Universidad de las Ciencias Informáticas

Sistema de Animaciones para CEGUI 0.7

Trabajo para optar por el Título de Ingeniería en Ciencias Informáticas

Autor:

Jaime González Pacheco

Tutores:

Ing. Jaime González Campistruz

Ing. Yesnier Domínguez Silva

Ciudad de La Habana, Cuba

2011

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de este trabajo y autorizo al <nombre área> de la Universidad de las Ciencias Informáticas; así como a dicho centro para que hagan el uso que estimen pertinente con este trabajo.

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

Jaime González Pacheco

_____

Firma del Autor

Jaime González Campistruz

_____

Firma del Tutor

Yesnier Domínguez Silva

_____

Firma del Tutor

Dedicatoria

A mi familia.

Agradecimientos

A mi familia, gracias a ellos he llegado a ser lo que soy.

A Yanet, incondicionalmente siempre a mi lado.

A los tutores de este trabajo por todo su apoyo.

A mis amigos y compañeros de estudio.

A Pedro, por la corrección y apoyo.

A todos los que de una manera u otra han ayudado en mi formación personal y profesional.

Resumen

Las interfaces gráficas de usuario funcionan como mediadoras entre los sistemas y los usuarios finales, por lo que su calidad es un punto crucial en la decisión de estos usuarios de usar dichos sistemas. Aunque el enriquecimiento artístico de estas interfaces no responde a necesidades funcionales importantes en los sistemas, ayudan a lograr una interacción más agradable con sus objetos.

El objetivo de este trabajo es desarrollar un sistema que le permita a los desarrolladores del proyecto Entrenadores Aduaneros mejorar la estética e interactividad de sus interfaces.

Como resultado de esta tesis de grado, se obtuvo Sistema de Animaciones para la biblioteca CEGUI 0.7, que permite el enriquecimiento estético de las interfaces creadas utilizando esta biblioteca con mejoras de animación.

Este sistema fue validado y es actualmente utilizado en el software desarrollado por el proyecto Entrenadores Aduaneros.

Índice

Introducción

Las interfaces gráficas de usuario (GUIs del inglés *Graphical User Interfaces*) como artefacto tecnológico, surgen por la necesidad de un método amigable de comunicación que superase la simple interfaz de línea de comandos y tienen como objetivo básico encontrar un modelo óptimo de interacción persona-ordenador.

A través de los años han transitado por un proceso de madurez que las ha convertido en un producto de consumo estético dentro de los sistemas interactivos. Hoy en día el desarrollo de GUIs se centra en la estética y en la adición de funcionalidades de decorado normalmente provocadas por necesidades mercantiles del producto frente a la competencia. Aunque estos procesos de enriquecimiento artístico no responden a necesidades funcionales importantes en los sistemas, ayudan a lograr una interacción más agradable con los objetos.

La Universidad de las Ciencias Informáticas (UCI), institución de altos estudios dedicada además a la producción de software, cuenta con un proyecto encargado de desarrollar una aplicación del tipo simulador, que responda a las necesidades de capacitación de los inspectores aduaneros en entrenamiento de la Aduana General de la República de Cuba.

La biblioteca de Interfaz Gráfica de Usuario de Crazy Eddies (CEGUI por sus siglas en inglés), en su versión 0.7, es la utilizada por este proyecto para la creación y control de sus interfaces. En esta versión la biblioteca incorpora características que permiten aplicar efectos basados en *shader* a los elementos de la interfaz, sin embargo, la inexistencia aún de un sistema que permita animar las propiedades de sus componentes de forma sencilla, entorpece el enriquecimiento de las interfaces utilizadas con mejoras de animación.

Es necesaria entonces la creación de un sistema que le permita a los desarrolladores del mencionado proyecto, mejorar la estética e interactividad de sus interfaces mediante animaciones.

En virtud de ello el **problema científico** que se plantea esta investigación es ¿Cómo incorporar animaciones a una interfaz creada utilizando la biblioteca CEGUI 0.7?

Cuyo **objeto de estudio** son las interfaces gráficas de usuarios como medio de comunicación usuario-aplicación, restringiendo el **campo de acción** a la animación de componentes en bibliotecas de interfaces gráficas de usuario.

El **objetivo** de esta investigación es desarrollar un Sistema de Animaciones para los componentes de la biblioteca CEGUI 0.7.

Para el cumplimiento del mismo se definen las siguientes **tareas investigativas**:

- Definir requisitos funcionales y no funcionales.
- Seleccionar y caracterizar una arquitectura flexible que guie el desarrollo del sistema.
- Implementar la solución.
- Documentar el código fuente de acuerdo a los estándares utilizados en el proyecto.
- Desplegar y validar la solución.

Se espera que el sistema desarrollado permita el uso de animaciones en las interfaces gráficas de usuario utilizadas en el proyecto Entrenadores Aduaneros (referido como EA en el resto de este documento).

El presente trabajo de diploma consta de tres capítulos y está estructurado de la siguiente manera:

Un primer capítulo, **Fundamentación teórica**, expone las matrices teóricas necesarias para el trabajo con interfaces gráficas de usuario y describe las principales características de la biblioteca CEGUI. Además se analizan brevemente las características de otros *frameworks* y bibliotecas GUIs que permiten animar sus componentes. En el segundo capítulo, **Características del sistema,** se exponen las particularidades que presentará el sistema, los requisitos funcionales y no funcionales a los que debe dar cumplimiento y se describen los casos de uso y el modelo de casos de uso del sistema. En el tercer y último capítulo, **Diseño e implementación del sistema**, se presentan los diagramas generados durante esta fase y el método utilizado para validar la solución.

Capítulo 1 Fundamentación teórica

Introducción

A continuación se presentan los conceptos fundamentales para el trabajo con interfaces gráficas de usuario y una descripción de las principales características de la biblioteca CEGUI. Además, con el objetivo de seleccionar un modelo arquitectónico que se ajuste a las necesidades de nuestro sistema e incorporar quizás otras funcionalidades útiles, se analizan brevemente las características de otros *frameworks* y bibliotecas GUIs que permiten animar sus componentes.

1.1 Interfaz gráfica de usuario

El concepto de interfaz posee una amplia significación y ha sido definido según el ámbito de conocimientos desde varios puntos de vista. En biología, *interfase,* constituye la ***capa*** de un organismo que separa su interior del exterior. En la electrónica y las telecomunicaciones define el ***puerto a través del que se envían o reciben señales desde un sistema o subsistema hacia otros***. En química define la ***superficie*** entre dos fases distintas en una mezcla heterogénea.

Si vamos a la etimología de la palabra interfaz encontramos que está compuesta por dos vocablos: ***Inter*** proveniente del latín *inter*, que significa, *entre* o en *medio*, y ***Faz*****,** del latín *facies*, cuyo significado es *superficie, vista o lado de una cosa*. Por lo tanto, una traducción literal del concepto de interfaz atendiendo a su etimología podría ser ***superficie, vista, o lado mediador***.

“En el contexto de la interacción persona-ordenador, hablamos de interfaz de usuario, para referirnos de forma genérica al espacio que media la relación de un sujeto y un ordenador o sistema interactivo.” [1]

El término interfaz gráfica de usuario es aún más específico y tiene una localización determinada y definida. Si la interfaz etimológicamente supone la cara o superficie mediadora, la interfaz gráfica de

usuario supone un tipo específico de interfaz que usa metáforas visuales y signos gráficos como paradigma interactivo entre la persona y el ordenador.

Según la *Computer Desktop Encyclopedia*, una GUI es el "método común de interactuar con un ordenador que permite mostrar elementos gráficos en la pantalla." [2]

Otro autores la definen como un "artefacto interactivo, que por su diseño y a través de ciertos interfaces humanos, posibilita la interacción de una persona con el sistema informático, haciendo uso de las gramáticas visuales y verbales (signos gráficos como íconos, botones, menús y verbales como tipografía)." [1]

1.1.1 Elementos interactivos en la interfaz gráfica

En este epígrafe se definen los elementos interactivos estandarizados actualmente en las GUIs, con el objetivo de detallar la gramática subyacente en la interacción con ordenadores y precisar conceptos que serán introducidos durante el resto del documento.

Ventanas

Las ventanas son recursos interactivos utilizados para la visualización, jerarquización y navegación de la información en una interfaz gráfica de usuario. A través de las ventanas pueden ser visualizados un conjunto de documentos, aplicaciones e íconos sobre los cuales es posible realizar diversas acciones.

"Las ventanas permiten una forma relativamente fácil de interacción con la información. Su comportamiento es como el de un objeto, y pueden ser abiertas, cerradas, movidas, escaladas, ampliadas (*zoom*) y navegadas (*scrolling*)." [1]

Menús

Los menús son listas de comandos, atributos o cualquier tipo de elementos agrupados de forma estructurada. Los ítems del menú normalmente constituyen descripciones textuales, aunque también incluyen en ocasiones signos adicionales que dan información sobre la posibilidad de ser ejecutado (apagado - encendido), el estado del ítem (activado - desactivado) o el tipo o clase a la que pertenece.

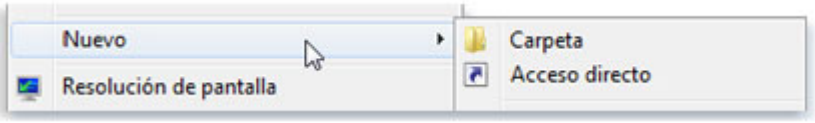

Figura 1. Menús.

Botones

Un botón es un objeto de control sobre la interfaz que posibilita introducir un dato de confirmación al sistema. Actúa como metáfora visual y funcional de los botones incluidos en los dispositivos tecnológicos.

Atendiendo a sus formas han sido catalogados en varios tipos:

Botón en relieve

Es el más común y el más usado en los sistemas operativos. Imita la gramática visual de un botón en un dispositivo físico, por lo que se le suele dar un tratamiento especial a los bordes de modo que simulen volumen. Suele incluir una descripción breve y soportar diversos estados.

Los estados son los posibles comportamientos de que dispone un ítem en relación a la interacción con el usuario. Los más habituales en los botones suelen ser:

- Activo: Estado normal de un botón sin que el usuario interactúe con él, aunque debe mostrar claramente la posibilidad de poder hacerlo.

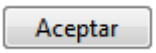

Figura 2. Botón en relieve activo.

- Inactivo: El botón muestra una apariencia difusa indicando que no puede ser seleccionado por el usuario en ese momento, pero que puede hacerlo bajo otras condiciones de selección de parámetros en el sistema.

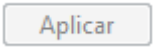

Figura 3. Botón en relieve inactivo.

- Seleccionado: Cuando el usuario ha posicionado o elegido el botón sin haberlo activado, el ítem suele cambiar de estado, normalmente indicando que está seleccionado en ese momento.

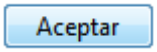

Figura 4. Botón en relieve seleccionado.

- Activado: El estado corresponde al momento en que el botón seleccionado es activado, ocurre cuando el usuario indica que desea ejecutar la acción asociada al botón cliqueando sobre él o pulsando una tecla que haga esta indicación al sistema.

Figura 5. Botón en relieve activado.

Botones en forma de radio [*RadioButton*]

Los botones en forma de radio o botones de opción única permiten realizar una selección entre dos o más opciones. Normalmente aparecen en cuadros de diálogo.

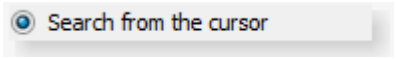

Figura 6. Botón de opción única.

Botones de confirmación [*CheckBox*]

Permiten seleccionar una o varias opciones independientes. La marca implica la aceptación de la afirmación que va enlazada a ella, y por consiguiente la falta de marca implica la negación de la afirmación.

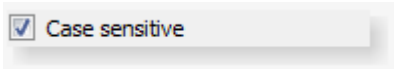

Figura 7. Botón de confirmación.

Elementos contenedores

Cuadros de diálogo [*GroupBox*]

Permite agrupar y mostrar elementos dentro de un marco, generalmente con un título asociado.

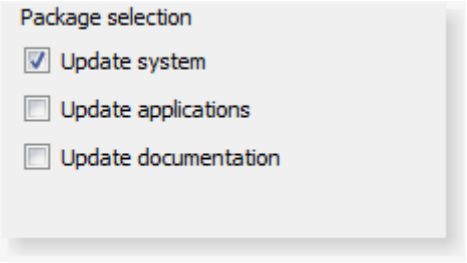

Figura 8. Cuadro de diálogo.

Fichas [*Tabs*]

Para una mejor organización de la información, las ventanas pueden estar divididas en dos o más fichas. Sólo se puede ver una ficha o un conjunto de opciones a la vez.

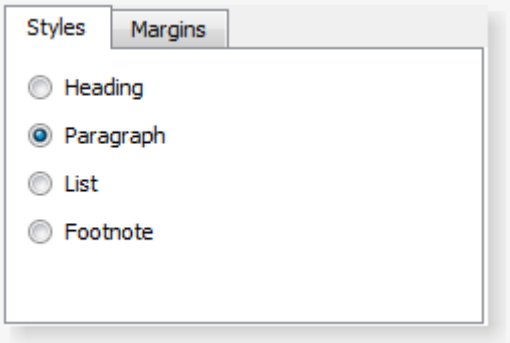

Figura 9. Fichas.

Elementos de entrada

Campo de texto [*EditBox*]

El campo de texto ha desarrollado también su propia gramática visual. Normalmente delimita un área en blanco e indica la posibilidad de introducir texto en la misma.

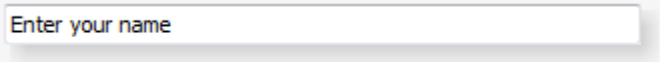

Figura 10. Campo de texto.

Control deslizante [*Slider*]

Un control deslizante permite ajustar una configuración entre un intervalo de valores.

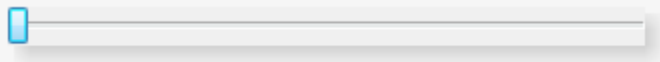

Figura 11. Control deslizante.

Elementos de salida

Los elementos de salida brindan información sobre el estado del sistema en un momento dado. Normalmente las aplicaciones reservan un área de la ventana para posicionar estos datos. Existen varios elementos de información de salida que vale la pena mencionar:

Barra de progreso [*ProgressBar*]

La barra de progreso es un elemento que indica al usuario el progreso de la acción que realiza el sistema. Todas las acciones del sistema no son realizadas de forma instantánea. Cuando el sistema requiere tiempo para realizar una acción, es fundamental dar información al usuario a través de la representación del proceso y por lo tanto del progreso de la acción.

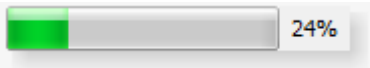

Figura 12. Barra de progreso.

Cuadro de consejo [*ToolTip*]

Es un recurso gráfico inspirado en los bocadillos de los cómics que surge en ciertos elementos de la interfaz para indicar información adicional sobre algún elemento o acción del usuario sobre el sistema.

Figura 13. Cuadro de consejo.

Barra de estado [*StatusBar*]

La barra de estado ofrece información al usuario sobre diferentes variables de la aplicación o del sistema. Normalmente es posicionada en la parte inferior de las ventanas y suele estar divida en

varias áreas de modo que en una misma horizontal se muestran diversos campos con diferentes informaciones.

Texto estático [*StaticText*]

Este componente muestra texto que el usuario no puede seleccionar o manipular. Se usa para mostrar textos de título, encabezados o incluso para mostrar resultados.

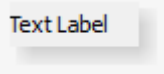

Figura 14. Texto estático.

Cuadros de lista [*ComboBox*]

Los cuadros de lista son elementos constituidos, en un estado inicial, por un campo de texto y una pestaña. El usuario puede introducir texto sobre el campo, pero si pulsa la pestaña despliega una ventana completa con elementos de navegación incluidos. Es un elemento compuesto que dispone de varias posibilidades de interacción y de acceso a la información introducida.

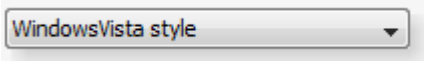

Figura 15. Cuadros de lista.

1.2 Biblioteca de interfaz gráfica de usuario CEGUI

Aquellos con experiencia en la creación de sus propios sistemas GUI, con al menos las funciones básicas, coincidirán con la idea de que esta tarea requiere tiempo, esfuerzo y presenta varios retos en la solución de problemas de programación. Los eventos, la entrada del ratón y teclado, la creación y organización en profundidad de controles, son asuntos complejos con los que se debe lidiar.

Por esta razón durante la creación de aplicaciones gráficas complejas, actividad que también requiere el empleo de gran cantidad de tiempo, debemos tomar en cuenta el uso de cualquier herramienta que reduzca costos y períodos de desarrollo.

En ciencias de la computación una biblioteca, “conjunto de subprogramas utilizados para desarrollar software que contiene código y datos que proporcionan servicios a programas independientes” [3], para el tratamiento de GUIs, puede abstraer al programador cliente de la complejidad intrínseca de estos sistemas y simplificar la programación y configuración de sus interfaces.

Ventajas del uso de bibliotecas de terceros:

- El tiempo que nos ahorra el uso de las funcionalidades ofrecidas por estas bibliotecas puede ser utilizado en otras tareas.
- Generalmente existe una comunidad de usuarios con experiencia en el uso de la biblioteca en la cual podemos apoyarnos para obtener información de valor.
- Probablemente la calidad, en términos computacionales, de estas bibliotecas sea mucho mayor que una de desarrollo propio.

Desventajas del uso de bibliotecas de terceros:

- La pérdida del control sobre los aspectos internos de la biblioteca puede resultar en dependencias no deseadas.
- A menudo las bibliotecas son proyectos que trabajan con sistemas de control de versiones, lo que vuelve más complejos sus archivos.
- Siempre se deben tener en cuenta los aspectos legales referentes a las licencias de las bibliotecas.

CEGUI es una de estas bibliotecas para la creación y control de GUIs en APIs y motores gráficos que no cuentan con esta característica de forma nativa. Fue diseñada con el objetivo de asistir a los programadores de videojuegos en la creación de interfaces, lo que no limita su uso a este tipo de aplicaciones, ya que la fuerza de su diseño reside en su configurabilidad.

“El sistema en sí mismo no carga directamente los archivos, ni realiza las tareas de dibujado o captura la entrada del usuario, CEGUI interconecta con otros módulos que realizan estas tareas usando código definido por el usuario”. [4] Esta libertad es la que permite que los usuarios utilicen CEGUI virtualmente en cualquier ambiente de codificación o sistema.

Sus principales características son:

- Código abierto bajo licencia MIT, licencia permisiva que posibilita su uso en aplicaciones tanto comerciales como no comerciales.
- Soporte multiplataforma con soluciones disponibles para las plataformas Windows, Linux y Mac OS.
- Habilidad de cargar interfaces completas o parciales desde archivos que las describen usando el lenguaje XML.
- *Renders* disponibles para OpenGL, DirectX 9, DirectX 10, DirectX 11, Irrlicht, Ogre3D, Crystal Space y Open Scene Graph.
- Interfaz clara que permite codificar *renders* para otras APIs o motores gráficos.
- Completo paquete de componentes (consulte el Anexo 1. para ver un listado de los componentes soportados).
- Varias herramientas para la edición de sus interfaces y temas.

Animaciones en CEGUI

La versión 0.7 de la biblioteca, la más reciente al comienzo de este proyecto, incluye nuevas características que permiten aplicar efectos basados en *shader* a los elementos de la interfaz. Aplicar animaciones utilizando esta técnica no es tarea sencilla. El programador debe modificar el conjunto de vértices que conforman la geometría del elemento utilizando complejos cálculos matemáticos, y además, ocuparse de hacer coincidir cada *textel* de la textura que utiliza el sistema para el componente con la nueva geometría modificada.

Si bien es cierto que con esta técnica se pueden lograr atractivos efectos, resulta ineficiente su aplicación para obtener las animaciones de más amplio uso en interfaces gráficas de usuario convencionales.

1.3 Animaciones en *frameworks* y bibliotecas GUIs

Con el objetivo de seleccionar un modelo arquitectónico que se ajuste a las necesidades de nuestro sistema e incorporar quizás otras funcionalidades útiles, se analizan brevemente las características de otros *frameworks* y bibliotecas GUIs que permiten animar sus componentes.

NaviLibrary

Licencia: LGPL.

Plataformas soportadas: Win32.

Breve descripción: Utiliza el motor Gecko para pintar interfaces creadas usando lenguajes de marcado. Soporta el uso de JavaScript y AJAX. Permite posicionar los componentes que soporta (ver componentes soportados por la biblioteca en el Anexo 1.) de forma relativa y/o absoluta.

Soporte de animaciones: El hecho de utilizar el motor Gecko para dibujar la interfaz, permite prácticamente utilizar todas las características de los contenidos web, entre ellas el uso de script animados para añadir interactividad.

Figura 16. Interfaz en una aplicación Ogre3D utilizando NaviLibrary.

Right BrainGUI Library

Licencia: LGPL.

Plataformas soportadas: Win32.

Breve descripción: Biblioteca creada por la empresa *Right Brain Games* para el desarrollo de interfaces gráficas de usuario a sus videojuegos que puede ser fácilmente portable para otros motores gráficos como Ogre3D.

Sistema de animaciones: Soporta el uso del hardware para el dibujado de múltiples ventanas en texturas separadas, lo que mejora el rendimiento y permite lograr complejos efectos de animación.

Framework Qt

Licencia: Licencias múltiples.

Plataformas soportadas: Win32/Linux/Mac OS.

Breve descripción: *Framework* multiplataforma principalmente utilizado para el desarrollo de interfaces gráficas de usuario. Presenta una arquitectura modular conformada por varios paquetes entre los cuales se encuentran *QtGui* (Componentes para el desarrollo de GUIs).

Sistema de animaciones: Incorpora un módulo que ofrece una API conveniente para el tratamiento de animaciones en Qt; permite definir, controlar y ejecutar animaciones, estados y transiciones.

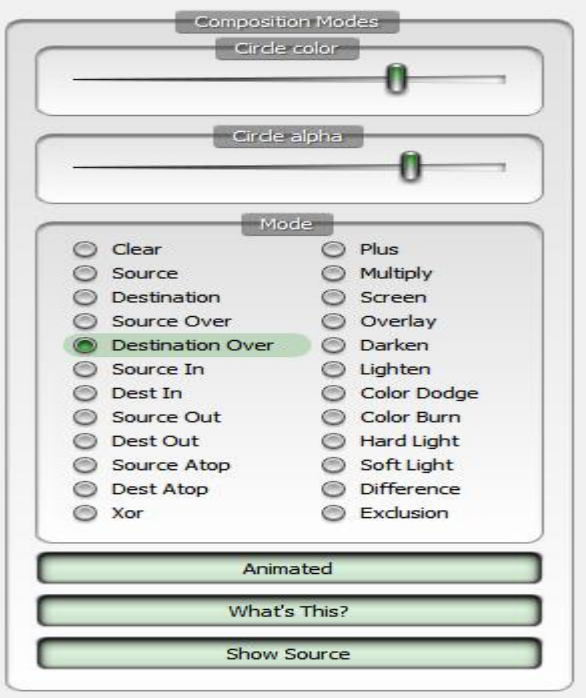

Figura 17. Interfaz utilizando Qt.

1.4 Metodologías, herramientas y lenguajes de programación a utilizar

Las metodologías, herramientas y lenguaje de programación a utilizar que se exponen a continuación no son objeto de selección en este trabajo. Estas fueron analizadas y evaluadas desde los inicios del proyecto EA.

- Metodología OpenUP.
- Lenguaje de Modelado Unificado (UML).
- Rational Rose como herramienta CASE.
- Microsoft Visual Studio 2008 como IDE de desarrollo.
- Lenguaje de programación C++.

Conclusiones

En este capítulo quedaron enunciados importantes conceptos estrechamente relacionados con el objeto de la investigación tratada. A través del análisis de *frameworks* y bibliotecas GUIs que soportan el uso de animaciones en sus interfaces, se obtuvieron los elementos necesarios para dar respuesta al problema científico de este trabajo.

Capítulo 2 Características del sistema

Introducción

En el presente capítulo se expone la aproximación técnica que más se ajusta a nuestro problema, basándonos en el anterior análisis de *frameworks* y bibliotecas GUIs que soportan el uso de animaciones. Se presentan los requisitos funcionales y no funcionales a los que debe dar cumplimiento la solución y se describen los casos de uso y el modelo de casos de uso del sistema.

2.1 Aproximación técnica

Los elementos y semántica de la arquitectura de máquina de estados utilizada por el *framework* de animaciones de Qt, nos permitirá modelar un sistema que soporte el uso de animaciones en una interfaz CEGUI.

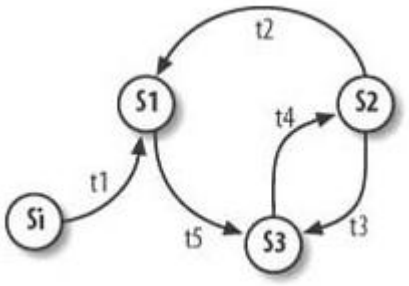

Figura 18. Diagrama de estados genérico.

Arquitectura de la animación

Cada animación contará con dos o más estados que actuarán como puntos de control durante la animación. Para cada estado se definirán propiedades y valores que describirán las condiciones de los componentes dentro de la interfaz (posición, rotación, transparencia, etc.). Las transiciones entre estos estados especificarán la forma en que variarán los valores de las propiedades que define cada estado, el tiempo y modo de ejecución de la transición.

El sistema será el encargado entonces de ejecutar las transiciones e interpolar los valores de las propiedades entre los estados de una animación.

Curvas de suavizado

Para adicionar más parámetros a la animación, se podrán definir curvas de suavizado que controlarán la velocidad de la interpolación entre los valores de inicio y finales de cada transición.

Las curvas de suavizado que se utilizarán serán las confeccionadas por Robert Penner bajo licencia BSD.

2.2 Información que se maneja

La información manejada por el sistema será la referente a los valores de las propiedades de los componentes a los que se le aplicarán animaciones en la interfaz.

2.3 Modelo de dominio

A partir del entendimiento del problema se obtiene el siguiente modelo de dominio. El glosario de términos de esta sección se incluye en el glosario de términos general.

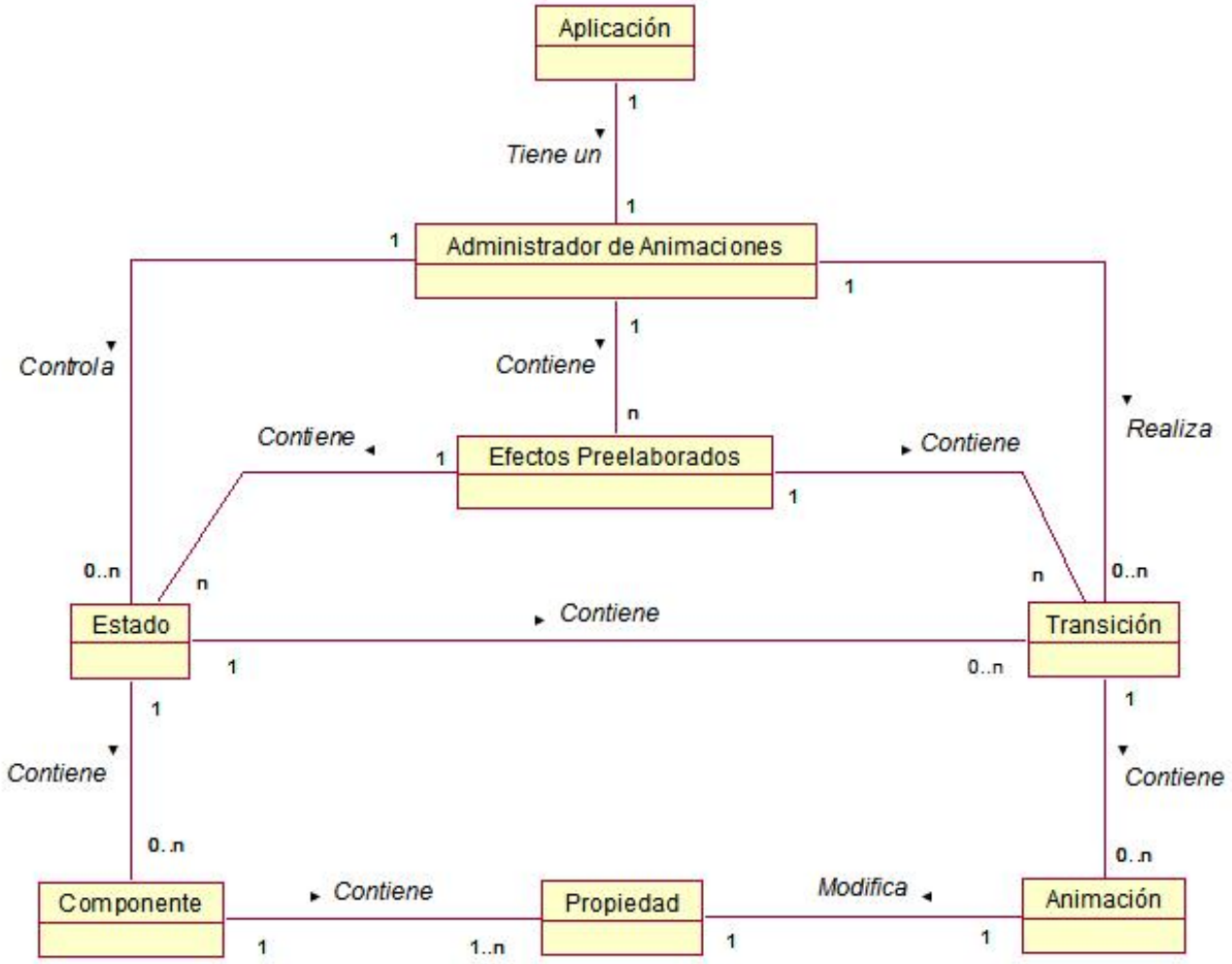

Figura 19. Modelo del dominio.

2.4 Especificación de los requisitos del software

2.4.1 Dependencias y relaciones con otro software

El sistema se incluirá en el módulo de clases BOGE que es utilizado como capa de abstracción en el desarrollo del proyecto EA, aunque puede ser utilizado también de forma independiente. Su única dependencia será la versión 0.7 de la biblioteca CEGUI.

2.4.2 Requisitos funcionales.

Teniendo en cuenta las necesidades del cliente (el proyecto EA) se elaboran los siguientes requisitos funcionales.

El sistema debe permitir:

RF1: Gestionar estados de las animaciones.

RF1.1 Agregar estado.

RF1.2 Obtener estado.

RF1.3 Eliminar estado.

RF2: Definir propiedades de los estados.

RF3: Gestionar transiciones entre los estados de las animaciones.

RF3.1 Agregar transiciones entre estados.

RF3.2 Obtener transiciones entre estados.

RF3.3 Eliminar transiciones entre estados.

RF4: Definir propiedades de las transiciones.

RF4.1 Seleccionar los tipos de propiedades que se animarán en los componentes durante una transición.

RF4.2 Definir la duración de la transición.

RF4.3: Seleccionar las curvas de suavizado que se aplicarán a las animaciones.

RF5: Controlar transiciones activas.

RF5.1 Detener las transiciones activas en un estado.

RF5.2 Pausar las transiciones activas en un estado.

RF5.3 Ejecutar una transición entre estados.

RF5.4 Seleccionar el modo de ejecución de la transición.

RF6: Aplicar efectos básicos comunes a cualquier componente en la interfaz de forma sencilla.

2.4.3 Requisitos no funcionales

El sistema debe cumplir con los siguientes requisitos no funcionales:

Soporte:

*RNF1***:** Sistema multiplataforma.

Diseño e Implementación:

*RNF2***:** Debe ser implementado usando el lenguaje de programación C++ y regido por la filosofía de la programación orientada a objetos.

Ayuda y Documentación:

RNF3: Documentación del código de acuerdo a los estándares y estilos definidos en el proyecto EA.

RNF4: Registro de las actividades del sistema en una bitácora para facilitar la depuración de errores lógicos.

2.5 Definición de los casos de uso del sistema

A partir de las funcionalidades que debe ofrecer la solución se reconoce al actor y los casos de uso del sistema a desarrollar.

2.5.1 Definición de los actores del sistema

Actores	Justificación
Programador	El sistema será diseñado para su uso por los programadores del proyecto EA en la programación de animaciones sobre los componentes de la biblioteca CEGUI 0.7.

Tabla 1. Justificación de los actores del sistema.

2.5.2 Casos de uso del sistema

CU_1	Gestionar estados.
Actor	Programador.
Propósito	Permitir al actor agregar, obtener y eliminar estados.
Referencia	RF1.

Tabla 2. CU Gestionar estados.

CU_2	Definir propiedades del estado.
Actor	Programador.
Propósito	Definir las propiedades de un estado en una animación.
Referencia	RF2.

Tabla 3. Definir propiedades del estado.

CU_3	Gestionar transiciones.
Actor	Programador.
Propósito	Permitir al actor agregar, obtener y eliminar transiciones.
Referencia	RF3.

Tabla 4. CU Gestionar transiciones.

CU_4	Definir propiedades de la transición.
Actor	Programador.
Propósito	Permitir al actor seleccionar los tipos de propiedades que se animarán en los componentes, la curva de suavizado a utilizar en cada animación y la duración de la transición.
Referencia	RF4.

Tabla 5. CU Definir propiedades de la transición.

CU_5	Controlar transiciones activas.
Actor	Programador.
Propósito	Permitir al actor detener, pausar, ejecutar y seleccionar el modo de ejecución de una transición.
Referencia	RF5.

Tabla 6. CU Controlar transiciones activas.

CU_6	Aplicar efecto básico.
Actor	Programador.
Propósito	Permitir al actor aplicar efectos básicos precodificados.
Referencia	RF6.

Tabla 7. CU Aplicar efecto básico.

2.5.3 Diagrama de actores y casos de uso del sistema

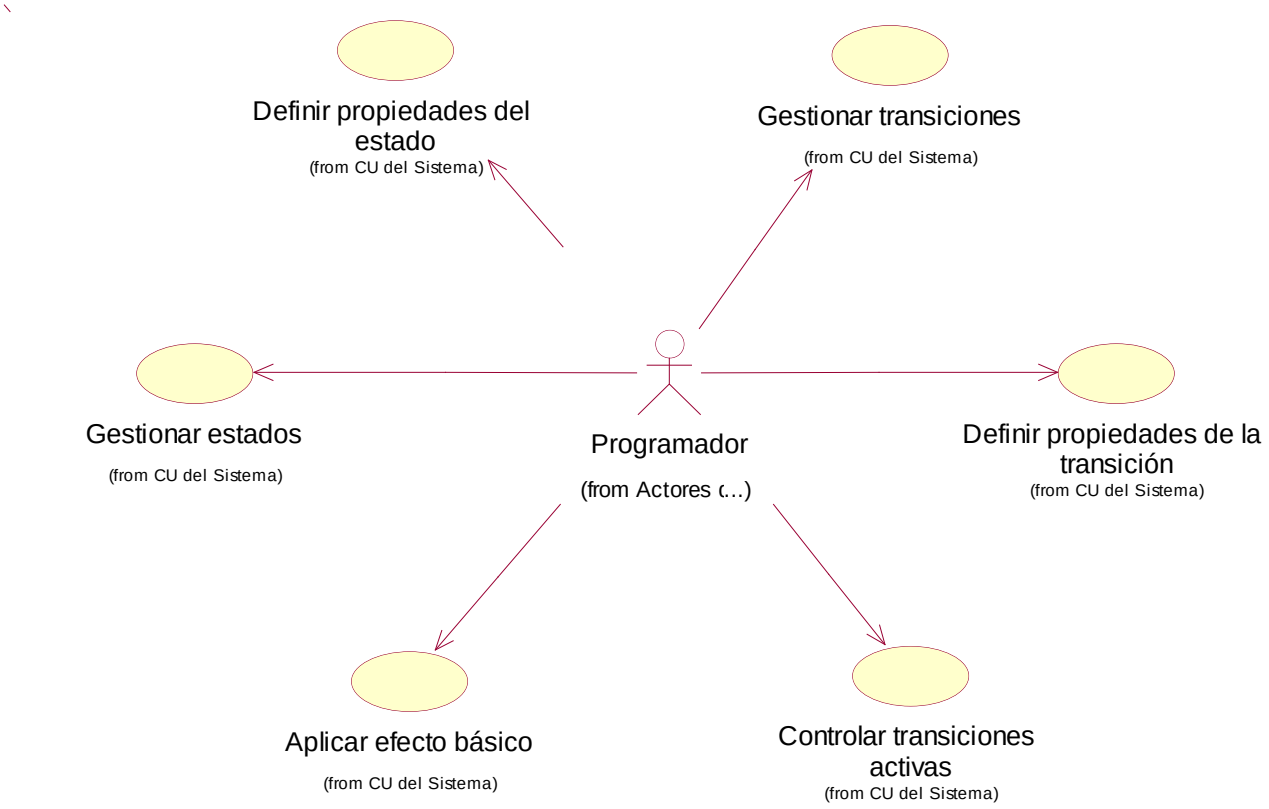

Figura 20. Diagrama de actores y casos de uso del sistema.

2.5.4 Expansión de los casos de uso del sistema

CU Gestionar estados

Caso de Uso	
CU_1	Gestionar estados.
Propósito	Permite al actor agregar, obtener y eliminar estados.
Actores: Programador. [Inicia]	
Resumen: El caso de uso inicia cuando el programador decide crear, obtener o eliminar un estado de animación y finaliza cuando el sistema lleva a cabo la acción solicitada.	
Referencia	RF1.
Flujo Normal de Eventos	
Acción del actor	**Respuesta del sistema**
1. El programador le indica al sistema agregar, obtener o eliminar un estado.	1.1 En el caso que la acción indicada sea: a) Agregar nuevo estado, ir a la Sección “Agregar Estado”. b) Obtener estado, ir a la Sección “Obtener Estado”. c) Eliminar estado, ir a la Sección “Eliminar Estado”.
Sección “Agregar Estado”	
Acción del actor	**Respuesta del sistema**
2. El programador proporciona un nombre único que identificará al estado en el	2.1 El sistema crea un nuevo estado si no hay otro estado

sistema.	suscrito al sistema con el identificador dado.
Sección "Obtener Estado"	
Acción del actor	**Respuesta del sistema**
2. El programador proporciona el nombre único que identifica al estado en el sistema.	2.1 El sistema devuelve el estado solicitado suscrito al sistema con el identificador dado, en caso que exista alguno.
Sección "Eliminar Estado"	
Acción del actor	**Respuesta del sistema**
2. El programador proporciona el nombre único que identifica al estado en el sistema.	2.1 El sistema elimina el estado suscrito con el identificador dado, en caso que exista alguno.
Precondiciones	
Poscondiciones	Se agrega, obtiene o elimina un estado del sistema.

Tabla 8. Expansión del CU Gestionar estados.

CU Definir propiedades del estado

Caso de Uso	
CU_2	Definir propiedades del estado.
Propósito	Definir las propiedades de un estado en una animación.
Actores: Programador. [Inicia]	
Resumen: El caso de uso inicia cuando el programador decide definir las propiedades de un estado en	

una animación y finaliza cuando el sistema lleva a cabo la acción solicitada.	
Referencia	RF2.
Flujo Normal de Eventos	
Acción del actor	**Respuesta del sistema**
1. El programador proporciona el nombre único del componente CEGUI, el tipo y valor de la propiedad que se desea definir.	1.1 Si un componente con el nombre dado, está registrado en CEGUI, el sistema asigna la propiedad al estado en cuestión.
Precondiciones	El componente cuya propiedad se agregará al estado debe estar registrado en CEGUI.
Poscondiciones	Se asigna una propiedad al estado en cuestión.

Tabla 9. Expansión del CU Definir propiedades del estado.

CU Gestionar transiciones

Caso de Uso	
CU_3	Gestionar transiciones.
Propósito	Permite al actor agregar, obtener y eliminar transiciones.
Actores: Programador. [Inicia]	
Resumen: El caso de uso inicia cuando el programador decide agregar, obtener o eliminar una transición y finaliza cuando el sistema lleva a cabo la acción solicitada.	
Referencia	RF3.

Flujo Normal de Eventos	
Acción del actor	**Respuesta del sistema**
1. El programador le indica al sistema agregar, obtener o eliminar una transición.	1.1 En el caso que la acción indicada sea: a) Agregar nueva transición, ir a la Sección "Agregar Transición". b) Obtener transición, ir a la Sección "Obtener Transición". c) Eliminar transición, ir a la Sección "Eliminar Transición".
Sección "Agregar Transición"	
Acción del actor	**Respuesta del sistema**
2. El programador proporciona el nombre único que identifica al estado destino de la transición en el sistema.	2.1 El sistema crea y devuelve la nueva transición en caso que no exista otra transición con las mismas características entre los dos estados en cuestión.
Flujo Alterno	
Acción del actor	**Respuesta del sistema**
2. El programador proporciona el nombre único que identifica al estado destino de la transición en el sistema.	2.1 En caso que exista otra transición con las mismas características entre los dos estados en cuestión, esta será devuelta por el sistema.
Sección "Obtener Transición"	
Acción del actor	**Respuesta del sistema**
2. El programador proporciona el nombre único que identifica al estado destino de la transición en el sistema.	2.1 El sistema devuelve la transición solicitada, en caso que exista.

Sección “Eliminar Transición”	
Acción del actor	**Respuesta del sistema**
2. El programador proporciona el nombre único que identifica al estado destino de la transición en el sistema.	2.1 El sistema elimina la transición nombrada, en caso que exista.
Precondiciones	
Poscondiciones	Se crea, obtiene o elimina una transición entre estados.

Tabla 10. Expansión del CU Gestionar transiciones.

CU Definir propiedades de la transición

Caso de Uso	
CU_4	Definir propiedades de la transición.
Propósito	Permitir al actor seleccionar los tipos de propiedades a animar en los componentes y la duración de la transición.
Actores: Programador. [Inicia]	
Resumen: El caso de uso inicia cuando el programador decide definir las propiedades de una transición y finaliza cuando el sistema lleva a cabo la acción solicitada.	
Referencia	RF4.
Flujo Normal de Eventos	
Acción del actor	**Respuesta del sistema**
1. El programador le indica al sistema que	1.1 En el caso que la acción indicada sea:

seleccionará los tipos de propiedades que se animarán durante una transición o que definirá el tiempo de una transición.	a) Seleccionar los tipos de propiedades a animar, ir a la Sección “Seleccionar Propiedades a Animar”. b) Definir tiempo de la transición, ir a la Sección “Definir Tiempo Transición”.
Sección “Seleccionar Propiedades a Animar”	
Acción del actor	**Respuesta del sistema**
2. El programador proporciona el nombre único del componente CEGUI, el tipo de animación y la curva de suavizado a utilizar.	2.1 Si un componente con el nombre proporcionado, está registrado en CEGUI, el sistema agrega una animación del tipo dado a la transición.
Sección “Definir Tiempo Transición”	
Acción del actor	**Respuesta del sistema**
2. El programador proporciona el tiempo de duración de la transición.	2.1 El sistema ajusta el tiempo de la transición en cuestión al valor proporcionado por el programador o a cero en caso que el valor proporcionado no sea válido.
Precondiciones	
Poscondiciones	Se lleva a cabo la acción solicitada.

Tabla 11. Expansión del CU Definir propiedades de la transición.

CU Controlar transiciones activas

Caso de Uso	
CU_5	Controlar transiciones activas.
Propósito	Permite al actor detener, pausar, ejecutar y seleccionar el modo de ejecución de una transición.
Actores: Programador. [Inicia]	
Resumen: El caso de uso inicia cuando el programador decide detener, pausar o ejecutar y seleccionar el modo de una transición.	
Referencia	RF5.
Flujo Normal de Eventos	
Acción del actor	**Respuesta del sistema**
1. El programador le indica al sistema que detendrá, pausará o ejecutará una transición entre estados.	1.1 En el caso que la acción indicada sea: a) Detener la transición, ir a la Sección “Detener Transición”. b) Pausar la transición, ir a la Sección “Pausar Transición”. c) Ejecutar la transición, ir a la Sección “Ejecutar Transición”.
Sección “Detener Transición”	
Acción del actor	**Respuesta del sistema**
2. El programador proporciona el nombre único que identifica al estado cuyas transiciones activas se detendrán.	2.1 El sistema detiene las transiciones activas en el estado suscrito al sistema con el identificador dado, en caso que exista alguno.
Sección “Pausar Transición”	

Acción del actor	Respuesta del sistema
2. El programador proporciona el nombre único que identifica al estado cuyas transiciones activas se pausarán.	2.1 El sistema pausa las transiciones activas en el estado suscrito al sistema con el identificador dado, en caso de que exista alguno.
Sección "Ejecutar Transición"	
Acción del actor	**Respuesta del sistema**
2. El programador proporciona los nombres únicos que identifican al estado inicial y final, y el modo de ejecución de la transición que utilizará el sistema.	2.1 El sistema busca la transición definida entre los dos estados proporcionados, en caso de que existan.
	2.2 El sistema comienza la ejecución de dicha transición utilizando el modo de ejecución proporcionado por el programador.
Precondiciones	
Poscondiciones	Se lleva a cabo la acción solicitada.

Tabla 12. Expansión del CU Controlar transiciones activas.

CU Aplicar efecto básico

Caso de Uso	
CU_6	Aplicar efecto básico.
Propósito	Permitir al actor aplicar efectos básicos precodificados.
Actores: Programador. [Inicia]	

Resumen: El caso de uso inicia al aplicar un efecto precodificado a un componente de la interfaz.	
Referencia	RF6.
Flujo Normal de Eventos	
Acción del actor	**Respuesta del sistema**
1. El programador proporciona el nombre del componente, el tipo de efecto que se le aplicará y la duración de la animación.	1.1 El sistema crea dos estados temporales para la animación y les asigna las propiedades en correspondencia con el componente al que se le aplica el efecto.
	1.2 El sistema crea una transición entre los estados temporales recién creados de acuerdo al tipo de efecto seleccionado por el programador.
	1.3 El sistema define el tiempo y ejecuta la transición entre los estados temporales.
	1.4 Al finalizar la transición el sistema elimina los estados temporales creados durante el proceso.
Precondiciones	
Poscondiciones	Se lleva a cabo la acción solicitada.

Tabla 13. Expansión del CU Aplicar efecto básico.

Conclusiones

En este capítulo se recopilan los requisitos funcionales y no funcionales, además de definir y describir los casos de uso de acuerdo a lo que espera el cliente que realice el sistema.

Capítulo 3 Diseño e implementación del sistema

Introducción

En este capítulo se describen todos los aspectos del sistema a construir mediante el diagrama de clases del diseño y los diagramas de secuencia correspondientes a los casos de uso. Presenta además los artefactos generados durante la posterior implementación de los casos de uso.

3.1 Diseño del sistema

3.1.1 Estándares de documentación y codificación

Con el objetivo de facilitar la comprensión, mantenimiento y distribución del código fuente se utilizan los siguientes estándares de codificación.

Estándares de código

- Se generará un fichero (*.h, *.cpp) por cada clase y tendrán el mismo nombre que la clase que se implemente en ellos.
- Formato de los nombres de las clases:

  <Módulo ID><Nombre de la clase>

  Donde <Módulo ID> es un identificador del módulo al cual pertenece la clase.

  Ejemplo: *class IAEntity*
- Formato de los atributos y variables de clases:

  <p><Name>

  Donde <Name> debe reflejar claramente el propósito del atributo.

Ejemplo: *pEntityName*

- Los identificadores de las variables locales o variables de funciones debe ser claro y preciso.
- Las funciones miembros tienen que cumplir con las siguientes especificaciones:
  - Nombres claros que identifiquen el propósito de la función y redactados en idioma inglés.
  - La primera palabra que identifica la función miembro siempre comenzará con letra minúscula; en caso que el identificador sea redactado utilizando más de una palabra, las palabras siguientes a la primera comenzarán con mayúscula y sin espacio entre ellas.

    Ejemplo: *void calculateForce()*
  - Formato de las funciones de tipo “eventos”:

    <on><Function Name>

    Ejemplo: *void onKeyPress()*
  - Formato de las funciones de tipo “privadas”:

    <_><Function Name>

    Ejemplo: *void _calculateForce()*
  - Formato de los parámetros de las funciones:

    <arg><Name>

    Ejemplo: *void calculateForce(float argVelocity)*
- Formato de los enumerados:

  < Módulo ID><_><Name>

  Ejemplo: *enum Behaviors { IA_FOLLOW_PATH, IA_SEEK};*

El módulo de clases al que se integra el sistema propuesto en el proyecto EA, utilizará la herramienta de auto documentación *Doxygen*. Se utilizarán los formatos de documentación requeridos por esta herramienta para mantener la coherencia en el código fuente y la documentación asociada en el mencionado proyecto.

Estándares de documentación

- Estilo de bloques de documentación JavaDoc:

  Se adopta el estilo de bloques de documentación JavaDoc, el cual consiste de un bloque de comentario de estilo C de la siguiente manera:

```
/**
*... text...
*/
```

- Uso de los comandos \brief o @ brief:

  Utilizados para mostrar una descripción del código.

```
/**
* @brief Descripción breve.
* Continuación de la descripción breve.
*
* Descripción detallada comienza aquí, nótese
* que se debe dejar una línea en blanco para lograr
* tener las dos descripciones (breve y detallada)
*/
```

3.1.2 Diagrama de clases del diseño

Diagrama de clases del diseño por paquetes

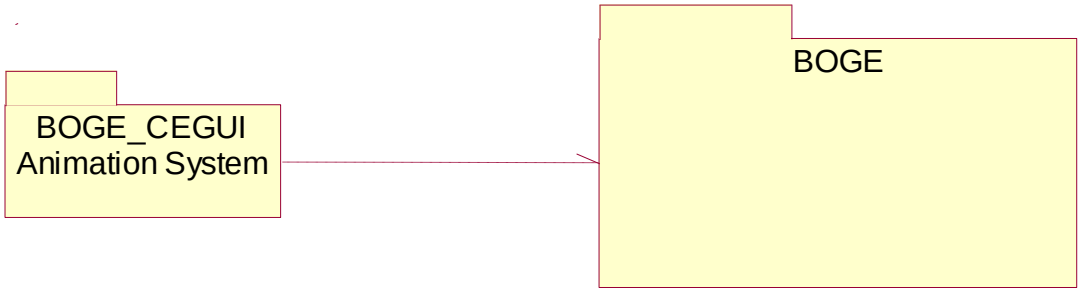

Figura 21. Diagrama de clases del diseño por paquetes.

Paquete BOGE_CEGUI Animation System

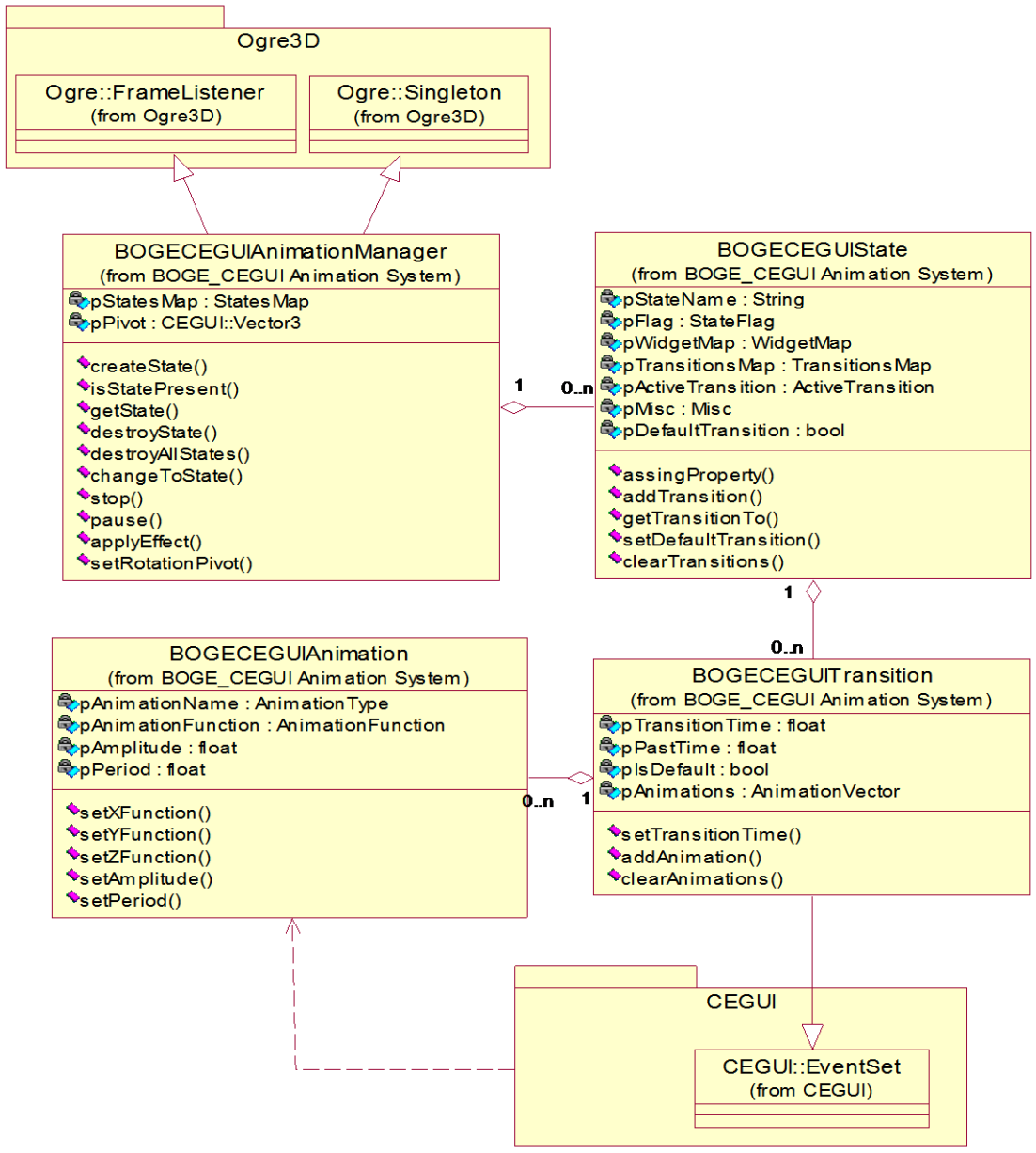

Figura 22. Paquete BOGE_CEGUI Animation System.

3.1.3 Definiciones de diseño que se aplican

Patrones Creacionales:

***Builder*:** Abstrae el proceso de creación de un objeto complejo, centralizando dicho proceso en un único punto.

***Singleton*:** Garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia.

Patrones Estructurales:

Composite: Permite tratar objetos compuestos como si de uno simple se tratase.

Patrones de Comportamiento:

Command: Encapsula una operación en un objeto, permitiendo ejecutar dicha operación sin necesidad de conocer el contenido de la misma.

Iterator: Permite realizar recorridos sobre objetos compuestos independientemente de la implementación de estos.

3.1.4 Diagramas de secuencia

A continuación se presentan los diagramas de secuencia agrupados por casos de uso. Corresponden a las relaciones que se establecen de forma secuencial entre los objetos de las principales funcionalidades descritas en los escenarios de los casos de uso del sistema.

CU Gestionar estados

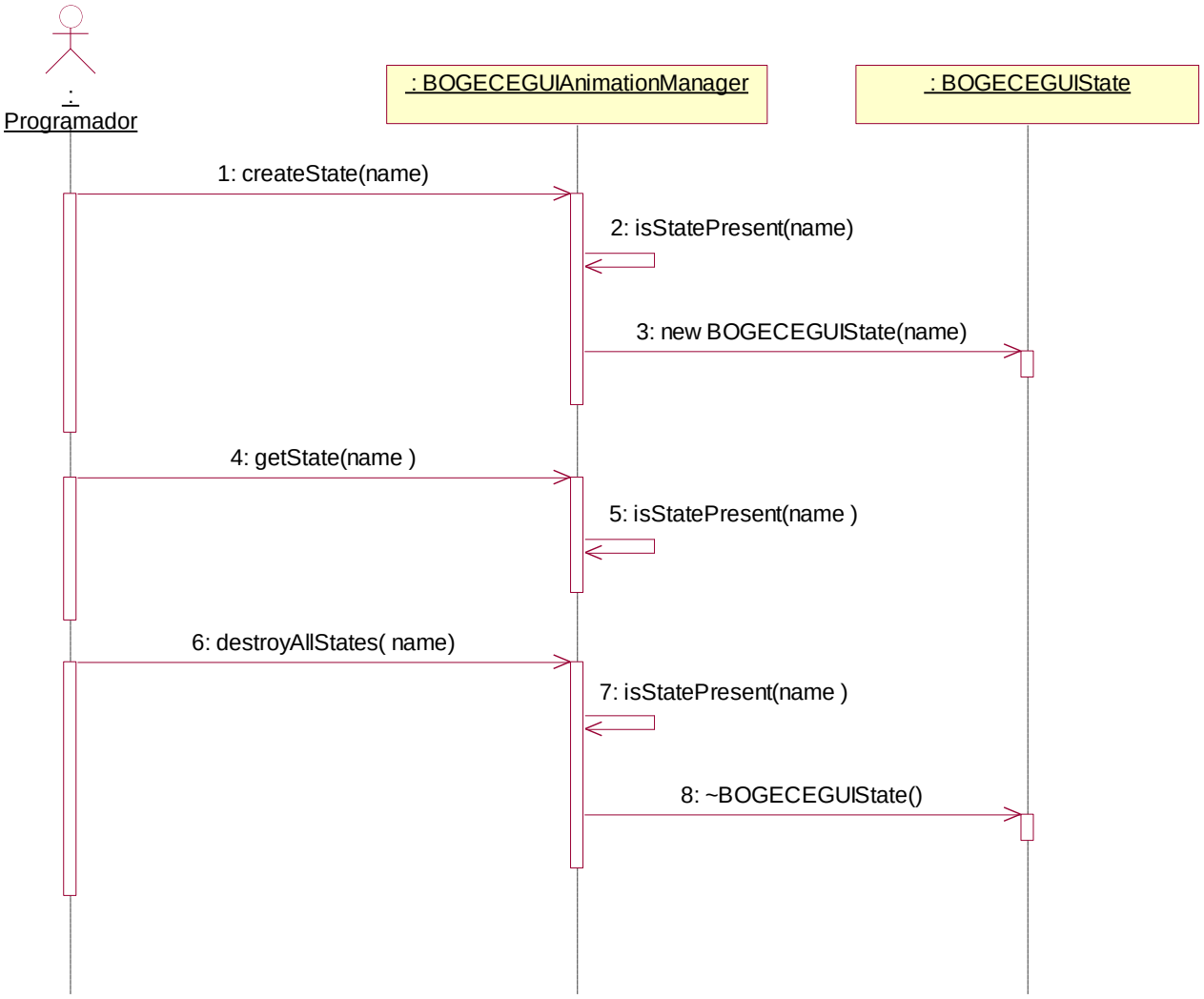

Figura 23. Diagrama de secuencia "CU Gestionar estados".

CU Definir propiedades de los estados

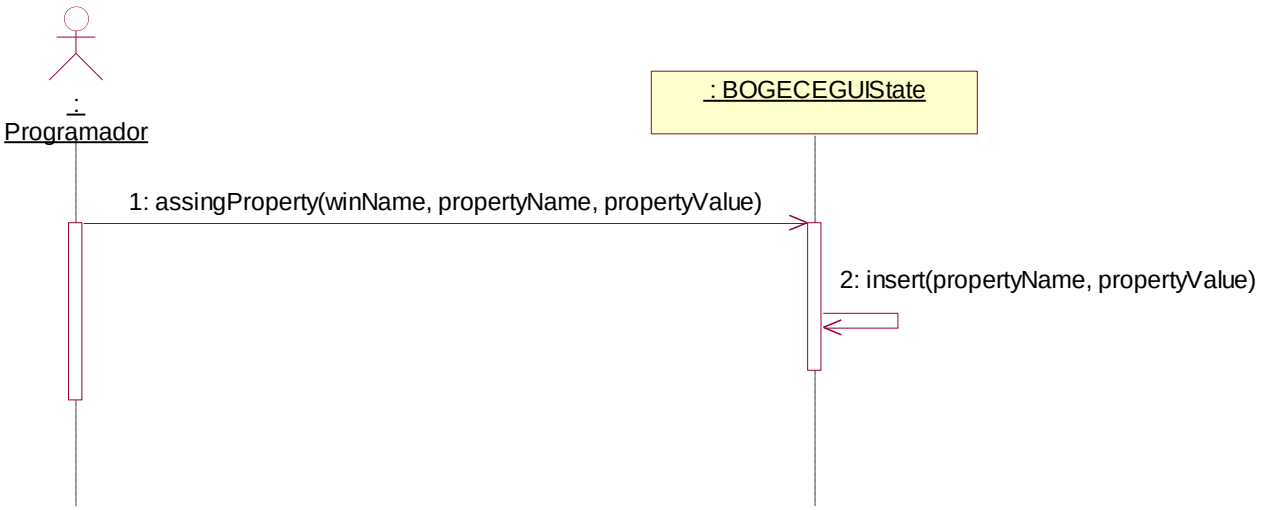

Figura 24. Diagrama de secuencia "CU Definir propiedades del estado".

CU Gestionar transiciones

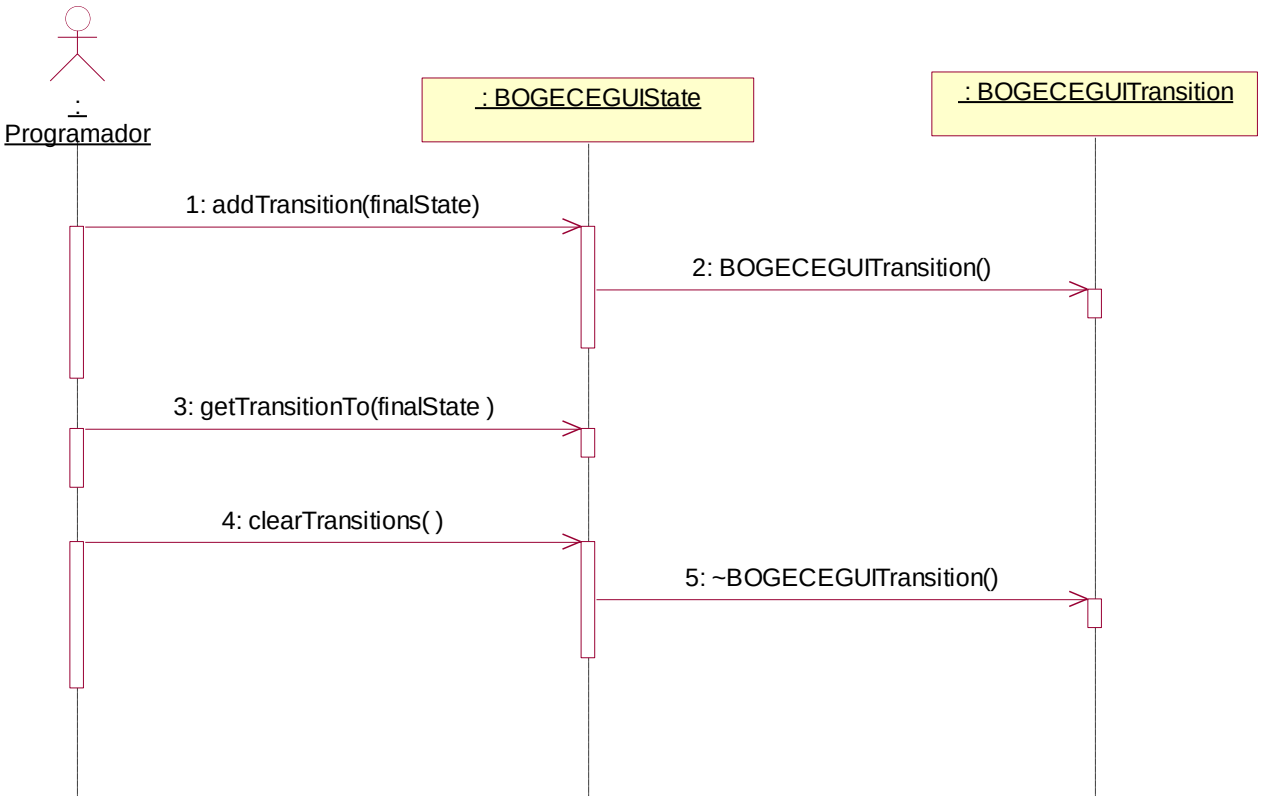

Figura 25. Diagrama de secuencia "CU Gestionar transiciones".

CU Definir propiedades de las transiciones

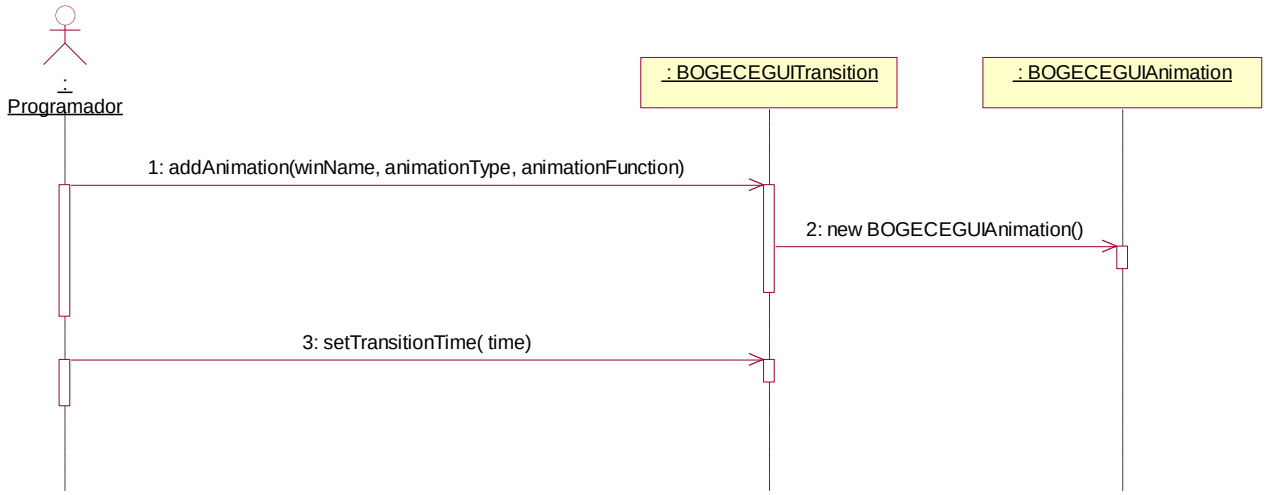

Figura 26. Diagrama de secuencia "CU Definir propiedades de la transición".

CU Controlar transiciones activas

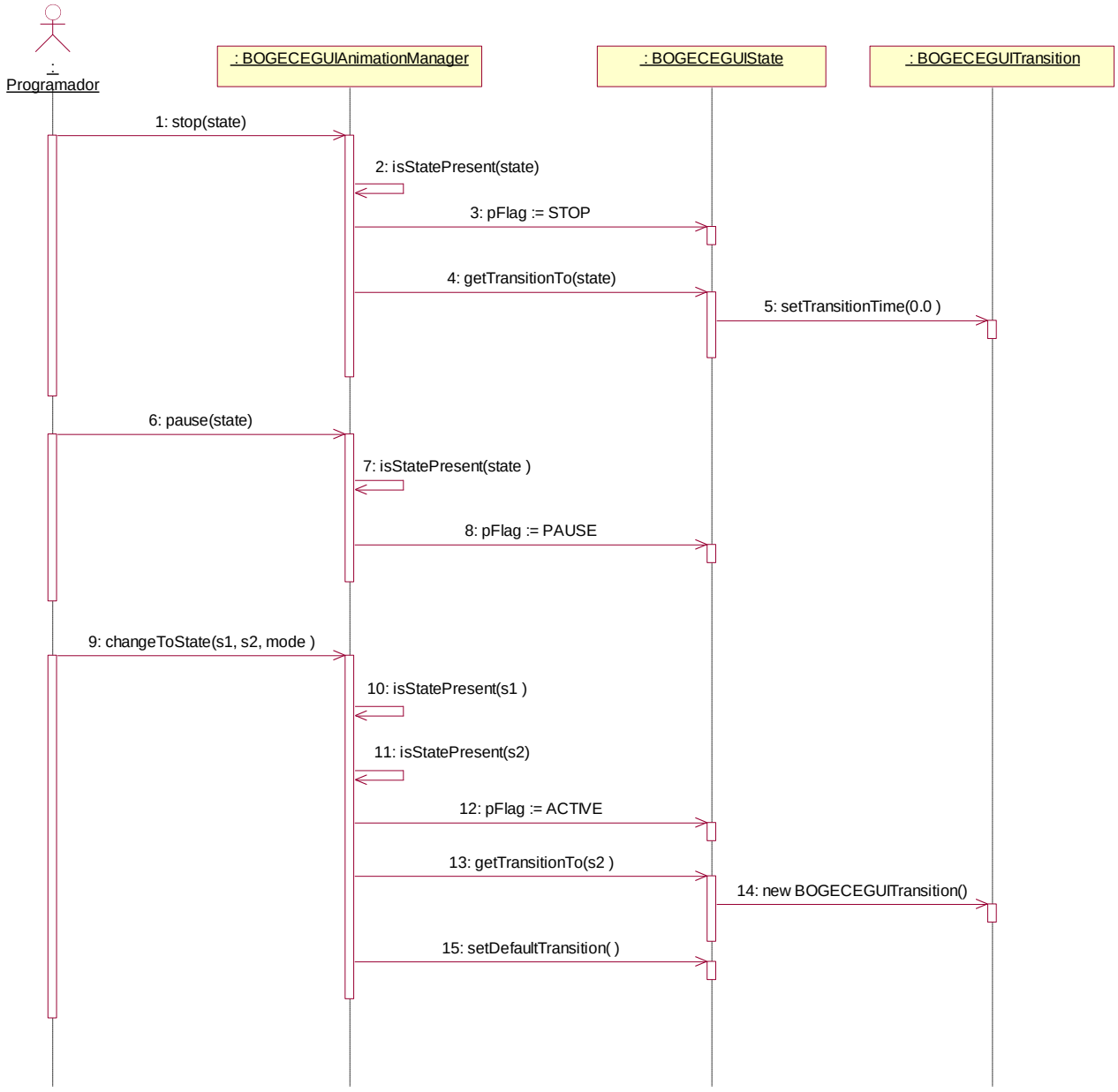

Figura 27. Diagrama de secuencia "CU Controlar transiciones activas".

CU Aplicar efecto básico

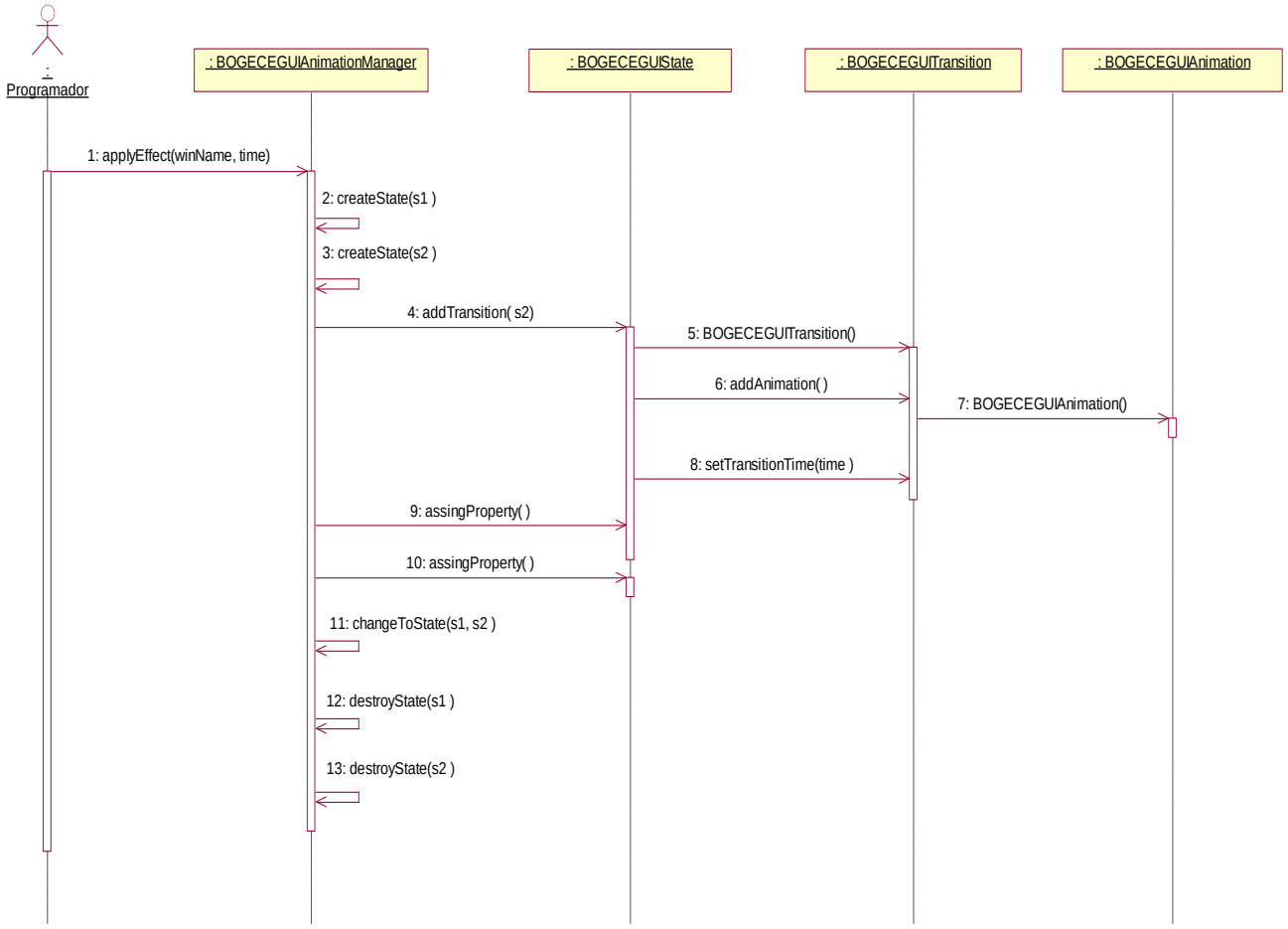

Figura 28. Diagrama de secuencia "CU Aplicar efecto básico".

3.2 Implementación del sistema

3.2.1 Diagrama de despliegue

Este diagrama consta de una sola computadora personal, por lo tanto se considera que no es necesario reflejarlo en este documento.

3.2.2 Diagrama de componentes

Este diagrama refleja la creación de los componentes físicos que formarán el sistema.

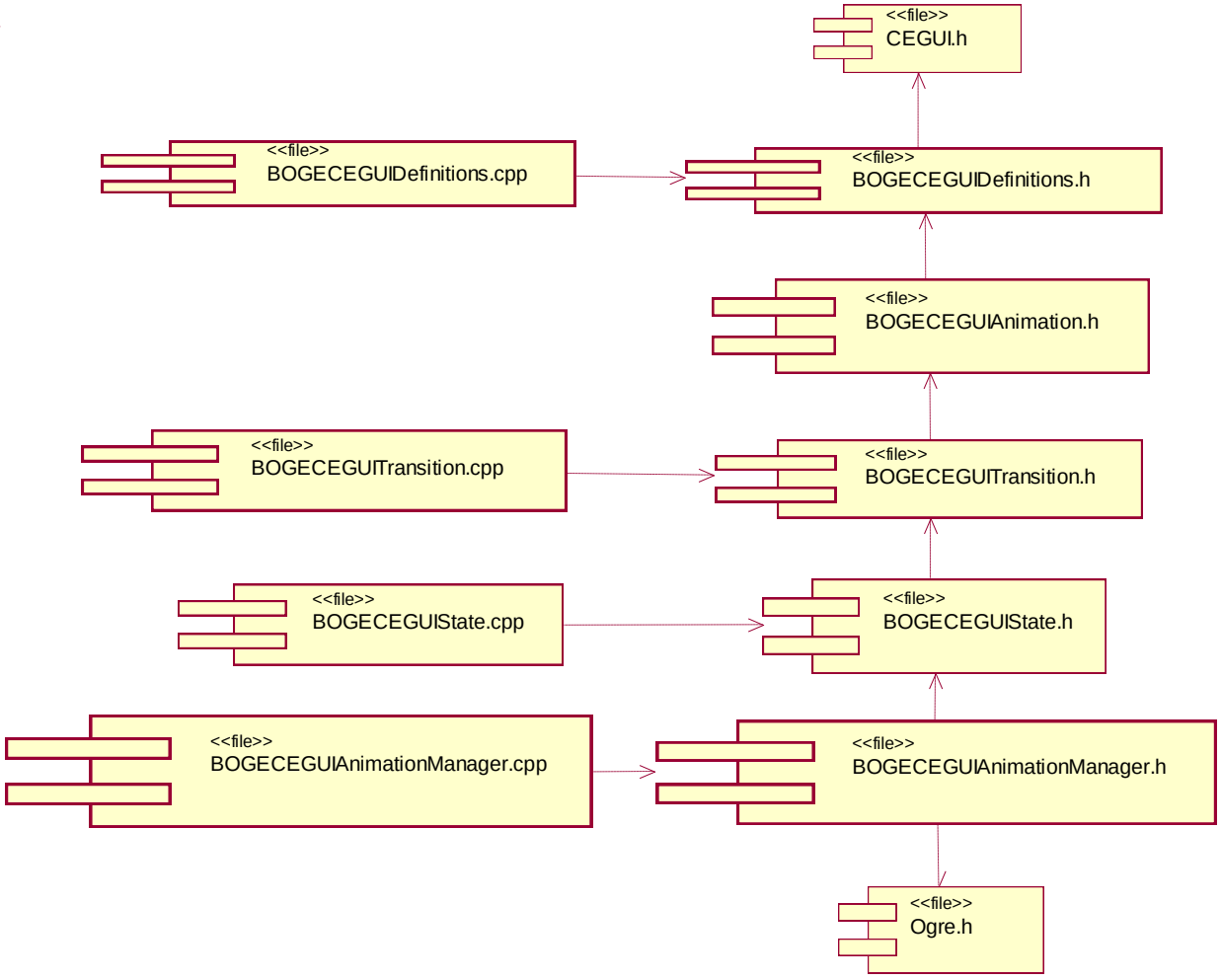

Figura 29. Diagrama de componentes.

3.3 Validación de la solución

La utilización de una metodología ágil y adaptable para la construcción del sistema, permite reducir los artefactos generados durante todo el proceso de desarrollo del software, especialmente durante las actividades de pruebas.

Durante la fase de construcción el programador será el responsable de ejecutar pruebas al concluir cada realización de casos de uso. Esto garantizará que la implementación funcione como fue diseñada.

Además, al finalizar cada iteración el sistema será sometido a pruebas unitarias que aseguren el correcto funcionamiento del módulo de forma individual y a pruebas de integración para comprobar que el sistema se ejecute correctamente en conjunción con el módulo BOGE donde se inserta en el proyecto EA.

Siguiendo uno de los principios de la metodología propuesta, *desarrollo evolutivo para obtener retroalimentación y mejoramiento continuo*, el sistema, desde sus primeras versiones funcionales, será entregado al cliente para su uso y aprobación. Esto permitirá obtener una temprana y continua retroalimentación y demostrarles el incremento progresivo de la funcionalidad.

Esta filosofía permitirá la corrección y refinamiento de errores e inconformidades del cliente durante todo el ciclo de vida, validando el cumplimiento de los requisitos del software. Para dar fe de ello se incluirá en el Anexo 2., la carta de aceptación del software por el cliente.

Conclusiones

Con la información obtenida durante la confección del capítulo que finaliza, un diseño detallado y los artefactos necesarios para el desarrollo del sistema, se puede proceder a la implementación de las funcionalidades más necesarias.

Conclusiones

Dando cumplimiento al objetivo de este proyecto:

- Se obtuvo un sistema para los componentes de la biblioteca CEGUI 0.7, que permite el uso de animaciones en las interfaces gráficas de usuario utilizadas en el proyecto EA.
- Se desplegó y validó la solución en la aplicación cliente.

Recomendaciones

- Mejorar e incorporar efectos visuales prefabricados al sistema.
- Incorporar efectos del tipo *Render* y curvas de suavizado que complementen y mejoren las existentes en el sistema.

Bibliografía

1. Marrero Expósito, C. Interfaz gráfica de usuario: Aproximación semiótica y cognitiva. Universidad de La Laguna. La Laguna, España. 2004-2006.

2. Nokia Corporation and/or its subsidiaries. Qt Reference Documentation. [En línea]. Nokia. 2008-2010. [Consultado el: 12 de enero de 2011]. Disponible en: http://qt.nokia.com/

3. Lyons, B. The Open Unified Process: A Brilliant, Collaborative March into Open Source. [En línea]. Disponible en: http://www.numbersix.com/news/n6articles/openUp.html

Referencias bibliográficas

1. Marrero Expósito, C. Interfaz gráfica de usuario: Aproximación semiótica y cognitiva. Universidad de La Laguna. La Laguna, España. 2004-2006.

2. GUI. [En línea]. Computer Desktop Encyclopedia. EBSCO - EBSCOhost Online Research Databases. [Consultado el: 10 de enero de 2011]. Disponible en: http://web.ebscohost.com

3. Biblioteca (Informática). [En línea]. Wikipedia.org, 2011. [Consultado el: 12 de enero de 2011]. Disponible en: http://es.wikipedia.org/wiki/Biblioteca_(informática)

4. Everything about CEGUI. [En línea]. CEGUI Wiki, 2011. [Consultado el: 17 de enero de 2011]. Disponible en: http://www.cegui.org.uk/wiki

Anexos

Anexo 1. Componentes soportados por los *frameworks* y bibliotecas GUIs analizadas.

Bibliotecas	**Texto estático**	**Barra de progreso**	**Botón en relieve**
CEGUI	Si	Si	Si
Navi	Si	Si	Si
Right Brain GUI	Si	Si	Si
Framework Qt	Si	Si	Si
	Botones de confirmación	**Campo de texto**	**Control deslizante**
CEGUI	Si	Si	Si
Navi	Si	Si	Si
Right Brain GUI	Si	Si	No
Framework Qt	Si	Si	Si
	Barra de desplazamiento	**Campo de texto multilínea**	**Botón de opción única**
CEGUI	Si	Si	Si
Navi	Si	Si	Si
Right Brain GUI	Si	Si	Si
Framework Qt	Si	Si	Si
	Cuadro de lista	**Menús**	**Fichas**
CEGUI	Si	Si	Si

Navi	Si	Si	Si
Right Brain GUI	Si	Si	No
Framework Qt	Si	Si	Si

Anexo 2. Acta de aceptación del producto

Acta de aceptación

ACTA DE ACEPTACIÓN

En función de la ejecución del proyecto: Entrenadores Aduaneros, se hace entrega del producto que se relaciona a continuación.

Lista de productos que serán aceptados:

- *Sistema de Animaciones para CEGUI 0.7*

Entrega	**Recibe**
Nombre y Apellidos: Jaime Glez. Pacheco	**Nombre y Apellidos:** Jaime Glez. Campistruz
Cargo: Estudiante	**Cargo:** Jefe de Proyecto
Firma:	**Firma:**

Comentarios:

La aceptación del producto relacionado anteriormente se realizó teniendo en cuenta el cumplimiento de los requisitos exigidos.

Glosario de abreviaturas

3D: Tercera Dimensión o en Tres Dimensiones.

AJAX – *Asynchronous JavaScript And XML.*

API – *Application Programming Interface* (Interfaz de Programación de Aplicaciones).

BOGE – *Basic Ogre3D Game Engine*.

BSD – *Berkeley Software Distribution License.*

CASE – *Computer Aided Software Engineering* (Ingeniería de Software Asistida por Ordenador).

CEDIN - Centro de Desarrollo Industrial.

CEGUI: *Crazy Eddies Graphical User Interfaces* (Interfaz Gráfica de Usuario de Crazy Eddies).

CSS – *Cascading Style Sheets* (Hojas de Estilo en Cascada).

GUI: *Graphical User Interfaces* (Interfaz Gráfica de Usuario).

HTML – *Hyper Text Markup Language* (Lenguaje de Marcado de Hipertexto).

IDE – *Integrated Development Environment* (Entorno de Desarrollo Integrado).

JS – JavaScript.

LGPL – *GNU Lesser General Public License* (Licencia Pública General Reducida de GNU).

MIT – *Massachusetts Institute of Technology License* (Licencia Del Instituto de Tecnologías de Massachusetts).

Ogre3D: *Object-Oriented Graphics Engine* (Motor Gráfico Orientado a Objetos).

OpenGL: *Open Graphics Library* (Biblioteca de gráficos abierta).

RAD – *Rapid Application Development* (Desarrollo rápido de aplicaciones).

UCI: Universidad de las Ciencias Informáticas.

UML – *Unified Modeling Lenguage* (Lenguaje Unificado de Modelado).

XML – *Extensible Markup Lenguage* (Lenguaje de Marcas Extensible).

Glosario de términos

Glosario de términos del dominio:

Animación – Variación de los valores de una propiedad de un componente de la interfaz en determinado tiempo.

Aplicación – Software que hace uso del sistema.

Componente – Objeto gráfico o control con el cual el usuario interactúa.

Efecto preelaborado – Efecto visual básico aplicable a un elemento en la interfaz sin necesidad de implementación. Incluye tres tipos de efectos de animación: Efectos de entrada, Efectos de salida y Efectos de énfasis.

Estado – Conjuntos de propiedades que determinan la condiciones de los componentes en la interfaz.

Administrador de animaciones – Entidad que controla los estados y que ejecuta las transiciones entre ellos.

Propiedades – Atributos o cualidades de los componentes de la interfaz.

Transición – Determina las propiedades que variarán al pasar de un estado a otro durante una animación. Controla la velocidad de los efectos de la animación.

Glosario de términos general:

ActionScript – Lenguaje de programación orientado a objetos, utilizado en especial en aplicaciones web animadas realizadas en el entorno Adobe Flash.

Crystal Space – *Framework* para el desarrollo de aplicaciones 3D.

Doxygen – Acrónimo de **dox** (document) **gen** (generator). Generador de documentación para código fuente.

Framework – Estructura conceptual y tecnológica en base a la cual otro proyecto de software puede ser organizado y desarrollado.

Gecko – Motor de *renderizado* libre escrito en C++.

Irrlicht – Motor 3D gratuito y de código abierto.

JavaDoc – Utilidad para la generación de documentación de APIs en formato HTML a partir de código fuente Java.

JavaScript – Lenguaje interpretado utilizado para acceder a objetos en aplicaciones.

Shader – Tecnología reciente destinada a proporcionar al programador interacción con la GPU.

Tema (Skin) – Serie de elementos gráficos que al aplicarse sobre un determinado software, modifican su apariencia externa.

Texel – Contracción del inglés *texture pixel*. Unidad mínima de una textura aplicada a una superficie usada en gráficos por ordenador.

Índice de figuras y tablas

Índice de figuras

Índice de tablas