructura de datos vacía. Definir los parámetros correspondientes a la transición que se desea crear.
Poscondiciones	La transición queda lista para utilizarse.
Prioridad	Crítico.

2.7 Consideraciones Finales

- Se especificaron las soluciones técnicas que se le dan a este proyecto.
- Partiendo del desarrollo de este capítulo se diseñará una estructura de clases en correspondencia con las técnicas citadas y la realización de los algoritmos planteados.
- Se establecieron además los requisitos funcionales y no funcionales, propiciando el paso a la siguiente etapa.
- Se realizaron las descripciones detalladas de los casos de uso a implementar, apoyados en el modelo de los casos de uso del sistema.

3 Diseño e Implementación

3.1 Introducción

Se presenta el diseño del sistema propuesto, con diagramas de clases, donde se definen las responsabilidades de estas y sus relaciones. Además se presentan otros artefactos involucrados en el diseño como los diagramas de secuencia por CU, facilitando con esto el entendimiento de la biblioteca. En esta etapa del producto además del paso del diseño de clases a la creación de componentes físicos, que se traducen en ficheros .cpp correspondiente a la implementación en C++, se enuncian los estándares para la nomenclatura.

3.2 Arquitectura de la biblioteca STE

La arquitectura de la biblioteca STE, está basada en el patrón arquitectónico N-Capas, el cual consiste en separar las funcionalidades en varias capas y ubicar en cada una de ellas los componentes correspondientes según sus responsabilidades. Permitiendo un diseño flexible y extensible, además de una escalabilidad considerable, ajustándose siempre a las necesidades del usuario.

Para el caso específico de la biblioteca STE, se implementan dos capas lógicas fundamentales. La capa de Presentación que contiene las clases que manejan la información de los efectos de transición que posee la biblioteca; y la capa de Lógica de Negocio, que engloba las clases que implementan cada uno de los efectos de transición. El objetivo de estas clases es recibir y ejecutar las solicitudes que se hagan de la capa Presentación y devolverle a esta un resultado.

Representación de la Arquitectura:

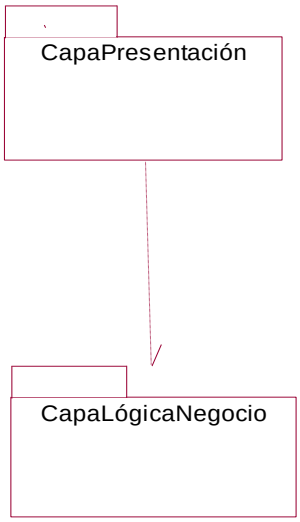

Fig. 14 Arquitectura dos capas.

3.3 Patrones de Diseño utilizados en la biblioteca STE

La utilización de los patrones de diseño son soluciones claves para lograr un buen diseño de clases, exhibe un amplio repertorio de principios que guían la creación de sistemas con calidad en el diseño. La arquitectura de la biblioteca STE impone el uso de patrones con el fin lograr una estructura y diseño adecuados, que se ajusten a los requisitos.

Los patrones **GRASP** son el acrónimo de General Responsibility Asignment Software Patterns (patrones generales de software para asignar responsabilidades), su objetivo es diseñar sistemas o módulos eficaces orientados a objetos.

Patrón de diseño **Experto**

Su objetivo fundamental es asignar responsabilidades de creación de un objeto a la clase que conoce toda la información necesaria para crearlo. Dentro de la biblioteca esta responsabilidad recae sobre la clase CLibrary, la cual contiene toda la información necesaria de todos los efectos de transiciones para crearlos y ejecutarlos.

Patrón de diseño **Creador**

Define quién es el encargado de la creación de nuevos objetos e instancias, en este caso la clase CLibrary es la encargada de la creación de instancias de cada uno de los efectos de transiciones, debido a que contiene toda la información de cada uno y la usa directamente.

Patrón de diseño **Controlador**

Funciona como intermediario entre una determinada interfaz y el algoritmo que la implementa. Es decir, es el encargado de recibir las peticiones y enviarlas a las distintas clases según el método llamado.

En este caso, es implementado por la clase CLibrary, la cual tiene la responsabilidad de realizar llamadas para lograr el renderizado de cada uno de los efectos de transiciones e implementa las funcionalidades de la biblioteca STE.

Patrón de diseño **Bajo Acoplamiento**

Propone un diseño donde la dependencia entre las clases es mínima estando lo menos ligadas entre sí, permitiendo la modificación de algunas sin afectaciones considerables en las otras clases. El diseño de la biblioteca STE tiene en cuenta este patrón.

Además de aplicar los patrones GRASP en la arquitectura de la biblioteca STE, también se da uso del patrón Fachada, patrón estructural perteneciente al grupo de patrones de diseño Gang-Of-Four (GoF), su nombre se refiere a los cuatros autores que lo crearon, este patrón describe soluciones habituales en el diseño de software.

Patrón de diseño **Fachada**

Provee una interfaz unificada sencilla que hace de intermediaria entre un cliente y una interfaz o grupo de interfaces más complejas .Es decir, simplifica el acceso a un grupo de objetos relacionados, proporcionando un objeto de comunicación de alto nivel. Una fachada, conoce qué componentes son responsables de qué peticiones y así, es capaz de delegarlas a los componentes apropiados.

En el caso de la biblioteca STE se utiliza dicho patrón para exponer todas las funcionalidades de la capa lógica de negocio que serán utilizadas por los componentes de la capa de presentación, dicha fachada será la única vía de comunicación entre las dos capas. Todos los componentes de la capa presentación que necesiten de los servicios de la capa lógica de negocio, tendrán visibilidad directa hacia las fachadas mediante la clase CLibrary, que será la encargada de la comunicación y ejecución de las transiciones.

3.4 Diagrama de Diseño de clases por CU

El diseño de clases, se desglosa en diagramas de clases por cada CU que se obtuvieron en la etapa de requerimientos para dar una mejor organización y estructuración.

3.4.1 Diagrama de diseño de clases por CU "Aplicar Transición Aparecer"

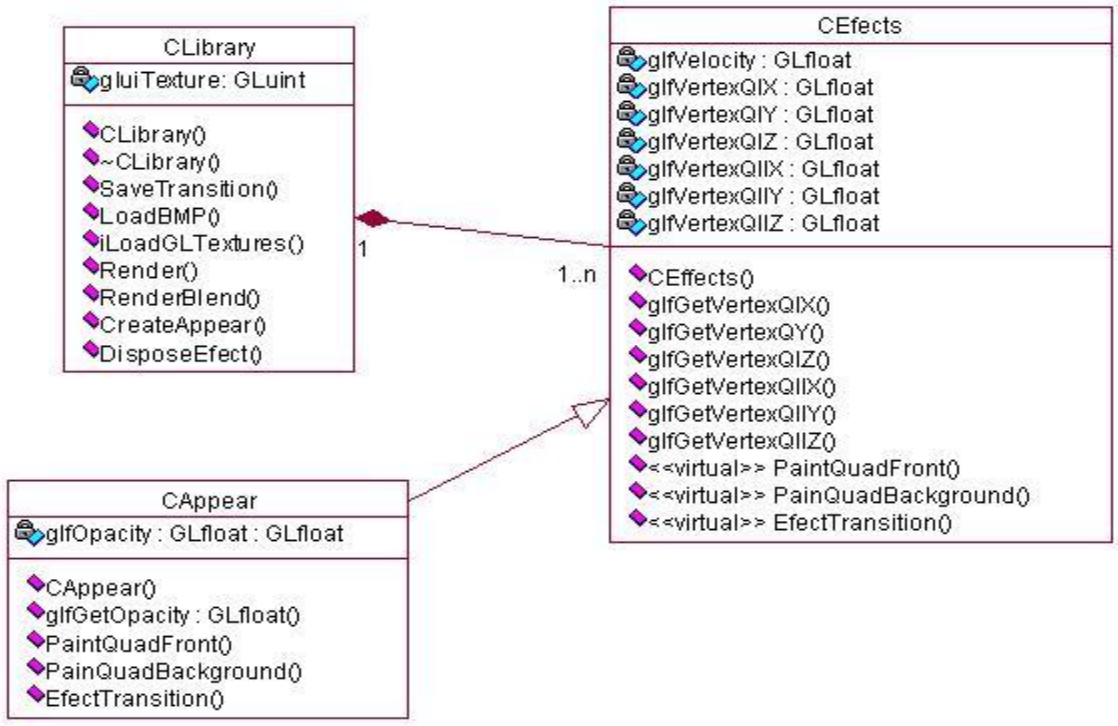

Fig. 15 Diagrama de clases CU "Aplicar Transición Aparecer".

3.4.2 Diagrama de diseño de clases por CU “Aplicar Transición Deslizar”

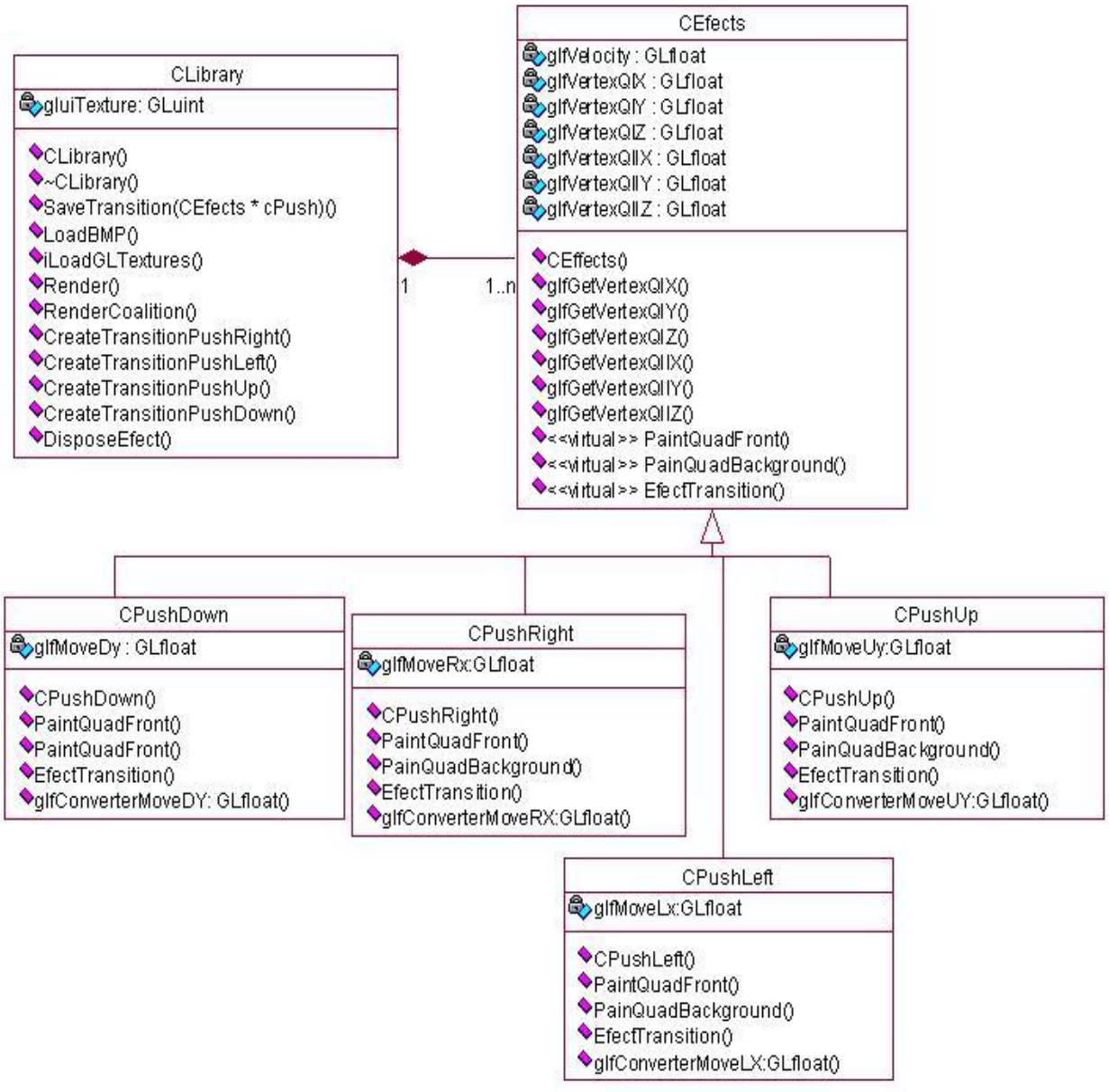

Fig. 16 Diagrama de clases CU "Aplicar Transición Deslizar".

3.4.3 Diagrama de diseño de clases por CU" Aplicar Transición Estirar"

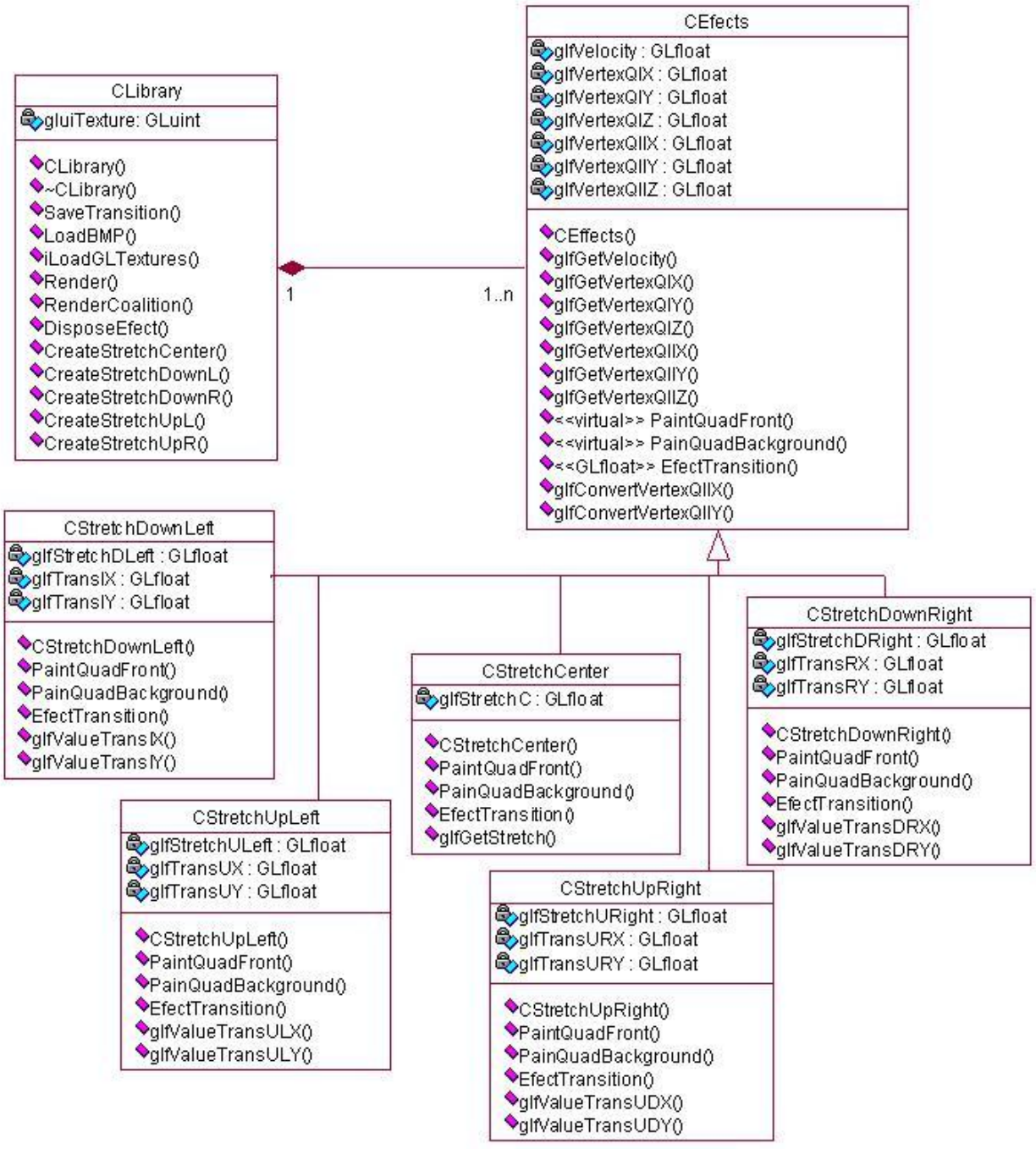

Fig. 17 Diagrama de clases CU "Aplicar Transición Estirar".

3.4.4 Diagrama de diseño de clases por CU" Aplicar Transición Girar"

Fig. 18 Diagrama de clases "Aplicar Transición Girar".

3.5 Descripción de las clases

Tabla 9 Descripción de la clase CLibrary.

Nombre: CLibrary		
Tipo de clase: Controladora		
Atributo		**Tipo**
vtPointEfect		vector<CEfects*>
gluiTexture [2]		GLuint
Para cada responsabilidad		
Nombre	**CLibrary()**	
Descripción	Constructor de la clase.	
Nombre	**~CLibrary**	
Descripción	Destructor de la clase.	
Nombre	**AUX_RGBImageRec *LoadBMP(**char *pcFilename**)**	
Descripción	Función auxiliar que carga el fichero de la textura.	
Nombre	**iLoadGLTextures (**char * pcTexture1, char * pcTexture2**)**	
Descripción	Parametriza las texturas cargadas y lo almacena.	
Nombre	**Render()**	
Descripción	Asigna el render que debe ejecutarse de acuerdo al efecto de transición que se seleccione.	
Nombre	**RenderBlend()**	

Descripción	Ejecuta la forma en que se va a pintar la escena de la transición Aparecer.
Nombre	**RenderCoalition()**
Descripción	Ejecuta la forma en que se va a efectuar las transiciones que no tengan vínculos con el Aparecer y el Girar- Contraer.
Nombre	**RenderSpin()**
Descripción	Ejecuta la forma en que se va a pintar la escena de la transición Girar – Contraer.
Nombre	**SaveTransition(**CEfects * tcEfect**)**
Descripción	Adiciona al vector el efecto de transición.
Nombre	**DisposeLibrary()**
Descripción	Limpia el vector de transiciones.
Nombre	**CreateTransitionPushRight(**GLfloat arg_glfVelocity, GLfloat arg_glfVertexQIX,GLfloat arg_glfVertexQIY, GLfloat arg_glfVertexQIZ, GLfloat arg_glfVertexQIIX, GLfloat arg_glfVertexQIIY,GLfloat arg_glfVertexQIIZ,GLfloat arg_glfMoveRx**)**
Descripción	Define los parámetros de la transición Deslizar a la derecha para almacenar dicha información.
Nombre	**CreateTransitionPushLeft (**GLfloat arg_glfVelocity, GLfloat arg_glfVertexQIX,GLfloat arg_glfVertexQIY, GLfloat arg_glfVertexQIZ, GLfloat arg_glfVertexQIIX, GLfloat arg_glfVertexQIIY, GLfloat arg_glfVertexQIIZ, GLfloat arg_glfMoveLx**)**
Descripción	Define los argumentos de la transición Deslizar a la izquierda para almacenar dicha información.
Nombre	**CreateTransitionPushUp(**GLfloat arg_glfVelocity, GLfloat arg_glfVertexQIX,

	GLfloat arg_glfVertexQIY,GLfloat arg_glfVertexQIZ, GLfloat arg_glfVertexQIIX, GLfloat arg_glfVertexQIIY, GLfloat arg_glfVertexQIIZ, GLfloat arg_glfMoveUy**)**
Descripción	Define los argumentos de la transición Deslizar desde arriba para almacenar dicha información.
Nombre	**CreateTransitionPushDown(**GLfloat arg_glfVelocity,GLfloat arg_glfVertexQIX,GLfloat arg_glfVertexQIY,GLfloat arg_glfVertexQIZ, GLfloat arg_glfVertexQIIX, GLfloat arg_glfVertexQIIY, GLfloat arg_glfVertexQIIZ, GLfloat arg_glfMoveDy**)**
Descripción	Define los argumentos de la transición Deslizar desde abajo para almacenar dicha información.
Nombre	**CreateTransitionSpinC(**GLfloat arg_glfVelocity,GLfloat arg_glfVertexQIX, GLfloat arg_glfVertexQIY, GLfloat arg_glfVertexQIZ, GLfloat arg_glfVertexQIIX, GLfloat arg_glfVertexQIIY, GLfloat arg_glfVertexQIIZ, GLfloat arg_glfAngle, GLfloat arg_glfMoveZ, GLfloat arg_glfIncrement**)**
Descripción	Define los argumentos de la transición Girar y ampliar desde el centro para almacenar dicha información.
Nombre	**CreateTransitionSpinCont(**GLfloat arg_glfVelocity, GLfloat arg_glfVertexQIX,GLfloat arg_glfVertexQIY, GLfloat arg_glfVertexQIZ, GLfloat arg_glfVertexQIIX, GLfloat arg_glfVertexQIIY, GLfloat arg_glfVertexQIIZ, GLfloat arg_glfAngleContract, GLfloat arg_glfMoveRote, GLfloat arg_glfContract**)**
Descripción	Define los parámetros de la transición Girar y encoger desde el centro para almacenar dicha información.

Nombre	**CreateStretchCenter(**GLfloat arg_glfVelocity, GLfloat arg_glfVertexQIX, GLfloat arg_glfVertexQIY, GLfloat arg_glfVertexQIZ, GLfloat arg_glfVertexQIIX, GLfloat arg_glfVertexQIIY,GLfloat arg_glfVertexQIIZ,GLfloat arg_glfStretchC**)**
Descripción	Define los parámetros de la transición Estirar desde centro para almacenar dicha información.
Nombre	**CreateStretchDownL(**GLfloat arg_glfVelocity, GLfloat arg_glfVertexQIX, GLfloat arg_glfVertexQIY, GLfloat arg_glfVertexQIZ,GLfloat arg_glfVertexQIIX,GLfloat arg_glfVertexQIIY, GLfloat arg_glfVertexQIIZ, GLfloat arg_glfStretchDLeft, GLfloat arg_glfTranslX, GLfloat arg_glfTranslY**)**
Descripción	Define los parámetros de la transición Estirar desde abajo empezando por el lado izquierdo, para almacenar dicha información.
Nombre	**CreateStretchDownR(**GLfloat arg_glfVelocity, GLfloat arg_glfVertexQIX, GLfloat arg_glfVertexQIY, GLfloat arg_glfVertexQIZ,GLfloat arg_glfVertexQIIX,GLfloat arg_glfVertexQIIY, GLfloat arg_glfVertexQIIZ,GLfloat arg_glfStretchDRight, GLfloat arg_glfTransRX, GLfloat arg_glfTransRY**)**
Descripción	Define los parámetros de la transición Estirar desde abajo empezando por el lado derecho, para almacenar dicha información.
Nombre	**CreateStretchUpL(**GLfloat arg_glfVelocity, GLfloat arg_glfVertexQIX, GLfloat arg_glfVertexQIY, GLfloat arg_glfVertexQIZ,GLfloat arg_glfVertexQIIX,GLfloat arg_glfVertexQIIY, GLfloat arg_glfVertexQIIZ,GLfloat arg_glfStretchULeft, GLfloat arg_glfTransUX, GLfloat arg_glfTransUY**)**
Descripción	Define los parámetros de la transición Estirar desde arriba empezando por el lado izquierdo, para almacenar dicha información.
Nombre	**CreateStretchUpR(**GLfloat arg_glfVelocity, GLfloat arg_glfVertexQIX, GLfloat arg_glfVertexQIY, GLfloat arg_glfVertexQIZ,GLfloat arg_glfVertexQIIX, GLfloat arg_glfVertexQIIY,GLfloat arg_glfVertexQIIZ,GLfloat arg_glfStretchURight,

	GLfloat arg_glfTransURX, GLfloat arg_glfTransURY**)**
Descripción	Define los parámetros de la transición Estirar desde arriba empezando por el lado derecha, para almacenar dicha información.
Nombre	**CreateAppear(**GLfloat arg_glfVelocity, GLfloat arg_glfVertexQIX, GLfloat arg_glfVertexQIY, GLfloat arg_glfVertexQIZ,GLfloat arg_glfVertexQIIX,GLfloat arg_glfVertexQIIY,GLfloat arg_glfVertexQIIZ,GLfloat arg_glfOpacity**)**
Descripción	Define los parámetros de la transición Aparecer, para almacenar dicha información.

Tabla 10 Descripción de la clase CEfects.

Nombre: CEfects	
Tipo de clase: Entidad	
Atributo	**Tipo**
glfVelocity	GLfloat
glfVertexQIX	GLfloat
glfVertexQIY	GLfloat
glfVertexQIZ	GLfloat
glfVertexQIIX	GLfloat
glfVertexQIIY	GLfloat
glfVertexQIIZ	GLfloat
Para cada responsabilidad	
Nombre	**CEfects()**

Descripción	Constructor de la clase CEfects.
Nombre	**glfGetVelocity()**
Descripción	Devuelve la velocidad del efecto.
Nombre	**glfConvertVertexQIIX()**
Descripción	Inicializa la posición inicial del vértice X del plano texturizado de fondo con respecto al valor del vértice X del plano texturizado frente.
Nombre	**glfConvertVertexQIIY()**
Descripción	Inicializa la posición inicial del vértice Y del plano texturizado de fondo con respecto al valor del vértice Y del plano texturizado frente.
Nombre	**PaintQuadFront()**
Descripción	Función virtual para definir el plano frente.
Nombre	**PainQuadBackground()**
Descripción	Función virtual para definir el plano fondo.
Nombre	**EfectTransition()**
Descripción	Función virtual para variar los parámetros que estén relacionados con la transición que se desea aplicar.

Tabla 11 Descripción de la clase CAppear.

Nombre: CAppear	
Tipo de clase: Entidad	
Atributo	**Tipo**
glfOpacity	GLfloat
Para cada responsabilidad	
Nombre	**CAppear()**
Descripción	Constructor de la clase CAppear.
Nombre	**glfGetOpacity()**
Descripción	Devuelve la opacidad del efecto de transición Aparecer.
Nombre	**PaintQuadFront()**
Descripción	Define el plano frente con sus parámetros y su función de blending.
Nombre	**PainQuadBackground()**
Descripción	Define el plano fondo con sus parámetros y su función de blending.
Nombre	**EfectTransition()**
Descripción	Calcula los valores de los parámetros que estén relacionados con la transición que se desea aplicar, en este caso el valor de la opacidad, redibujándolo con ese parámetro.

Tabla 12 Descripción de la clase CPushUp.

Nombre: CPushUp		
Tipo de clase: Entidad		
Atributo		**Tipo**
glfMoveUy		GLfloat
Para cada responsabilidad		
Nombre	**CPushUp()**	
Descripción	Constructor de la clase CPushUp	
Nombre	**PaintQuadFront()**	
Descripción	Define el plano frente con sus parámetros.	
Nombre	**PainQuadBackground()**	
Descripción	Define el plano fondo con sus parámetros y su función de traslación con respecto a la variación del parámetro Y.	
Nombre	**EfectTransition()**	
Descripción	Calcula la variación de los parámetros que estén implicado en esta transición de Deslizar desde arriba, en este caso el movimiento con respecto a Y, redibujándolo con ese parámetro.	
Nombre	**glfConverterMoveUY()**	
Descripción	Inicializa el movimiento con respecto al eje Y.	

Tabla 13 Descripción de la clase CPushDown.

Nombre: CPushDown	
Tipo de clase: Entidad	
Atributo	**Tipo**
glfMoveDy	GLfloat
Para cada responsabilidad	
Nombre	**CPushDown()**
Descripción	Constructor de la clase CPushDown.
Nombre	**PaintQuadFront()**
Descripción	Define el plano frente con sus parámetros.
Nombre	**PainQuadBackground()**
Descripción	Define el plano fondo con sus parámetros y su función de traslación con respecto a la variación del parámetro Y en sentido negativo.
Nombre	**EfectTransition()**
Descripción	Calcula la variación de los parámetros que estén implicado en esta transición de Deslizar desde abajo, en este caso el movimiento con respecto a Y, redibujándolo con ese parámetro.
Nombre	**glfConverterMoveDY()**
Descripción	Inicializa el valor movimiento con respecto al eje Y.

Tabla 14 1 Descripción de la clase CPushRight.

Nombre: CPushRight		
Tipo de clase: Entidad		
Atributo		**Tipo**
glfMoveRx		GLfloat
Para cada responsabilidad		
Nombre	**CPushRight()**	
Descripción	Constructor de la clase CPushRight.	
Nombre	**PaintQuadFront()**	
Descripción	Define el plano frente con sus parámetros.	
Nombre	**PainQuadBackground()**	
Descripción	Define el plano fondo con sus parámetros y su función de traslación con respecto a la variación del parámetro X.	
Nombre	**EfectTransition()**	
Descripción	Calcula la variación de los parámetros que estén implicados en esta transición de Deslizar desde la derecha, en este caso el movimiento con respecto a X en sentido negativo, redibujándolo con ese parámetro.	
Nombre	**glfConverterMoveRX()**	
Descripción	Inicializa el valor del movimiento con respecto al eje X.	

Tabla 15 Descripción de la clase CPushLeft.

Nombre: CPushLeft		
Tipo de clase: Entidad		
Atributo		**Tipo**
glfMoveLx		GLfloat
Para cada responsabilidad		
Nombre	**CPushLeft()**	
Descripción	Constructor de la clase CPushLeft.	
Nombre	**PaintQuadFront()**	
Descripción	Define el plano frente con sus parámetros.	
Nombre	**PainQuadBackground()**	
Descripción	Define el plano fondo con sus parámetros y su función de traslación con respecto a la variación del parámetro X.	
Nombre	**EfectTransition()**	
Descripción	Calcula la variación de los parámetros que estén implicados en esta transición Deslizar desde la izquierda, en este caso el movimiento con respecto a X en sentido positivo, redibujándolo con ese parámetro.	
Nombre	**glfConverterMoveLX()**	
Descripción	Inicializa el valor del movimiento con respecto al eje X.	

Tabla 16 2 Descripción de la clase CSpinCenter.

Nombre: CSpinCenter	
Tipo de clase: Entidad	
Atributo	**Tipo**
glfAngle	GLfloat
glfMoveZ	GLfloat
glfIncrement	GLfloat
Para cada responsabilidad	
Nombre	**CSpinCenter()**
Descripción	Constructor de la clase CSpinCenter.
Nombre	**PaintQuadFront()**
Descripción	Define el plano frente con sus parámetros.
Nombre	**PainQuadBackground()**
Descripción	Define el plano fondo con sus parámetros y su función de rotación con respecto a la variación de los parámetros ángulo y el eje de rotación Z.
Nombre	**EfectTransition()**
Descripción	Calcula la variación de los parámetros que estén implicados en esta transición de Giro, en este caso el ángulo, eje *Z* de rotación, y los vértices del plano de fondo, redibujándolo con esos parámetros.

Tabla 17 Descripción de la clase CSpinContract.

Nombre: CSpinContract	
Tipo de clase: Entidad	
Atributo	**Tipo**
glfAngleContract	GLfloat
glfContract	GLfloat
glfMoveRote	GLfloat
Para cada responsabilidad	
Nombre	**CSpinContract()**
Descripción	Constructor de la clase CSpinContract.
Nombre	**PaintQuadFront()**
Descripción	Define el plano frente con sus parámetros y su función de rotación con respecto a la variación de los parámetros ángulo y el eje de rotación Z.
Nombre	**PainQuadBackground()**
Descripción	Define el plano fondo con sus parámetros.
Nombre	**EfectTransition()**
Descripción	Calcula la variación de los parámetros que estén implicados en esta transición de Giro, en este caso el ángulo y el eje *Z* de rotación, redibujándolo con esos parámetros.

Tabla 18 Descripción de la clase CStretchCenter.

Nombre: CStretchCenter	
Tipo de clase: Entidad	
Atributo	**Tipo**
glfStretchC	GLfloat
Para cada responsabilidad	
Nombre	**CStretchCenter()**
Descripción	Constructor de la clase CStretchCenter.
Nombre	**glfGetStretch()**
Descripción	Devuelve el valor de incremento del estiramiento.
Nombre	**PaintQuadFront()**
Descripción	Define el plano frente con sus parámetros.
Nombre	**PainQuadBackground()**
Descripción	Define el plano fondo con sus parámetros.
Nombre	**EfectTransition()**
Descripción	Calcula la variación de los parámetros que estén implicados en esta transición de Estiramiento, variando así los vértices X e Y del plano fondo, redibujándolo con esos parámetros.

Tabla 19 Descripción de la clase CStretchDownRight.

Nombre: CStretchDownRight	
Tipo de clase: Entidad	
Atributo	**Tipo**
glfStretchDRight	GLfloat
glfTransRX	GLfloat
glfTransRY	GLfloat
Para cada responsabilidad	
Nombre	**CStretchDownRight()**
Descripción	Constructor de la clase CStretchDownRight.
Nombre	**PaintQuadFront()**
Descripción	Define el plano frente con sus parámetros.
Nombre	**PainQuadBackground()**
Descripción	Define el plano fondo con sus parámetros y su función de traslación con respecto a la variación de los parámetros del eje X positivo, Y en sentido negativo.
Nombre	**EfectTransition()**
Descripción	Calcula la variación de los parámetros que estén implicados en esta transición de Estiramiento abajo desde la derecha, en este caso el eje de traslación X e Y, los vértices del plano fondo, redibujándolo con esos parámetros.
Nombre	**glfValueTransDRX()**

Descripción	Inicializa el valor del atributo fTransRX (traslación con respecto al eje X).
Nombre	**glfValueTransDRY()**
Descripción	Inicializa el valor del atributo fTransRY (traslación con respecto al eje Y).

Tabla 20 Descripción de la clase CStretchUpLeft.

Nombre: CStretchUpLeft	
Tipo de clase: Entidad	
Atributo	**Tipo**
glfStretchULeft	GLfloat
glfTransUX	GLfloat
glfTransUY	GLfloat
Para cada responsabilidad	
Nombre	**CStretchUpLeft()**
Descripción	Constructor de la clase CStretchUpLeft
Nombre	**PaintQuadFront()**
Descripción	Define el plano frente con sus parámetros.
Nombre	**PainQuadBackground()**
Descripción	Define el plano fondo con sus parámetros y su función de traslación con respecto a la variación de los parámetros del eje X negativo e Y en sentido positivo.
Nombre	**EfectTransition()**

Descripción	Calcula la variación de los parámetros que estén implicados en esta transición de Estiramiento arriba desde la izquierda, en este caso el eje de traslación X e Y, y los vértices del plano de fondo, redibujándolo con esos parámetros.
Nombre	**glfValueTransULX()**
Descripción	Inicializa el valor del atributo fTransUX (traslación con respecto al eje X).
Nombre	**glfValueTransULY()**
Descripción	Inicializa el valor del atributo fTransUY (traslación con respecto al eje Y).

Tabla 21 Descripción de la clase CStretchUpRight.

Nombre: CStretchUpRight	
Tipo de clase: Entidad	
Atributo	**Tipo**
glfStretchURight	GLfloat
glfTransURX	GLfloat
glfTransURY	GLfloat
Para cada responsabilidad	
Nombre	**CStretchUpRight()**
Descripción	Constructor de la clase CStretchUpRight
Nombre	**PaintQuadFront()**
Descripción	Define el plano frente con sus parámetros.

Nombre	**PainQuadBackground()**
Descripción	Define el plano fondo con sus parámetros y su función de traslación con respecto a la variación de los parámetros del eje X e Y en sentido positivo.
Nombre	**EfectTransition()**
Descripción	Calcula la variación de los parámetros que estén implicados en esta transición de Estiramiento arriba desde la izquierda, en este caso el eje de traslación X e Y, y los vértices del plano de fondo, redibujándolo con esos parámetros.
Nombre	**glfValueTransUDX()**
Descripción	Inicializa el valor del atributo fTransURX (traslación con respecto al eje X).
Nombre	**glfValueTransUDY()**
Descripción	Inicializa el valor del atributo fTransURY (traslación con respecto al eje Y).

Tabla 22 Descripción de la clase CStretchDownLeft.

Nombre: CStretchDownLeft	
Tipo de clase: Entidad	
Atributo	**Tipo**
glfStretchDLeft	GLfloat
glfTranslX	GLfloat
glfTranslY	GLfloat
Para cada responsabilidad	
Nombre	**CStretchDownLeft()**

Descripción	Constructor de la clase CStretchDownLeft
Nombre	**PaintQuadFront()**
Descripción	Define el plano frente con sus parámetros.
Nombre	**PainQuadBackground()**
Descripción	Define el plano fondo con sus parámetros y su función de traslación con respecto a la variación de los parámetros del eje X e Y en sentido negativo.
Nombre	**EfectTransition()**
Descripción	Calcula la variación de los parámetros que estén implicados en esta transición de Estiramiento arriba desde la izquierda, en este caso el eje de traslación X e Y, y los vértices del plano de fondo, redibujándolo con esos parámetros.
Nombre	**glfValueTranslX()**
Descripción	Inicializa el valor del atributo fTranslX (traslación con respecto al eje X).
Nombre	**glfValueTranslY()**
Descripción	Inicializa el valor del atributo fTranslY (traslación con respecto al eje Y).

3.6 Diagramas de Secuencia

Muestran el flujo de mensajes de un objeto a otro y como tales, representan los métodos y los eventos soportados por un objeto o clase. Seguidamente se muestran los diagramas de secuencias correspondientes a los CU del módulo a implementar.

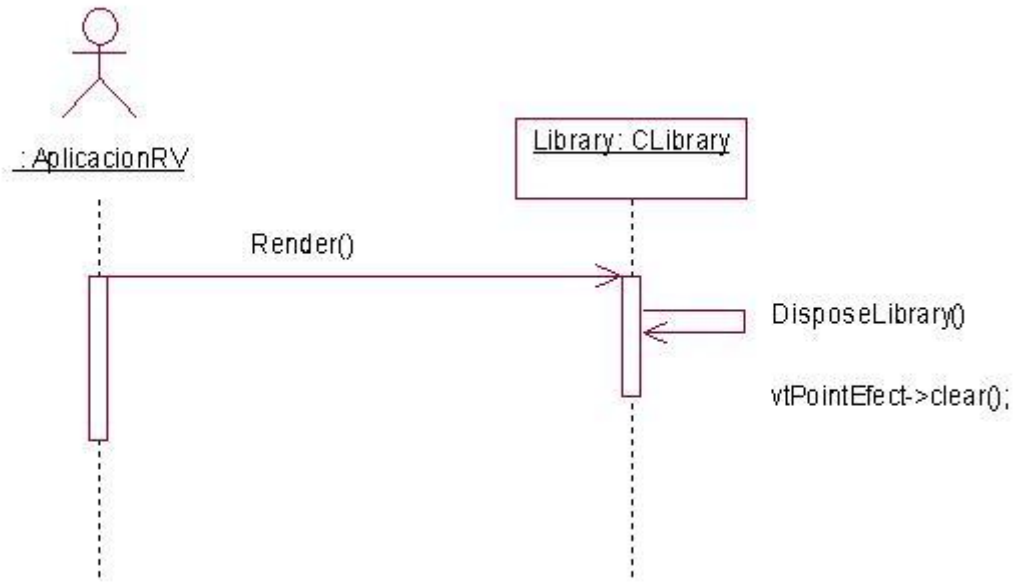

Fig. 19 Diagrama de Secuencia "Eliminar Transición".

En este diagrama se muestra el flujo de secuencia para eliminar la transición. Inicialmente se llama al visualizador gráfico, luego se elimina la transición que está ejecutándose, para darle paso a la ejecución de otra transición.

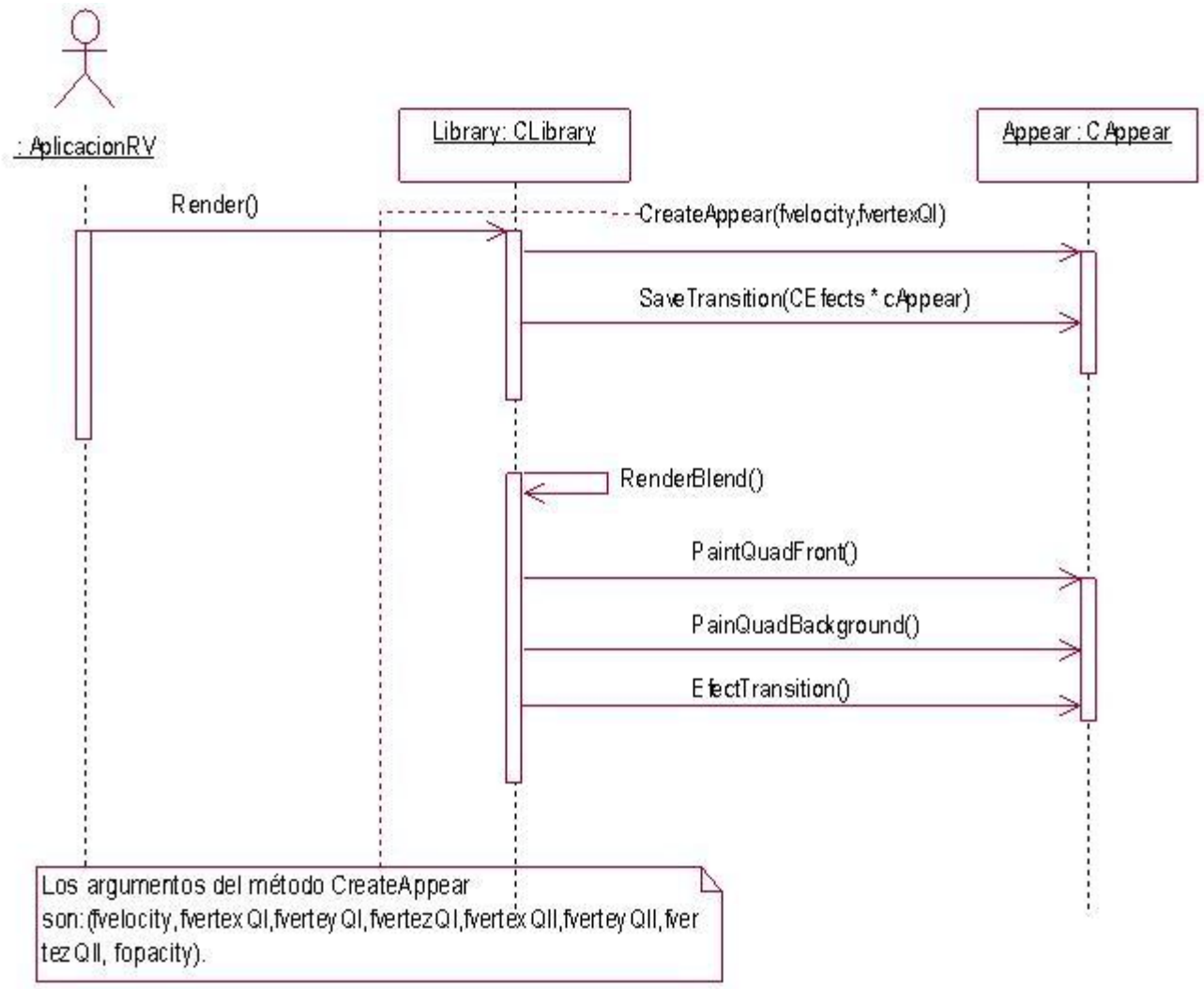

Fig. 20 Diagrama de Secuencia "Aplicar Transición Aparecer".

En este diagrama se refleja el flujo de secuencia para aplicar la transición aparecer. Para ejecutarla se debe llamar al visualizador gráfico, luego se crea la transición con los parámetros correspondientes que se desea que tenga. Con estos datos se construyen los planos que van ser escenario de la transición y se le calcula los valores que van estar relacionado con el efecto de transición (opacidad).

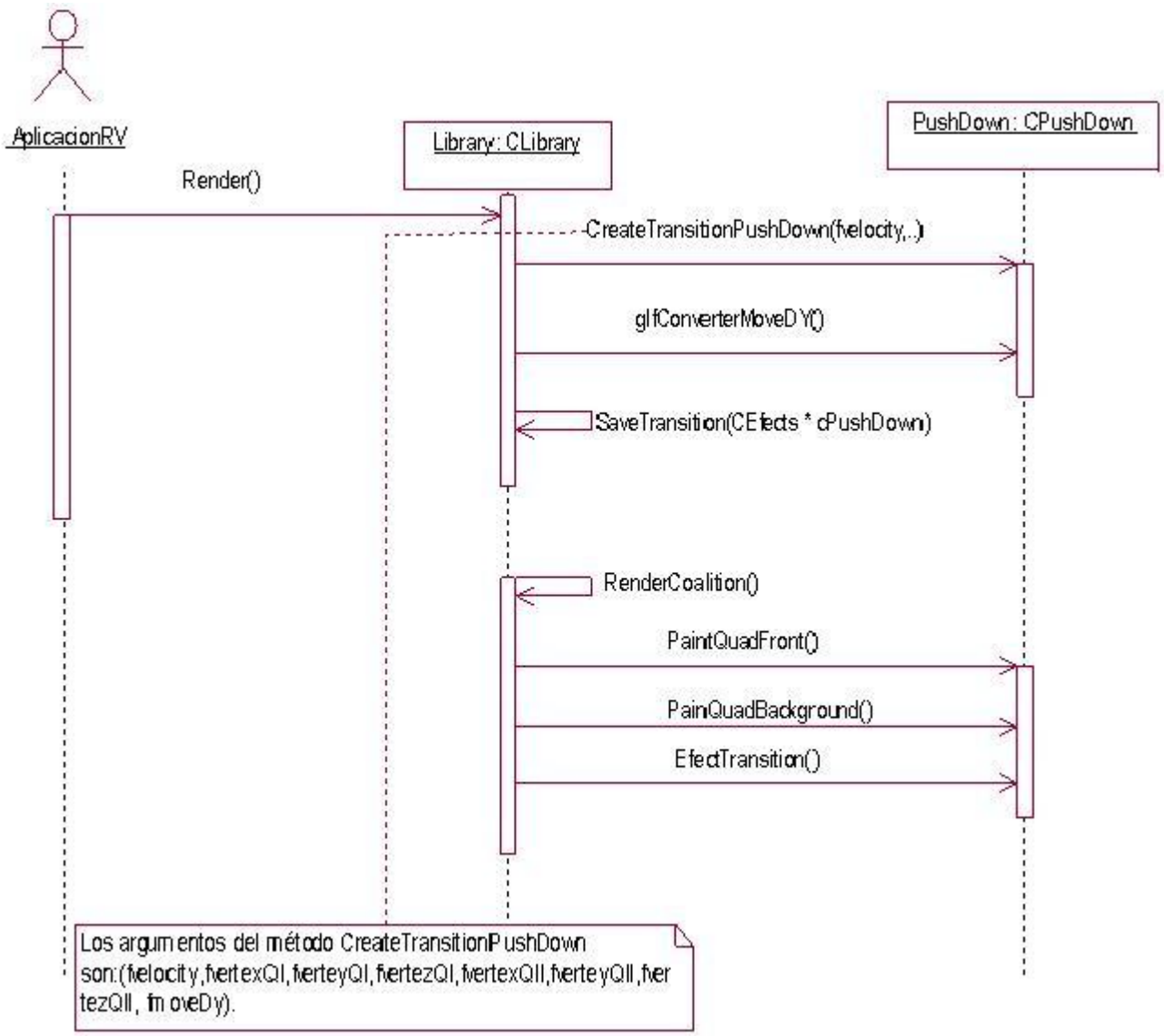

Fig. 21 Diagrama de Secuencia "Aplicar Transición Deslizar" (Sección "Deslizar desde Abajo").

El diagrama anterior muestra el flujo de secuencia para aplicar la transición Deslizar desde Abajo. Para llevar a cabo el efecto se llama al visualizador gráfico, luego se crea la transición con los parámetros correspondientes. Con estos datos se actualizan la posición inicial donde se va encontrar el plano del background con respeto al eje *y* negativo, se construyen los planos de front y background que van a ser escenario de la transición y se le calculan los valores del efecto que se va a aplicar.

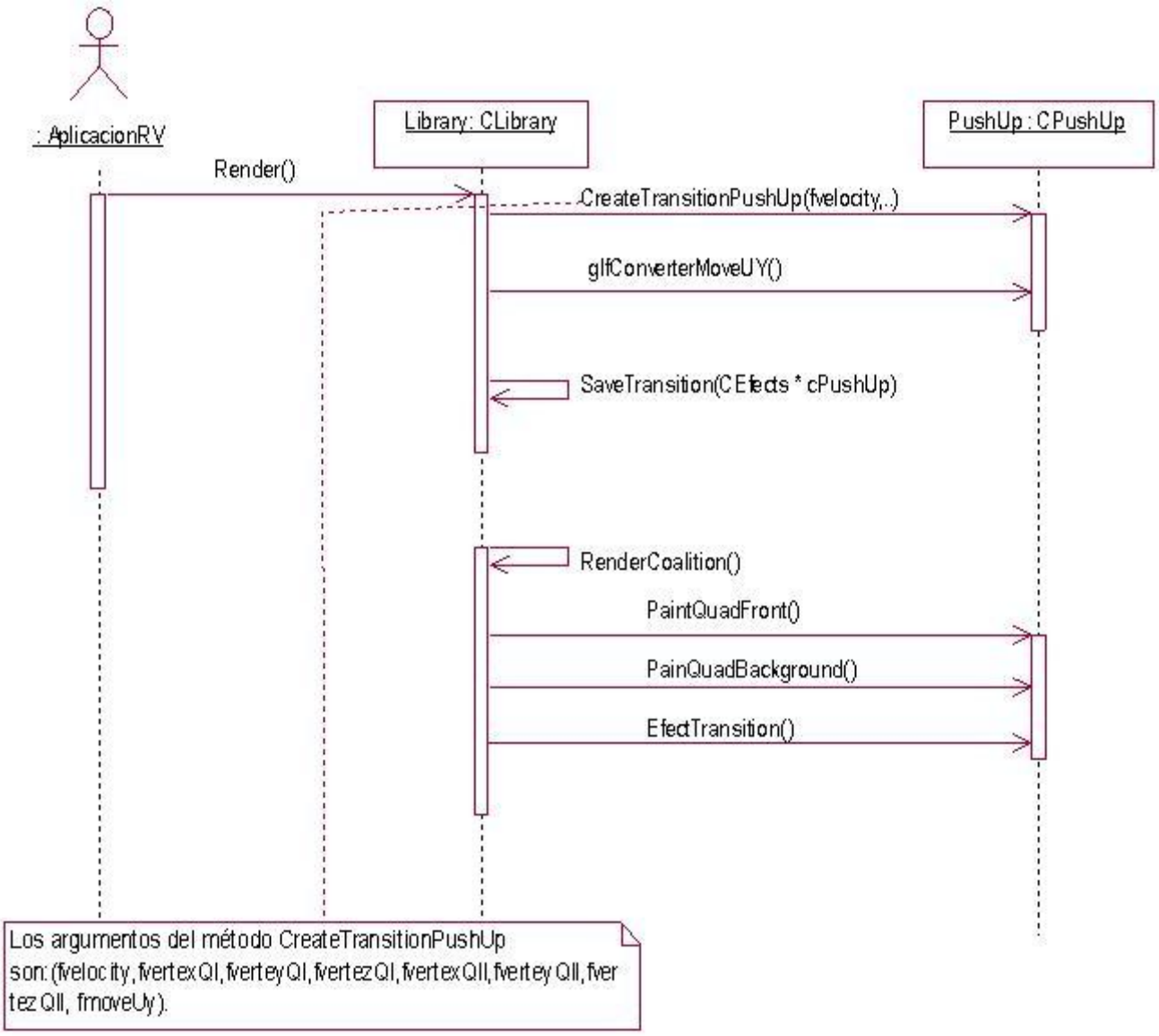

Fig. 22 Diagrama de Secuencia "Aplicar Transición Deslizar" (Sección "Deslizar desde Arriba").

En este diagrama se describe la secuencia de operaciones para aplicar la transición Deslizar desde Arriba. Para iniciarla se llama al visualizador gráfico, luego se crea la transición con los valores correspondientes a esta. Con estos datos se actualizan la posición inicial donde va aparecer el plano de background con respeto al eje *y* positivo, se construyen los planos front y background, se procede a calcular los valores de la transición para efectuarla.

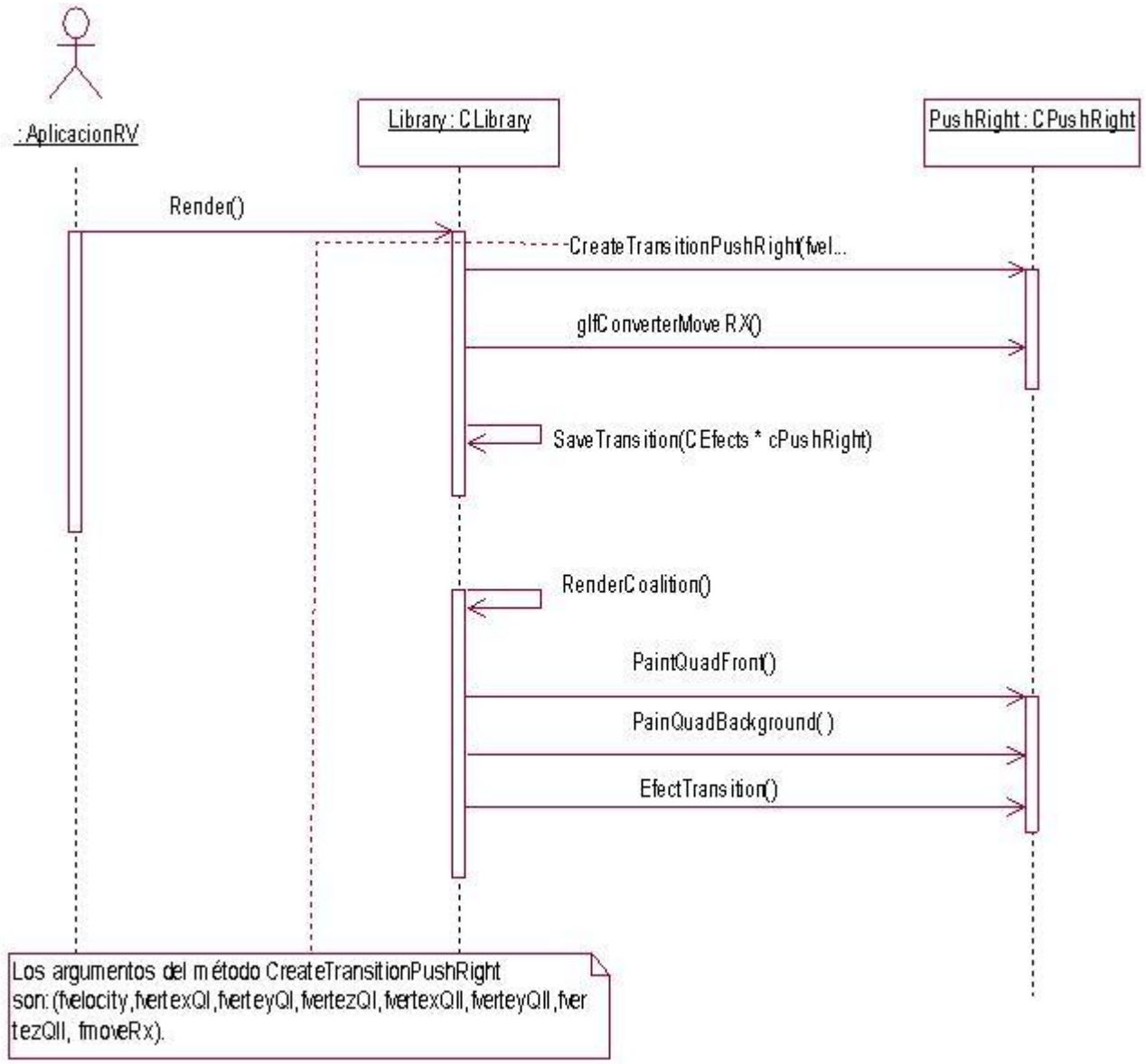

Fig. 23 Diagrama de Secuencia "Aplicar Transición Deslizar" (Sección "Deslizar Derecha").

En este diagrama se refleja la secuencia de operaciones para aplicar la transición Deslizar Derecha a Izquierda. Simplemente lo que se hace es llamar al visualizador gráfico, él mismo a su vez invoca al crear transiciones con los argumentos correspondientes a esta. Con estos datos se actualizan la posición inicial en que se va encontrar el plano background con respeto al eje *x* positivo, sucesivamente se construyen los planos front y background, se procede a calcular los valores correspondientes a la operación que se quiere ejecutar.

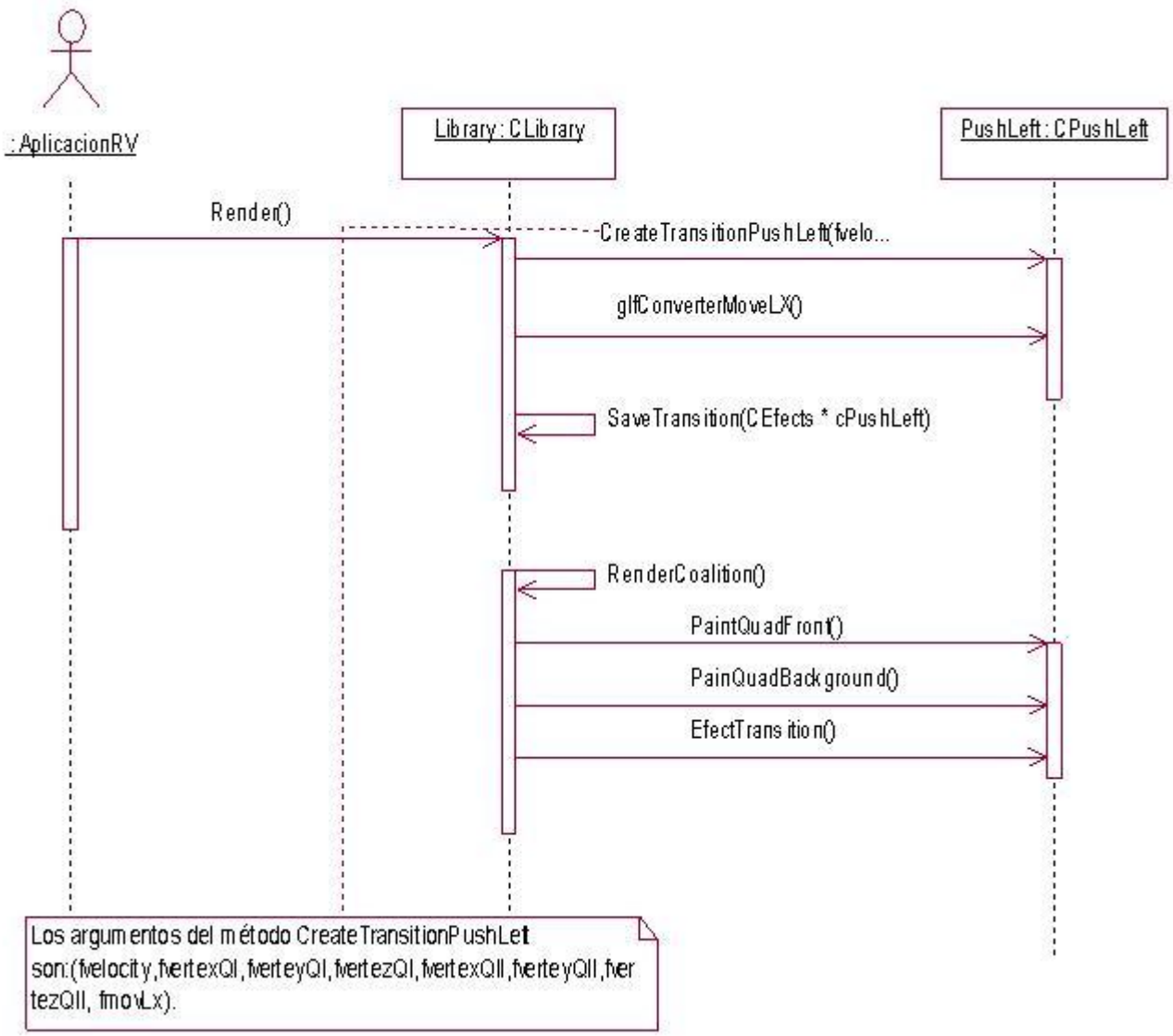

Fig. 24 Diagrama de Secuencia "Aplicar Transición Deslizar" (Sección "Deslizar Izquierda").

En este diagrama se muestra el flujo de secuencia a seguir para aplicar la transición Deslizar Izquierda a Derecha. Para comenzar se llama al visualizador gráfico, él mismo a su vez invoca al crear transiciones con los valores correspondientes. Con estos datos se actualizan la posición inicial donde se va encontrar el plano background con respecto al eje *x* negativo, posteriormente se construyen los planos front y background, posteriormente se calcula los valores de la operación a ejecutar.

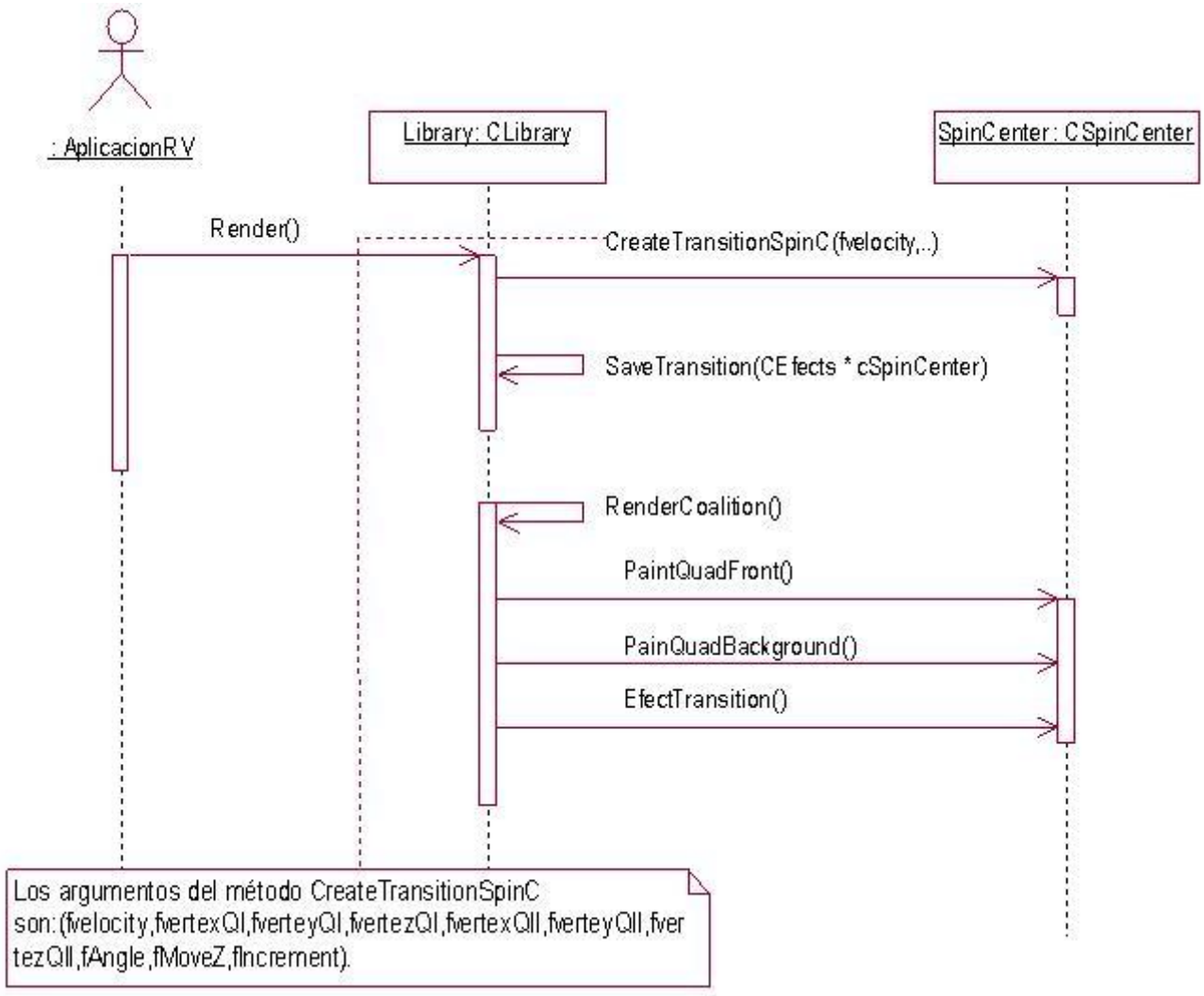

Fig. 25 Diagrama de Secuencia "Aplicar Transición Girar" (Sección "Girar-ampliar-centro").

En el diagrama se refleja la secuencia de operaciones para aplicar la transición Girar- desde el centro y ampliar. Se inicia cuando se invoca al visualizador gráfico, posteriormente se llama al crear transición pasándole los parámetros correspondientes, con estos datos se construyen los planos front y background que van ser escenario del efecto, y se calcula los valores de la operación a ejecutar (ángulo de rotación, eje de rotación, vértices del plano background).

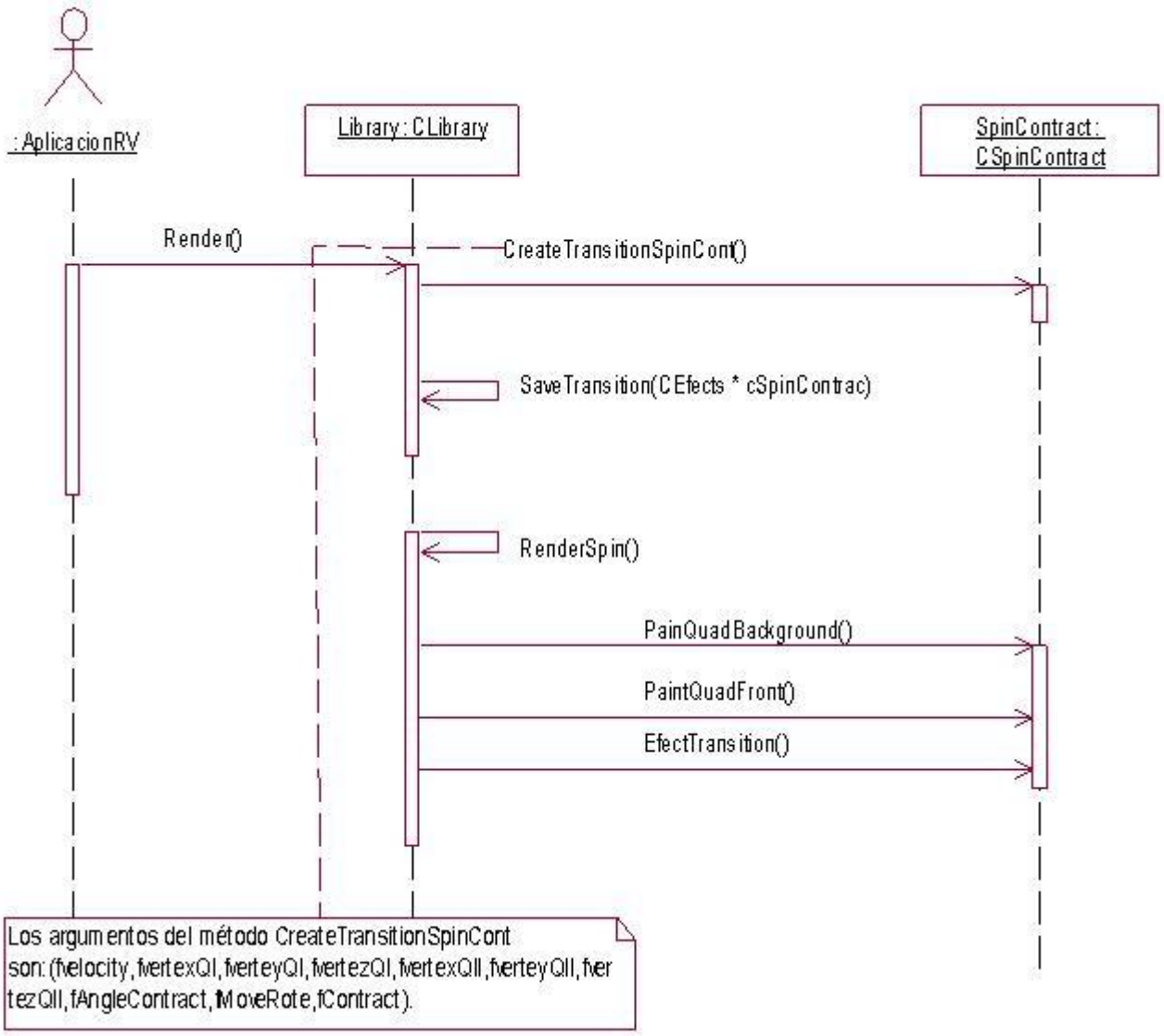

Fig. 26 Diagrama de Secuencia "Aplicar Transición Girar" (Sección "Girar-encoger-centro").

En este diagrama se expone el flujo de secuencia para ejecutar la transición Girar desde el centro y encoger a la vez. Comienza cuando se llama al visualizador gráfico, luego se invoca al crear transición, pasándole los parámetros correspondientes, con dichos datos se construyen los planos de background y front que van ser escenario del efecto, y se calcula los valores de la función a ejecutar (ángulo, ejes de traslación y vértices front).

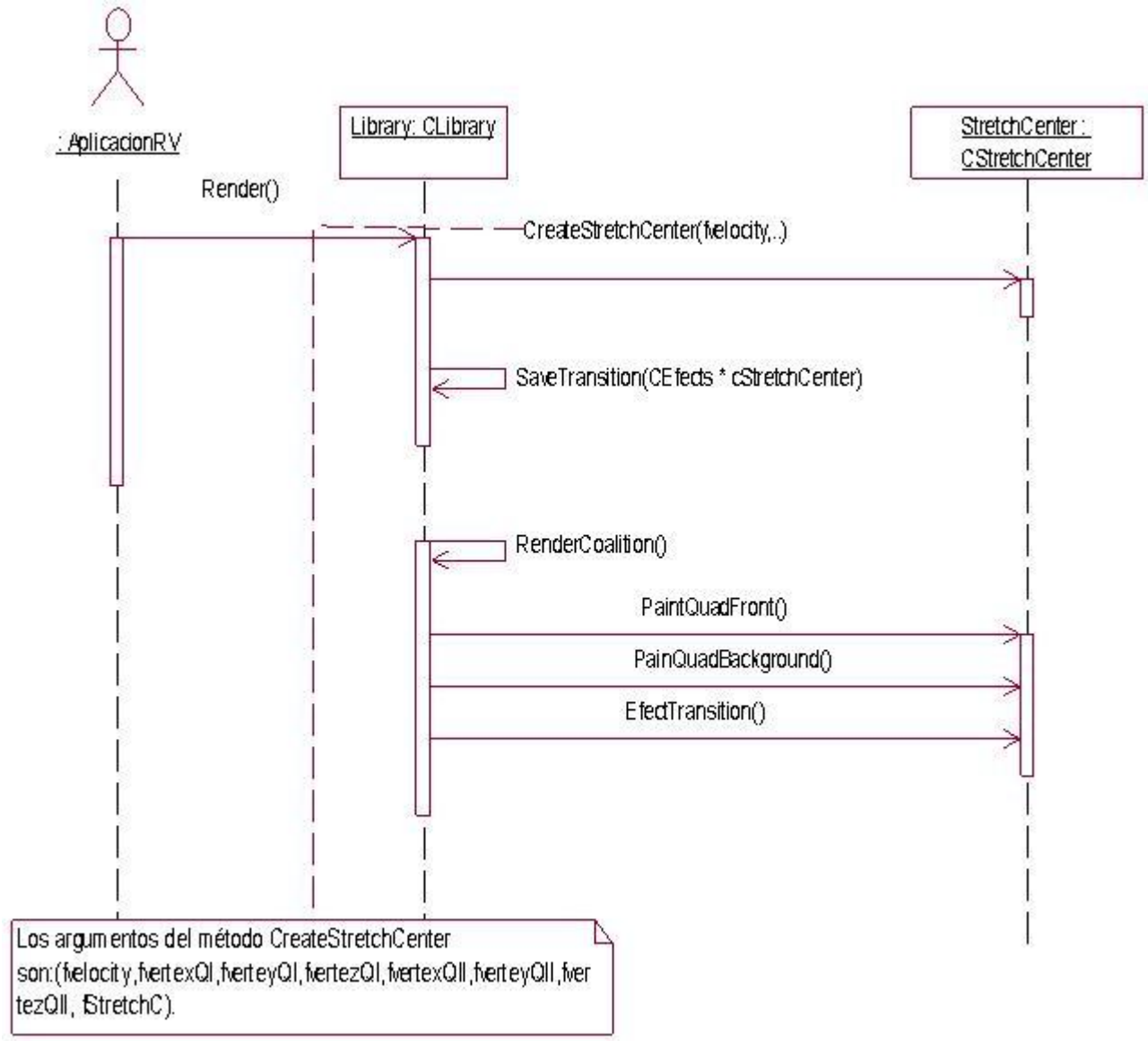

Fig. 27 Diagrama de Secuencia "Aplicar Transición Estirar" (Sección "Estirar-centro").

En el diagrama se muestra la secuencia de operaciones para aplicar la transición Estirar desde el centro. Se inicia cuando se invoca al visualizador gráfico, posteriormente se llama al crear transición pasándole los parámetros correspondientes. Con esos datos se construyen los planos front y background que van ser escenario del efecto y se calculan los valores de la operación (vértices del plano que va ser estirado background) para ser ejecutado.

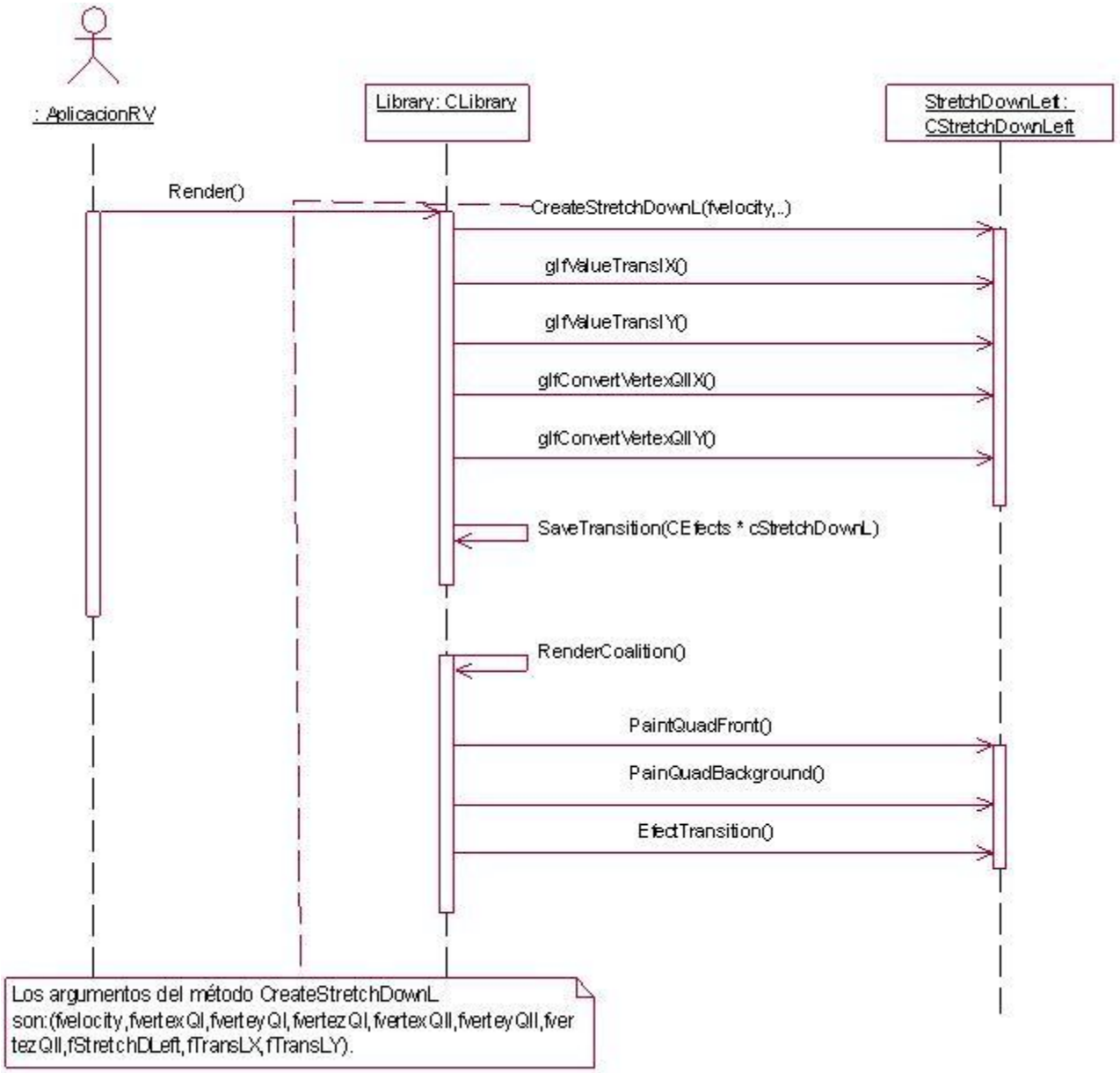

Fig. 28 Diagrama de Secuencia "Aplicar Estirar" (Sección "Estirar abajo izquierda-derecha").

En este diagrama se expone el flujo de secuencia para aplicar la transición Estirar abajo izquierda-derecha. Comienza invocando al visualizador gráfico, luego se llama al crear transición pasándole los argumentos correspondientes a esta, con dichos valores, se inicializan la dimensión, la posición inicial con respecto al eje *x* y *y* que se va encontrar el plano background, se construyen los planos front y

background que van ser escenario del efecto y se calcula los valores de la función (ejes traslación, vértices) para ser aplicado.

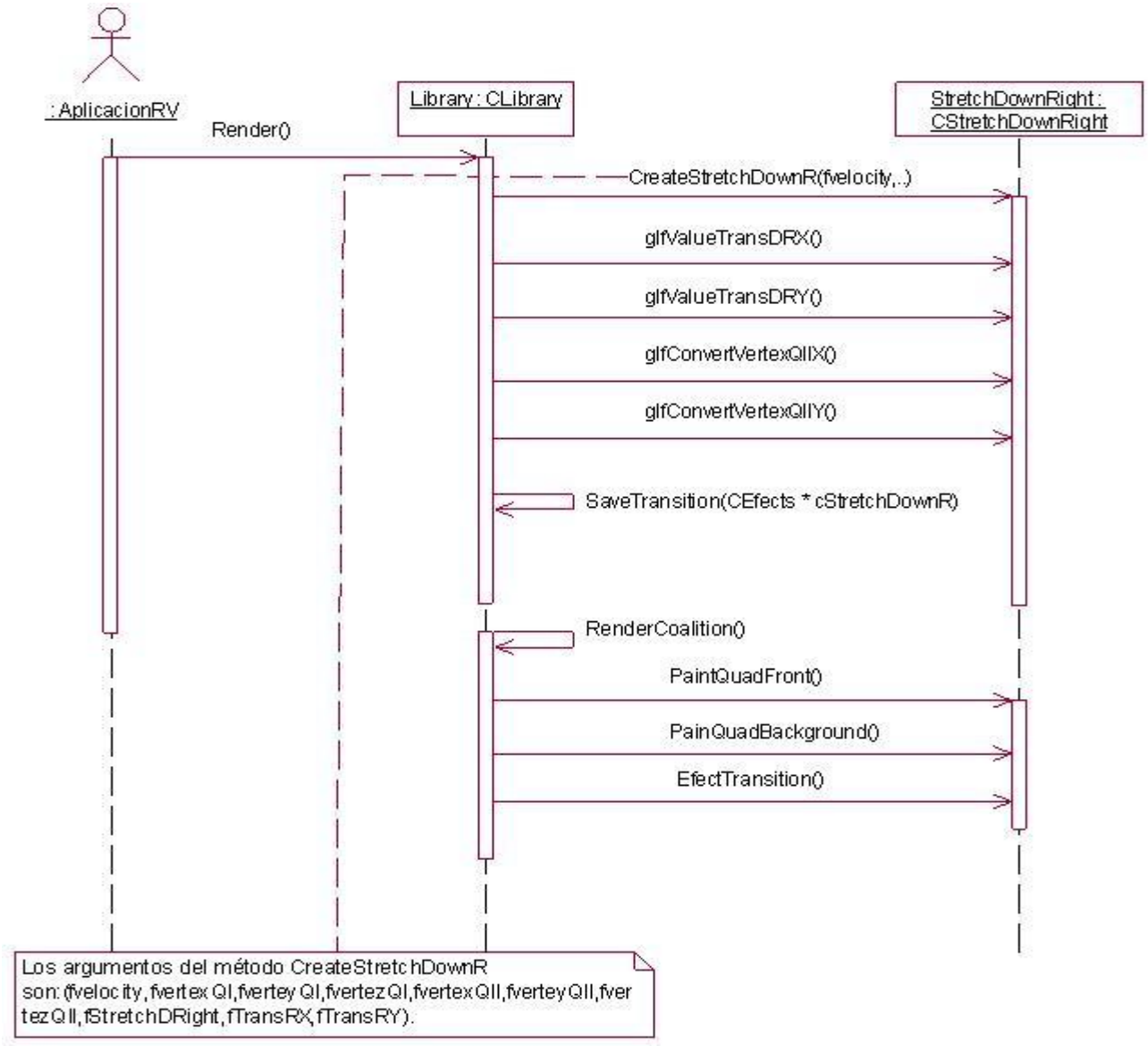

Fig. 29 Diagrama de Secuencia "Aplicar Estirar" (Sección "Estirar abajo derecha-izquierda").

Este diagrama refleja el flujo de operaciones para ejecutar la transición Estirar abajo derecha-izquierda, se inicia llamando al visualizador gráfico, posteriormente se crea la transición pasándole los parámetros correspondientes, con estos valores se inicializan la posición inicial (a la derecha-abajo del

plano front), la dimensión del plano background, luego se construyen los planos front y background, se calcula los valores de la transición (ejes traslación, vértices background) para ser ejecutado.

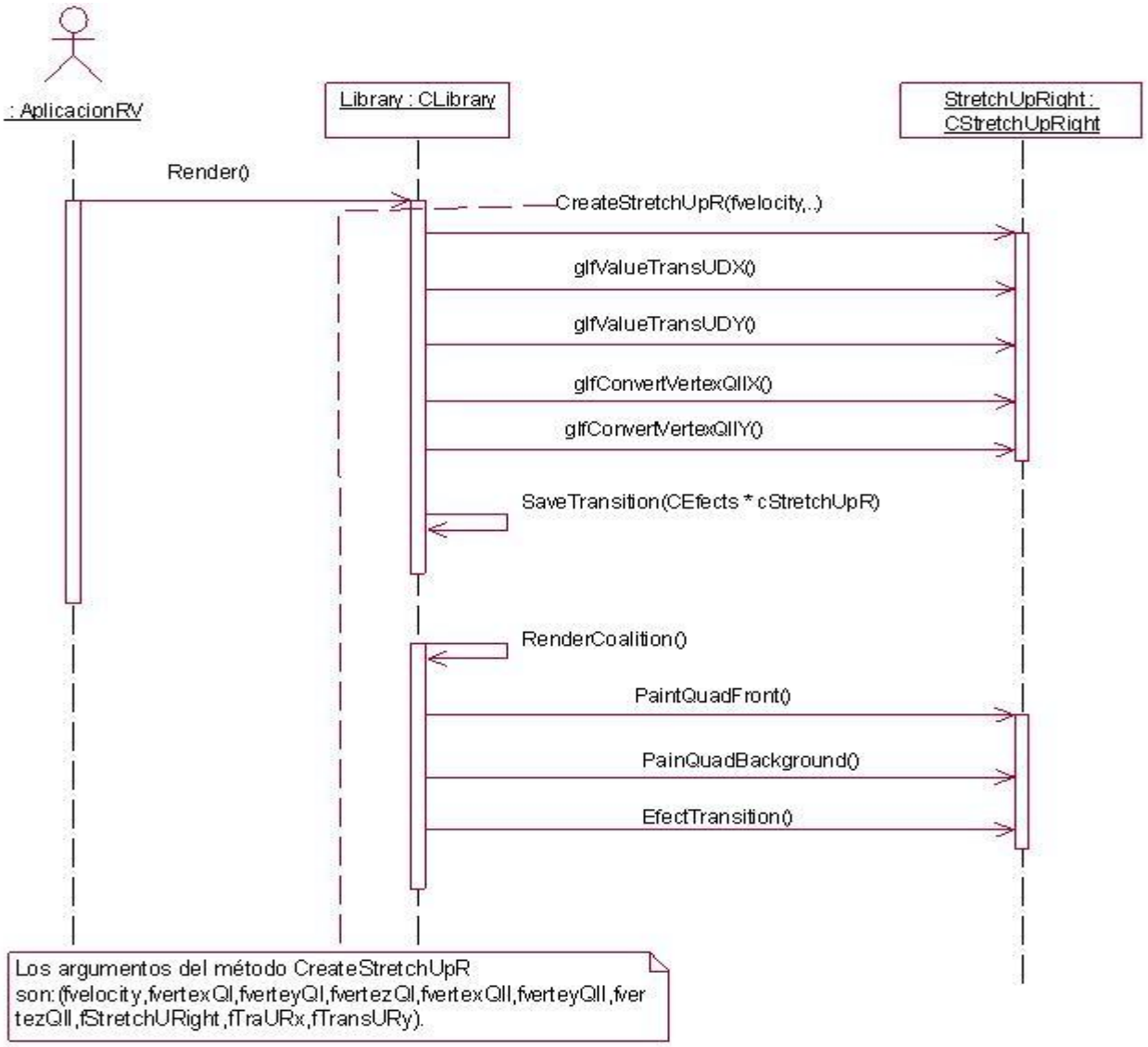

Fig. 30 Diagrama de Secuencia "Aplicar Estirar" (Sección "Estirar arriba derecha-izquierda").

El diagrama anterior muestra el flujo de secuencia para efectuar la transición Estirar arriba derecha-izquierda. Comienza invocando al visualizador gráfico, luego se crea la transición con los argumentos correspondientes, con estos datos se inicializan la posición inicial (arriba a la derecha del plano front) y

la dimensión que tendrá el plano background, construyéndose los planos front y background, y por último se calcula los valores correspondientes a la función que se va aplicar.

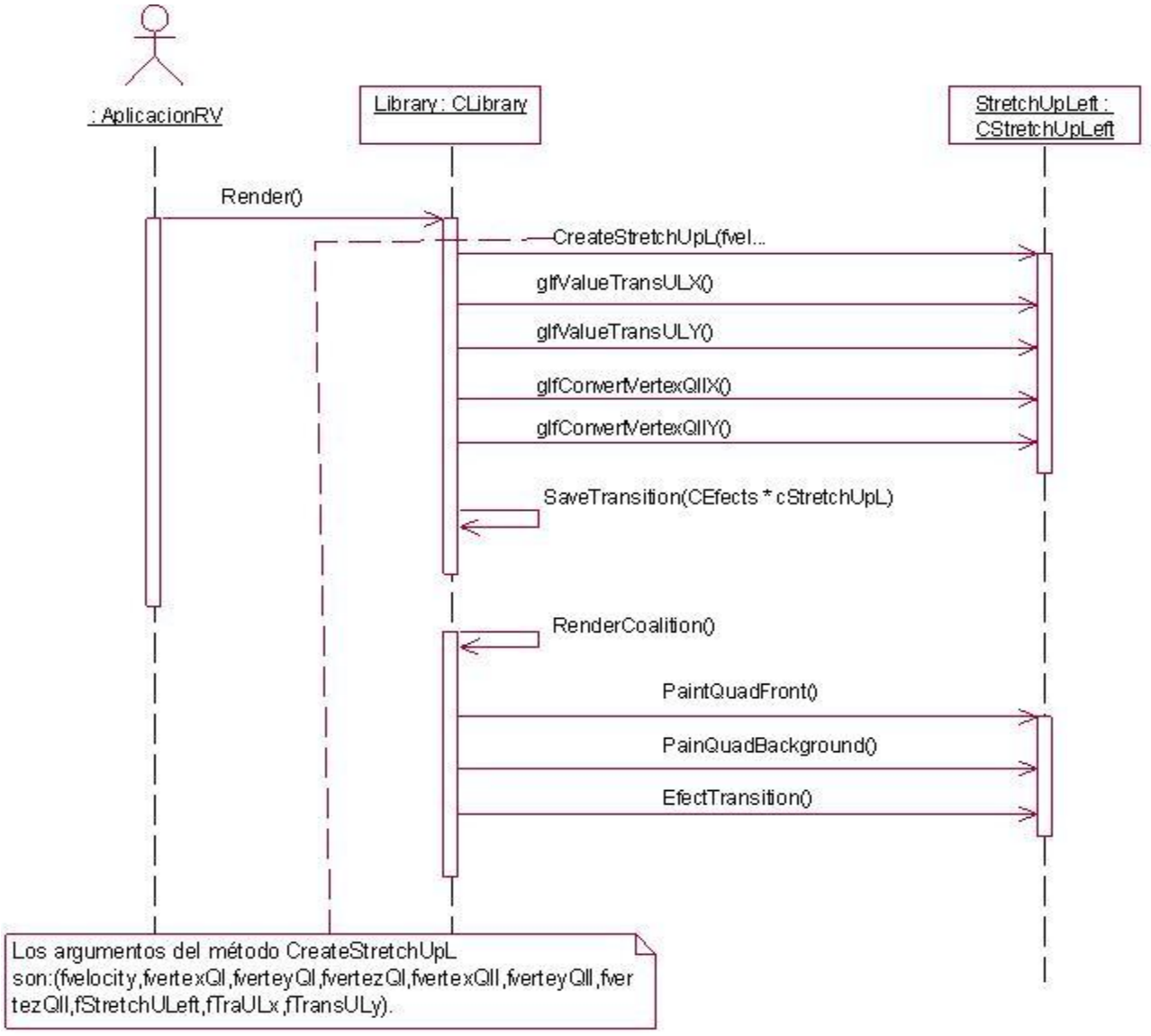

Fig. 31 Diagrama de Secuencia "Aplicar Estirar" (Sección "Estirar arriba izquierda-derecha").

El diagrama anterior refleja las operaciones para efectuar la transición Estirar arriba izquierda-derecha. Se inicia con la llamada al visualizador gráfico, luego este invoca al crear transición

pasándole los valores correspondientes, con estos parámetros se actualiza la posición inicial (arriba a la izquierda del plano frente) y la dimensión que tendrá el plano fondo, posteriormente se construyen los planos frente y fondo, y por último se calcula los valores correspondientes a la operación que se va ejecutar.

3.7 Estándares de codificación

Esta etapa de confección de la biblioteca constituye el paso del diseño de clases a la creación de componentes físicos que se traducen en ficheros .h y .cpp correspondientes a la implementación en C++.

Estándares de Codificación.

Está diseñado en inglés debido a que las palabras son simples, no se acentúan y es un idioma muy difundido en el mundo informático. El conocimiento de los estándares seguidos para el desarrollo de la misma permitirán un mayor entendimiento del código, y es una exigencia de los autores de la misma que cualquier módulo que se añada debe estar codificado siguiendo estos estándares.

Nombre de los ficheros:

Se nombrarán los ficheros .h y .cpp de la siguiente manera:

NameOfUnits.cpp

Clases: class **C**ClassName;

Indicando con "**C**" que es una clase.

Constantes:

Las constantes se nombrarán con mayúsculas, utilizándose el "**_**" para separar las palabras: **CONST_ZERO** = 0;

Tipos de datos:

Los tipos se nombrarán siguiendo el siguiente patrón:

Enumerados: enum **E**_Enum {**E**_VALUE, **E**_OTHER_VALUE, indicando con "**E**" que es de tipo enumerado};

Estructuras: struct **S**MyStruct {…}; Indicando con "**S**" que es una estructura.

Declaración de variables:

Los nombres de las variables comenzarán con un identificador del tipo de dato al que correspondan, como se muestra a continuación. En el caso de que sean variables miembros de una clase, se le antepondrá el identificador "**m**_" (en minúscula), si son globales se les antepondrá la letra "**g**", y en caso de ser argumentos de algún método, se les antepondrá el prefijo "**arg**_".

Tipos simples:

bool **b**VarName;

GLfloat **glf**Var

GLuint **glui**Var;

int **i**Name;

unsigned int **ui**Name;

float **f**Name;

char **c**Name;

char* **ac**Name; // arreglo de caracteres

char* **pc**Name; // puntero a un char

char** **aac**Name; // bidimensional

char** **apc**Name; // arreglo de punteros

bool **m_b**MemberVarName; //variable miembro

char **gc**GlobalVarName; //variable global

short **s**Name;

long **l**Name;

unsigned long **ul**Name;

size **s**Name;

Instancias

Instancias de tipo creados, se usa "**t**" para diferenciarlos.

EMyEnumerated **te**Name;

SMyStructure **ts**Name;

CClassName **tc**ObjectName;

CClassName* **ptc**Name; //puntero a objeto

CClassName* **atc**Name; //arreglo de objetos

CClassName* **tc**Name; // variable miembro de clase

Métodos

En el caso de los métodos, se les antepondrá el identificador del tipo de dato de devolución, y en caso de no tenerlo (void), no se les antepondrá nada.

En el caso de los argumentos se les antepone el prefijo "arg_"

Constructor y destructor:

CClassName (bool **arg**_bVarName, float& **arg**_fVarName);

~**C**ClassName ();

Funciones:

bool **b**Function1 (…);

int * **pi**Function2 (…);

CClassName* **pc**Function3 (…);

Métodos de acceso a miembros:

Los métodos de acceso a los miembros de las clases se nombrarán "**Gets**" y "**Sets**", como los demás métodos, pero con el nombre de la variable a la que se accede:

int iGetVar; //variable

Obtención del valor:

int **i**GetVar ();

{return **i**Var;}

Establecimiento del valor:

void SetVar (int **arg**_iVar)

{

iVar = **arg**_ i Var;

}

Obtención y establecimiento del valor:

int & **i**Var ();

{

return **i**Var;}

3.8 Diagrama de componentes

3.8.1 Diagrama de Componentes “Aplicar Transición Aparecer”

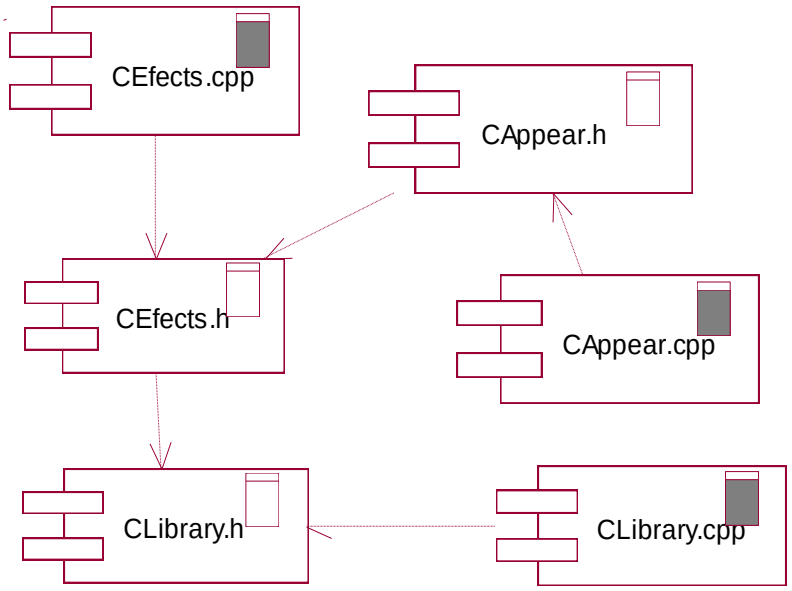

Fig. 32 Diagrama de Componentes “Aplicar Transición Aparecer”.

3.8.2 Diagrama de Componentes “Aplicar Transición Deslizar”.

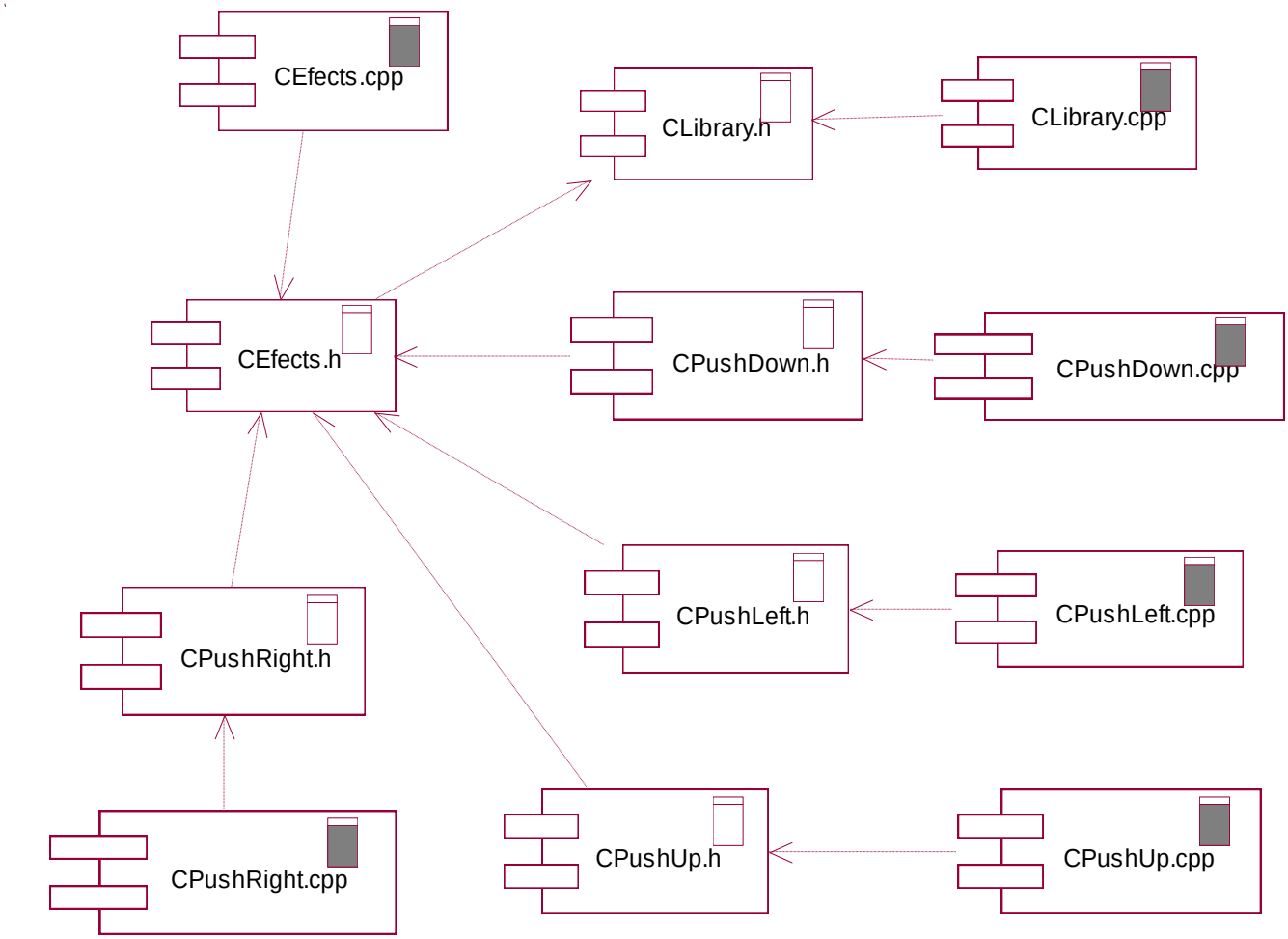

Fig. 33 Diagrama de Componentes “Aplicar Transición Deslizar”.

3.8.3 Diagrama de Componentes “Aplicar Transición Deslizar”.

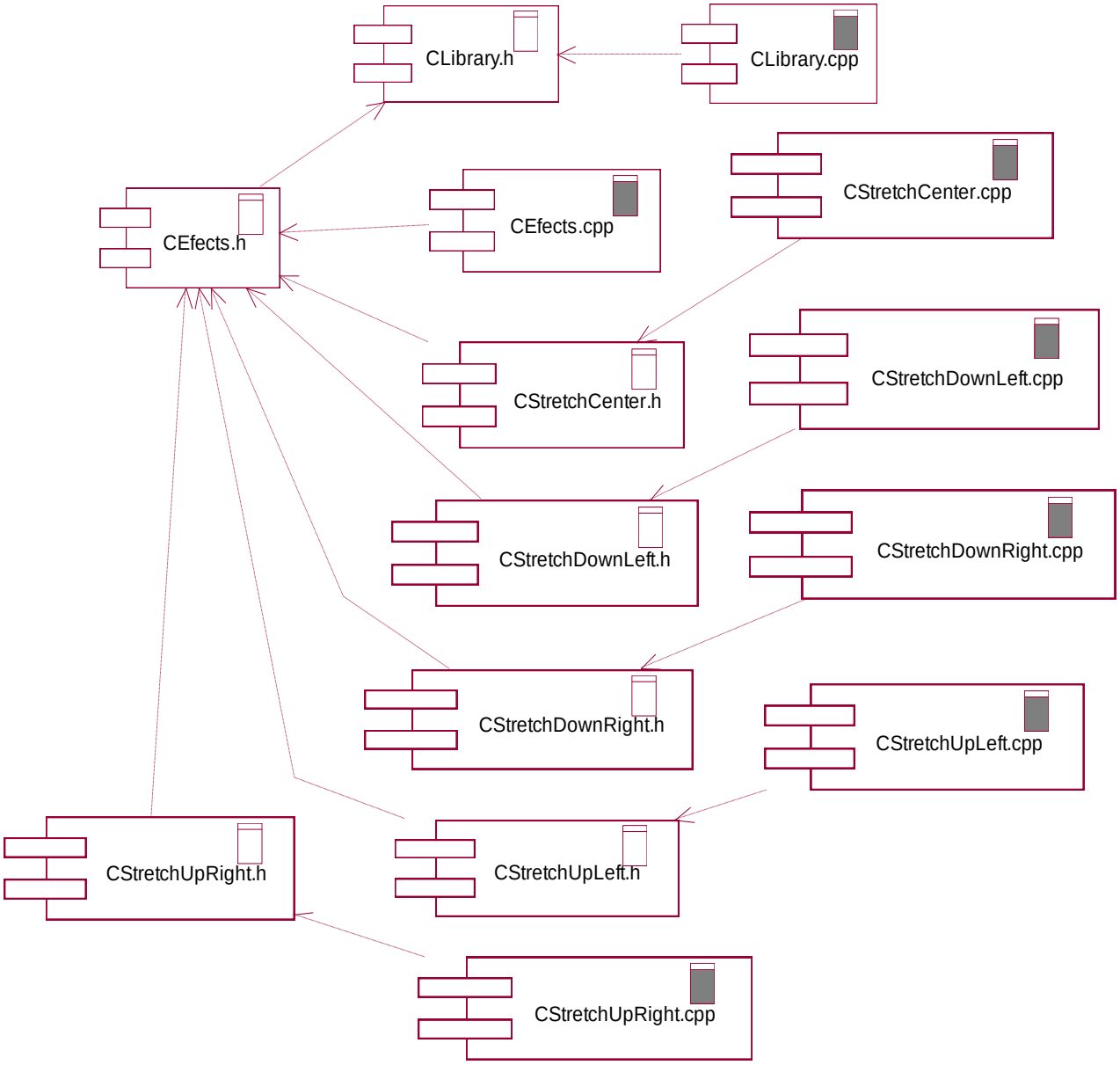

Fig. 34 Diagrama de Componentes “Aplicar Transición Estirar”.

3.8.4 Diagrama de Componentes “Aplicar Transición Girar”.

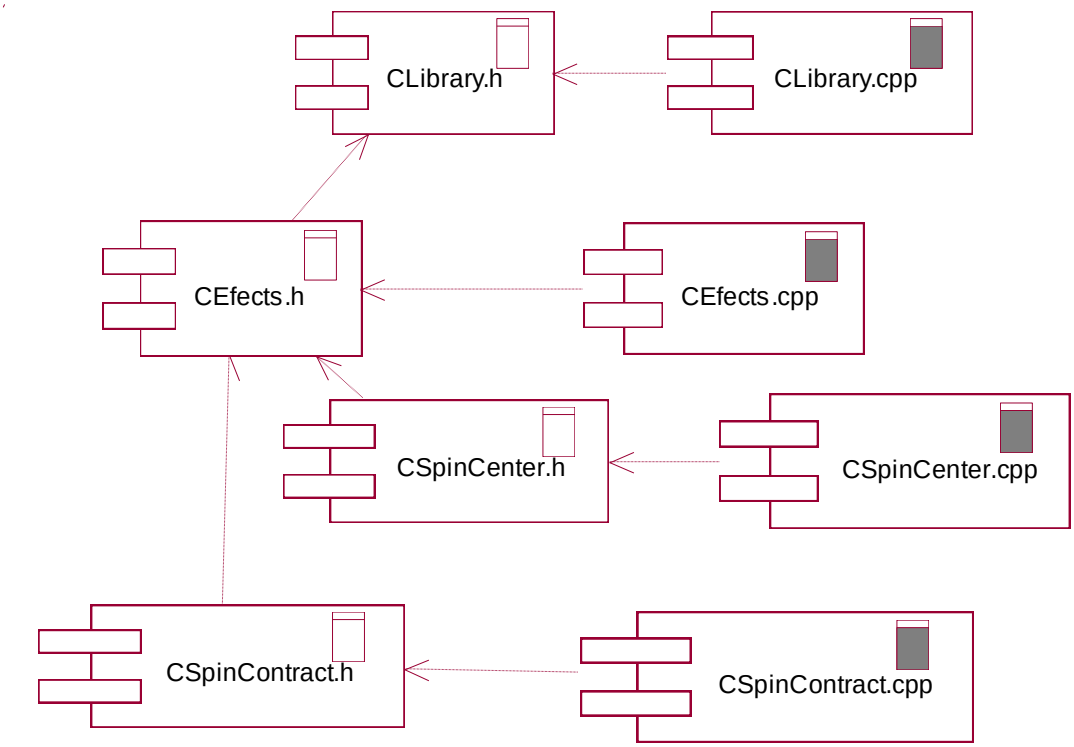

Fig. 35 Diagrama de Componentes “Aplicar Transición Girar”.

3.9 Consideraciones Finales

- Se dieron a conocer los diagramas más importantes de los flujos de trabajo Diseño e Implementación como son el Diagrama de clases, en los que se describen los atributos y métodos de cada clase, Diagrama de Despliegue y el Diagrama de Componentes.
- El Diagrama de Componentes da a conocer las interacciones entre los .h y los .cpp de cada clase.
- Las condiciones están creadas para pasar a la etapa de programación de los casos de uso.

Conclusiones

El trabajo desarrollado, en torno a la satisfacción de la necesidad científica de crear una biblioteca de transición de pantallas para elevar el funcionamiento de las ARV en la F5 de la UCI logra cumplir el propósito trazado por sus autores, en tanto que las técnicas de transición seleccionadas con el fin de conformar la librería de referencia contribuirán a impregnar un gran dinamismo en las apariencias gráficas de las interfaces y de facto optimizará el trabajo de los desarrolladores de video- juegos, los cuales están altamente necesitados de este tipo de ayuda, que son de gran utilidad en el mundo contemporáneo, producto al gran auge alcanzado por las ciencias informáticas.

Para dar cumplimiento a los objetivos de este proyecto, se hizo necesario, primeramente, valorar las técnicas, tecnologías y tendencias en cuanto al uso de los efectos de transición de pantallas lo cual contribuyó a revelar nuevas características para su uso.

En el progreso de la propuesta se propone la descripción de la solución, con el objetivo de ubicar a toda persona interesada en el tema, donde se brindan las funcionalidades necesarias para percibir lo relacionado con las transiciones, específicamente, de pantallas.

En este sentido, se efectuó la captura de los requisitos funcionales y no funcionales, agrupándose en los casos de uso del sistema. Se transitó por las etapas de diseño e implementación, utilizando los artefactos de RUP, donde se crearon y consolidaron las clases, se concluyó con la elaboración del Diagrama de Componentes que contendrá las interacciones entre los ficheros .h y .cpp de las clases de la biblioteca.

Una vez finalizadas cada unas de las tareas anteriores se concluyó con la implementación y visualización de varios efectos de transición de pantallas para una primera versión, propiciando que se puedan ampliar a futuras implementaciones, así como el número de transiciones en la biblioteca para enriquecer el mundo virtual en este campo de la Informática.

Recomendaciones

A este trabajo se le recomienda los siguientes aspectos:

- Utilizar la biblioteca desarrollada para la confección de menú con efectos de transición para juegos y/o simuladores.
- Extender la biblioteca STE a otra cantidad de transiciones, utilizando los métodos implementados en este trabajo.
- Vincular la biblioteca realizada con la herramienta de desarrollo de SceneToolKit.
- Retomar el trabajo como bibliografía para el estudio de los efectos de transiciones y los algoritmos que lo desarrollan.

Referencias Bibliográficas

-*Analisis de Lenguajes de Programación.* (marzo de 2008). Citado el: 5 de marzo de 2008
Disponible en: http://www.vjuegos.org/modules.php?name=Content&pa=showpage&pid=4

-**Bravo Ramos, J. L. (Junio de 2002).** *Elaboración de presentaciones con Ordenador*. (Universidad Politécnica de Madrid) Citado el: 10 de Noviembre de 2007
Disponible en: http://www.ice.upm.es/av/html/TecnoRec/Temario/PRESENdib02.pdf

-*Características de Software*. (2006). Citado el: 19 de junio de 2008
Disponible en: http://www.taringa.net/posts/downloads/1021803/Programadores.html

-*Características de Visualización C++.Net*. (Octubre de 2007). Citado el: 4 de Diciembre de 2007
Disponible en: http://www.microsoft.com

-*Efectos gráficos*. (noviembre de 2007). Citado el: 7 de Diciembre de 2007
Disponible en:] http://grafics.pina.cat/definicio.php?id=21

-Grupo de Desarrollo en Allegro, G. (abril de 2004). *Ingeniería de Sistemas y Computación*. (Universidad Tecnológica de Pereira) Citado el: 14 de Diciembre de 2007
Disponible en: http://gda.utp.edu.co:9673/gda/documentacion/programacion_3d/directx/tutorial08

-**Guerrero Tala, F.** (diciembre de 2006). *Lenguaje C++. Guía para Programadores*. Citado el: 6 de Diciembre de Disponible en: http://www.mailxmail.com/curso/informatica/cplusplus2

-*iPod ABC "Configurar un pase de diapositivas".* (Noviembre de 2007). Citado el: 5 de Noviembre de 2007 Disponible en: http://docs.info.apple.com/article.html?artnum=304679-es

-*Manual de Cinelerra CV*. (noviembre de 2007). Citado el: 15 de Noviembre de 2007
Disponible en: http://cvs.cinelerra.org/docs/cinelerra_cv_manual_es.txt

-**Moreno Vozmediano, A.** (29 de Marzo de 2008). *Compilación y enlace .* Citado el: 30 de Mayo de 2008 Disponible en: http://profeblog.es/blog/alfredo/2008/03/29/compilacion-y-enlace/

-**Pérez, M.** (noviembre de 2007). Citado: 8 de Diciembre de 2007, de Animación y efectos
Disponible en: http://roble.pntic.mec.es/~mperez8/pagweb3/animacion.htm

-*Teoría RUP*. (2006). Citado: 19 de junio de 2008
Disponible en: http://prof.usb.ve/lmendoza/Documentos/PS-6116/Teoria%20PS6116%20O-O%20y%20RUP.pdf.

-Valverde Berrocoso, J. (Noviembre de 2004). Citado el: 9 de Diciembre de 2007, de Diseño de materiales educativos multimedia
Disponible en: http://www.unex.es/didactica/Tecnologia_Educativa/guion10.htm

-*Wikipedia "Enlace dinámico"*. (Noviembre de 2007). Citado el: 1 de Noviembre de 2007
Disponible en: http://es.wikipedia.org/wiki/Enlace_din%C3%A1mico
-*Wikipedia "Transiciones"*. (Noviembre de 2007). Citado el: 3 de Noviembre de 2007
Disponible en: http://es.wikipedia.org/wiki/Transici%C3%B3n

Bibliografía

-**ASTLE, D., & HAWKING, K. (2001).** *OpenGL Game Programming.* USA.: Prima Tech Publishing.

-**HIEBERT, G. & CHARLEY, K. (2006).** *OpenGL Programmer's Guide.*

-**JACOBSON, I., BOOCH, G. & Rumbaugh, J. (2000).** *El Lenguaje Unificado de Modelado.* Addison Wesley.

-**JACOBSON, I., BOOCH, G. & Rumbaugh, J. (1999).** *El Proceso Unificado de Desarrollo de Software.* Addison Wesley.

-**McCuskey, M. (2002).** *Special Effects Game Programming with DirectX.* (S. Doell, Ed.) Premier Press, Game Development.

-**Svarcas, A. (Abril de 2004).** *Efectos de transición de Imágenes (Actualizado).* Citado el: 12 de Noviembre de 2007.
Disponible en: http://www.elguille.info/colabora/puntoNET/anibal_EfectosTrans.htm

-**Wolfgang, F. Engel. (2003).** *ShaderX2 Shader Programming Tips and Ticks with DirectX 9.* United States of America.

.

APÉNDICES

Glosario de Términos.

Alpha Blending: Técnica utilizada para lograr transiciones entre imágenes, proceso de combinar dos objetos en pantalla teniendo en cuenta los valores alfa que designa su grado de transferencia.

Animación: Simulación de un movimiento creada por la muestra de una serie de imágenes o fotografías.

Buffer: Espacio de memoria para almacenamiento temporal de datos.

C: Es un lenguaje de programación creado en 1969 por Ken Thompson y Dennis M. Ritchie en los Laboratorios Bell. C es apreciado por la eficiencia del código que produce y es el lenguaje de programación más popular para crear software de Sistemas, aunque también se utiliza para crear aplicaciones.

C++: Lenguaje de programación orientado a objetos.

DirectX: Colección de APIs creadas para facilitar tareas relacionadas con la programación de juegos en la plataforma Microsoft Windows.

Escena: Espacio gráfico generado por un ordenador.

Glide: API propietaria para que los desarrolladores de videojuegos la utilicen para programar videojuegos 3D, es un pequeño subconjunto de OpenGL implementado en el hardware.

Matriz de transformación: Matriz definida para calcular nuevas coordenadas a partir de coordenadas existentes según una determinada transformación gráfica (traslación, rotación, escalado y reflexión).

Matriz*:* Arreglo de elementos.

OpenGL: Es una biblioteca gráfica desarrollada originalmente por Silicon Graphics Incorporated

(SGI).OpenGL significa Open Graphics Library, cuya traducción es biblioteca abierta de gráficos.

Píxel: Es la menor unidad de información de una imagen digital.

Quad: Quadrangle, abreviatura de cuadrángulo, en nuestro caso Cuadrado.

Renderizar: Visualización grafica de objetos en el espacio 2D o 3D.

Requisito: Condición o capacidad que debe cumplir un sistema.

Requisito funcional: Requisito que especifica una acción que debe ser capaz de realizar un sistema, sin considerar restricciones físicas; requisito que específica comportamiento de entrada/salida de un sistema.

Requisito no funcional: Requisito que especifica propiedades del sistema, como restricciones del entorno o de implementación, rendimiento, dependencias de la plataforma, mantenimiento, extensibilidad o viabilidad. Requisito que especifica restricciones físicas sobre un requisito funcional.

Sistema de realidad virtual: Sistema informático interactivo que brinda una percepción al usuario de un mundo tridimensional sintético que suplanta al real.

Software libre: Software que, una vez obtenido, puede ser usado, copiado, estudiado, modificado y redistribuido libremente.

Tecnología: Conjunto de teorías y de técnicas que permiten el aprovechamiento práctico del conocimiento científico.

Textura: Imagen que sirve de “cubrimiento” a las primitivas de un mundo virtual.

Texturizar: Aplicar texturas.

Transición: En nuestro caso es la variación de cambio en una imagen.

Vector: Cantidad que expresa magnitud y dirección.

Vértices*:* Punto en el que se unen los lados de un ángulo o las caras de un poliedro. Punto de una curva en el que la curvatura es máxima o mínima.

Viewport: Porción de la ventana que establece una correspondencia entre las coordenadas espaciales transformadas sobre el plano de proyección y las coordenadas en pantalla.

Glosario de Abreviaturas

API: Applications Programming Interface (Interfaz de programación de aplicaciones).

ARV: Aplicaciones de Realidad Virtual.

CU: Caso de Uso

DLL: Dynamic Linking Library (Bibliotecas de Enlace Dinámico).

F5: Facultad 5

G3D: Librería Open Source orientada a objetos, programadas en C++ y de alto nivel usada para la programación de juegos 3D.

HW: Hardware.

IDE: Integrated Development Environment (entorno de desarrollo integrado) es un programa compuesto por un conjunto de herramientas para un programador.

JavaScript: Lenguaje de programación interpretado, orientado a objetos, que no requiere compilación, utilizado principalmente en páginas web, con una sintaxis semejante a la del lenguaje Java y el lenguaje C.

RUP: Rational Unified Process (Proceso Unificado de Software).

RV: Realidad Virtual.

SRV: Sistema de Realidad Virtual.

STL: Standard Template Library (Librería de Plantilla Estándar).

STE: Screen Transitions Effects.

UML: Unified Modeling Language (Lenguaje Unificado de Modelado).

2D: Dos dimensiones.

3D: Tres dimensiones.

Índice de figuras y tablas.

Índice de tablas.

Índice de figuras.