UNIVERSIDAD NACIONAL DE LOJA

Área de la Energía, las Industrias y los Recursos Naturales no Renovables

Carrera de Ingeniería en Sistemas

TEMA:

"*Construcción de una Herramienta Multimedia para el desarrollo de diapositivas tridimensionales bajo licencia GPL, haciendo uso de la tecnología Java 3D, orientada al sector académico y profesional del Área de La Energía las Industrias y los Recursos Naturales No Renovables*"

Tesis previa a la obtención del
Título de Ingeniero en Sistemas

Autores:

Alfredo Rolando Macas Chocho

Klever Daniel Macas Flores

Director:

Ing. Luis Antonio Chamba Eras

LOJA - ECUADOR

2010

CERTIFICACIÓN

Ing. Luis Antonio Chamba Eras

DOCENTE DE LA CARRERA DE INGENIERÍA EN SISTEMAS DE LA UNIVERSIDAD NACIONAL DE LOJA, DIRECTOR DE TESIS

C E R T I F I C A:

Que los señores egresados Alfredo Rolando Macas Chocho y Klever Daniel Macas Flores, realizaron el trabajo de investigación titulado **"Construcción de una herramienta multimedia para el desarrollo de diapositivas tridimensionales bajo licencia GPL, haciendo uso de la tecnología Java 3D, orientada al sector académico y profesional del Área de La Energía las Industrias y los Recursos Naturales No Renovables"** bajo mi dirección y asesoramiento, mismo que fue revisado, enmendado y corregido minuciosamente. En virtud que la Tesis reúne, a satisfacción, las cualidades de fondo y forma exigidas para un trabajo de este nivel, autorizo su presentación, sustentación y defensa ante el tribunal respectivo.

Loja, Septiembre del 2010

…….

Ing. Luis Antonio Chamba Eras

DIRECTOR DE TESIS

AUTORIA

Las ideas conceptos y opiniones que se exponen en el trabajo de investigación que se presenta, son de absoluta responsabilidad de los autores, excepto las que se encuentran debidamente citadas.

……. …….

Alfredo Rolando Macas Chocho **Klever Daniel Macas Flores**

AGRADECIMIENTO

Agradecemos primeramente a Dios por habernos permitido llegar hasta donde estamos ya que Él es la fortaleza y la luz que nos guía en los senderos de la vida, ya que sin Él no somos nada.

A la Universidad Nacional de Loja, al Área de la Energía, las Industrias y los Recursos Naturales no Renovables, por haber permitido culminar nuestros estudios en la Carrera de Ingeniería en Sistemas, en especial a su planta Docente y Administrativa quien hace posible que al igual que nosotros, nuevos jóvenes cumplan sus sueños de superación.

Al Ing. Luis Antonio Chamba Eras, Director del presente trabajo investigativo, gracias a su colaboración dedicación pudimos finalizar con éxito esta meta que nos planteamos.

A la planta docente del Área de la Energía, las Industrias y los Recursos no Renovables, quienes supieron colaborar de manera desinteresada en nuestra investigación, brindándonos un poco de su tiempo para poder evaluar nuestra aplicación.

A nuestros familiares que estuvieron ahí cuando más los necesitamos, amigos, compañeros y todas las personas que de una u otra manera ha contribuido en el desarrollo de este gran sueño.

Gracias a todos Uds.

Los Autores

DEDICATORIA

Dedico el presente trabajo a las personas que más quiero y admiro mis padres, a mis hermanos que me han brindado su apoyo y han confiado en mí todo el tiempo, gracias por haberme apoyado y lograr que alcance mis metas.

Klever

A mis padres, que en todo momento de la vida me brindaron su apoyo incondicional y me enseñaron que en este mundo todo se puede alcanzar mientras seas constante y no te des por vencido jamás, a mis hermanos, amigos y todos los que estuvieron presentes cuando más los necesite.

Alfredo

1. RESUMEN

La aplicación para crear diapositivas tridimensionales está orientada al sector tanto académico como profesional del Área de la Energía las Industrias y los Recursos Naturales No Renovables de la Universidad Nacional de Loja, la cual permitirá crear presentaciones de diapositivas con varios componentes tales como texto 3d, formas(cono, cilindro, pirámides, etc...), sonido, carrusel y objetos 3D que pueden ser configurados de acuerdo a las necesidades del usuario, además se podrá agregar animaciones de rotación, posición, escala, color y transparencia dependiendo del componente, también se manejaran texturas y material para el texto 3D y las formas lo cual enriquecerá aún más el diseño de las diapositivas.

Con la presente aplicación se pretende mejorar la forma de exposiciones de trabajos, proyectos o investigaciones, de manera que el interés del público sea mayor logrando así un mejor aprendizaje, ya que se podrá manipular de distintas maneras los diferentes componentes que contengan la presentación, representando de esta manera en una solo diapositiva lo que con otro software se lograría con varias. Además la aplicación será de código abierto lo cual servirá como material de estudio y podrán hacer uso de ella ya sea para mejorarla o como base para realizar nuevos trabajos con relación al mismo campo.

Como parte fundamental para el desarrollo de este proceso de investigación, se empezó con la determinación del tema, planeamiento de los objetivos y la selección de una metodología la cual permitió la planificación de procedimientos mediante métodos y técnicas. La metodología de desarrollo de software seleccionada para el presente proyecto fue ICONIX, la misma que hace un fuerte uso del Lenguaje Unificado de Construcción de Modelos (UML), facilitando de esta manera el desarrollo del proyecto.

Cabe destacar que el software empleado para el desarrollo del proyecto en su totalidad es no propietario, el lenguaje de programación empleado es Java JDK 1.6 con sus entorno de programación Netbeans 6.8, apoyándose en las librerías de java3D 1.5 los mismos que se los puede encontrar de forma gratuita en sus respectivos sitios web.

2. ABSTRACT

The application to create tridimensional slides is addressed to the academic or professional personnel of the Area of Energy, Industries and Non-Renewable Natural Resources of the National University of Loja, which will allow creating presentations whit several slides to which we could add components such as 3D text, shapes(cone, cylinder, pyramids, etc.), sound, carrousel, and 3D objects that can be configured according to the user, moreover, it could be established animations of rotation, position, scale, color and transparency dependent of component, likewise it will manage texture and material for 3D text and shapes which will improved the final presentation.

Through the present application it pretends to improve the way of works, projects or researches, obtaining more interest from the people and achieving better learning outcomes. As in this program it works with 3D objects, these will be manageable in different ways in only one slide instead of using several ones using other software; also, being an application of open code, it will serve as study material to the students who could use it to improve or take it as a model to carry out new projects.

As a main part to the development of this research process, it began with selection of the theme, objectives setting, and the methodology selection which let the planning of the procedures carried out throughout methods, techniques, analysis, design, programming and approval of software. The software methodology development selected to this project was ICONIX, which makes a hard use of UML helping to make a better application design.

It is important to highlight that the software employed to the project development in this totality has no owners; the programing language employed is Java JDK 1.6 with its programing environments Netbeans 6.8, supporting themselves in the Java 3D 1.5 libraries findable for free in their corresponding web sites.

3. INDICE GENERAL

INDICE DE FIGURAS

INDICE DE TABLAS

4. INTRODUCCION

La Universidad Nacional de Loja ha integrado la investigación como parte fundamental en la formación de sus profesionales. A través de ella, se propone brindar soluciones efectivas a las complejas problemáticas del mundo actual.

Apoyados en los conocimientos adquiridos durante la etapa de estudios se ha establecido dar solución a un problema que se ha presentado en base a lo que tiene que ver con la presentación de trabajos en diapositivas, lo cual se encuentra limitado a lo que se refiere al manejo y manipulación de los componentes en tres dimensiones por ello se ha creído conveniente realizar una aplicación de escritorio para el desarrollo de diapositivas tridimensionales bajo licencia GPL, haciendo uso de la tecnología JAVA 3D, la cual está orientada para el sector académico y profesional del Área de la Energía las Industrias y los Recursos Naturales No Renovables.

Debido a la complejidad y el tiempo que llevaría la construcción de una aplicación completa para el manejo de presentación de diapositivas se ha creído conveniente realizarla con las funciones más importantes las mismas que permitirán la elaboración de presentaciones capaces de representar lo mismo e incluso más de lo que se podría realizar con cual cualquier software del mercado dedicado al mismo fin.

Para llevar a cabo este proyecto se planteó el objetivo principal:

"Construir una herramienta multimedia para el desarrollo de diapositivas tridimensionales, haciendo uso de la tecnología JAVA 3D, la cual está orientada al sector académico y profesional del Área de la Energía las Industrias y los Recursos No Renovables".

Para cumplir con el objetivo propuesto nos apoyamos en base a los siguientes puntos:

- Desarrollar una galería básica de objetos tridimensionales del mundo real.
- Desarrollar un asistente de software configurable para la creación de diapositivas tridimensionales.
- Desarrollar un algoritmo que permita visualizar múltiples objetos en una misma diapositiva.
- Desarrollar un componente de software configurable para los efectos tridimensionales.

- Integrar un componente de software para el sonido.
- Desarrollar un árbol de navegación para los objetos tridimensionales agregados a la diapositiva.
- Implantación y liberación del software.

5. METODOLOGIA

Para el desarrollo de presente proyecto se tomó en cuenta los diferentes métodos investigativos, técnicas y una metodología de software que permita cumplir los objetivos propuestos.

5.1. Métodos.

Los Métodos que utilizamos en el presente trabajo de investigación son los siguientes:

Método Científico: Lo utilizamos como guía principal de toda la investigación, ya que a través de este se planteó el problema, los objetivos: general y específico, además nos permitió la organización, procesamiento, análisis, e interpretación de la información obtenida.

Método Inductivo-Deductivo: Los cuales siguen el proceso analítico sintético que satisfacen los requerimientos propios de las ciencias informáticas (recolección de datos, análisis de la información e interpretación de los hechos y descubrimiento de nuevos procedimientos).

5.2. Técnicas

Considerando la importancia y pertinencia del tema de esta investigación y para que los métodos empleados tengan mayor eficacia, fue indispensable utilizar las siguientes técnicas:

Observación: En base a la técnica de observación pudimos recolectar las características básicos que debe poseer un software de este tipo lo cual permitió determinar los requerimientos básicos del sistema tomando como modelos los programas de desarrollo de presentaciones Power Point e Impress, ya que son los más utilizados por los usuarios.

Encuestas: Las cuales estuvieron dirigidas a la planta docente del Área de la Energía las Industrias y los Recursos No Renovables, con la finalidad de validar nuestra aplicación lo cual fue de gran importancia ya que así pudimos recolectar diferentes opiniones y sugerencias de parte de los encuestados para de esta manera corregir errores o hacer nuevas mejoras a la aplicación.

5.3. Materiales

Consulta Bibliográfica: Se determinó el sustento teórico-científico de las consecuencias de esta investigación. Por medio de esta técnica se procedió a la utilización de resúmenes y gráficos para acotar la información necesaria.

Software y Hardware: Para el desarrollo de esta aplicación en lo que se refiere a software se empleó lo siguiente:

- Lenguaje de programación JAVA.
- Entorno de Programación IDE Netbeans 6.8.
- Librerías de JAVA 3D 1.5.
- Para la etapa de diseño se utilizó Enterprise Architec 7.0.
- El modelado de objetos 3D se realizó mediante Blender 2.9.
- Sistemas Operativos: Windows XP, Windows 7, Windows Vista y Ubuntu Ultimate Edition 8.1. Esto con la finalidad de probar que la aplicación funcione correctamente en cada uno de ellos.

El hardware empleado que posibilito el correcto desarrollo del proyecto fue siguiente:

- Computador Intel Pentium IV, CPU de 2.0 GHz, RAM 1GB.
- Computador Intel Core i5, CPU de 2.27 GHZ, RAM 4GB
- Impresora Canon IP1000.

5.4. Metodología para el proceso de desarrollo de software.

En el campo de la informática y especialmente en el desarrollo de software, es de vital importancia tomar en cuenta ciertas metodologías de trabajo, las mismas que permitan conseguir los objetivos propuestos desde el inicio del proyecto hasta su culminación.

Por tal razón la metodología que se consideró importante con respecto al desarrollo del proyecto es la ICONIX ya que es una metodología ágil que permite un mejor desarrollo de proyectos y ha sido empleado en la mayoría de proyectos realizados por parte de los desarrolladores.

Dentro de la metodología de desarrollo de software se presentan algunas tareas las cuales se citan a continuación:

Análisis de Requisitos: En esta etapa se inició con la fase de especificación de requisitos que en principio deberían ser parte del sistemas para lo que se tomó en cuenta las necesidades de los usuarios que utilizan por lo menos algún tipo de software para el desarrollo de presentaciones, los cuales brindaron ideas de las opciones básicas que debe poseer este tipo de software, posteriormente tomando como base las diferentes ideas propuestas se generó una tabla de requerimientos tanto funcionales y no funcionales que permitieron el desarrollo del proyecto. Posteriormente se construyó el diagrama de clases o modelo del dominio el cual representa el modelo estático del sistema.

Para simular el diseño del sistema se elaboró un prototipación rápida del sistema lo cual se repitió y se finalizó una vez que en el sistema se incluyó todas las características necesarias y expuestas en los requisitos.

A continuación se procedió a realizar el modelo de casos de uso, el que permitió comprender de mejor manera los requerimientos que debe cumplir la aplicación, y a precisar que información se quiere intercambiar y a describir lo que debe hacerse para obtener el resultado esperado.

Análisis y Diseño Preliminar: Una vez identificados los diferentes casos de uso se procedió a la descripción de cada uno de ellos cuyo propósito es la delineación de los procesos, curso normal y alterno de eventos que se dan entre el sistema y los actores involucrados.

Seguidamente se elaboró los diagramas de robustez los cuales sirven para comprobar que los casos de uso estén formulados correctamente y de forma completa.

Diseño: En esta etapa se elaboró los diagramas de secuencia los mismos que muestran como interactúa los diferentes objetos de la aplicación con usuario a través del tiempo. Seguidamente completamos el modelo inicial del dominio con los diferentes atributos y métodos apoyándonos en los diagramas de secuencia.

Implementación: En esta etapa se llevó a cabo la construcción de los componentes de la solución y creación de código fuente que cumplirá con los requerimientos planteados en la etapa de análisis. Además se debe tener en cuenta las siguientes características:

La reusabilidad ya que al ser una aplicación de código abierto sus componentes deberán ser útiles para diferentes aplicaciones, la extensibilidad para de esta forma poder modificar con facilidad el software y la confiabilidad descartando de esta forma la posibilidad de errores.

Terminada la etapa de implementación se detalló los diferentes diagramas como son de componentes y despliegue, los mismos que permiten tener una mejor perspectiva del sistema, dando así por terminado el desarrollo de la aplicación.

6. FUNDAMENTACIÓN TEÓRICA

CONTENIDO

La diapositiva

Java 3d

Metodología de Desarrollo de Software

Arquitectura del software

Licencias de Software

Herramientas de desarrollo

CAPITULO 1: LA DIAPOSITIVA

6.1. La Diapositiva

Las diapositivas son cada uno de los elementos que constituyen la presentación y cada una de ellas podría identificarse con una lámina o página. Se pueden crear y modificar de manera individual.

El número de diapositivas varía en función del contenido de la presentación, pero en general, podemos decir que es aconsejable que cada diapositiva contenga una única idea o elemento de información.

Tradicionalmente, las diapositivas eran "las del proyector" que podían obtenerse a través de fotografías o incluso hacerse manualmente. Hoy en día existen también las diapositivas informatizadas, elaboradas en varios programas con estos fines, que incluso ofrecen la posibilidad de movimiento y sonido.

La diapositiva es un instrumento idóneo para la explicación de determinadas materias donde el elemento visual tiene especial protagonismo: arte, presentación de datos estadísticos, imágenes del mundo natural, esquemas educativos y, en definitiva, la diapositiva debe esta donde interese mantener una imagen fija para flexionar sobre ella**.**

La diapositiva le ofrece la posibilidad de adaptarse a su estilo personal de enseñar y al nivel de la clase. Por su bajo coste y por su simplicidad de uso, la diapositiva es una herramienta insustituible en la enseñanza.

6.1.1. Características de las diapositivas[1]

- Son mucho más luminosas, brillantes, comunicativas y sugestivas que las fotografías.
- Se proyectan en grande y así puede participar toda la clase.
- Las imágenes son de buena calidad y no cansan la vista.
- Se pueden mostrar al ritmo que el profesor necesite.
- Se puede cambiar su colocación, añadir o eliminar, según convenga.

[1] María José Ballesteros, Gloria María Gallego. La Diapositiva [En línea], Características de las diapositivas, [http://www.uclm.es/profesorado/ricardo/Diapositivas/Diapositivas_Fca_2004.doc], [Consulta: 20 de enero del 2010]

- Se adapta a todas las áreas de enseñanza.
- Es un recurso que favorece la motivación.

6.1.2. Diapositivas informatizadas

Las diapositivas informatizadas son documentos informáticos que pueden incluir textos, esquemas, sonidos, animaciones, fragmentos de video…etc., y pueden visionarse por la pantalla del ordenador como si se tratara de una proyección de diapositivas.

Si se dispone de un cañón proyector de video o de una pantalla de cristal líquido y un retroproyector, las diapositivas informáticas pueden proyectarse sobre una pantalla externa.

6.1.3. Clasificación de las diapositivas[2]

De acuerdo con el tipo de apoyo que brinde al expositor, la diapositiva se clasifica como: de reafirmación, ampliación y complementación.

Cuando un texto describe lo más objetivamente posible lo que representa una imagen, se está en presencia de una diapositiva de reafirmación. El texto reafirma lo que muestra la imagen. Si el texto amplía el mensaje que la imagen muestra, se tiene una diapositiva de ampliación. El texto va más allá de la mera descripción de la imagen. Por último, cuando el texto agrega a la imagen información que esta no muestra ni permite deducir, se trata de una diapositiva de complementación. La imagen actúa como un complemento del texto.

Dentro de los tres tipos de diapositivas señaladas, pueden existir variantes, así sucede con la forma negativa de la diapositiva de reafirmación. El texto de la diapositiva afirma que la limpieza dentro de un laboratorio tiene que ser escrupulosa; sin embargo, la imagen representa a un investigador que trabaja en un laboratorio donde predomina el

---

[2] Leonardo Cruz Verduit, 2002. Aspectos metodológicos básicos para la preparación y el empleo de las diapositivas. [En línea], Clasificación de las diapositivas, Editorial CENIC, La Habana, Cuba, [http://bvs.sld.cu/revistas/aci/vol10_4_02/aci030402.htm], [Consulta: 20 de Noviembre del 2009]

caos y la desorganización. La imagen, aparentemente, contradice lo que afirma el texto. La imagen en todos los casos apoya al texto, complementa o refuerza el mensaje.

6.1.4. Diapositiva de Texto

La forma más común de escribir un texto para una diapositiva, antes de la introducción de las computadoras, era mediante las máquinas de escribir. Para lograr una diapositiva de texto aceptable a partir de estos medios, era necesario que el texto fuera escrito en letras mayúsculas, que los tipos de la máquina estuvieran bien limpios, que se golpearan las teclas con la misma fuerza, y se empleara una cinta nueva y de buena calidad.

Durante la última década del siglo XX y hasta el presente, la computadora ha sustituido a la máquina de escribir en la elaboración de diapositivas de texto, como resultado de la aparición en el mercado de múltiples paquetes de programas avanzados que facilitan esa operación, como Microsoft Power Point, Corel Presentations, Lotus Smart Suit, Margi Systems Presenter-to-go, Astound Presentation, entre otros. Dichos programas permiten obtener infinitas variantes, debido a sus muchas prestaciones y herramientas entre las que se encuentran la ilimitada cantidad de fuentes tipográficas, de paletas de colores, el amplio rango de presentaciones y plantillas, así como las extensas bibliotecas con otros muchos recursos.

No obstante, es recomendable de igual forma, escribir el texto en altas y que se presente de forma centrada. Los textos justificados a la derecha o a la izquierda pudieran quedar muy próximos a los bordes de la diapositiva. En el caso que el texto se escriba en altas y bajas, debe utilizarse un tamaño de letra igual o superior a los 12 puntos.

6.1.5. Diapositivas de Gráficos

Las diapositivas de texto son las de uso más frecuente, pero las de gráficos han adquirido notable importancia en los últimos tiempos, debido a la posibilidad que ofrecen de transmitir eficazmente los resultados de complejas investigaciones, sin la necesidad de engorrosas explicaciones. No en vano se ha afirmado que un gráfico transmite más de mil palabras. Un diseño apropiado de las diapositivas se basa en el empleo mesurado de los textos y la utilización profusa de las ilustraciones. Las

diapositivas se distinguen por los tipos de gráficos que presentan, entre las principales tenemos:

a) De línea

b) De bola o pastel

c) De barras o columnas.

Los gráficos de línea se utilizan preferentemente para dar una idea inmediata de los objetivos o de las variables cambiantes. Son los más empleados y regularmente establecen una relación entre dos o más variables. Frecuentemente se utilizan para indicar porcentajes ascendentes o descendentes. Se aconseja utilizar más de dos líneas para realizar comparaciones entre variables. La combinación de colores en estos casos resulta de mucha utilidad. Siempre que sea posible, deben usarse patrones (notación o simbología) que ayuden a identificar las líneas. Se debe evitar el empleo de abreviaturas, acrónimos, siglas, etcétera, salvo las de reconocimiento internacional; las tipografías (letras y números) deben tener un tamaño tal que asegure su adecuada legibilidad y debe evitarse la aglomeración de cifras y datos comparativos.

En el caso de los gráficos de bola o pastel se utilizan principalmente para mostrar el peso relativo de los elementos que forman un sistema. Se emplean unidades relativas, especialmente, el tanto por ciento. El pastel debe tener una proporción acorde a la diapositiva (se debe dar un tamaño adecuado a los márgenes), las secciones del pastel no deben ser más de siete, y ninguna debe representar menos del 5 % del total, se debe estar alerta y comprobar que las partes sumen el 100 %, el tamaño de las rebanadas debe responder al porcentaje que representan (las cifras respectivas deben colocarse preferentemente en su interior), los textos necesarios pueden situarse dentro del pastel o fuera de este (preferiblemente fuera cuando las secciones representan valores pequeños) y el color debe usarse para dar énfasis y estética a la diapositiva, de acuerdo con su influencia psicofisiológica sobre el espectador.

Los gráficos de columnas o barras son ideales para representar datos comparativos; ellos ofrecen gran flexibilidad en la ubicación de textos y cifras, así como en el empleo del color. También deben cumplir con requisitos fundamentales para su utilización: se

deben comparar las barras horizontales y verticales en la representación de los datos (seleccionar las que se usarán en la diapositiva), nunca se debe hacer un gráfico con más de siete barras, ellas deben ser bien proporcionadas (ni muy estrechas, ni muy anchas). En cuanto a los textos se deben situar fuera de las columnas y las cifras se pueden colocar fuera o dentro, según sea más conveniente desde el punto de vista estético. Sobre los colores, deben usarse tantos como categorías de barras, y si el gráfico es en blanco y negro, las barras deben estar identificadas con toda claridad. El empleo de relleno no es útil en estos casos.

Otro de los aspectos esenciales en la utilización de diapositivas como medio audiovisual, es la retención del espectador durante la presentación. Se sabe que la capacidad máxima de retención de la atención de un espectador adulto, con coeficiente de inteligencia normal y escolaridad promedio de noveno grado, ante una diapositiva de calidad, no es mayor de 30 segundo. La curva de atención comienza a decrecer pasado ese tiempo, que es cuando el espectador empieza a buscar deficiencias en la presentación o en el contenido de la imagen. Si una buena diapositiva puede retener la atención del espectador durante ese lapso de tiempo, una mala diapositiva es posible que no lo logre por más de tres o cuatro segundos; o lo que es peor, que distraiga en lugar de fijar la atención, ello tiene una relación directa con la calidad de la diapositiva.

6.1.6. Ventajas y Desventajas[3]

- Las diapositivas permiten ser proyectadas en grandes pantallas, todos pueden verlas claramente.
- Los esquemas, demás elementos audiovisuales, como en el caso de diapositivas informatizadas, llaman la atención de los alumnos, son motivantes.
- Son ideales para dar clase a grandes grupos.
- Para la proyección de las diapositivas la sala debe estar sin luz, siendo esto un inconveniente pero, en cambio para las diapositivas informatizadas no hace falta, y se pueden tomar apuntes…

[3] María José Ballesteros, Gloria María Gallego. La Diapositiva [En línea], Ventajas y Desventajas, [http://www.uclm.es/profesorado/ricardo/Diapositivas/Diapositivas_Fca_2004.doc], [Consulta: 20 de enero del 2010]

- Se puede utilizar con cualquier tema o nivel educativo.
- Se pueden facilitar copias en papel, para las diapositivas informatizadas.
- El profesor puede mantenerse cara a los alumnos a la vez que explica, esto mejora la comunicación
- Ayuda al profesor o al exponente recordando mediante los esquemas o fotografías.
- La elaboración resulta fácil y el control es sencillo, quizá se puede presentar alguna dificultad en las diapositivas informatizadas.

6.1.7. Aprendizaje mediante el uso de diapositivas.

En el proceso de enseñanza-aprendizaje es de vital importancia la utilización de herramienta, técnicas y métodos que permitan que se realice de la mejor manera para de esta forma lograr una comunicación bidireccional mucho más efectiva entre los protagonistas. El uso de diapositivas para la enseñanza se encuentra entre las herramientas multimedia ya que utiliza diferentes tipos de datos dentro de una misma presentación como puede ser texto, audio, animaciones, videos, imágenes tanto fijas como en movimiento.

El empleo de diapositivas en la enseñanza favorece en gran escala al docente ya que mezcla diferentes tecnologías de difusión de la información, impactando varios sentidos a la vez para lograr un efecto mayor en la comprensión del mensaje. Hoy en día la utilización de este tipo de recurso es muy común y a esto le favorece la aparición del computador y software diseñado con este fin lo cual facilita aún más su uso.

El computador es considerado como una de las mejores herramientas multimedia ya que este posee los componentes esenciales para la elaboración de presentaciones, el óptimo uso de la misma depende de la creatividad y destreza que posea el docente.

Entre los usos que se da a este tipo de herramienta multimedia esta la creación de presentaciones de diapositivas mediante software dedicado a este propósito, para el desarrollo de estas presentaciones es necesario conocer el uso de estos programa que tiene muchas virtudes, entre las cuales se encuentran: edición de textos, colocación de imágenes fijas y en movimiento, edición de dibujos libres, colocación de música y todo

tipo de sonido así como también colocación de videos tipo películas de cine, colocación de fondos de pantalla especiales, colocación de efectos especiales de sonido y movimiento, etc., las que pueden ser colocadas en cada diapositiva, a conveniencia de la persona que las edita.

Para presentar los archivos diseñados, para pequeños o grandes auditorios, se necesita contar con los siguientes elementos:

- Proyector multimedia, que permite ver las imágenes en pantalla gigante.
- Parlantes, que permite escuchar en alto volumen la música y otros sonidos.
- Equipo de sonido y micrófonos que permite escuchar la voz del expositor en altoparlantes para que pueda ser escuchado por todo el auditorio, cuando se trata del dictado de conferencias.

CAPITULO 2: JAVA 3D[4]

[4] SUN, 2010.[en línea] Api Java 3D,[http://java.sun.com/developer/onlineTraining/java3d/],[Consulta:12 Febrero 2010]

6.2. Que es Java3D

Java3D es una interfaz de programación utilizada para realizar aplicaciones y applets con gráficos en tres dimensiones. Proporciona a los desarrolladores un alto nivel para crear y manipular objetos geométrios 3D y para construir las estructuras utilizadas en el **renderizado** de dichos objetos. Se pueden describir grandes **mundos virtuales** utilizando estos constructores, que proporcionan a Java3D la suficiente información para hacer un renderizado de forma eficiente.

Los objetos geométricos creados por los constructores residen en un **universo virtual**, que luego es renderizado. El API está diseñado con flexibilidad para crear universos virtuales precisos de una amplia variedad de tamaños, desde astronómicos a subatómicos.

A pesar de toda esta funcionalidad, la API es sencilla de usar. Los detalles de renderizado se manejan automáticamente. Aprovechándose de los Threads, Java3D es capaz de renderizar en paralelo.

6.2.1. Características

El modelado de Java3D se basa en múltiples objetivos, siendo el principal el rendimiento. Se tomaron diferentes decisiones relativas al modelado de tal forma que las implementaciones de Java3D proporcionaran el mejor rendimiento posible a las aplicaciones de usuario. En particular, cuando se realizan distribuciones, se elige la alternativa que permite obtener mejores prestaciones en tiempo de ejecución. Otros objetivos importantes de Java3D son:

- Proporcionar un amplio conjunto de utilidades que permitan crear mundos en 3D interesantes. Se evitó incluir características no esenciales o que se podrían colocar directamente sobre Java3D.
- Proporcionar un paradigma de programación orientado a objeto de alto nivel para permitir a los desarrolladores generar sofisticadas aplicaciones y applets de forma rápida.
- Proporcionar soporte a cargadores en tiempo de ejecución. Esto permite que Java3D se adapte a un gran número de formatos de ficheros, como pueden ser formatos específicos de distintos fabricantes de CAD, formatos de intercambio o VRML 1. y VRML 2.0.

Las aplicaciones en Java3D construyen los distintos elementos gráficos como objetos separados y los conectan unos con otros mediante una estructura en forma de árbol denominada **grafo de escena**. La aplicación manipula los diferentes objetos utilizando los métodos de acceso, de modificación y de unión definidos en su interfaz.

6.2.2. Geometrías en java 3D

Hay tres formas principales de crear contenidos geométricos. Una forma es usar las clases de utilidades geométricas para box, cone, cylinder, y sphere. Otra forma es especificar coordenadas de vértices para puntos, segmentos de líneas y/o superficies poligonales. Una tercera forma es usar un cargador geométrico.

6.2.3. Sistema de Coordenadas del Mundo Virtual

Un ejemplar de la clase VirtualUniverse sirve como raíz para el escenario gráfico de todos los programas Java 3D. El término 'Universo Virtual' comúnmente se refiere al espacio virtual de tres dimensiones que rellenan los objetos Java 3D. Cada objeto del universo virtual establece un sistema de coordenadas Cartesianas. Un objeto Locale sirve como punto de referencia para los objetos visuales en un universo virtual. Con un Locale en un SimpleUniverse, hay un sistema de coordenadas en el universo virtual.

El sistema de coordenadas del universo virtual Java 3D es de mano derecha. El eje X es positivo hacia la derecha, el eje Y es positivo hacia arriba y el eje Z es positivo hacia el espectador, con todas las unidades en metros.

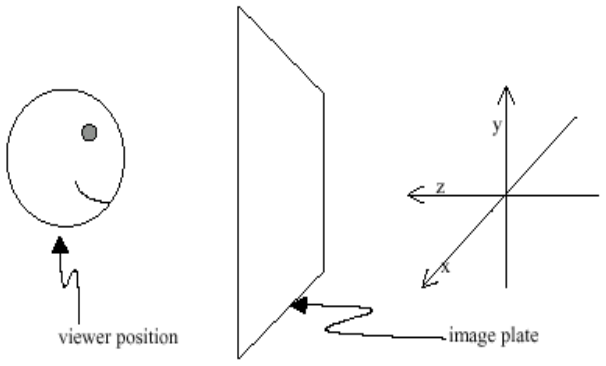


Figura 1 Orientación de Ejes en un Mundo Virtual

6.2.4. Definición Básica de Objeto Visual

Un ejemplar de Shape3D Define un Objeto Visual, un nodo Shape3D de escenario gráfico define un objeto visual. Shape3D es una de las subclases de la clase Leaf: por lo tanto, los objetos Shape3D sólo pueden ser hojas en un escenario gráfico. El objeto Shape3D no contiene información sobre la forma o el color de un objeto visual. Esta información está almacenada en los objetos NodeComponent referidos por el objeto Shape3D. Un objeto Shape3D puede referirse a un componente nodo Geometry y a un componente nodo Appearance.

En los escenarios gráficos de la página anterior, el símbolo de objeto genérico (rectángulo) fue utilizado para representar el objeto ColorCube. El sencillo escenario gráfico de la Figura 2 muestra un objeto visual representado como una hoja Shape3D (triángulo) y dos NodeComponents (óvalos) en lugar del rectángulo genérico.

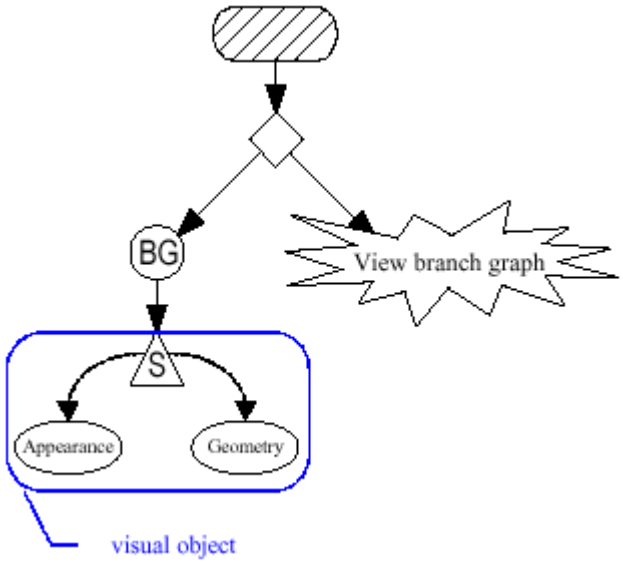


Figura 2 Un Objeto Shape3D Define un Objeto Visual en un Escenario Gráfico.

Un objeto visual se puede definir usando sólo un objeto Shape3D y un nodo componente Geometry. Opcionalmente, el objeto Shape3D también se refiere a un nodo componente Appearance. Los constructores de Shape3D muestran que se pueden crear sin referencias a componentes nodos, con sólo una referencia a un nodo Geometry, o con referencias a ambos tipos de componentes.

Mientras que el objeto Shape3D no esté vivo o compilado, las referencias a los componentes pueden modificarse con varios. Estos métodos pueden usarse sobre objetos Shape3D vivos o compilados si se configuran las capacidades del objeto primero.

6.2.5. NodeComponent

Los objetos NodeComponent contienen las especificaciones exactas de los atributos de un objeto visual. Cada una de las muchas subclases de NodeComponent define ciertos atributos visuales. La Figura 3 muestra una parte del árbol del API Java 3D que contiene las clases NodeComponent y sus descendientes.

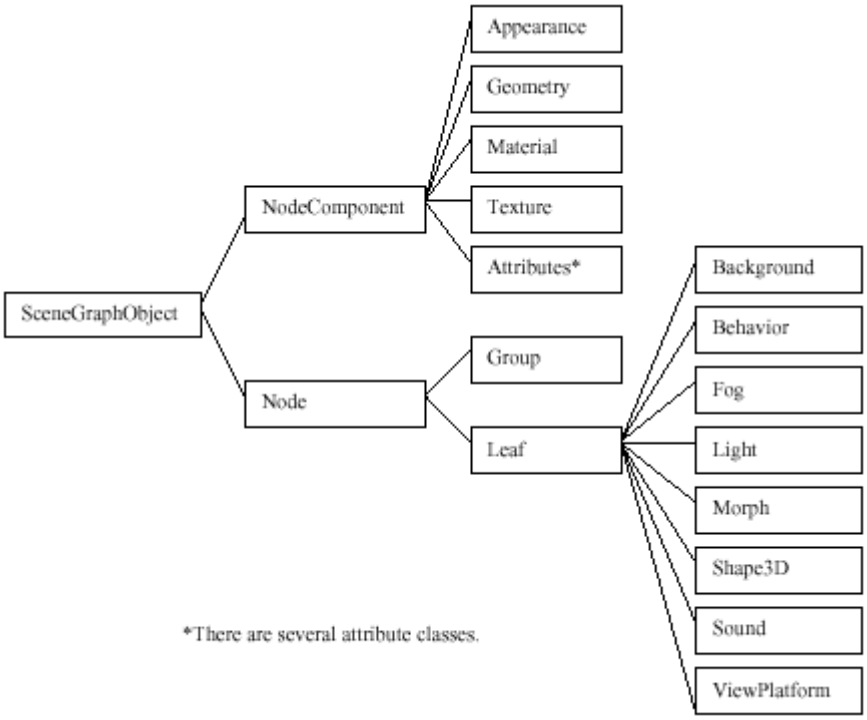


Figura 3 Clases de Java 3D Mostrando las Subclases de NodeComponent.

6.2.6. Clases de Utilidades Geométricas

Esta sección cubre las clases de utilidad para crear gráficos primitivos geométricos como cajas, conos, cilindros y esferas. Los primitivos geométricos son la segunda forma

más fácil para crear contenidos en un universo virtual. La más fácil es usar la clase ColorCube.

Las clases primitivas proporcionan al programador más flexibilidad que la clase ColorCube. Un objeto ColorCube define la geometría y el color en un componente Geometry. Consecuentemente, todo en el ColorCube es fijo, excepto su tamaño. El tamaño de un ColorCube sólo se especifica cuando se crea.

Un objeto primitivo proporciona más flexibilidad especificando la forma sin especificar el color. En una clase de utilidad geométrica primitiva, el programador no puede cambiar la geometría, pero puede cambiar la apariencia. Las clases primitivas le dan al programador la flexibilidad de tener varios ejemplares de la misma forma geométrica primitiva donde cada una tiene una apariencia diferente haciendo una referencia a un NodeComponent de apariencia diferente.

Las clases de utilidad Box, Cone, Cylinder y Sphere están definidas en el paquete com.sun.j3d.utils.geometry. En la Figura 4 podemos ver la porción del paquete com.sun.j3d.utils.geometry que contiene las clases primitivas.

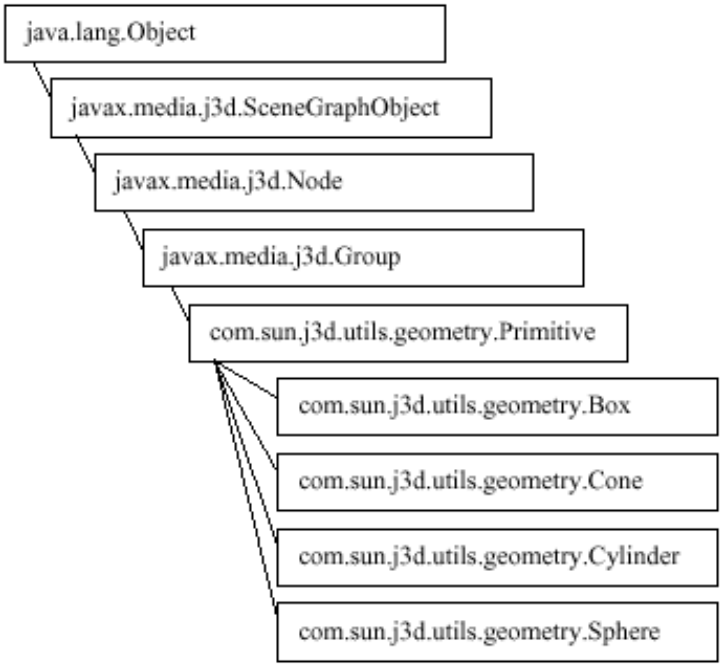


Figura 4 Árbol de Clases de las Clases Geométricas Primitivas

6.2.6.1. Box

La clase geométrica Box crea cubos de 3 dimensiones. Los valores por defecto para la longitud, anchura y altura son 2 metros, con el centro en el origen, resultando en un cubo con esquinas en (-1, -1, -1) y (1, 1, 1). La longitud, la anchura y la altura pueden especificarse en el momento de la creación del objeto.

6.2.6.2. Cone

La clase Cone define conos centrados en el origen y con el eje central alineado con el eje Y. Los valores por defecto para el radio y la altura son 1,0 y 2,0 metros respectivamente.

6.2.6.3. Cylinder

La clase Cylinder crea objetos cilíndricos con sus eje central alineado con el eje Y. Los valores por defecto para el radio y la altura son 1,0 y 2,0 metros respectivamente.

6.2.6.4. Sphere

La clase Sphere crea objetos visuales esféricos con el centro en el origen. El radio por defecto es de 1,0 metros.

6.2.7. Clases Matemáticas

Para crear objetos visuales, se necesitan la clase Geometry y sus subclases. Muchas de éstas subclases describen primitivos basados en vértices, como puntos, líneas y polígonos rellenos. A continuación veremos varias clases matemáticas (Point*, Color*, Vector*, TexCoord*) usadas para especificar datos relacionados con los vértices

Todas estas clases matemáticas están en el paquete javax.vecmath.*. Este paquete define varias clases Tuple* como superclases genéricas abstractas. Otras clases más útiles descienden de las clases Tuple. En la Figura 5, podemos ver algunas de las clases del árbol.

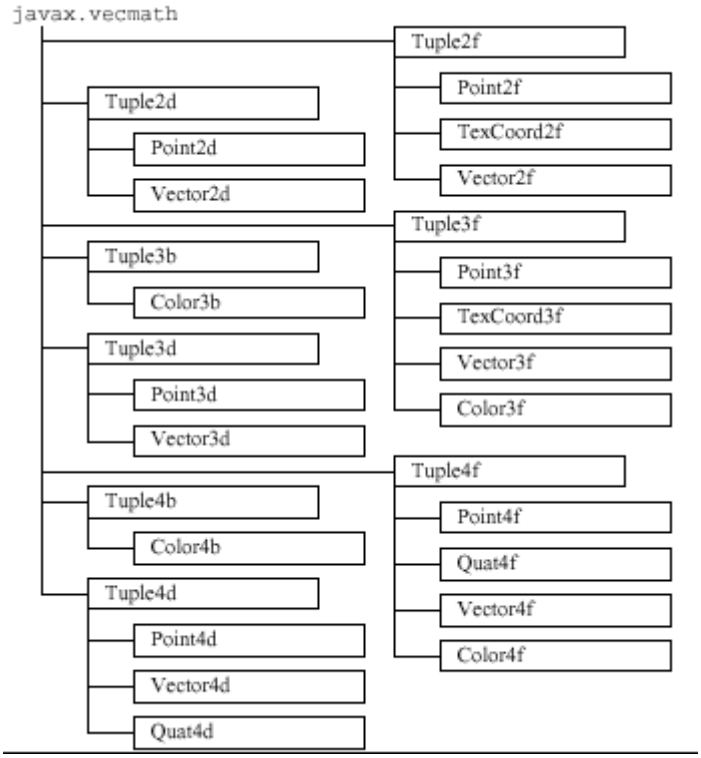


Figura 5 Árbol de clases del paquete de clases matemáticas.

Cada vértice de un objeto visual podría especificar hasta cuatro objetos javax.vecmath, representado coordenadas, colores, superficies normales, y coordenadas de textura. Normalmente se usan las siguientes clases:

- Point* (para coordenadas)
- Color* (para colores)
- Vector* (para superficies normales)
- TexCoord* (para coordenadas de textura)

Observa que las coordenadas (objetos Point*) son necesarios para posicionar cada vértice. Los otros datos son opcionales, dependiendo de cómo se renderice el primitivo. Por ejemplo, se podría definir un color (un objeto Color*) para cada vértice y los colores del primitive se interpolan entre los colores de los vértices. Si se permite la iluminación, serán necesarias las superficies normales (y por lo tanto los objetos Vector*). Si se permite el mapeo de texturas, podrían necesitarse las coordenadas de texturas.

6.2.7.1. Clases Point

Los objetos Point* normalmente representan coordenadas de un vértice, aunque también pueden representar la posición de una imagen, fuente de un punto de luz, localización espacial de un sonido, u otro dato posicional. Los constructores de las clases Point* son muy similares a los de Tuple*, excepto en que devuelven objetos Point*.

6.2.7.2. Clases Color

Los objetos Color* representan un color, que puede ser para un vértice, propiedad de un material, niebla, u otro objeto visual. Los colores se especifican con Color3* o Color4*, y sólo para datos de byte o coma flotante de simple precisión. Los objetos Color3* especifican un color como una combinación de valores rojo, verde y azul (RGB). Los objetos Color4* especifican un valor de transparencia, además del RGB. Para los tipos de datos de tamaño byte, los valores de colores van desde 0 hasta 255. Para tipos de datos de coma flotante de simple precisión, los valores van entre 0,0 y 1,0.

6.2.7.3. Clases Vector

Los objetos Vector* frecuentemente representan superficies normales en vértices aunque también pueden representar la dirección de una fuente de luz o de sonido. De nuevo, los constructores de las clases Vector* son similares a los de Tuple*. Sin embargo, los objetos Vector* añaden muchos métodos que no se encuentran en las clases Tuple*.

6.2.7.4. Clases TexCoord

Hay sólo dos clases TexCoord* que pueden usarse para representar las coordenadas de textura de un vértice: TexCoord2f y TexCoord3f. TexCoord2f mantiene las coordenadas de textura como una pareja de coordenadas (s, t); TexCoord3f como un trio (s, t, r).

Los constructores para las clases TexCoord* también son similares a los de Tuple* como las clases Color*, las clases TexCoord* tampoco tienen métodos adicionales, por eso sólo tratan con los métodos que heredan de sus superclases.

6.2.8. Clases Geometry

En los gráficos 3D por ordenador, todo, desde el más sencillo triángulo al más complicado de los modelos Jumbo, está modelado y renderizado con datos basados en

vértices. Con Java 3D, cada objeto Shape3D debería llamar a su método setGeometry () para referenciar uno y sólo uno objeto Geometry. Para ser más preciso, Geometry es una superclase abstracta, por eso los objetos referenciados son ejemplares de una de sus subclases.

Estas subclases se dividen en tres categorías principales:

- Geometría basada en vértices no indexados (cada vez que se renderiza un objeto visual, sus vértices sólo podrían usarse una vez)
- Geometría basada en vértices indexados (cada vez que se renderiza un objeto visual, sus vértices se podrían reutilizar)
- Otros objetos visuales (las clases Raster, Text3D, y CompressedGeometry)

Esta sección cubre las dos primeras categorías. El árbol de clases de Geometry y sus subclases se muestran en la Figura 6.

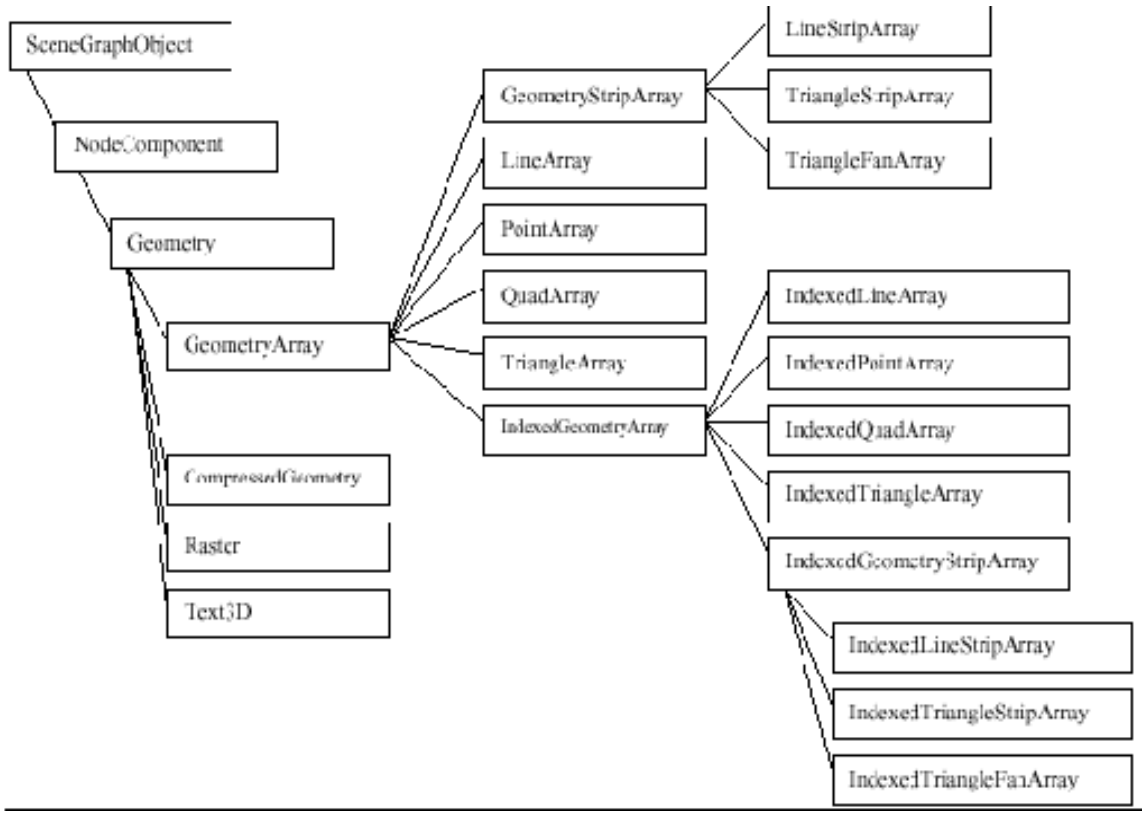


Figura 6 Árbol de clases de Geometry.

6.2.9. Clase GeometryArray

Como se podría deducir de los nombres de las clases, las subclases de Geometry podrían usarse para especificar puntos, líneas y polígonos rellenos (triángulos y cuadriláteros). Estos primitivos basados en vértices son subclases de la clase abstracta GeometryArray, que indica que cada una tiene un array que mantiene los datos de los vértices.

Por ejemplo, si se usa un objeto GeometryArray para especificar un triángulo, se define un array de tres elementos: un elemento para cada vértice. Además de la localización de las coordenadas, se pueden definir otros tres arrays opcionales para almacenar el color, la superficie normal, y las coordenadas de textura. Estos arrays que contienen las coordenadas, los colores, las superficies normales y las coordenadas de texturas, son los "arrays de datos".

Hay tres pasos en la vida de un objeto **GeometryArray**:

Paso 1: Construcción de un objeto GeometryArray vacío.

Cuando se construye por primera vez un objeto GeometryArray, se deben definir dos cosas:

- El número de vértices (arrays de elementos) necesarios.
- El tipo de datos (coordenadas de localización, color, superficie normal, y/o coordenadas de textura) a almacenar en cada vértice. Esto se llama formato del vértice.

Paso 2: Rellenar con Datos el Objeto GeometryArray.

Después de construir el objeto GeometryArray, asignamos valores a los arrays, correspondiendo a los formatos de los vértices asignados. Esto se podría hacer vértice por vértice, o usando un array para asignar datos a muchos vértices con una sola llamada del método.

Paso 3: Hacer que los Objetos Shape3D Referencien a los Objetos GeometryArray.

Los objetos Shape3D se añaden a un BranchGroup, que en algún lugar se añade al escenario gráfico general. (Al contrario que los objetos GeometryArray, que son

NodeComponents, Shape3D es una subclase de Node, por eso pueden ser añadidos como hijos al escenario gráfico.)

6.2.9.1. Subclases de GeometryArray

La clase GeometryArray es una superclase abstracta para subclases más útiles, como LineArray. La Figura 7 muestra el árbol de subclases de GeometryArray. La principal distinción entre estas subclases es cómo el renderizador Java 3D decide renderizar sus vértices.

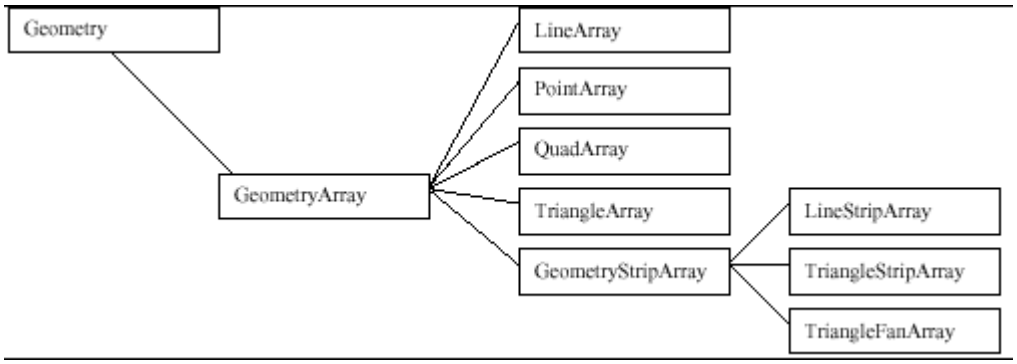


Figura 7 Subclases no indexadas de GeometryArray.

La Figura 8 muestra ejemplos de las cuatro subclases de GeometryArray: PointArray, LineArray, TriangleArray, y QuadArray. En esta figura, los tres conjuntos más a la izquierda muestran los mismos seis vértices renderizando seis puntos, tres líneas, o dos triángulos. La cuarta imagen muestra cuatro vértices definiendo un cuadrilátero.

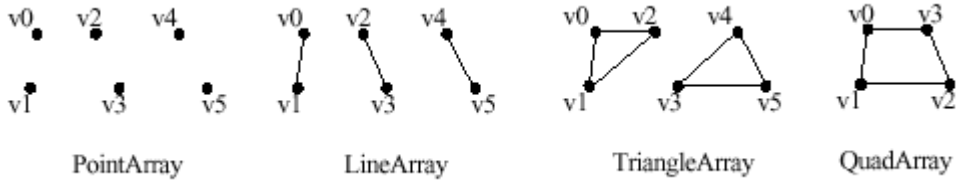


Figura 8 Subclases de GeometryArray.

Por defecto, se rellena el interior de los triángulos y cuadriláteros. En la última sección, aprenderemos los atributos que pueden influenciar en el modo de renderizado de los primitivos rellenos.

6.2.10. Contenidos sencillos en java 3D

6.2.10.1. Cargadores

Una clase Loader lee ficheros de escenas 3D (no ficheros Java 3D) y crea representaciones Java 3D de sus contenidos que pueden ser añadidos selectivamente a un mundo Java 3D y argumentados por otro código Java 3D. El paquete com.sun.j3d.loaders proporciona el contenido principal para convertir los ficheros creados en otras aplicaciones en aplicaciones Java 3D. Las clases cargadoras implementan el interface Loader definido en el paquete com.sun.j3d.loaders.

Como hay una gran variedad de formatos de ficheros para propósitos de representación de escenas 3D (por ejemplo, .obj, .vrml, etc.) y siempre habrá más formatos de ficheros, el código real para cargar un fichero no forma parte de Java 3D o del paquete loaders; sólo se incluye el interface para el mecanismo de carga. Con la definición del interface, el usuario de Java 3D puede desarrollar clases cargadoras de ficheros con el mismo interface que las otras clases cargadoras.

6.2.10.2. Cargadores Disponibles Públicamente

En Java 3D existen varias clases cargadoras. A continuación se listan algunos ejemplos:

Formato de Fichero	Descripción
3DS	3D-Studio
DXF	AutoCAD Drawing Interchange File
OBJ	Wavefront
SLD	Solid Works (prt and asm files)
WRL	Virtual Reality Modeling Language

Tabla 1 Cargadores Java 3D disponibles Públicamente

Esta gran variedad de cargadores existe para hacer más sencilla la escritura de cargadores para los diseñadores Java 3D. Las clases Loader son implementaciones del interface Loader que baja el nivel de dificultad para escribir un cargador. Como en el ejemplo, un programa que carga un fichero 3D realmente usa un cargador y un objeto escena. El cargador lee, analiza y crea la representación Java 3D de los contenidos del fichero. El objeto escena almacena el escenario gráfico creado por el cargador. Es posible cargar escenas desde más de un fichero (del mismo formato) usando el mismo

objeto cargador y crear múltiples objetos escena. Los ficheros de diferentes formatos pueden combinarse en un programa Java 3D usando las clases cargadoras apropiadas.

La clase LoaderBase proporciona una implementación para cada uno de los tres métodos load () del interface Loader. LoaderBase también implementa dos constructores.

6.2.11. Texto 3D

La forma de añadir texto a un mundo virtual Java 3D es crear un objeto Text3D para texto. El Text3D crea texto usando geometría. La geometría textual de un objeto Text3D es una extrusión de la fuente.

Crear un objeto Text3D es un poco más complicado que crear un objeto Text2D. El primer paso es crear un objeto Font3D con el tipo de fuente, el tamaño y el estilo seleccionado. Luego se crea un objeto Text3D para una cadena particular usando el objeto Font3D. Como la clase Text3D es una subclase de Geometry, el objeto Text3D es un NodeComponent que es referenciado por uno o más objetos Shape3D:

Figura 9 Punto de Referencia por Defecto y Extrusión de un Objeto 3DText

El texto de un objeto Text3D puede orientarse de una gran cantidad de formas. La orientación se especifica como el camino de dirección. Las direcciones son right, left, up, y down.

Cada objeto Text3D tiene un punto de referencia. El punto de referencia para un objeto Text3D es el origen del objeto. El punto de referencia de cada objeto se define por la combinación del camino y la alineación del texto. La Tabla 2 muestra los efectos de las especificaciones del camino y la alineación sobre la orientación del texto y la situación del punto de referencia.

	ALIGN_FIRST (Default)	ALIGN_CENTER	ALIGN_LAST
PATH_RIGHT (Default)	Text3D	Text3D	Text3D
PATH_LEFT	D3txeT	D3txeT	D3txeT
PATH_DOWN	T e x t 3 D	T e x t 3 D	T e x t 3 D
PATH_UP	D 3 t x e T	D 3 t x e T	D 3 t x e T

Tabla 2 Alineación y Orientación del texto 3D

Los objetos Text3D tienen superficies. La adicción de un paquete de apariencia que incluye un objeto Material a un objeto Shape3D referenciando la geometría Text3D permitirá la iluminación del objeto Text3D.

6.2.12. Clases Usadas en la Creación de Objetos Text3D

Esta sección presenta el material de referencia para tres clases usadas en la creación de objetos Text3D: Text3D, Font3D, y FontExtrusion, en este orden. La Figura 10 muestra el árbol de clases de Text3D.

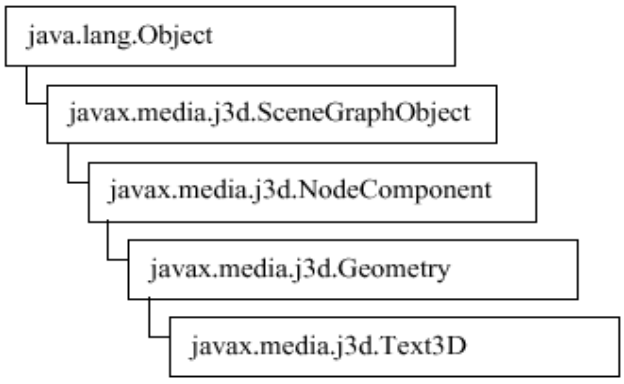


Figura 10 Árbol de Clases de Text3D

Cada objeto Text3D se crea desde un objeto Font3D. Un sólo objeto Font3D puede usarse para crear un número ilimitado de objetos Text3D. Un objeto Font3D contiene la extrusión geométrica de cada carácter en el tipo de letra. Un objeto Text3D copia las geometrías para formar la cadena especificada. Los objetos Font3D pueden ser recolectados por el recolector de basura sin afectar a los objetos Text3D creados a partir de él.

6.2.13. Fondo

Por defecto, el fondo de un universo virtual Java 3D es negro sólido. Sin embargo, podemos especificar otros fondos para nuestros mundos virtuales. El API Java 3D proporciona una forma fácil de especificar un color sólido, una imagen, una geometría o una combinación de éstos como fondo.

Cuando especificamos una imagen para el fondo, se sobrescribe la especificación del color de fondo, si existe. Cuando se especifica una geometría, se dibuja sobre el color de fondo o la imagen.

La única parte espinosa es la especificación de un fondo geométrico. Toda la geometría de fondo se especifica como puntos en una esfera. Si nuestra geometría es un PointArray, que podría representar estrellas a años luz, o un TriangleArray, que podría representar montañas en la distancia. La geometría de fondo se proyecta sobre el infinito cuando se renderiza.

Los objetos Background tienen límites de aplicación, lo que nos permite que se puedan especificar diferentes fondos para diferentes regiones del mundo virtual. Un nodo Background está activo cuando su región de aplicación se interseca con el volumen de activación del ViewPlatform.

Si están activos varios nodos Background, el nodo que está más "cercano" al ojo será el utilizado. Si no hay ningún nodo Background activo, la ventana se mostrará en negro. Sin embargo, la definición de "más cercano" no está especificada. Por cercano, se elige el fondo con los límites de aplicación más internos que encierra la ViewPlatform.

Es improbable que nuestra aplicación necesite iluminar la geometría del fondo en realidad el sistema visual humano no puede percibir los detalles visuales a grandes distancias. Sin embargo, una geometría de fondo si puede ser sombreada. La geometría del fondo podría no contener luces, pero las luces definidas en el escenario gráfico pueden influenciar en la geometría del fondo.

6.2.13.1. La clase Background

La Figura 11 muestra el árbol de clase de la clase Background. Como una extensión de la clase Leaf, un ejemplar de la clase Background puede ser un hijo de un objeto Group.

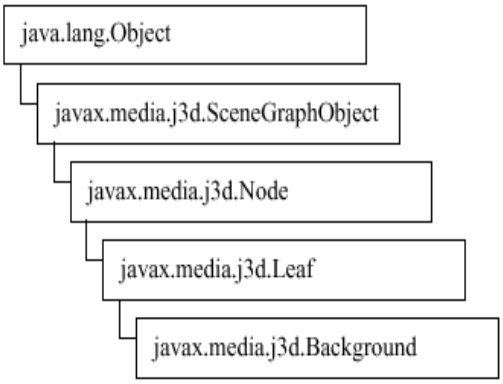


Figura 11 Árbol de Clases de Background.

6.2.14. BoundingLeaf

Los Bounds (límites) se usan con luces, comportamientos, fondos y una gran variedad de otras aplicaciones en Java 3D. Los Bounds permiten al programador variar la acción, la apariencia, y/o el sonido sobre el campo virtual. La especificación de Bounds también permite al sistema de renderizado de Java 3D mejorar la ejecución del recortado y por lo tanto mejorar el rendimiento.

La especificación típica de límites utiliza un objeto Bounds para limitar una región. En el escenario gráfico resultante, los objetos Bounds se mueven con los objetos que lo referencian. Esto está bien para muchas aplicaciones; sin embargo, podría haber situaciones en las que fuera deseable tener la región límite que se moviera independientemente de los objetos que usan los límites.

Por ejemplo, si un mundo incluye una fuente de luz estacionaria que ilumina unos objetos en movimiento, los límites de la luz deberían incluir el objeto en movimiento. Una forma de manejar esto podría ser crear los límites lo suficientemente grandes como para incluir todos los lugares donde se mueve el objeto. Esta no es la mejor respuesta en muchos casos. Una mejor solución es usar un BoundingLeaf. Situado en el escenario gráfico con el objeto visual, el BoundingLeaf se mueve con el objeto visual independientemente de la fuente de luz. La Figura 12 muestra un escenario gráfico con un una luz que usa un nodo BoundingLeaf.

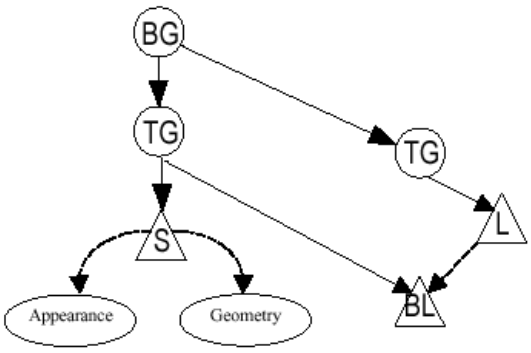


Figura 12 Movimiento de un objeto BoundlingLeaf

6.2.15. Interacción en java 3D

La interacción y la animación se especifican con objetos Behavior. La clase Behavior es una subclase abstracta que proporciona el mecanismo para incluir código que modifique el escenario gráfico. La clase Behavior, y sus descendientes, son enlaces a código del usuario que proporciona las modificaciones para los gráficos y los sonidos del universo virtual.

El propósito del objeto Behavior en un escenario gráfico es modificar el propio escenario gráfico, o los objetos que hay dentro de él, en respuesta a algunos estímulos. Un estímulo puede ser una pulsación de tecla, un movimiento del ratón, la colisión de objetos, el paso del tiempo, algún otro evento, o una combinación de estos. Los cambios producidos incluyen la adicción de objetos al escenario gráfico, la eliminación de objetos, cambio de atributos de los objetos del escenario gráfico, reordenación de los objetos del escenario gráfico, o una combinación de estos. Las posibilidades sólo están limitadas por las capacidades de los objetos del escenario gráfico.

6.2.15.1. Animación contra Interacción

Si un usuario navega en un programa donde se proporciona un comportamiento, la vista se moverá en respuesta a eventos del teclado y/o ratón. El movimiento de la plataforma de la vista es una interacción porque es el resultado directo de una acción del usuario. Sin embargo, otras cosas podrían cambiar como resultado del movimiento de la plataforma de la vista, (por ejemplo, comportamientos de cartelera o LOD). Los cambios causados como resultado del movimiento de la plataforma de vista son indirectamente causados por el usuario y por lo tanto son animaciones.

6.2.16. Behavior Básico

Las clases Behavior se usan en muchas aplicaciones Java 3D y de muchas formas. Es importante entender las consideraciones de funcionamiento y programación de estas clases. Esta sección explica la clase Behavior, ofrece una receta para programar clases de comportamientos personalizadas, y muestra una aplicación de ejemplo que usa una clase Behavior.

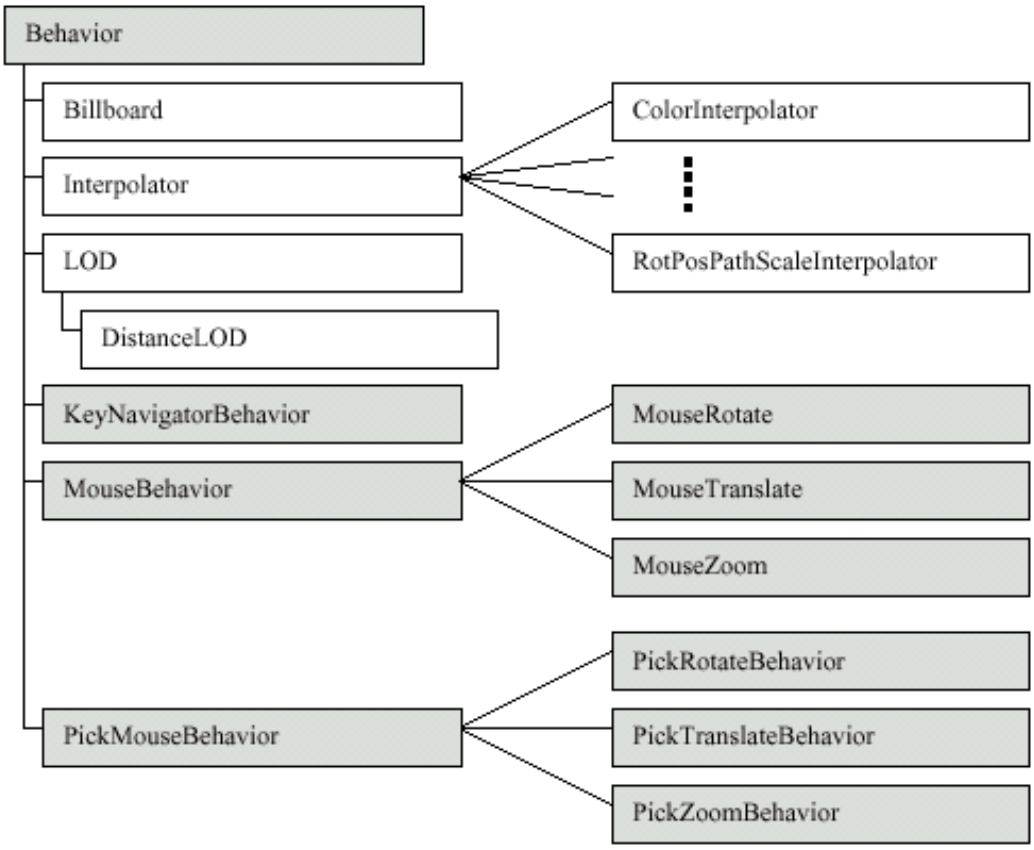


Figura 13 Árbol de Subclases de Behavior

6.2.16.1. Mecanismo de Behaviors

Una clase de comportamiento personalizado implementa los métodos de inicialización y processStimulus de la clase abstracta Behavior. Por supuesto, la clase de comportamiento personalizado, también tiene al menos un constructor y también podría tener otros métodos.

La mayoría de los comportamientos actuarán sobre un objeto del escenario gráfico para afectar al comportamiento. El objeto sobre el que actúa un comportamiento es referido como el objeto del cambio. Es a través de este objeto, u objetos, que el comportamiento afecta al mundo virtual. Aunque es posible tener un comportamiento que no tenga un objeto del cambio, la mayoría lo tienen.

El comportamiento necesita una referencia a su objeto(s) de cambio para poder realizar los cambios de comportamiento. Se puede usar el constructor para seleccionar la referencia del objeto de cambio. Si no se hace, otro método de la clase de comportamiento personalizado debe almacenar esta información. En cualquier caso, la referencia se hace en el momento en que se construye el escenario gráfico, que es el primer cálculo de comportamiento.

El método de inicialización se invoca cuando el escenario gráfico que contiene la clase de comportamiento se vuelve vivo. Este método de iniciación es responsable de seleccionar el evento de disparo inicial para el comportamiento y seleccionar la condición inicial de las variables de estado del comportamiento. El disparo se especifica como un objeto WakeupCondition, o una combinación de objetos WakeupCondition.

El método processStimulus se invoca cuando ocurre el evento de disparo especificado para el comportamiento. Este método es responsable de responder al evento. Como se pueden codificar muchos eventos en un sólo objeto WakeupCondition (por ejemplo, varias acciones de teclado podrían estár codificados en un WakeupOnAWTEvent), esto incluye la descodificación del evento. El método processStimulus responde al estímulo, normalmente modificando el objeto de cambio, y, cuando es apropiado, reseteando el disparo.

6.2.16.2. Disparo de los Comportamientos

Los comportamientos activados se disparan por la ocurrencia de uno o más estímulos especificados. El estímulo de disparo para un comportamiento se especifica usando descendientes de la clase WakeupCondition.

La clase abstracta, WakeupCondition, es la base para todas las clases de disparo del API Java 3D. Cinco clases extienden WakeupCondition, una es la clase abstracta WakeupCriterion, las otras cuatro permiten la composición de múltiples condiciones de disparo en una única condición de disparo. La Figura 14 muestra el árbol de clases.

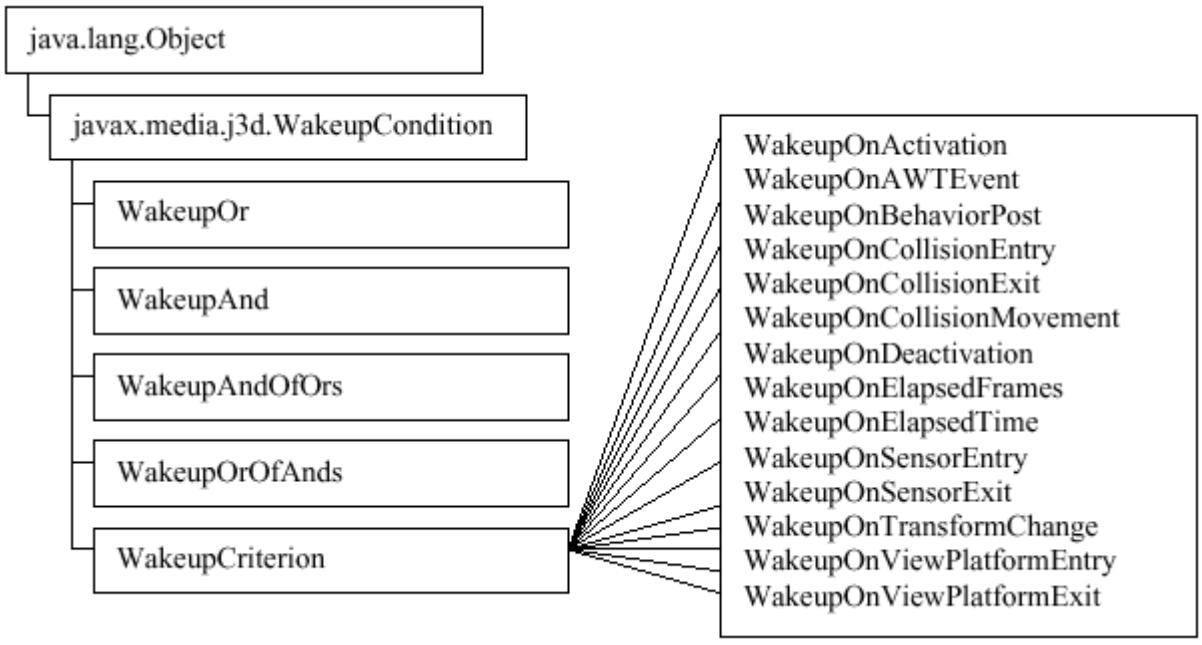


Figura 14 Árbol de Clases para WakeupCondition y clases relacionadas.

6.2.16.3. Clases WakeupCriterion Específicas

La Tabla 4 presenta las 14 clases WakeupCriterion específicas. Estas clases se usan para especificar las condiciones de disparo de los objetos behavior. Los ejemplares de estas clases se usan individualmente o en combinaciones.

Clase Criterio	Disparo
WakeupOnActivation	En la primera detección de una intersección del volumen de activación de un ViewPlatform con la región límite del objeto.
WakeupOnAWTEvent	Cuando ocurre un evento AWT específico.
WakeupOnBehaviorPost	Cuando un objeto behavior envía un evento específico.
WakeupOnCollisionEntry	En la primera detección de colisión del objeto especificado con otro objeto del escenario gráfico.
WakeupOnCollisionExit	Cuando el objeto especifico no colisiona con ningún otro objeto del escenario gráfico.
WakeupOnCollisionMovement	Cuando el objeto especificado se mueve mientras colisiona con otro objeto del escenario gráfico.
WakeupOnDeactivation	Cuando el volumen de activación de un ViewPlatform deja de interactuar con los límites del objeto.
WakeupOnElapsedFrames	Cuando ha pasado un número determinado de frames.
WakeupOnElapsedTime	Cuando ha pasado un número de segundos determinado.
WakeupOnSensorEntry	En la primera detección de cualquier sensor que intersecciones con los límites especificados.
WakeupOnSensorExit	Cuando un sensor que intersecciona va con los límites del objeto deja de interseccionar con los límites especificados.
WakeupOnTransformChange	Cuando cambia la transformación dentro de un TransformGroup especificado.
WakeupOnViewPlatformEntry	En la primera detección de intersección del volumen de activación de un ViewPlatform con los límites especificados.
WakeupOnViewPlatformExit	Cuando el volumen de activación de una vista deja de interseccionar con los límites especificados.

Tabla 3 Clases WakeupCriterion específicas.

6.2.16.4. Clases de Comportamientos Útiles para la Navegación por Teclado

Hasta este momento, el espectador ha estado en una localización fija con una orientación fija. El poder mover el espectador es una capacidad importante en muchas aplicaciones de los gráficos 3D. Java 3D es capaz de mover el espectador.

La Figura 15 muestra la rama gráfica básica para un universo virtual de Java 3D. En esta figura, se considera la plataforma de la visión transform. Si se cambia la transformación, el efecto es mover, o reorientar, o ambas, al espectador. De esto, podemos ver que el diseño básico de la navegación del teclado es simple: hacemos que un objeto behavior cambie la transformación de la vista de la plataforma en respuesta a los movimientos dominantes.

Este diseño simple es exactamente el modo de trabajo de las clases utilitarias del teclado de Java 3D.

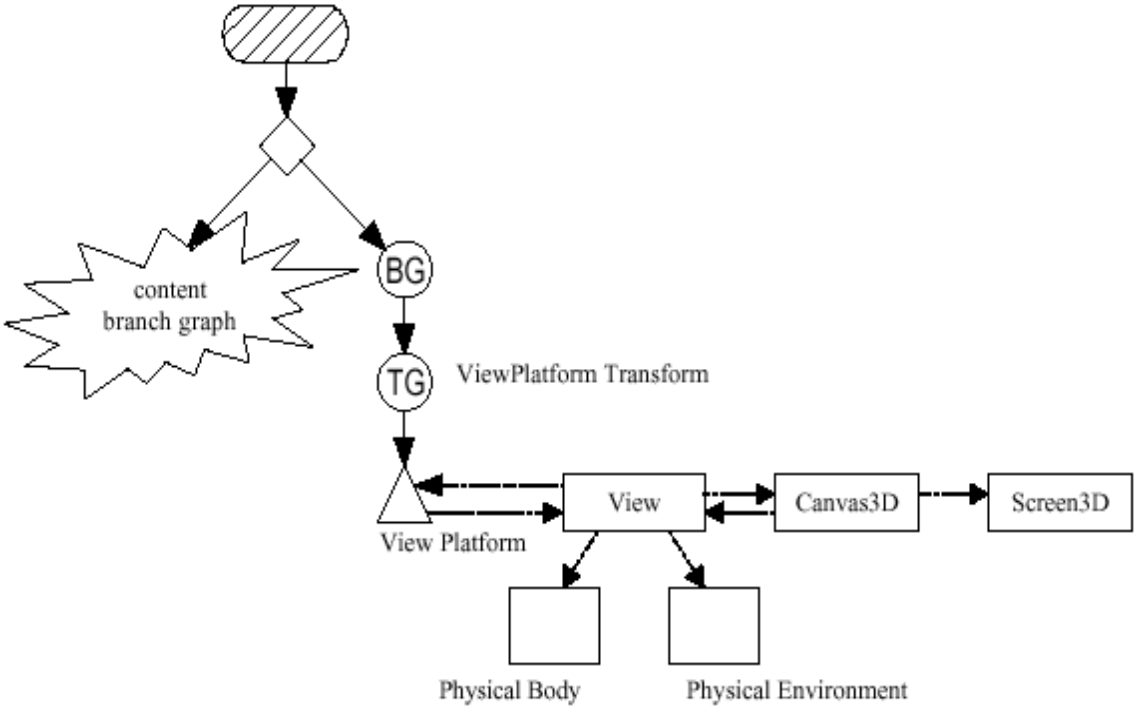


Figura 15 Rama gráfica básica para un universo virtual de Java 3D

6.2.16.5. Clases KeyNavigatorBehavior y KeyNavigator

La utilidad de navegación del teclado se implementa como dos clases. En el tiempo de ejecución hay dos objetos. El primer objeto es el objeto KeyNavigatorBehavior, el segundo es un objeto KeyNavigator. El objeto KeyNavigatorBehavior realiza todas las funciones típicas de una clase de comportamiento, excepto que llama al objeto KeyNavigator para realizar la función de processStimulus. La clase KeyNavigator toma el AWTEvent y lo procesa bajo al nivel de pulsaciones de teclas individuales. La Tabla siguiente muestra el efecto de las pulsaciones de teclas individuales. KeyNavigator implementa el movimiento con aceleración.

Tecla	Movimiento	Movimiento Alt-tecla
<-	rotar a la izquierda	traslación lateral izquierda
->	rotar a la derecha	traslación lateral derecha
^	mover hacia adelante	
V	mover hacia atrás	
PgUp	rotar arriba	translación hacia arriba
PgDn	rotar abajo	translación hacia abajo
+	aumenta la distancia de salto (y vuelve al origen)	
-	reduce la distancia de salto	
=	vuelve al centro del universo	

Tabla 4 Movimientos de KeyNavigatorBehavior

6.2.16.6. Clases de Utilidad para Interactuar con el Ratón

El paquete de comportamientos de ratón (com.sun.j3d.utils.behaviors.mouse) contiene las clases del comportamiento en las cuales el ratón se utiliza como entrada de información para la interacción con los objetos visuales. Incluyendo las clases para traslaciones (moviéndose en un plano paralelo a la placa de la imagen), enfocando (que mueve hacia atrás y adelante), y los objetos visuales que rotan en respuesta a los movimientos del ratón.

La siguiente tabla resume las tres clases específicas del comportamiento del ratón incluidas en el paquete. Además de estas tres clases, está la clase abstracta MouseBehavior, y el interface MouseCallback. Esta clase abstracta y el interface se utilizan en la creación de las clases específicas del comportamiento del ratón y son útiles para crear comportamientos personalizados del ratón.

Clase MouseBehavior	Acción en Respuesta a la Acción del Ratón	Acción del ratón
MouseRotate	Rota el objeto visual sin moverlo.	Botón izquierdo pulsado con movimiento del ratón.
MouseTranslate	Traslada el objeto visual en un plano paralelo al plato de imagen.	Botón derecho pulsado con movimiento del ratón.
MouseZoom	Traslada el objeto visual en un plano orthogonal al plato de imagen.	Botón central pulsado con movimiento del ratón.

Tabla 5 Sumario de las clases específicas de MouseBehavior

6.2.16.7. MouseNavigation

Las tres clases específicas de comportamiento del ratón se pueden utilizar para crear un universo virtual en el cual el ratón se utilice para la navegación. Cada una de las clases específicas del comportamiento del ratón tiene un constructor que toma un solo parámetro entero para las banderas. Cuando se utiliza MouseBehavior.INVERT_INPUTS como argumento a este constructor, el comportamiento del ratón responde en la dirección opuesta. Este comportamiento inverso es apropiado para cambiar el transform ViewPlatform. Es decir las clases del comportamiento del ratón se pueden utilizar para el control de navegación.

6.2.16.8. Picking y sus clases de Utilidad

Picking (Elección) le da al usuario una forma de interactuar con objetos visuales individuales en la escena.

Picking está implementado por un comportamiento típicamente disparado por eventos de botones del ratón. En la selección de un objeto visual, el usuario sitúa el puntero del

ratón sobre el objeto elegido y pulsa el botón del ratón. El objeto behavior se dispara por la pulsación de este botón y empieza la operación de selección. Se proyecta un rayo dentro del mundo virtual desde la posición del puntero del ratón paralela con la proyección. Se calcula la intersección de este rayo con los objetos del mundo virtual. El objeto visual que intersecciona más cerca al plato de la imagen se selecciona para interacción. La Figura 16 muestra un rayo de selección proyectado en un mundo virtual.

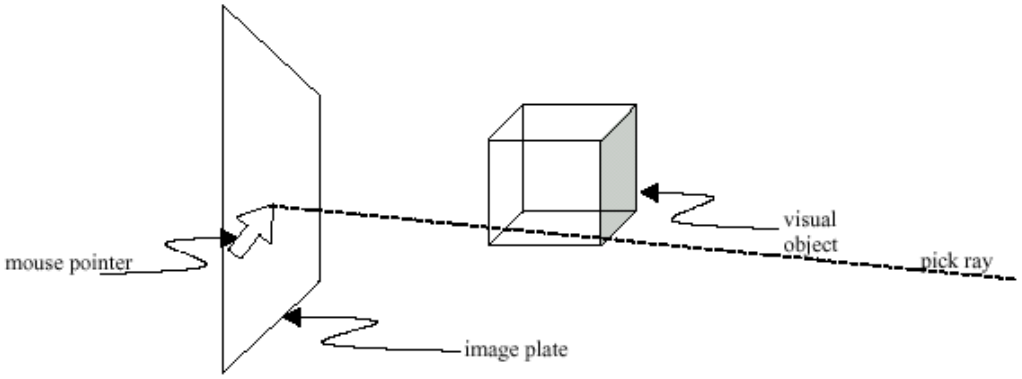


Figura 16 Proyección de un PickRay en el Mundo Virtual

En algunos casos la interacción no se hace directamente con el objeto seleccionado, sino con un objeto en el camino del escenario gráfico hasta el objeto. La determinación del objeto para un posterior procesamiento no siempre es sencilla. Si un objeto visual cúbico que va a ser rotado está compuesto por seis objetos Shape3D individuales junto con seis objetos TransformGroup, como en el escenario gráfico de la Figura 17, no es el objeto TransformGroup sobre el objeto Shape3D interseccionado el que necesita ser modificado. El 'cubo' se rota por la manipulación del objeto TransformGroup que es hijo del objeto BranchGroup en el escenario gráfico. Por esta razón, el resultado de algunas operaciones de selección es devolver el path del escenario gráfico para su posterior procesamiento.

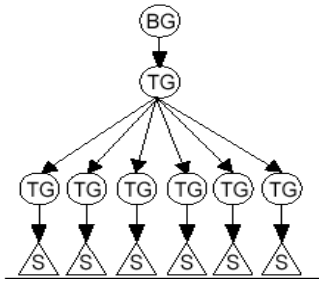


Figura 17 Diagrama del Escenario Gráfico del Cubo de Shape3D planos.

La comprobación de intersecciones es necesita mucho cálculo. Por lo tanto, la selección es cara y se vuelve más cara con la complejidad de la escena.

Hay dos aproximaciones básicas para usar las características de selección de Java 3D, usar objetos de clases picking, o crear clases picking personalizadas y usar ejemplares de estas clases. El paquete picking incluye clases para pick/rotate, pick/translate, y pick/zoom.

Cómo un objeto comportamiento picking operará sobre cualquier objeto del escenario gráfico (con las capacidades apropiadas), sólo se necesita proporcionar un objeto picking.

PickPoint

Los objetos PickPoint representan un punto para selección. Como una subclase de PickShape, los objetos PickBounds se usan con BranchGroup y Locale así como con clases del paquete picking.

PickRay

Los objetos PickRay representan un rayo (un punto y una dirección) para selección. Como una subclase de PickShape, los objetos PickBounds se usan con BranchGroup y Locale así como con clases del paquete picking.

PickSegment

Los objetos PickSegment representan un segmento de línea (definida por dos puntos) para selección. Como una subclase de PickShape, los objetos PickBounds se usan con BranchGroup y Locale así como con clases del paquete picking.

SceneGraphPath

La clase SceneGraphPath se usa en la mayoría de las aplicaciones de selección. Esto es porque normalmente la selección implica encontrar un camino de escenario gráfico en el que se encuentra un objeto para permite la manipulación del objeto o de un objeto TransformGroup en el path.

Un objeto SceneGraphPath representa el camino del escenario gráfico hacia el objeto elegido permitiendo la manipulación del objeto o de un objeto TransformGroup en el camino del objeto.

6.2.16.9. Clases Generales del Paquete Picking

Incluidas en el paquete com.sun.j3d.utils.behaviors.picking hay varias clases de comportamientos generales y específicos. Las clases generales son útiles para crear nuevos comportamientos de selección, entre las que se incluyen PickMouseBehavior, PickObject, y PickCallback.

Clase PickMouseBehavior

Esta es la clase base para los comportamientos de selección específicos proporcionados en el paquete. También es útil para extender clases de comportamientos de selección personalizados.

Clase PickObject

La clase PickObject proporciona métodos para determinar qué objeto fue seleccionado por una operación de selección del usuario. Una amplia variedad de métodos resulta de las distintas formas posibles de aplicaciones de selección. Es útil crear clases de selección personalizadas.

Interface PickingCallback

El interface PickingCallback proporciona un marco de trabajo para extender una clase de selección existente. En particular cada una de las clases específicas implementa este interface permitiendo al programador proporcionar un método que sea llamado cuando la operación de selección tenga lugar.

Clase Intersect

La clase Intersect proporciona varios métodos para comprobar la intersección de un objeto PickShape (clase corazón) y geometrías primitivas. Esta clases es útil para la creación de clases picking personalizadas.

6.2.16.10. Clases de Comportamientos Picking Específicas

Incluidas en el paquete com.sun.j3d.utils.behaviors.picking hay clases de comportamientos específicas: PickRotateBehavior, PickTranslateBehavior, y PickZoomBehavior. Estas clases permiten al usuario interactuar con un objeto seleccionado con el ratón. Los comportamientos individuales responden a los diferentes botones del ratón (izquierdo=rotar, derecho=trasladar, central=zoom). Estas clases son subclases de PickMouseBehavior.

PickRotateBehavior

Esta clase permite al usuario seleccionar y rotar interactivamente un objeto visual. El usuario usa el botón izquierdo del ratón para seleccionar y rotar. Se puede usar un ejemplar de PickRotateBehavior en conjunción con otras clases de selección específicas.

PickTranslateBehavior

Esta clase permite al usuario seleccionar y trasladar interactivamente un objeto visual. El usuario usa el botón derecho del ratón para seleccionar y trasladar. Se puede usar un ejemplar de PickTranslateBehavior en conjunción con otras clases de selección específicas.

PickZoomBehavior

Esta clase permite al usuario seleccionar y hacer zoom interactivamente un objeto visual. El usuario usa el botón central del ratón para seleccionar y hacer zoom. Se puede usar un ejemplar de PickZoomBehavior en conjunción con otras clases de selección específicas.

6.2.17. Animación en java 3D

Al igual que las interacciones, las animaciones Java 3D se implementan usando objetos Behavior. Como podrás imaginar, se puede crear cualquier animación personalizada

usando objetos Behavior. Sin embargo, el API Java 3D proporciona varias clases útiles para crear animaciones sin tener que crear una nueva clase.

Los Interpoladores y los Objetos Alpha Proporcionan Animaciones Basadas en el Tiempo. Las acciones de Interpolator incluyen el cambio de localización, orientación, tamaño, color o transparencia de un objeto visual. Todos los comportamientos interpoladores podrían implementarse creando una clase behavior personalizada; sin embargo, usar un interpolador hace más sencilla la creación de estas animaciones. Las clases Interpolator existen para otras acciones, incluyendo algunas combinaciones de estas acciones.

6.2.18. Iluminación en java 3D

Java 3D sombrea los objetos visuales basándose en la combinación de sus características materiales y en las luces del universo virtual. El sombreado resulta de aplicar un modelo de iluminación a un objeto visual en presencia de fuentes de luz. La siguiente sección da una descripción del modelo de iluminación usado en el renderizador de Java 3D y cómo la luz interactúa con las características materiales para proporcionar sombreado.

6.2.19. Sombreado en Java 3D

Sombrear objetos visuales en Java 3D depende de muchos factores. Esta sección proporciona una descripción abreviada del modelo de iluminación de Java 3D, del modelo de color, y del modelo de sombreado.

6.2.20. Modelo de Iluminación

En el mundo real, los colores que percibimos son una combinación de las características físicas del objeto, de las características de las fuentes de luz, de las posiciones relativas de los objetos a las fuentes de luz, y del ángulo desde el cual se ve el objeto. Java 3D utiliza un modelo de iluminación para aproximar la física del mundo real.

La ecuación del modelo de iluminación depende de tres vectores: la superficie normal (n), la dirección de la luz (l), y la dirección al ojo del espectador (e) además de las características materiales del objeto y de las características de la luz. La Figura 18 muestra los tres vectores para dos vértices de una superficie esférica. Los vectores para cada vértice pueden tener distintas direcciones dependiendo de especificidades de la

escena. Cuando los vectores de la luz y del ojo varían, se calculan en tiempo de ejecución. Por lo tanto, cada vértice de la esfera potencialmente se renderiza como una sombra diferente.

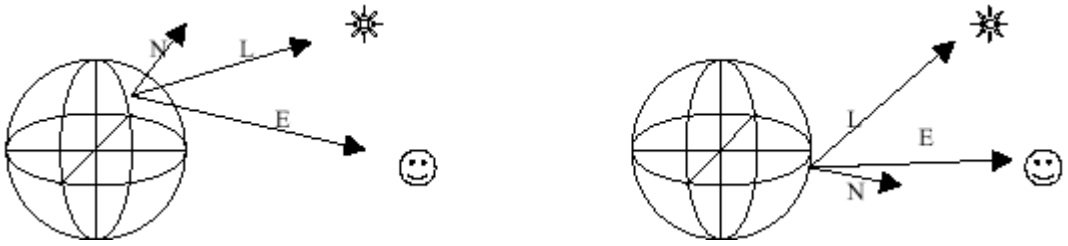


Figura 18 Vectores de Luz, Superficie Normal y Ojo del espectador.

El modelo de iluminación incorpora tres tipos de reflexiones de luz del mundo real: ambiente, difuso, y especular. La reflexión ambiente resulta de la luz ambiente, luz constante de bajo nivel, en una escena. La reflexión difusa es la reflexión normal de una fuente de luz desde un objeto visual. Las reflexiones especulares son las reflexiones sobre iluminadas de una fuente de luz sobre un objeto, que ocurren en ciertas situaciones.

La Figura 19 muestra una esfera y un plano renderizado por Java 3D... La parte más oscura de la esfera exhibe sólo la reflexión ambiente. El centro de la esfera está iluminado por la luz difusa y ambiente. Con una esfera azul y una luz blanca, la reflexión difusa es azul. La parte más brillante de la esfera es el resultado de la reflexión especular con reflexiones ambiente y difusa.

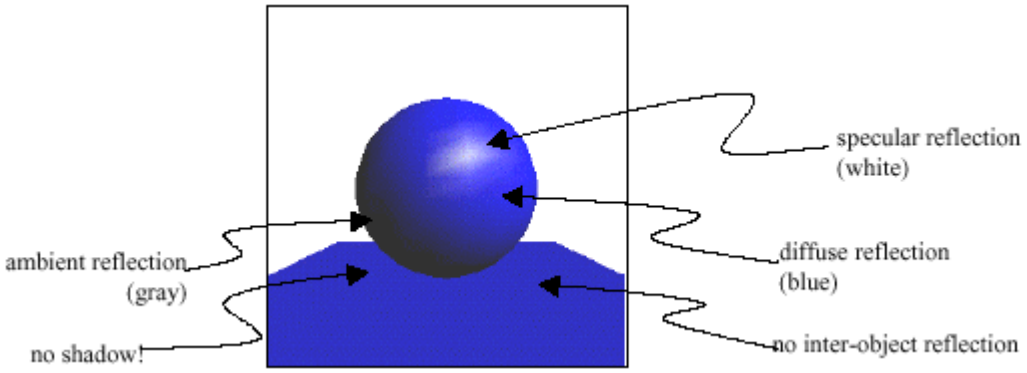


Figura 19 Esfera sombreada y un Plano.

6.2.21. Clase Light

El API Java 3D proporciona cuatro clases para luces. Todas se derivan de la clase Light. La Figura 20 muestra la jerarquía de clases de Java 3D relacionada con las luces. Light, una clase abstracta, proporciona los métodos y las constantes de capacidades asociadas para manipular el estado, color, y los límites de un objeto Light. El estado de la luz es un booleano que activa y desactiva la luz.

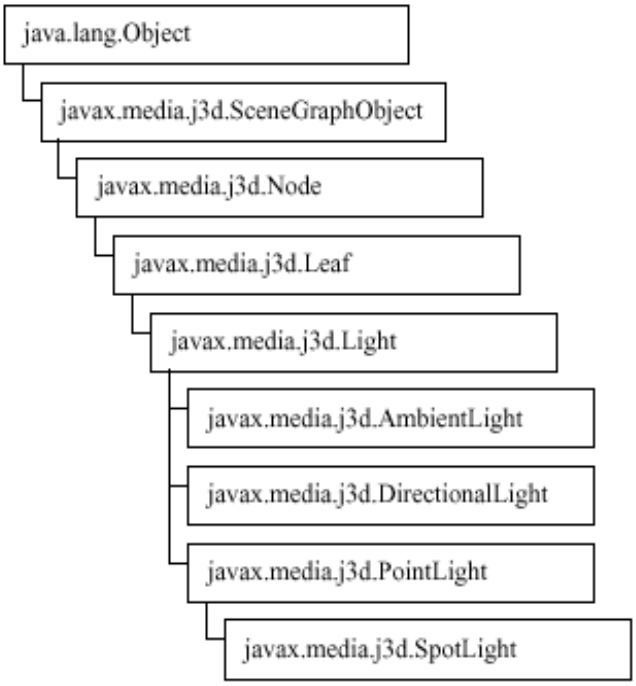


Figura 20 El Árbol de Clases del API Java 3D con las clases Light

El siguiente bloque de referencia lista los métodos y las constantes de la clase Light. Debemos recordar que los límites seleccionados con setInfluencingBounds () activan una luz cuando el objeto bounds referenciado intersecciona con la vista.

6.2.21.1. Luz Ambiente

Los objetos de luz ambiente proporcionan luz de la misma intensidad en todas las localizaciones y en todas las direcciones. Los objetos de luz ambiente modelan la luz reflejada desde otros objetos visuales. Si miramos la superficie inferior de nuestro escritorio, veremos la parte inferior del escritorio aunque ninguna fuente de luz esté dando directamente en esa superficie (a menos que tengamos una lámpara bajo el escritorio). La luz que brillaba hacia arriba en el fondo del escritorio se reflejó en el

suelo y en otros objetos. En ambientes naturales con muchos objetos, la luz se refleja desde muchos objetos para proporcionar la luz ambiente. La clase AmbientLight de Java 3D simula este efecto.

6.2.21.2. Luz Direccional

Una fuente DirectionalLight aproxima fuentes de luz muy distantes tales como el sol. Al contrario que las fuentes AmbientLight, las fuentes DirectionalLight proporcionan una sola dirección al brillo de luz. Para los objetos iluminados con una fuente DirectionalLight, el vector de luz es constante.

La Figura 21 muestra dos vértices de la misma esfera que están siendo iluminados por una fuente DirectionalLight. El vector de luz es igual para estos dos y para todos los vértices. Puesto que todos los vectores de luz de una fuente DirectionalLight son paralelos, la luz no se atenúa. En otras palabras, la intensidad de una fuente DirectionalLight no varía con la distancia al objeto visual y la fuente DirectionalLight.

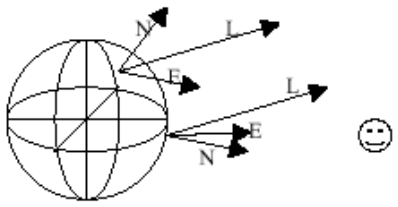


Figura 21 El Vector de Luz es Constante para Fuentes DirectionalLight.

6.2.21.3. Punto de Luz

Un PointLight es el contrario de un DirectionalLight. Es una fuente de luz omnidireccional cuya intensidad se atenúa con la distancia y tiene una localización. Los objetos PointLight se aproximan a bombillas, velas, u otras fuentes de luz sin reflectores o lentes.

Un modelo de ecuación cuadrática modela la atenuación de las fuentes PointLight. La Figura 22 ilustra la relación de un objeto PointLight con una esfera. Observa que los vectores de luz no son paralelos.

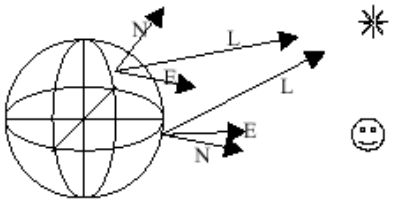


Figura 22 El vector de Luz varía para una fuente PointLight.

6.2.22. Texturas en java 3D

El texturado, también llamado mapeo de textura, es una manera de añadir riqueza visual a una superficie sin la adición de los detalles geométricos finos. La riqueza visual la proporciona una imagen, también llamada textura, que da el aspecto del detalle superficial para el objeto visual. La imagen se mapea dentro de la geometría del objeto visual en el momento de la renderización. De ahí el término mapeo de textura.

El aspecto de muchos objetos del mundo real depende de su textura. La textura de un objeto es realmente la geometría relativamente fina de la superficie de un objeto. Para apreciar la superficie del papel las texturas juegan con el aspecto de los objetos del mundo real, consideremos una alfombra. Incluso cuando todas las fibras de una alfombra son del mismo color la alfombra no aparece con un color constante debido a la interacción de la luz con la geometría de las fibras. Aunque Java 3D es capaz de modelar la geometría de las fibras individuales de la alfombra, los requisitos de memoria y el funcionamiento de la renderización para una alfombra del tamaño de una habitación modelada a tal detalle harían dicho modelo inútil. Por otra parte, tener un polígono plano de un solo color no hace un reemplazo convincente para la alfombra en la escena renderizada.

La Figura 23 muestra algunas de las texturas como podemos ver, una textura puede proporcionar riqueza visual a objetos de distintos tamaños.

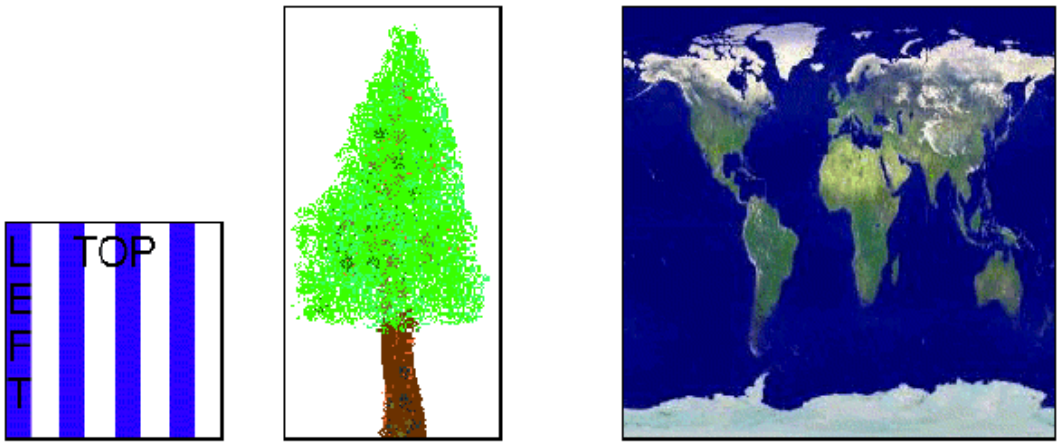


Figura 23 Imágenes usadas como Texturas en los Programas de Ejemplo.

6.2.23. Texturado Básico

El texturado de polígonos en un programa de Java 3D se consigue a través de la creación del manojo de apariencia apropiado y cargando la imagen de la textura dentro de él, especificando la localización de la imagen de la textura en la geometría, y fijando los atributos de texturado. Como veremos, especificar texturas puede ser muy complejo. Afortunadamente, hay clases de utilidad para ayudarnos en el proceso y las configuraciones de los valores por defecto para las opciones texturado son las apropiadas para las aplicaciones de texturado básicas.

6.2.24. La Clase NewTextureLoader

Los programas Java 3D que usan texturas pueden tener una gran cantidad de líneas sólo para cargar las texturas y crear los manojos de apariencia. Se puede ahorrar algo de programación y, más importante, memoria en tiempo de ejecución compartiendo manojos de apariencia cuando sea apropiado. Sin embargo, esto no reduce mucho la cantidad de programación. Se pueden conseguir otras reducciones de programación creando una clase para crear los manojos de apariencia de textura. El desafío de crear esta clase consiste en el requisito del observador de imagen para el objeto TextureLoader.

El objeto Canvas3d o un Applet pueden servir como el observador de imagen, pero tener una referencia a un cierto componente por todas partes en el programa puede ser fastidioso. Para tratar esta inconveniencia, se ha extendido la clase TextureLoader que elimina la necesidad de un componente observador de imagen. En su lugar se utiliza un

solo método para especificar un observador de imagen para todas las aplicaciones futuras del cargador de textura.

Los constructores de NewTextureLoader son iguales a los de TextureLoader excepto en que ninguno requiere un componente observador de imagen.

6.2.25. Coordenadas de Textura

La imagen de la textura se crea para caber en la geometría basándose en la especificación de las coordenadas de textura. El proceso real es asociar los texels de la textura a los pixeles de la geometría cuando es renderizada. Cada pixel de una textura se llama un texel, o un 'elemento de textura'. Éste es el proceso de mapeado de la textura.

El mapeo de textura comienza con la especificación de las coordenadas de textura para los vértices de la geometría. Mientras se renderiza cada pixel del triángulo texturado, se calculan las coordenadas de la textura en el pixel desde los vértices del triángulo. La interpolación Trilinear de las coordenadas de textura de los vértices determina las coordenadas de textura para el pixel y por lo tanto, el texel de la imagen de la textura usada en el color final del pixel.

La Figura 24 ilustra el proceso de interpolación Trilinear para un pixel del ejemplo. La representación se hace en orden de scan. El pixel, P, para el mapeo de textura está justo en el centro del scan actual en el triángulo de la izquierda de la ilustración. Se han asignado las coordenadas de textura para cada uno de los vértices del triángulo. Se han etiquetado TC1, TC2, y TC3. Estas coordenadas de textura son el punto de partida para la interpolación Trilinear. Las primeras dos interpolaciones lineares determinan las coordenadas de la textura a lo largo de las caras del triángulo en el scan (puntos etiquetados A y B en la figura). La tercera interpolación se hace entre estos dos puntos.

Las coordenadas de textura que resultan para P son (0,75, 0,6). En la derecha de la figura está la textura. Usando las coordenadas de textura calculadas para P se selecciona el texel.

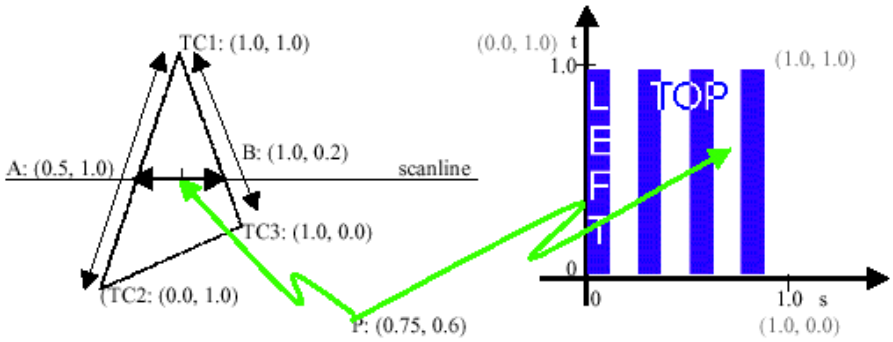


Figura 24 Imagen del Mapeo de Textura

La selección del Texel no se explica completamente en el ejemplo anterior. La especificación de la sección de filtración da más detalles sobre la selección del texel. Otro detalle todavía no explicado es la interacción entre el color del texel, otras fuentes del color, y el color final del pixel. El modo de valor por defecto se 'substituye' con el cuál se utiliza el color del texel como el color del pixel, pero hay otros modos.

En este punto del capítulo se han utilizado todas las texturas en su orientación ordinaria. La Figura 25 muestra planos con algunas de las posibles orientaciones de textura sólo seleccionando las coordenadas de textura en los vértices.

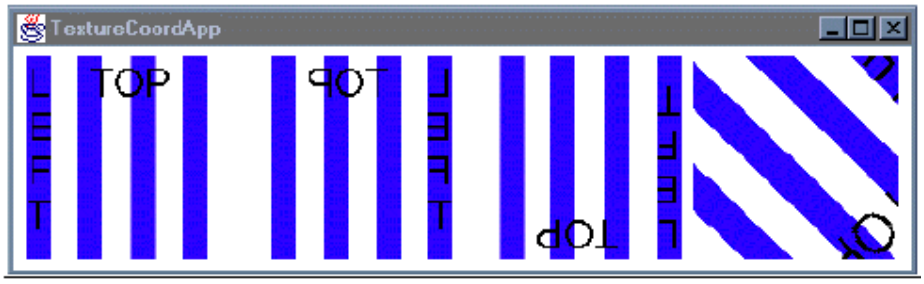


Figura 25 Algunas de las posibles orientaciones de una Textura en un Plano.

6.2.26. Opciones de Textura

Texture2D, es una extensión de Texture. Algunas de las opciones básicas para el texturado se implementan en esta clase. Puesto que Texture es una clase abstracta, sus configuraciones se harán a través de un objeto Texture2D o Texture3d. Las configuraciones son el modo de límites, filtros, y el formato de la textura.

6.2.27. Modo de Límites: Envolver o Abrazar

La configuración del modo de límites determina lo que ocurre cuando el mapeo tiene lugar si las coordenadas de textura van más allá del rango 0 a 1 del espacio de la

imagen. Las opciones son envolver la imagen, o abrazar la imagen. Envolver, significa repetir la imagen según sea necesario, es el valor por defecto. Abrazar utiliza el color del borde de la imagen en cualquier lugar fuera del rango 0 a 1. Estas configuraciones se hacen independientemente en las dimensiones s y t.

6.2.28. Formato de Textura

La última configuración de la clase Texture es la del formato de textura. El formato de textura es una declaración de cuántos valores hay por texel y cómo esos valores afectan a los pixeles. Por ejemplo, una configuración del formato de textura de INTENSITY indica que solo el valor del texel será utilizado para rojo, verde, azul, y los valores de alpha del pixel. Una configuración del formato de textura de RGB indica que los tres valores del texel serán utilizados para los valores rojo, verde, y azul del pixel mientras que el valor de alpha del pixel sigue siendo igual.

Formato de Textura	Valores por Texel	Modifica Color del Pixel	Modifica alpha del Pixel
INTENSITY	1	si, R=G=B	si, R=G=B=A
LUMINANCE	1 (sólo color)	si, R=G=B	No
ALPHA	1 (sólo alpha)	no	Si
LUMINANCE_ALPHA	2	si, R=G=B	Si
RGB	3	si	No
RGBA	4	si	Si

Tabla 6 Formato de textura

6.2.29. Texture3d

Como el nombre implica, un objeto Texture3d contiene una imagen tridimensional de la textura. Puede ser que pensemos en él como un volumen de color. La clase Texture3d es una extensión de Texture, así que todas las características de la clase Texture se aplican a Texture3d. La única característica que Texture3d tiene y que Texture2D no tiene, es una especificación para el modo de límite de la tercera dimensión.

CAPITULO 3: METODOLOGIAS DE DESARROLLO DE SOFTWARE

6.3. Introducción

El desarrollo de software no es una tarea fácil. Prueba de ello es que existen numerosas propuestas metodológicas que inciden en distintas dimensiones del proceso de desarrollo. Por una parte tenemos aquellas propuestas más tradicionales que se centran especialmente en el control del proceso, estableciendo rigurosamente las actividades involucradas, los artefactos que se deben producir, y las herramientas y notaciones que se usarán. Estas propuestas han demostrado ser efectivas y necesarias en un gran número de proyectos, pero también han presentado problemas en otros muchos. Una posible mejora es incluir en los procesos de desarrollo más actividades, más artefactos y más restricciones, basándose en los puntos débiles detectados. Sin embargo, el resultado final sería un proceso de desarrollo más complejo que puede incluso limitar la propia habilidad del equipo para llevar a cabo el proyecto.

6.3.1. ICONIX

ICONIX es una metodología del desarrollo del software que maneja casos de uso como el Proceso unificado racional (RUP), es pequeño y firme como la Programación extrema (XP) y pretende el Desarrollo ágil del software. Este proceso también hace uso aerodinámico del UML mientras guarda un enfoque afilado en el seguimiento de requisitos.

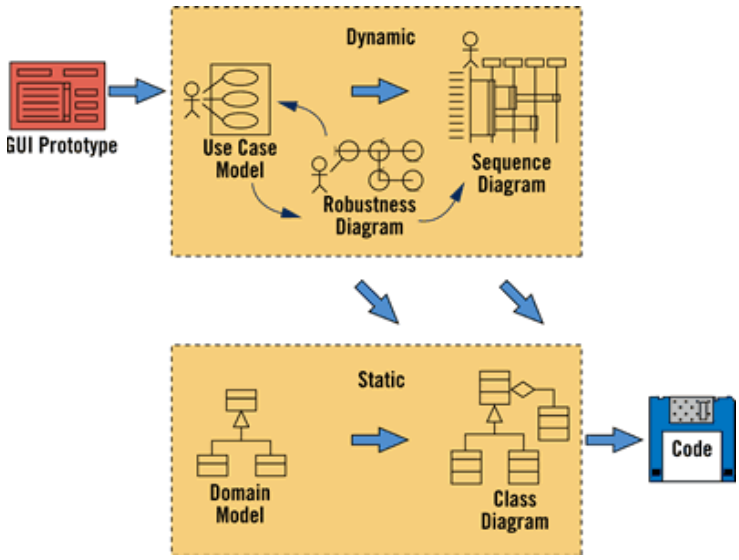


Figura 26 Diagrama de ICONIX

La metodología de desarrollo de software ICONIX es un proceso simplificado en comparación con otros procesos más tradicionales, que unifica un conjunto de métodos

de orientación a objetos con el objetivo de abarcar el ciclo de vida de un proyecto, además presenta claramente las actividades de cada etapa y exhibe una secuencia de pasos que deben ser seguidos.

Una distinción del principio de ICONIX es su uso de análisis de la robustez, un método para tender un puente sobre el boquete entre el análisis y el diseño. El análisis de la robustez reduce las descripciones funcionando del caso de la ambigüedad, asegurándose de que están escritos en el contexto de un acompañamiento modelo del dominio. Este proceso hace los casos del uso mucho más fáciles diseñar, probar y estimar.

6.3.1.1. Características de ICONIX

Iterativo e incremental: Varias iteraciones ocurren entre el desarrollo del modelo y la identificación de los casos de uso. El modelo estático es incrementalmente refinado por los modelos dinámicos.

Trazabilidad: cada paso está referenciado por algún requisito. Se define trazabilidad como la capacidad de seguir una relación entre los diferentes artefactos de software producidos.

Dinámica del UML: La metodología un uso dinámico del UML por que utiliza alguno diagramas sin exigir la utilización de todos, como en el caso del RUP.

6.3.1.2. Tareas de ICONIX

Análisis de requisitos

- Modelo del dominio
- Prototipación rápido
- Modelo de casos de uso

Análisis y diseño preliminar

- Descripción de casos de uso
- Diagramas de robustez

Diseño

- Diagramas de Secuencia

Implementación

- Escribir y generar el código.

6.3.2. Comparación entre las metodologías agiles y tradicionales

La siguiente tabla recoge esquemáticamente las principales diferencias de las metodologías agiles respecto a las tradicionales. Estas diferencias que afectan no solo al proceso sino también al contexto del equipo así como a su organización.

Metodologías Agiles	Metodologías Tradicionales
Basadas en heurística provenientes de prácticas de producción de código.	Basadas en normas provenientes de estándares seguidos por el entorno de desarrollo.
Especialmente preparados para cambios durante el proyecto.	Cierta resistencia a los cambios.
Impuestas internamente (por el equipo)	Impuestas externamente.
Proceso menos controlado, con pocos principios.	Proceso mucho más controlado, con numerosos políticas/normas.
No existe contrato tradicional o al menos bastante flexible.	Existe un contrato prefijado.
El cliente es parte del equipo de desarrollo.	El cliente interactúa con el equipo de desarrollo mediante reuniones.
Grupos pequeños y trabajando en el mismo sitio.	Grupos grandes y posiblemente distribuidos.
Pocos artefactos y roles.	Mas artefacto y roles.
Menos énfasis en la arquitectura del software.	La arquitectura del software es esencial y se expresa mediante roles.

Tabla 7 Diferencias entre metodologías ágiles y tradicionales

6.3.3. Lenguaje Unificado de Modelado - UML

El Lenguaje Unificado de Modelado prescribe un conjunto de notaciones y diagramas estándar para modelar sistemas orientados a objetos, y describe la semántica esencial de lo que estos diagramas y símbolos significan. Mientras que ha habido muchas notaciones y métodos usados para el diseño orientado a objetos, ahora los modeladores sólo tienen que aprender una única notación. UML se puede usar para modelar distintos tipos de sistemas: sistemas de software, sistemas de hardware, y organizaciones del mundo real. UML ofrece nueve diagramas en los cuales modelar sistemas.

- Diagramas de Casos de Uso para modelar los procesos 'business'.
- Diagramas de Secuencia para modelar el paso de mensajes entre objetos.

- Diagramas de Colaboración para modelar interacciones entre objetos.
- Diagramas de Estado para modelar el comportamiento de los objetos en el sistema.
- Diagramas de Actividad para modelar el comportamiento de los Casos de Uso, objetos u operaciones.
- Diagramas de Clases para modelar la estructura estática de las clases en el sistema.
- Diagramas de Objetos para modelar la estructura estática de los objetos en el sistema.
- Diagramas de Componentes para modelar componentes.
- Diagramas de Implementación para modelar la distribución del sistema.

UML es una consolidación de muchas de las notaciones y conceptos más usados orientados a objetos. Empezó como una consolidación del trabajo de Grade Booch, James Rumbaugh, e Ivar Jacobson, creadores de tres de las metodologías orientadas a objetos más populares.

En 1996, el Object Management Group (OMG), un pilar estándar para la comunidad del diseño orientado a objetos, publicó una petición con propósito de un metamodelo orientado a objetos de semántica y notación estándares. UML, en su versión 1.0, fue propuesto como una respuesta a esta petición en enero de 1997. Hubo otras cinco propuestas rivales. Durante el transcurso de 1997, los seis promotores de las propuestas, unieron su trabajo y presentaron al OMG un documento revisado de UML, llamado UML versión 1.1. Este documento fue aprobado por el OMG en Noviembre de 1997. El

OMG llama a este documento OMG UML versión 1.1. El OMG está actualmente en proceso de mejorar una edición técnica de esta especificación, prevista su finalización para el 1 de abril de 1999.

6.3.3.1. Funciones del UML

- Permite que los desarrolladores representen gráficamente un sistema, de tal manera que otro u otros lo puedan leer.
- Permite especificar las características de un sistema antes de su construcción.
- Permite documentar el sistema desarrollado a través de sus elementos gráficos.

- A partir de los modelos realizados en éste lenguaje, se pueden construir los sistemas diseñados.

6.3.3.2. Diagramas

Un diagrama es la representación gráfica de un sistema o subsistema, que permite comprender de mejor manera cómo interactúan sus elementos en la realidad (sistema actual) y cómo se desea que interactúen en el sistema a desarrollar. Gráficamente un diagrama es un conjunto de vértices y arcos. Sin embargo, lo realmente valioso de un diagrama es su significado y no su imagen gráfica. Los diagramas son usados para visualizar un sistema desde diferentes perspectivas, puesto que ningún sistema complejo puede entenderse totalmente desde un único punto de vista. Si están bien elaborados, los diagramas pueden lograr que el sistema en desarrollo sea entendible y accesible.

6.3.3.3. Diagramas de casos de uso

Este diagrama muestra un conjunto casos de uso, actores y sus relaciones. Ayuda a documentar el comportamiento que tiene el sistema actual (análisis) o el comportamiento que va a tener el sistema a desarrollar (Diseño), a través de la identificación de los actores y escenarios que se van a presentar.

El Diagrama de Casos de Uso es especialmente importante para el modelado y la organización del comportamiento de un sistema.

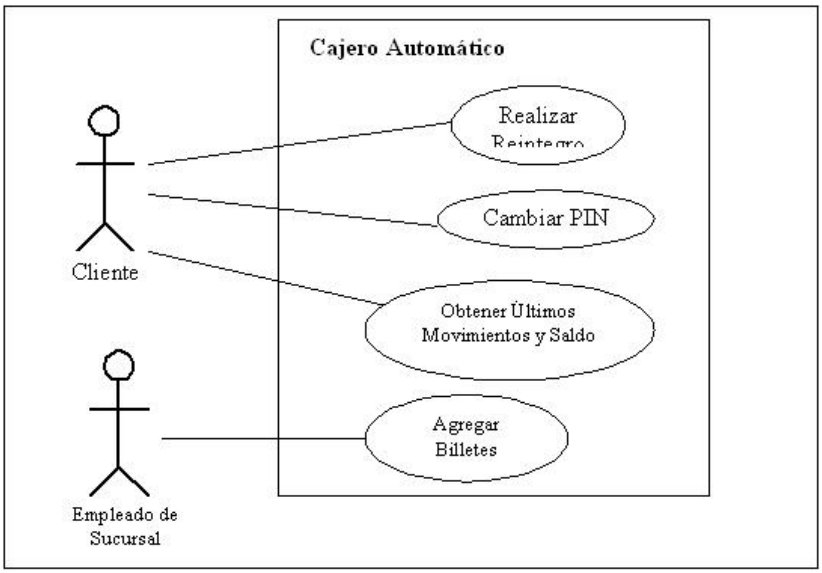


Figura 27 Diagrama de casos de uso

6.3.3.4. Diagramas de Secuencia

Este tipo de diagrama representa la secuencia ordenada y lógica que se da entre los objetos de un caso de uso dentro de un escenario específico. Se enfoca en ordenar los mensajes de acuerdo a su tiempo de envío.

Los Diagramas de Secuencia constituyen el mayor trabajo que debe realizarse durante la etapa de diseño, puesto que marcan el comportamiento que tendrá el sistema en tiempo de ejecución. Para lograrlo, se deberá incluir con gran detalle, cómo los objetos van a realizar todos y cada uno de los procesos.

En los diagramas de secuencia existen dos tipos de mensajes: sincrónicos y asincrónicos. Los mensajes sincrónicos se corresponden con llamadas a métodos del objeto que recibe el mensaje. El objeto que envía el mensaje queda bloqueado hasta que termina la llamada. Este tipo de mensajes se representan con flechas con la cabeza llena. Los mensajes asincrónicos terminan inmediatamente, y crean un nuevo hilo de ejecución dentro de la secuencia. Se representan con flechas con la cabeza abierta.

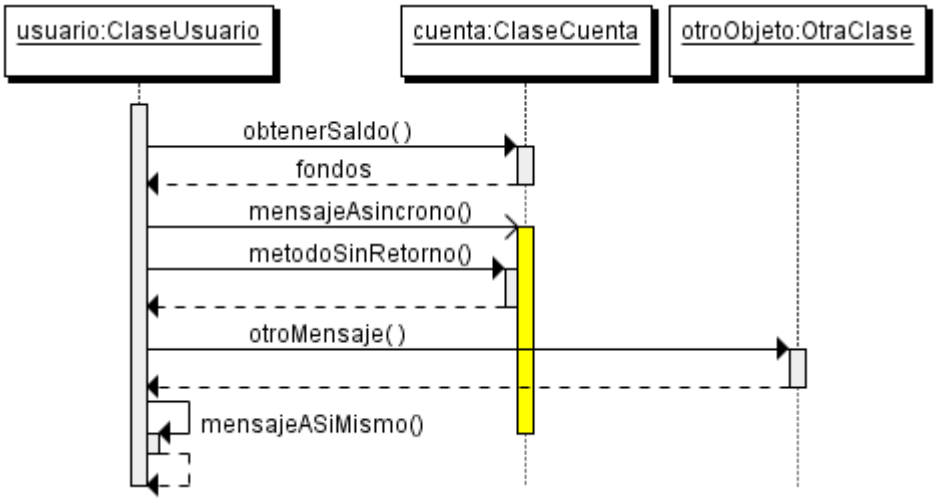


Figura 28 Diagrama de Secuencia

6.3.3.5. Diagrama de Colaboración

Un diagrama de colaboración en las versiones de UML 1.x es esencialmente un diagrama que muestra interacciones organizadas alrededor de los roles. A diferencia de los diagramas de secuencia, los diagramas de comunicación muestran explícitamente las relaciones de los roles. Por otra parte, un diagrama de comunicación no muestra el

tiempo como una dimensión aparte, por lo que resulta necesario etiquetar con números de secuencia tanto la secuencia de mensajes como los hilos concurrentes.

- Muestra cómo las instancias específicas de las clases trabajan juntas para conseguir un objetivo común.
- Implementa las asociaciones del diagrama de clases mediante el paso de mensajes de un objeto a otro. Dicha implementación es llamada "enlace".

Un diagrama de comunicación es también un diagrama de clases que contiene roles de clasificador y roles de asociación en lugar de sólo clasificadores y asociaciones. Los roles de clasificador y los de asociación describen la configuración de los objetos y de los enlaces que pueden ocurrir cuando se ejecuta una instancia de la comunicación. Cuando se instancia una comunicación, los objetos están ligados a los roles de clasificador y los enlaces a los roles de asociación. El rol de asociación puede ser desempeñado por varios tipos de enlaces temporales, tales como argumentos de procedimiento o variables locales del procedimiento. Los símbolos de enlace pueden llevar estereotipos para indicar enlaces temporales.

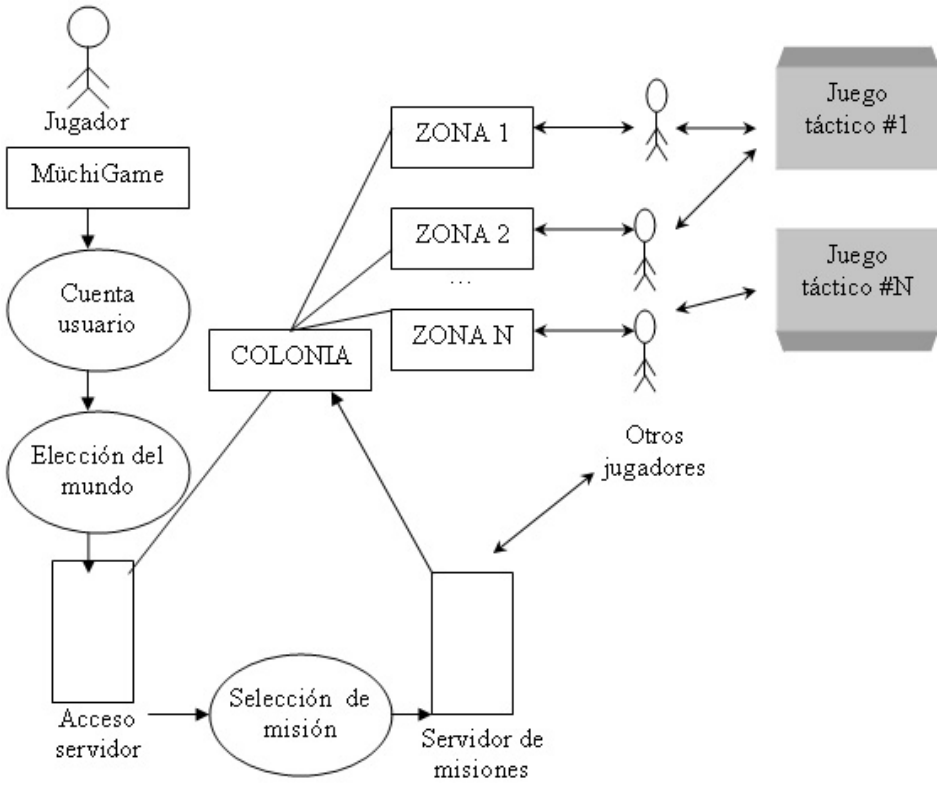


Figura 29 Diagrama de Colaboración

6.3.3.6. Diagrama de Estado

Capturan el ciclo de vida de uno o de más objetos y expresa los estados que va a tener el o los objetos durante su tiempo de ejecución. Estos diagramas son especialmente importantes en el modelado del comportamiento de una interfaz, clase o colaboración.

6.3.3.7. Diagrama de Actividades

En el Lenguaje de Modelado Unificado, un diagrama de actividades representa los flujos de trabajo paso a paso de negocio y operacionales de los componentes en un sistema. Un Diagrama de Actividades muestra el flujo de control general.

En UML 1.x, un diagrama de Actividades es una variación del Diagrama de estados UML donde los "estados" representan operaciones, y las transiciones representan las actividades que ocurren cuando la operación es completa.

El diagrama de Actividades UML 2.0, mientras que es similar en aspecto al diagrama de Actividades UML 1.x, ahora tiene semánticas basadas en redes de Petri. En UML 2.0, el diagrama general de Interacción está basado en el diagrama de Actividades.

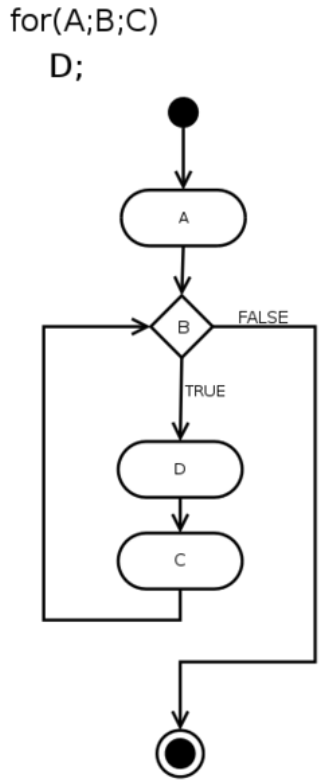


Figura 30 Diagrama de Actividades para un loop

6.3.3.8. Diagrama de Clases

Un diagrama de clases es un tipo de diagrama estático que describe la estructura de un sistema mostrando sus clases, atributos y las relaciones entre ellos. Los diagramas de clases son utilizados durante el proceso de análisis y diseño de los sistemas, donde se crea el diseño conceptual de la información que se manejará en el sistema, y los componentes que se encargaran del funcionamiento y la relación entre uno y otro.

En el diagrama de Clases se representa los Requerimientos en entidades y actuaciones, la arquitectura conceptual de un dominio, las soluciones de diseño en una arquitectura y los Componentes de software orientados a objetos

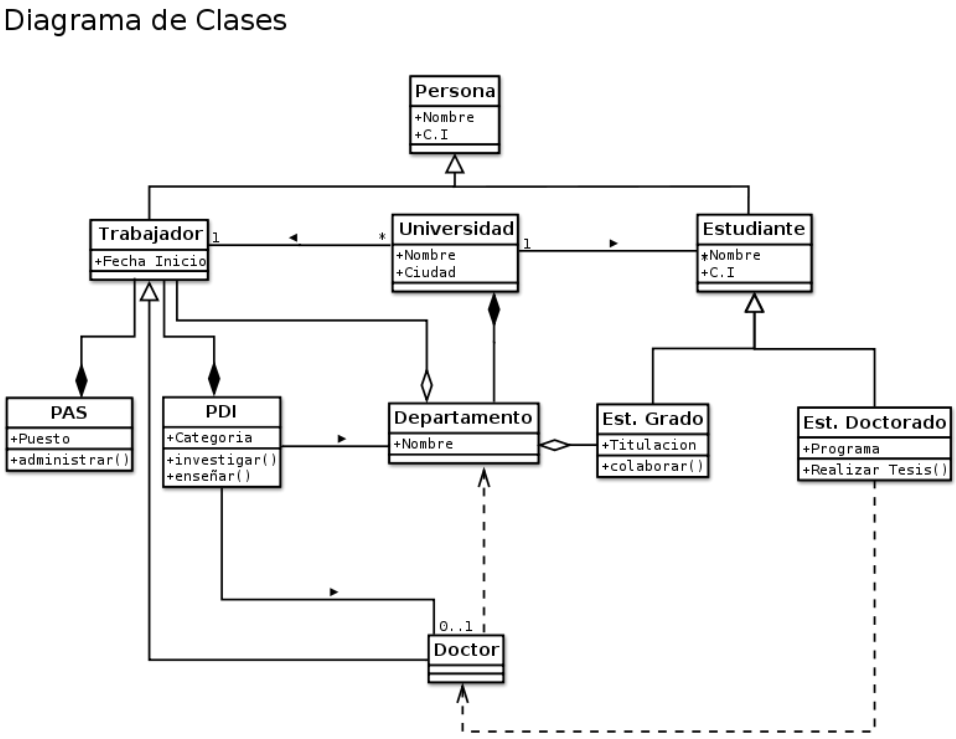


Figura 31 Diagrama de Clases de una Universidad

6.3.3.9. Diagrama de Objetos

Muestra un conjunto de objetos y la relación existente entre ellos en un momento determinado de la aplicación representan una instantánea estática de los objetos que forman parte de un Diagrama de Clases, motivo por el cual, el analista deberá decidir previamente cuál es la situación del sistema que se desea representar.

6.3.3.10. Diagrama de Componentes

Este tipo de diagrama muestra un conjunto de componentes y sus relaciones, permitiendo a los analistas realizar el diseño de cómo quedará un sistema en su parte física. Normalmente los Diagramas de Componentes se utilizan para modelar código fuente, código ejecutable y bases de datos físicas. Estos se relacionan con los Diagramas de Clase en donde un componente típico mapea a una o a más clases, interfaces o colaboraciones.

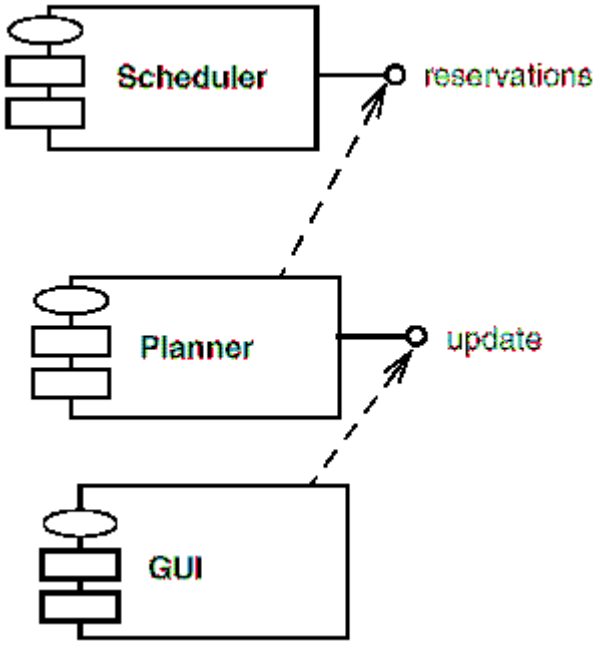


Figura 32 Diagrama de Componentes

6.3.3.11. Diagrama de Implementación

Los Diagramas de Implementación se usan para modelar la configuración de los elementos de procesado en tiempo de ejecución (run-time processing elements) y de los componentes, procesos y objetos de software que viven en ellos. En el diagrama 'deployment' (despliegue), empiezas modelando nodos físicos y las asociaciones de comunicación que existen entre ellos. Para cada nodo, puedes indicar qué instancias de componentes viven o corren (se ejecutan) en el nodo. También puedes modelar los objetos que contiene el componente.

Los Diagramas de Implementación se usan para modelar sólo componentes que existen como entidades en tiempo de ejecución; no se usan para modelar componentes solo de

tiempo de compilación o de tiempo de enlazado. Puedes también modelar componentes que migran de nodo a nodo u objetos que migran de componente a componente usando una relación de dependencia con el estereotipo 'becomes' (se transforma).

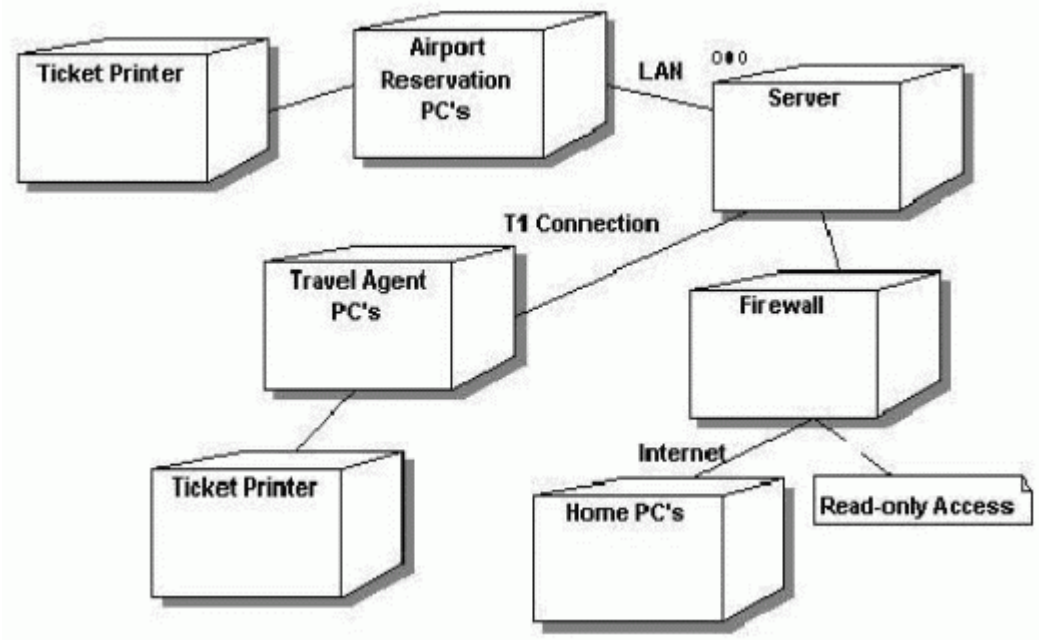


Figura 33 Diagrama de Implementación

CAPITULO 4: ARQUITECTURA DEL SOFTWARE

6.4. Arquitectura

La Arquitectura de Software, también se denomina Arquitectura lógica que consiste en un conjunto de patrones y abstracciones coherentes que proporcionan el marco de referencia necesario para guiar la construcción del software para un sistema de información. Esta establece los fundamentos para que analistas, diseñadores, programadores, etc. trabajen en una línea común que permita alcanzar los objetivos del sistema de información, cubriendo todas las necesidades, unas arquitecturas son más recomendables de implementar con ciertas tecnologías mientras que otras tecnologías no son aptas para determinadas arquitecturas. La arquitectura de software define, de manera abstracta, los componentes que llevan a cabo alguna tarea de computación, sus interfaces y la comunicación entre ellos.

6.4.1. Modelos o Vistas

Toda arquitectura de software debe describir diversos aspectos del software. Generalmente, cada uno de estos aspectos se describe de una manera más comprensible si se utilizan distintos modelos o vistas. Es importante destacar que cada uno de ellos constituye una descripción parcial de una misma arquitectura y es deseable que exista cierto solapamiento entre ellos. Esto es así porque todas las vistas deben ser coherentes entre sí, evidente dado que describen la misma cosa.

Cada paradigma de desarrollo exige diferente número y tipo de vistas o modelos para describir una arquitectura. No obstante, existen al menos tres vistas absolutamente fundamentales en cualquier arquitectura:

- La visión **estática**: describe qué componentes tiene la arquitectura.
- La visión **funcional**: describe qué hace cada componente.
- La visión **dinámica**: describe cómo se comportan los componentes a lo largo del tiempo y cómo interactúan entre sí.

Las vistas o modelos de una arquitectura de software pueden expresarse mediante uno o varios lenguajes. El más obvio es el lenguaje natural, pero existen otros lenguajes tales como los diagramas de estado, los diagramas de flujo de datos, etc. Estos lenguajes son apropiados únicamente para un modelo o vista. Afortunadamente existe cierto consenso

en adoptar UML (*Unified Modeling Language*, lenguaje unificado de modelado) como lenguaje único para todos los modelos o vistas. Sin embargo, un lenguaje generalista corre el peligro de no ser capaz de describir determinadas restricciones de un sistema de información (o expresarlas de manera incomprensible).

6.4.2. Tipos de Arquitecturas

Generalmente, no es necesario inventar una nueva arquitectura de software para cada sistema de información. Lo habitual es adoptar una arquitectura conocida en función de sus ventajas e inconvenientes para cada caso en concreto. Así, las arquitecturas más universales son:

- **Monolítica.** Donde el software se estructura en grupos funcionales muy acoplados.
- **Cliente-servidor**. Donde el software reparte su carga de cómputo en dos partes independientes pero sin reparto claro de funciones.
- **Arquitectura de tres niveles**. Especialización de la arquitectura cliente-servidor donde la carga se divide en tres partes (o capas) con un reparto claro de funciones: una capa para la presentación (interfaz de usuario), otra para el cálculo (donde se encuentra modelado el negocio) y otra para el almacenamiento (persistencia). Una capa solamente tiene relación con la siguiente.

6.4.3. Arquitectura por capas

La Arquitectura por capas es un estilo de programación en el que el objetivo primordial es la separación de la lógica de negocios de la lógica de diseño; un ejemplo básico de esto consiste en separar la capa de datos de la capa de presentación al usuario.

La ventaja principal de este estilo es que el desarrollo se puede llevar a cabo en varios niveles y, en caso de que sobrevenga algún cambio, sólo se ataca al nivel requerido sin tener que revisar entre código mezclado. Un buen ejemplo de este método de programación sería el modelo de interconexión de sistemas abiertos. Además, permite distribuir el trabajo de creación de una aplicación por niveles; de este modo, cada grupo de trabajo está totalmente abstraído del resto de niveles, de forma que basta con conocer la API que existe entre niveles.

En el diseño de sistemas informáticos actual se suele usar las arquitecturas multinivel o Programación por capas. En dichas arquitecturas a cada nivel se le confía una misión simple, lo que permite el diseño de arquitecturas escalables (que pueden ampliarse con facilidad en caso de que las necesidades aumenten).

El diseño más utilizado actualmente es el diseño en tres niveles (o en tres capas).

6.4.3.1. Capas o Niveles

Dentro del modelado en tres niveles existen capas las cuales están bien definidas de la siguiente forma:

Capa de presentación: es la que ve el usuario (también se la denomina "capa de usuario"), presenta el sistema al usuario, le comunica la información y captura la información del usuario en un mínimo de proceso (realiza un filtrado previo para comprobar que no hay errores de formato). También es conocida como interfaz gráfica y debe tener la característica de ser "amigable" (entendible y fácil de usar) para el usuario. Esta capa se comunica únicamente con la capa de negocio.

Capa de negocio: es donde residen los programas que se ejecutan, se reciben las peticiones del usuario y se envían las respuestas tras el proceso. Se denomina capa de negocio (e incluso de lógica del negocio) porque es aquí donde se establecen todas las reglas que deben cumplirse. Esta capa se comunica con la capa de presentación, para recibir las solicitudes y presentar los resultados, y con la capa de datos, para solicitar al gestor de base de datos almacenar o recuperar datos de él. También se consideran aquí los programas de aplicación.

Capa de datos: es donde residen los datos y es la encargada de acceder a los mismos. Está formada por uno o más gestores de bases de datos que realizan todo el almacenamiento de datos, reciben solicitudes de almacenamiento o recuperación de información desde la capa de negocio.

Todas estas capas pueden residir en un único ordenador, si bien lo más usual es que haya una multitud de ordenadores en donde reside la capa de presentación (son los clientes de la arquitectura cliente/servidor). Las capas de negocio y de datos pueden residir en el mismo ordenador, y si el crecimiento de las necesidades lo aconseja se

pueden separar en dos o más ordenadores. Así, si el tamaño o complejidad de la base de datos aumenta, se puede separar en varios ordenadores los cuales recibirán las peticiones del ordenador en que resida la capa de negocio.

Si, por el contrario, fuese la complejidad en la capa de negocio lo que obligase a la separación, esta capa de negocio podría residir en uno o más ordenadores que realizarían solicitudes a una única base de datos. En sistemas muy complejos se llega a tener una serie de ordenadores sobre los cuales corre la capa de negocio, y otra serie de ordenadores sobre los cuales corre la base de datos.

En una arquitectura de tres niveles, los términos "capas" y "niveles" no significan lo mismo ni son similares.

El término "capa" hace referencia a la forma como una solución es segmentada desde el punto de vista lógico:

Presentación/
Lógica de Negocio/
Datos.

En cambio, el término "nivel" corresponde a la forma en que las capas lógicas se encuentran distribuidas de forma física. Por ejemplo:

Una solución de tres capas (presentación, lógica del negocio, datos) que residen en un solo ordenador (Presentación+lógica+datos). Se dice que la arquitectura de la solución es de tres capas y *un nivel*.

Una solución de tres capas (presentación, lógica del negocio, datos) que residen en dos ordenadores (presentación - lógica, lógica - datos). Se dice que la arquitectura de la solución es de tres capas y *dos niveles*.

CAPITULO 5: LICENCIAS DEL SOFTWARE

6.5. Introducción

La licencia de software es una especie de contrato, en donde se especifican todas las normas y cláusulas que rigen el uso de un determinado programa, principalmente se estipulan los alcances de uso, instalación, reproducción y copia de estos productos.

El negocio del software se basa en licencias binarias. La propiedad intelectual de los distribuidores de software comercial nace del código fuente. Las licencias de software se crean con diversos fines empresariales y para afrontar diversos tipos de relaciones (como distribuidor/cliente y partner/partner). Los desarrolladores de software tanto comercial como no comercial utilizan decenas de licencias que abarcan una gran variedad de términos y condiciones.

Es común que las grandes empresas dispongan de sistemas que poseen altos costos de mantenimiento, actualización, capacitación, soporte, etc. que muchas veces superan el costo de obtención de la licencia. Por otra parte, han surgido cada vez con mayor fuerza programas de código libre amigables para el "usuario del hogar" que le permiten abaratar costos en desmedro de otro software comercial con altos costos en licencias.

Conocer las ventajas, desventajas, derechos y deberes de las empresas y de los usuarios finales, además de todas las otras personas que se relacionan con el software, de las licencias de software más utilizadas, tanto el software libre como el software comercial, es imprescindible para que las empresas y los usuarios finales puedan tomar las mejores decisiones acerca de los sistemas que utilizarán. Es importante también conocer cómo afectan estas licencias al trabajo de otras personas, como por ejemplo a los desarrolladores, vendedores, distribuidores, etc... Las licencias de uso de software generalmente caen en alguno de estos tipos:

- Licencia propietaria. Uso en una computadora por el pago de un precio.
- Shareware. Uso limitado en tiempo o capacidades, después pagar un precio.
- Freeware. Usar y copiar ilimitado, precio es cero.
- Software libre. Usar, copiar, estudiar, modificar, redistribuir. Código fuente incluido.

Es posible dividir las licencias de software libre en dos grandes familias. Una de ellas está compuesta por las licencias que no imponen condiciones especiales, sólo

especifican que el software se puede redistribuir o modificar. Estas son las llamadas licencias permisivas. La otra familia, denominadas licencias robustas o licencias copyleft, imponen condiciones en caso de que se quiera redistribuir el software, condiciones que van en la línea de forzar a que se sigan cumpliendo las condiciones de la licencia después de la primera redistribución.

Mientras que el primer grupo hace énfasis en la libertad de quien recibe un programa, ya que le permite hacer casi lo que quiera con él (en términos de las sucesivas redistribuciones), el segundo obliga a que las modificaciones y redistribuciones respeten los términos de la licencia original.

6.5.1. Licencia GPL

La Licencia Pública General (inglés: General Public License o GPL) otorga al usuario la libertad de compartir el software licenciado bajo ella, así como realizar cambios en él. Es decir, el usuario tiene derecho a usar un programa licenciado bajo GPL, modificarlo y distribuir las versiones modificadas de éste.

La licencia GPL adopta el principio de la no ocultación, respaldando el concepto moral que establece que todo software desarrollado con el uso de material licenciado bajo GPL debe estar disponible para ser compartido con el resto de la humanidad.

GPL fue creada para mantener la libertad del software y evitar que alguien quisiera apropiarse de la autoría intelectual de un determinado programa. La licencia advierte que el software debe ser gratuito y que el paquete final, también debe ser gratuito, asegurándose siempre de mantener los nombres y créditos de los autores originales.

Como aspecto curioso, se debe considerar que si se reutiliza un programa "A", licenciado bajo GPL, y se reutiliza un programa "B", bajo otro tipo de licencia libre, el programa final "C", debe de estar bajo la licencia GPL. Este concepto se introduce con el denominado copyleft a fin de garantizar que cualquier aprovechamiento de un programa bajo licencia GPL redunde sobre la comunidad.

6.5.1.1. Ventajas de GPL

- Cualquier código fuente licenciado bajo GPL, debe estar disponible y accesible, para copias ilimitadas y a cualquier persona que lo solicite.

- De cara al usuario final, el software licenciado bajo GPL es totalmente gratuito, pudiendo pagar únicamente por gastos de copiado y distribución.
- Se ha establecido la idea global que GPL contribuye al mejoramiento y evolución del software, ya que la disponibilidad y acceso global de los programas permite la expansión del conocimiento depositado en cada pieza de software.

6.5.1.2. Desventajas de GPL

- Si el desarrollador incluye código fuente bajo GPL en otro programa, todo el programa final está obligado a seguir las condiciones y términos de la licencia GPL.
- El software licenciado bajo GPL carece de garantía. El autor del software no se hace responsable por el malfuncionamiento del mismo.
- De cara al desarrollador, no se puede establecer ningún cobro por las modificaciones realizadas. Únicamente se pueden establecer cobros asociados a copiado y distribución.
- Aunque GPL posibilita la modificación y redistribución del software, obliga a que se haga únicamente bajo esa misma licencia.

6.5.2. Licencia LGPL

La Licencia Pública General Menor (inglés: Lesser General Public License o LGPL) es una modificación de la licencia GPL descrita anteriormente. La LGPL reconoce que muchos desarrolladores de software no utilizarán el código fuente que se distribuya bajo la licencia GPL, debido a su principal desventaja que determina que todos los derivados tendrán que seguir los dictámenes de esa licencia. La LGPL permite que los desarrolladores utilicen programas bajo la GPL o LGPL sin estar obligados a someter el programa final bajo dichas licencias.

La licencia LGPL permite entonces la utilización simultánea de software con este tipo de licencia tanto en desarrollos libres como en desarrollos privativos. Entonces, LGPL es una licencia de software libre que no tiene un copyleft fuerte, porque permite que el software se enlace con módulos no libres.

6.5.2.1. Ventajas de LGPL

- Si en el desarrollo de un producto se utiliza código fuente licenciado bajo GPL o LGPL, no es obligatorio licenciar dicho producto final bajo dichas licencias.
- LGPL es menos restrictiva que la licencia GPL, ya que sólo se ocupa en impedir el realizar versiones comerciales del producto licenciado bajo LGPL.
- Ahora bien, LGPL permite realizar versiones comerciales de un producto final que contenga como herramienta adicional un programa LGPL. Por lo tanto, LGPL puede ser utilizada o enlazada con software propietario.
- LGPL exige registrar todos los cambios realizados por terceros, a manera de no afectar la reputación del autor original del software.

6.5.2.2. Desventajas de LGPL

Otras actividades que no sean copia, distribución o modificación no están cubiertas en esta licencia y están fuera de su alcance

6.5.3. Licencia BSD

La Licencia de Distribución de Software de Berkeley (inglés: Berkeley Software Distribution o BSD) no impone ninguna restricción a los desarrolladores de software en lo referente a la utilización posterior del código en derivados y licencias de estos programas. Este tipo de licencia permite a los programadores utilizar, modificar y distribuir a terceros el código fuente y el código binario del programa de software original con o sin modificaciones. Los trabajos derivados pueden optar a licencias de código abierto o comercial.

La licencia BSD es un buen ejemplo de una licencia permisiva, que casi no impone condiciones sobre lo que un usuario puede hacer con el software. La licencia BSD permite la redistribución, uso y modificación del software.

Esta licencia permite el uso del código fuente en software no libre, con lo que es muy similar a la LGPL descrita anteriormente. La diferencia consiste en que en la licencia BSD no es obligatorio mencionar a los autores ni proporcionar el código fuente.

El autor, bajo esta licencia, mantiene la protección de copyright únicamente para la renuncia de garantía y para requerir la adecuada atribución de la autoría en trabajos derivados, pero permite la libre redistribución y modificación.

6.5.3.1. Ventajas de BSD

- La licencia BSD permite el uso de código fuente en software propietario.
- Una aplicación licenciada con BSD permite que otras versiones puedan tener otros tipos de licencias, tanto propietarias como libres.
- BSD permite que los redistribuidores puedan hacer casi cualquier cosa con el software, incluyendo usarlo para productos propietarios.
- De cara al desarrollador, BSD permite el cobro por la distribución de objetos binarios. Así mismo, el desarrollador no está en la obligación de incluir el código fuente.
- Se argumenta que la licencia BSD asegura el verdadero software libre, en el sentido que el usuario tiene libertad ilimitada con respecto al software, y que puede decidir incluso redistribuirlo como no libre.
- La utilización de las licencias BSD ha contribuido al mantenimiento de un ecosistema de software sano, ya que ha permitido la investigación con fondos públicos y su posterior comercialización, con lo cual se mejora al sector privado del software.

6.5.3.2. Desventajas de BSD

- Las distribuciones del software bajo licencia BSD deben incluir copias literales de la licencia, anuncio de copyright y una “negación de responsabilidad" (inglés: disclaimer).
- Debe incluirse reconocimiento del origen del software (la Universidad de California) en cualquier anuncio, con el aviso publicitario de BSD, más no es obligatorio la inclusión de los autores.
- BSD no incluye ninguna restricción orientada a garantizar que los trabajos derivados sigan siendo libres.
- De cara al usuario final, BSD permite establecer el pago por la distribución de objetos binarios. Así mismo, el usuario puede no recibir el código fuente.
- Aunque se argumenta que BSD segura el software libre, existen tendencias que destacan que BSD no contribuye al desarrollo de nuevo software libre, ya que el mismo puede ser utilizado en desarrollos propietarios o de distribución no libre.

6.5.4. Licencia MPL

La Licencia Pública de Mozilla (inglés: Mozilla Public License o MPL) es una licencia de código abierto y software libre utilizada por el navegador de Internet Mozilla y sus productos derivados. Cumple completamente con los postulados del open source y del software libre. Sin embargo, la MPL deja abierto el camino a una posible reutilización comercial y no libre del software, si el usuario así lo desea, sin restringir la reutilización del código ni el relicenciamiento bajo la misma licencia.

Aunque el uso principal de la MPL es servir como licencia de control para el navegador Mozilla y el software relacionado con él, esta licencia es ampliamente utilizada por desarrolladores y programadores que quieren liberar su código.

6.5.4.1. Ventajas de MPL

- MPL es una licencia de código abierto y software libre utilizada por desarrolladores y programadores para la liberación de código fuente.

6.5.4.2. Desventajas de MPL

- No se puede, legalmente, enlazar un módulo cubierto por la licencia GPL con un módulo cubierto por la licencia MPL.

6.5.5. Licencia NPL

La Licencia Pública de Netscape (inglés: Netscape Public License o NPL), es una licencia de software libre sin un copyleft fuerte. NPL está basada en la Licencia Pública de Mozilla revisada anteriormente.

NPL agrega una cláusula que permite a Netscape utilizar el código que un desarrollador agregue a un programa con bajo licencia NPL. Netscape podrá utilizar este código fuente sin importar si el desarrollador lo estableció de tipo privativo. En contrapartida, la licencia NPL no permite al desarrollador hacer uso del código fuente Netscape.

6.5.5.1. Ventajas de NPL

- NPL presenta como ventaja básica que se trata de una licencia de software libre.

6.5.5.2. Desventajas de NPL

- LPN no otorga derechos iguales a Netscape y al resto de los desarrolladores, ya que permite utilizar el código de Netscape sólo como se especifica en la licencia, pero Netscape puede utilizar los cambios realizados por los desarrolladores en cualquier forma posible, incluso en versiones del software bajo licencia propietaria.
- LPN no se esfuerza en asegurar que las modificaciones hechas por los usuarios queden disponibles como software libre.
- Aunque todas las modificaciones hechas por los desarrolladores se deben liberar bajo la licencia LPN, esto sólo aplica a las modificaciones realizadas al código existente, no a subrutinas añadidas colocadas en archivos diferentes.

6.5.6. Otras Licencias Reconocidas

6.5.6.1. Licencia de JAVA

Es una licencia particular con características muy especiales dirigidas a la protección del lenguaje de programación JAVA de SUN Microsystems. Pone énfasis en evitar que aparezcan extensiones incompatibles con el lenguaje JAVA.

6.5.6.2. Licencia de Distribución y Desarrollo Común (CDDL)

Recientemente publicada por SUN Microsystems, tiene por objeto liberar parte del software de esta compañía y que sea integrado con otras herramientas open source. Su objetivo principal apunta a permitir la integración de Java con las diferentes distribuciones de Linux del mercado. La licencia CDDL está basada en la MPL, y por tanto es muy similar en sus términos a la LGPL, permitiendo compartir el código utilizado por el programa con otros de diferente procedencia y guardarse para el autor la libertad de publicar o no los resultados. Permite la inclusión de cualquier otro tipo de código, sea cual sea la licencia del mismo, en la solución completa.

6.5.6.3. Licencia Creative Commons

Es una licencia de reciente creación, dirigida básicamente para trabajos multimedia. No permite la alteración del producto original, ni tampoco su comercialización. Sólo permite su reproducción tal cual, mencionando al autor.

6.5.7. Tabla Comparativa De Las Principales Licencias De Software

LICENCIA	DESCRIPCIÓN	COMPATIBLE GPL	CERTIFICADA OPEN SOURCE INITIATIVE
BSD Modificada	Simple, libre, abierta	Sí	Sí
BSD Original (BSD)	Permisiva, sin copyleft,, con cláusula de advertencia.	No	No
GNU Public (GPL)	Libre, abierta, con copyleft.	Sí	Sí
GNU Reducida (LGPL)	GPL sin copyleft, permite enlazar con módulos no libres.	Sí	Sí
Mozilla Public (MPL)	Libre, copyleft limitado, no enlazable con GPL.,	No	Sí
Netscape Public (NPL)	Como MPL pero puede usar código propietario.	No	No
Apache Software	Libre y abierta, con patentes.	No	Sí
Common Development and Distribution (CDDL)	Libre, sin copyleft, con patentes, con propiedad intelectual.	No	Sí
MIT/X Window	Libre, permisiva, copyleft limitado.	Sí	Sí
Eiffel Forum (EFL)	Libre y abierta (la versión 1 no es compatible con GPL).	v2	Sí
Expat	Libre, simple, permisiva y si copyleft (similar a la MIT).	Sí	Sí
IBM Public	Libre, con patentes.	No	Sí
Intel Open Software	Libre (ha dejado de usarse).	Sí	Sí
Perl	Licencia dual AL/GPL.	Sí	
PHP	Libre, sin copyleft (similar a BSD Original).	No	Sí
Python	Libre (compaible GPL).	Sí	Sí
Zope Public (ZPL)	Abierta, simple, copyleft reducido.	v2	Sí
W3C Software	Libre, compatible con GPL.	Sí	Sí
OpenLDAP	Libre, permisiva, sin copyleft.	v2.7	No

Tabla 8 Tabla de comparación de Licencias

CAPITULO 6: HERRAMIENTAS DE DESARROLLO

6.6. Herramientas de desarrollo

Entre las diferentes herramientas utilizadas para el desarrollo del proyecto tenemos:

6.6.1. Blender

Es un programa que se emplea para el modelamiento, creación y animación de gráficos tridimensionales, el cual es compatible con todas las versiones de Windows, Mac OS X, Linux, Solaris, FreeBSD e IRIX. Es del tipo software libre el cual se puede conseguir en su sitio web.

El programa posee un interfaz que es catalogada muy poco amigable lo cual para nuevos usuarios podría tomarse como un problema al momento de iniciarse en el uso de esta herramienta, pero a su vez posee sus ventajas ya que le permitirá una configuración personalizada de la distribución de los menús y vistas de cámara.

Blender tiene sus orígenes a partir del año de 1998 el principal autor, Ton Roosendaal, fundó la empresa "Not a Number Technologies" (NaN) en junio de 1998 para desarrollar y distribuir el programa.

El domingo 13 de octubre de 2002, Blender fue liberado al mundo bajo los términos de la Licencia Pública General de GNU (GPL). Actualmente Blender continúa en desarrollo a través de voluntarios procedentes de todas partes del mundo los cuales están liderados por el creador de Blender, Ton Roosendaal.

6.6.1.1. Características[5]

- Multiplataforma, libre, gratuito y con un tamaño de origen realmente pequeño comparado con otros paquetes de 3D, dependiendo del sistema operativo en el que se ejecuta.
- Capacidad para una gran variedad de primitivas geométricas, incluyendo curvas, mallas poligonales, vacíos, NURBS, metaballs.

[5] Wikipedia. 2010. Blender [en línea]. Características, [http://es.wikipedia.org/w/index.php?title=Blender],[Consulta:20 de Septiembre 2010]

- Junto a las herramientas de animación se incluyen cinemática inversa, deformaciones por armadura o cuadrícula, vértices de carga y partículas estáticas y dinámicas.
- Edición de audio y sincronización de video.
- Características interactivas para juegos como detección de colisiones, recreaciones dinámicas y lógica.
- Posibilidades de renderizado interno versátil e integración externa con potentes trazadores de rayos o "raytracer" libres como kerkythea, YafRay o Yafrid
- Lenguaje Python para automatizar o controlar varias tareas.

6.6.1.2. Herramientas

Blender cuenta con herramientas las cuales facilitan el trabajo del modelador entre las cuales tenemos las más importantes como son:

Cámara: Indica donde se encuentra situada la cámara y desde que ángulo se observa el objeto en el renderizado.

Cursor 3D: Indica donde aldrán los objetos que se crearon, inicialmente sale en el centro del plano 3d.

Lámpara: Es la que controla la iluminación de la imagen se puede convertir a su vez en múltiples opciones como son lámpara, sol, etc.

Blender a pesar de ser una herramienta nueva ha sido aceptada de la mejor manera por muchos animadores, además posee un sin número de ejemplos y documentación que se encuentra en internet la cual está realizada por usuarios que desean compartir sus experiencias con esta herramienta. Aunque la inicialización en Blender puede ser un poco compleja debido a su complicada interfaz de usuario, el animador no debe darse por vencido ya que una vez acostumbrado le será muy fácil realizar sus trabajos ya que es una potente aplicación y como anteriormente se dijo posee gran cantidad de información conjuntamente con ejemplos que pueden ser aprovechados de la mejor manera. Algunos de los sitios donde puede encontrar información acerca de Blender

son: La comunidad de Blender en español cuya dirección es http://www.g-blender.org/ y la página de la fundación Blender en inglés http://www.blender.org/.

6.6.2. Mplayer

Es un reproductor multimedia multiplataforma liberado bajo la licencia GPL. Reproduce la mayoría de los archivos MPEG, VOB, AVI, OGG/OGM, MKV, VIVO, ASF/WMA/WMV, QT/MOV/MP4, FLI, RM, NuppelVideo, YUV4MPEG, FILM, RoQ, PVA, soportados por algunos códecs nativos, XAnim, y DLL's Win32. Además puede reproducir VideoCD, SVCD, DVD, 3ivx y DivX 3/4/5.

El reproductor puede funcionar en la mayoría de las plataformas, incluyendo Linux, derivados de Unix, Mac OS X, Syllable, Pegasos y también en Windows. Hay también derivados de DOS y FreeDOS y en la consola de juegos Wii (v. Mplayer CE).

La utilización de mplayer como reproductor se debe a que su implementación con java resulta mucho más sencilla de realizar y además reproduce la mayoría de los formatos de audio conocidos.

6.6.3. Firma digital de un Applet

Un applet es un programa Java que, se ejecuta dentro de un navegador Web, los cuales pueden ser Internet Explorer, Firefox, Safari, etc... En algunos de los casos es necesario que un applet necesite escribir en el disco local algún tipo de información, lo cual por las restricciones de la JVM no es posible ya que de esta manera evita ataques maliciosos puesto que el applet se ejecuta en un entorno controlado.

Para que un applet pueda ejecutarse es necesario que este firmado por una entidad de confianza para el cliente, lo cual se puede hacer usando un certificado, el cual lo puede proporcionar una entidad reconocida mundialmente (como VeriSign), o también generar un certificado firmado por nosotros mismos lo cual suele servir dentro de una intranet o de prueba, pero si queremos usarlo en internet es recomendado que lo firme una entidad certificadora con autoridad o reconocimiento mundial.

6.6.3.1. Como firmar un Applet

Para firmar un applet en java se utiliza las utilidades:

- keytool: la usaremos para generar los certificados.
- jarsigner: la usaremos para firmar el applet con el certificado que hemos generado.

En internet se pueden encontrar múltiples ejemplos de cómo aplicar estas utilidades, lo cual será de gran ayuda al momento de querer generar nuestros propios certificados.

6.6.4. XML: Persistencia de Datos en java

Existen diferentes tipos de persistencia, sin embargo, en la plataforma java uno de los más utilizados es la serialización por la cual se entiende que un objeto puede transformarse en un conjunto de bytes, el mismo que se podrá reconstruir cuando se desee, esto permitirá guardarlo en un archivo o mandarlo por la red.

El estándar XML es un formato de texto simple y muy flexible, fue diseñado para describir datos y hoy en día es de suma importancia para el intercambio de datos a través de la web.

En el proyecto utilizamos las clases XMLEncoder y XMLDecoder para serializar y reconstruir objetos java, lo cual se logra a partir de versiones superiores a la 1.4 la cual solo permitía realizarlo a través de la interfaz serializable usando el formato binario.

XMLEncoder trabaja reproduciendo el grafo del objeto y registrando los pasos que fueron necesarios para crear la copia. Así este posee una copia que XMLDecoder tomaría para descifrar el archivo.

Java ofrece la posibilidad de la clase java.beans.XMLEncoder para realizar la persistencia de objetos como documentos XML, la cual se encarga de convertir el objeto con todos sus datos a un formato XML. Entre las ventajas que posee tenemos:

- Soporta cambios en las versiones lo que permite que sean intercambiables entre entornos que tienen versiones diferentes de alguna clase o incluso máquinas virtuales distintas.
- Es sumamente compacto ya que XMLEncoder utiliza un algoritmo de eliminación de redundancia.

- Tolerante a fallas ya que los errores no estructurales permanecen localizados lo que permite que el proceso continúe con las partes del documento que no se ven afectadas por el error.

6.6.5. Movimiento Rectilíneo Uniforme

Cuando la velocidad es constante, es decir recorre en tiempos iguales espacios iguales.

6.6.5.1. Ecuaciones del movimiento[6]

La trayectoria de una partícula es rectilínea cuando su aceleración es nula (sin serlo la velocidad) o cuando su aceleración no tiene componente normal a la velocidad. El movimiento rectilíneo es, pues, un caso particular del movimiento general en el espacio, pero debido a la abundancia de problemas y situaciones en que lo encontraremos, le dedicaremos una atención especial. Puesto que los vectores V y A están dirigidos a lo largo de la trayectoria, será conveniente escoger el origen O sobre ella de modo que el vector de posición 1ª también estará situado sobre ella. Entonces, al ser paralelos entre sí todos los vectores que nos describen el movimiento de la partícula podemos prescindir de la notación vectorial. Si tomamos el eje *x* en la dirección de la trayectoria y especificamos un cierto sentido como positivo, las ecuaciones de definición de la velocidad y de la aceleración se reducen a la componente *x*, o sea

$$V = \frac{dx}{dt} \qquad a = \frac{dv}{dt} = \frac{d^2x}{dt^2}$$

de modo que, si conocemos x=x(t) podemos obtener la velocidad y la aceleración de la partícula, i.e., v=v(t) y a=a(t) , mediante dos derivaciones sucesivas. En algunos casos conoceremos a=a (t) y, entonces, por integración (y conociendo las condiciones iniciales V_0 y X_0) podemos obtener v=v (t) y x=x (t).

Podemos encontrar otra relación cinemática importante aplicando a la definición de la aceleración la regla de derivación de una función de función. Así, obtenemos la expresión:

[6] Wikipedia, 2010. Movimiento Rectilíneo,[En línea], Ecuaciones del Movimiento, [http://es.wikipedia.org/wiki/Movimiento_rectil%C3%ADneo], [Consulta: 15 de mayo del 2010]

$$a = \frac{dv}{dt} = \frac{dv}{dx}\frac{dx}{dt} = v\frac{dv}{dx}$$

que nos resultará de gran utilidad cuando conozcamos a=a(x) o v=v(x).

Conocemos	Se aplica	Se Obtiene	Es decir
$a = a(t)$	$dv = a.dt$	$v = v_0 + \int a.dt$	$v = v(t)$
$v = v(t)$	$dx = v.dt$	$x = x_0 + \int v.dt$	$x = x(t)$
$a = a(x)$	$v.dv = a.dx$	$v^2 = v_0^2 + 2\int a.dx$	$v = v(x)$
$v = v(x)$	$dt = dx/v$	$t = t_0 + \int dx/v$	$t = t(x)$
$a = a(v)$	$dx = v.dv/a$	$x = x_0 + \int v.dv/a$	$x = x(v)$
	$dt = dv/a$	$t = t_0 + \int dv/a$	$t = t(v)$

Figura 34 Expresiones para el movimiento rectilíneo uniforme

7. EVALUACION DEL OBJETO DE INVESTIGACION

El presente trabajo investigativo denominado **"Construcción de una Herramienta Multimedia para el desarrollo de diapositivas tridimensionales bajo licencia GPL, haciendo uso de la tecnología Java 3D, orientada al sector académico y profesional del Área de La Energía las Industrias y los Recursos Naturales No Renovables"**, dio como resultado final la Herramienta Multimedia P3D, misma que fue concebida con el afán de mejorar la calidad del aprendizaje, imponiéndose así a las limitaciones existentes con el software dedicado a la elaboración de presentaciones de diapositivas que existe en el mercado ya que en algunos casos es necesario visualizar objetos del mundo real como son, piezas mecánicas, partes de computadores, aparatos, animales, etc., de manera que el expositor pueda interactuar con ellos para así lograr una mejor explicación del tema.

El objetivo general así como cada uno de los objetivos específicos lograron ser abarcados en su totalidad, gracias a la aplicación ordenada de metodologías orientadas al desarrollo de la investigación y al desarrollo de software:

- Desarrollar una galería básica de objetos tridimensionales del mundo real.

  Para llevar a cabo el cumplimiento exitoso del objetivo la herramienta empleada para el modelado fue Blender la cual que es sumamente potente para el diseño además es gratuita y multiplataforma. Puesto que el modelado de toda una galería llevaría gran cantidad de tiempo también se optó por la opción de obtener objetos prediseñados desde la web, ya que existen varios sitios que permiten descargar modelos de forma gratuita y en diferentes formatos.
- Desarrollar un asistente de software configurable para la creación de diapositivas tridimensionales.

  La creación del asistente de software cumplió con las fases necesarias que hicieron que su desarrollo concluya de la mejor manera posible, para el cual en su diseño se planteó que inicie siempre al arrancar la aplicación permitiéndole elegir de tres opciones posibles que son: presentación vacía, a partir de una plantilla y abrir una presentación existente. Con esto se pretende brindar una ayuda al usuario y así agilitar la creación de proyectos.
- Desarrollar un algoritmo que permita visualizar múltiples objetos en una misma diapositiva.

Para el correcto desarrollo del algoritmo se utilizó la ecuación de la circunferencia la cual permitió ordenar en forma circular los objetos agregados, distribuyéndose uniformemente alrededor de 360 grados.

- Desarrollar un componente de software configurable para los efectos tridimensionales.

  Java 3D trae consigo clases que permiten trabajar con efectos, pero para un mejor control se optó por desarrollar un componente propio para cumplir este objetivo, tomando en cuenta los efectos de posición, rotación, escala, transparencia y color, para lo cual se aplicó conceptos de física y geometría dando como resultado un módulo estable capaz de brindar al usuario diferentes opciones de configuración y combinación de los efectos programados.

- Integrar un componente de software para el sonido.

  Este objetivo se pudo ejecutar exitosamente puesto que se empleó el componente prediseñado mplayer debido a que acepta una gama amplia de formatos para su reproducción y su integración con java resulta más sencilla comparada con otros componentes; además tomándose en cuenta que la aplicación es multiplataforma se debía integrar un componente compatible para los dos sistemas operativos, quedando al desarrollador la tarea de realizar la configuración necesaria para que funcione de manera óptima con el programa.

- Desarrollar un árbol de navegación para los objetos tridimensionales agregados a la diapositiva.

  Este objetivo se cumplió con éxito ya que se desarrolló una clase extendida de la DefaultTreeNode, agregando un campo más de modo que permita identificar con exactitud cada uno de los componentes integrados en la presentación.

- Implantación y liberación del software.

  Para cumplir este objetivo se creó una cuenta en sourceforgen.net, la cual brinda la posibilidad de subir proyectos que pueden ser descargados por cualquier persona o desarrollador, entre los archivos de descarga se encuentran la aplicación, código fuente, manuales y la ayuda. Para la descarga se puede ingresar al sitio www.sourceforge.net y colocar en el buscador el nombre del proyecto P3Dimensiones.

8. DESARROLLO DE LA PROPUESTA ALTERNATIVA

8.1. DETERMINACION DE REQUERIMIENTOS

8.1.1. Determinación de requerimientos funcionales

CODIGO	DESCRIPCION	CATEGORIA
RF001	El sistema permitirá al usuario crear, modificar y eliminar diapositivas.	Evidente
RF002	El sistema permitirá al usuario crear, y modificar las presentaciones.	Evidente
RF003	El sistema permitirá al usuario agregar nuevos objetos 3D a la galería prediseñada, los mismos que pueden ser en formato .3ds o .obj.	Evidente
RF004	El sistema permitirá al usuario eliminar objetos de la galería prediseñada.	Evidente
RF005	El sistema permitirá exportar las presentaciones en extensiones .png y scorm.	Evidente
RF006	El sistema permitirá al usuario configurar el asistente de software	Evidente
RF007	El sistema permitirá al usuario agregar, configurar y eliminar efectos a los objetos.	Evidente
RF008	El sistema permitirá al usuario agregar, configurar y eliminar efectos al texto.	Evidente
RF009	El sistema permitirá al usuario agregar, configurar y eliminar los efectos a las formas y el componente carrusel.	Evidente
RF010	El sistema permitirá al usuario agregar y eliminar el sonido en los formatos más conocidos como es .mp3, .WAV, .wma a las diapositivas.	Evidente
RF011	El sistema permitirá al usuario agregar, modificar y eliminar material a los componentes gráficos.	Evidente
RF012	El sistema permitirá al usuario generar una vista previa de la presentación.	Evidente
RF013	El sistema permitirá al usuario agregar, modificar y eliminar texto en la diapositiva.	Evidente
RF014	El sistema permitirá al usuario manipular los objetos en las presentaciones.	Evidente
RF015	El sistema permitirá al usuario modificar los parámetros de configuración de los objetos en las diapositivas	Evidente
RF016	El sistema permitirá agregar, modificar y eliminar formas, objetos 3D a las diapositivas de la presentación.	Evidente
RF017	El sistema permitirá exportar diapositivas en formato png y gif.	Evidente
RF018	El sistema permitirá imprimir la presentación realizada.	Evidente
RF019	El sistema permitirá pre visualizar el efecto agregado.	Evidente

Tabla 9 Requerimientos Funcionales

8.1.2. Determinación de requerimientos no funcionales

CODIGO	DESCRIPCION
RNF001	El programa contara con una interfaz gráfica amigable con un menú de navegación.
RNF002	El programa deberá ser desarrollado en la plataforma java.
RNF003	El programa será multiplataforma

Tabla 10 Requerimientos no Funcionales

8.2. MODELO DEL DOMINIO

8.2.1. Glosario de términos

TÉRMINO	DESCRIPCIÓN
Presentación	Conjunto de diapositivas destinada a la proyección.
Diapositiva	Las diapositivas son cada uno de los elementos que constituyen la presentación y cada una de ellas podría identificarse con una lámina o página.
Animación	Efecto visual que se configura a cada componente gráfico.
Componente Gráfico	Formas de líneas, objetos, palabras.
Formas	Moldes, figuras geométricas.
Carrusel	Componente que distribuye un número de objetos en forma circular.
Cilindro	Solido limitado por una superficie cilíndrica cerrada y dos planos que forman su base.
Cubo	Poliedro regular limitado por seis cuadrados iguales.
Cono	Solido limitado por una superficie cónica.
Esfera	Lugar geométrico de los puntos del espacio que equidistan de otro interior llamado centro.
Plano	Figura de forma cuadrada de un solo lado.
Punto	Término genérico que designa los elementos de cualquier espacio geométrico.
Imagen	Figura, representación, semejanza y apariencia de una cosa.
Texto3D	Conjunto de letras que poseen características de profundidad y proyección.

Tabla 11 Glosario de términos

8.2.2. Modelo conceptual del dominio

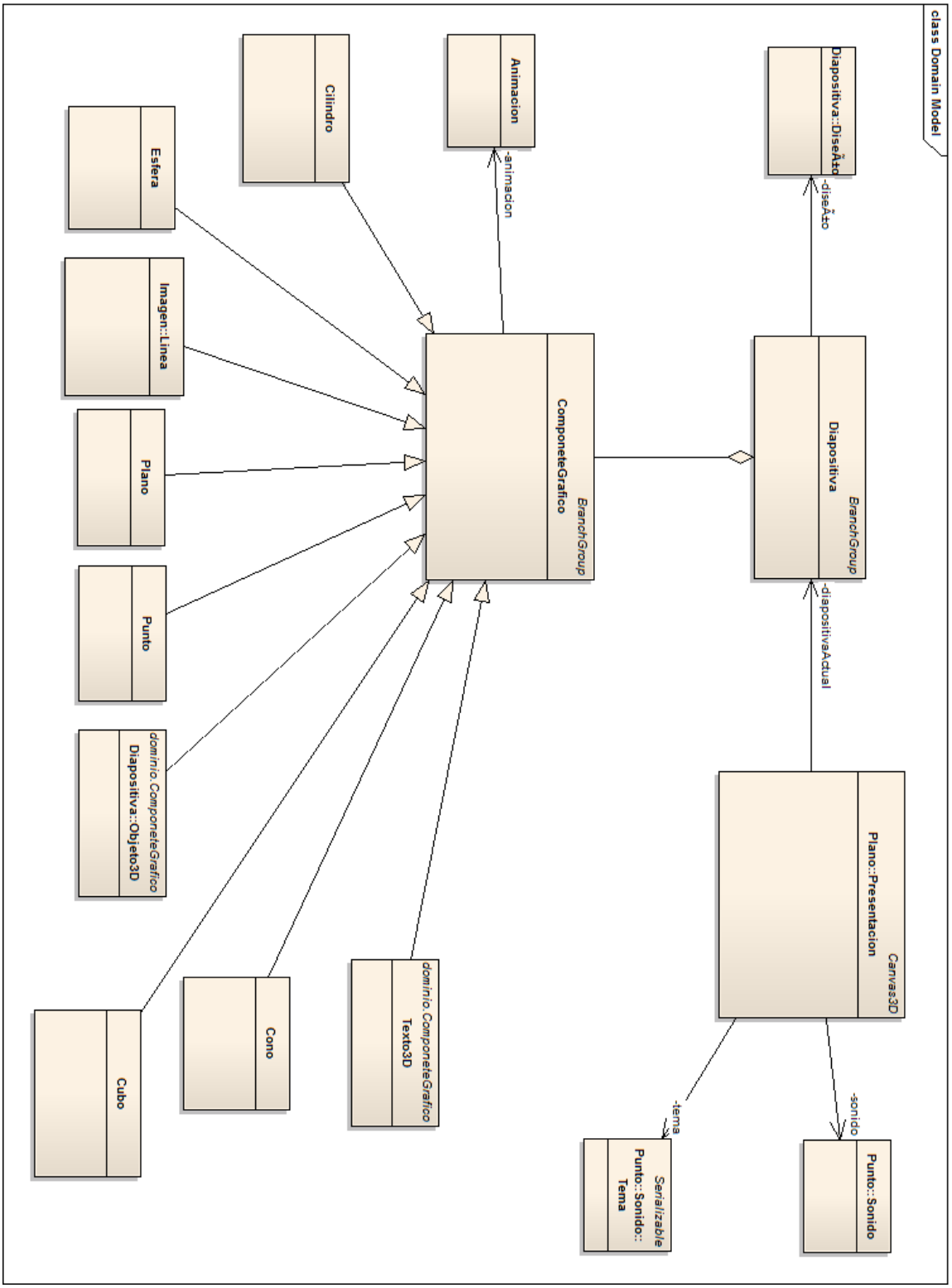


Figura 35 Modelo inicial del dominio

8.3. MODELADO DE CASOS DE USO

8.3.1. Determinación de casos de uso

Actor	Meta	Caso de uso
Usuario	Crear nuevas presentaciones.	Crear presentación vacía
	Crear nuevas presentaciones.	Crear presentación de plantilla
	Realizar cambios en presentaciones previamente creadas.	Modificar presentación
	Guardar las presentaciones creadas.	Guardar presentación
	Guardar en formato de imagen png la presentación creada.	Exportar presentación en formato png
	Guardar en formato scorm la presentación creada.	Exportar presentación en formato scorm
	Fijar un fondo predeterminado cada vez que se cree una diapositiva.	Seleccionar fondo para la presentación
	Realizar una previsualización de la presentación creada.	Generar vista previa
	Generar un documento físico de la presentación con las diapositivas que la conforman.	Imprimir presentación
	Crear diapositiva en blanco	Agregar diapositiva en blanco
	Crear una diapositiva a partir de un diseño establecido.	Agregar diapositiva con diseño
	Fijar un color de fondo a la diapositiva seleccionada.	Agregar color de fondo
	Fijar una imagen de fondo a la diapositiva seleccionada.	Agregar imagen de fondo
	Remover diapositivas innecesarias de la presentación.	Eliminar diapositiva
	Generar una imagen en formato png o gif a partir de la diapositiva creada.	Exportar diapositiva en formato png o gif
	Previsualizar los efectos agregados a los diferentes componentes dentro de una diapositiva.	Generar vista previa efecto
	Insertar un componente gráfico a la diapositiva.	Agregar componente grafico
	Realizar cambios en los diferentes componentes gráficos agregados en las diapositivas.	Modificar componente grafico
	Remover el componente gráfico de la diapositiva.	Eliminar componente grafico

	Agregar nuevos fondos, texturas y objetos 3D a la galería.	Importar recursos
	Remover de la galería los recursos no deseados.	Eliminar recursos
	Incorporar sonido a una diapositiva.	Agregar sonido
	Remover el sonido de la diapositiva seleccionada.	Eliminar sonido
	Agregar diferentes clases de material al componente gráfico.	Agregar material
	Realizar cambios en la configuración del material.	Modificar configuración del material
	Remover el material configurado del componente gráfico.	Eliminar material
	Fijar animaciones a los componentes gráficos.	Agregar efecto a componente grafico
	Realizar cambios en las animaciones agregadas.	Modificar configuración del efecto
	Remover la animación o efecto del componente gráfico.	Eliminar efecto de componente grafico
	Fijar una textura al componente gráfico desde un directorio externo de la galería.	Agregar textura desde archivo
	Fijar una textura al componente gráfico desde galería prediseñada.	Agregar textura desde galería
	Realizar cambios en la textura agregada.	Configurar textura
	Remover la textura del componente gráfico.	Eliminar textura

Tabla 12 Determinación de Requerimientos

8.3.2. Diagramas de casos de uso

Caso de uso: Administrar Presentación.

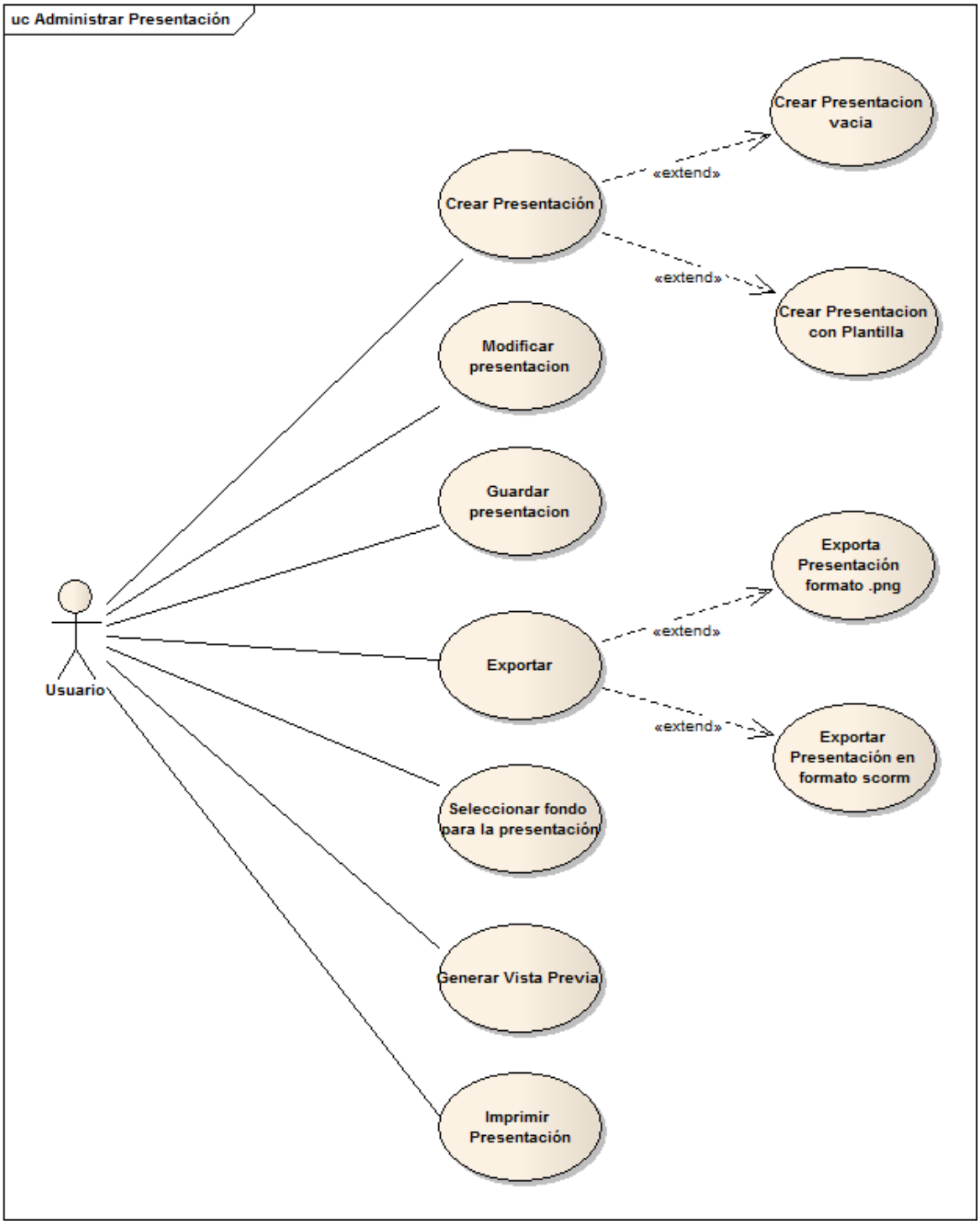


Figura 36 Diagrama del Caso de Uso Administrar Presentación

Caso de uso: Administrar Diapositiva.

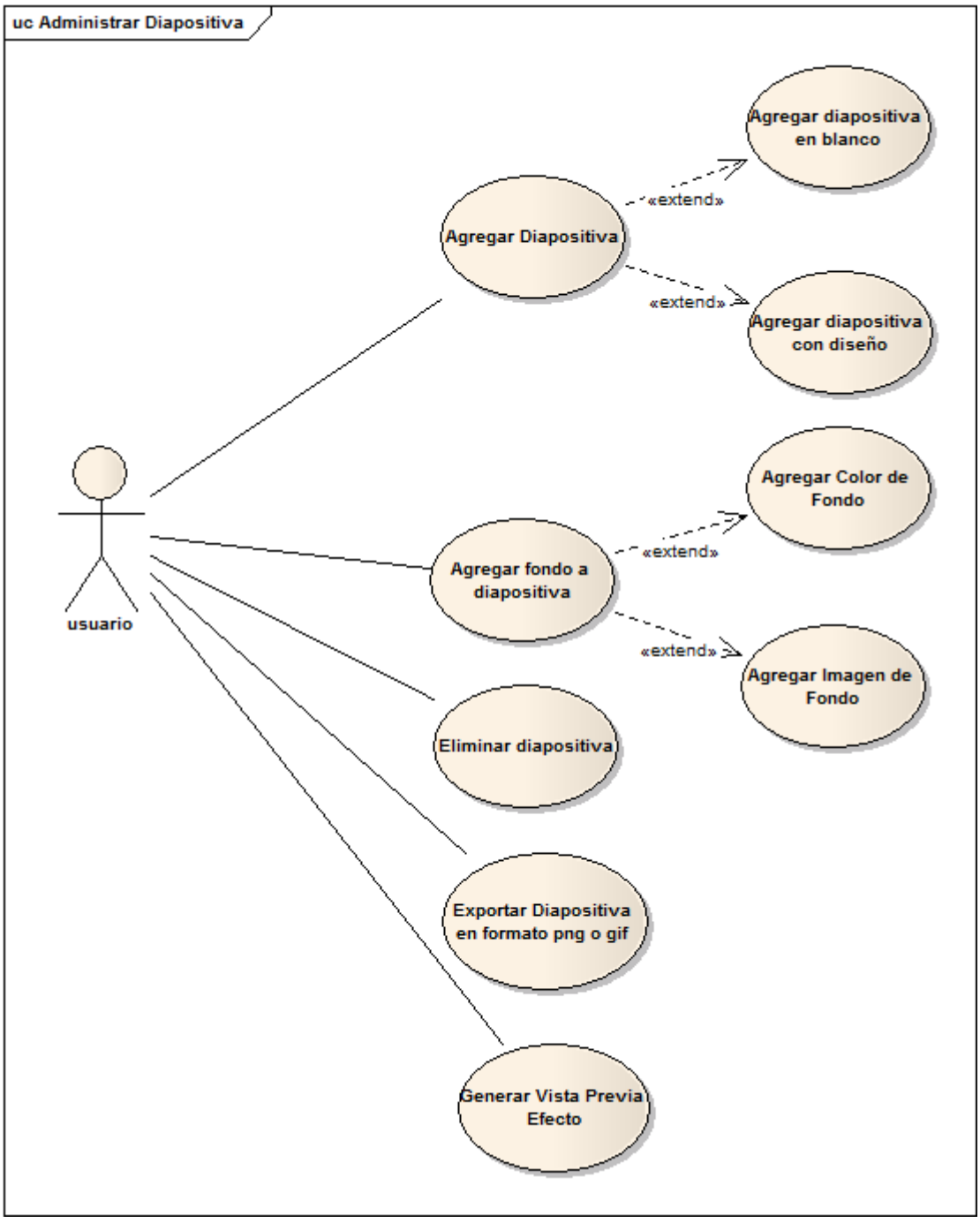


Figura 37 Diagrama del Caso de Uso Administrar Diapositiva

Caso de uso: Administrar Componentes Gráficos.

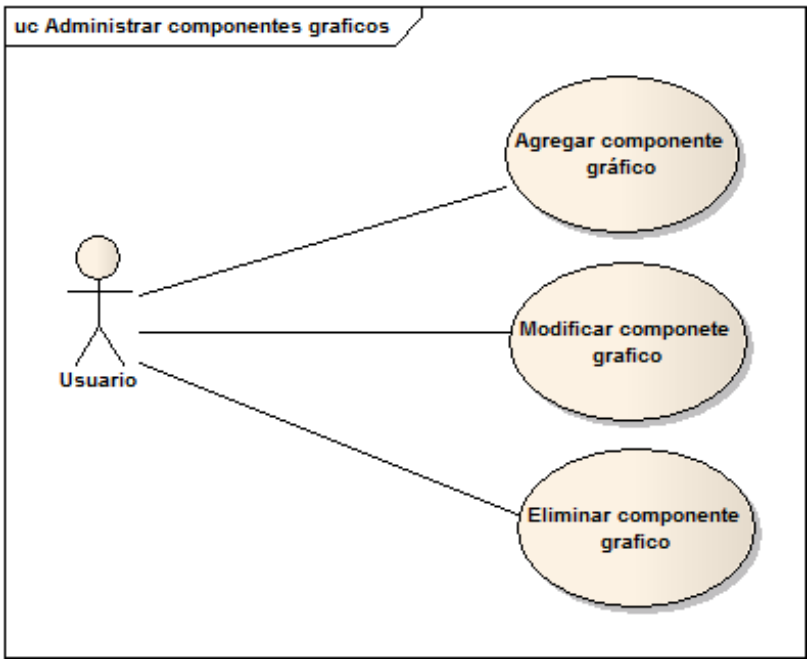


Figura 38 Diagrama del Caso de Uso Administrar Componentes Gráficos

Caso de uso: Administrar Galería de Recursos.

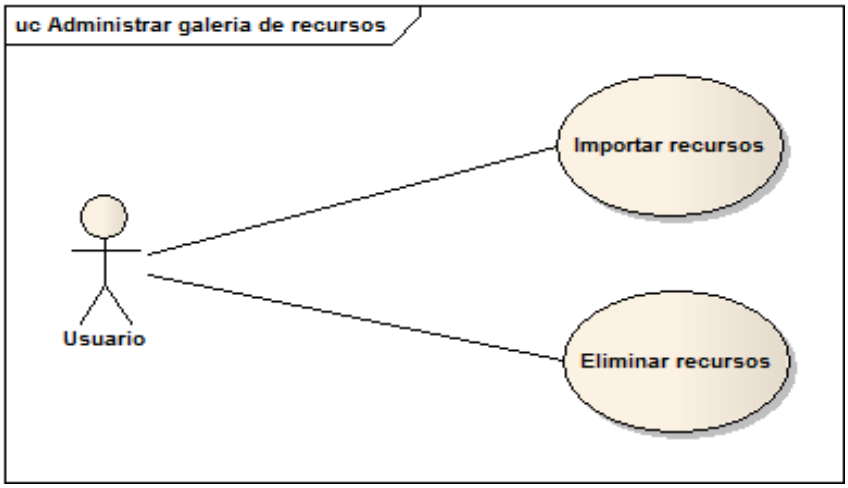


Figura 39 Diagrama del Caso de Uso Administrar Galería de Recursos

Caso de uso: Administrar Sonido.

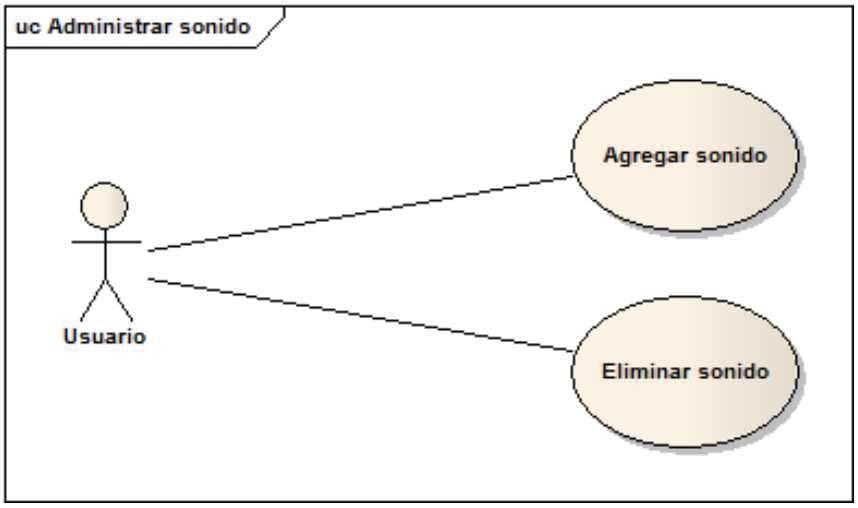


Figura 40 Diagrama del Caso de Uso Administrar Sonido

Caso de uso: Administrar Material.

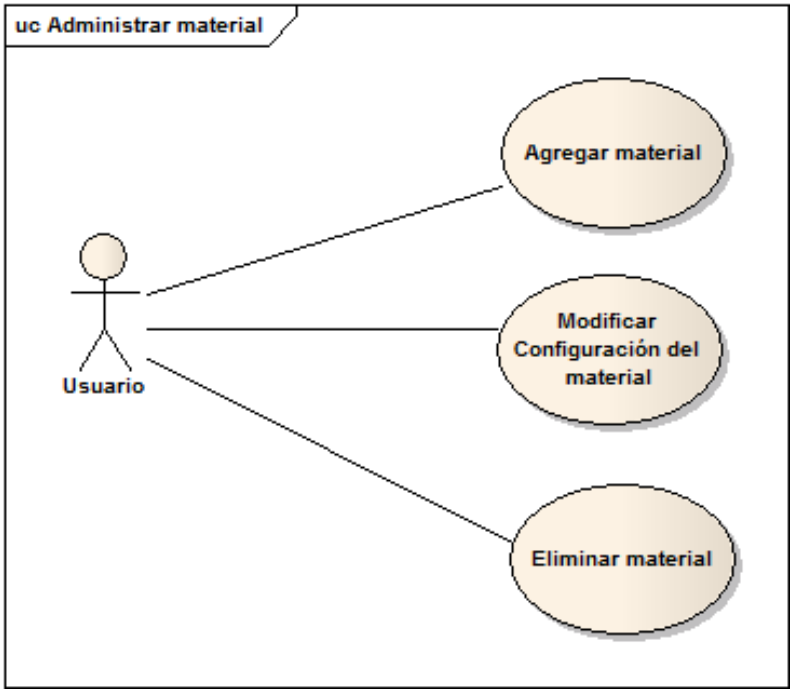


Figura 41 Diagrama del Caso de Uso Administrar Material

Caso de uso: Administrar Efectos Visuales.

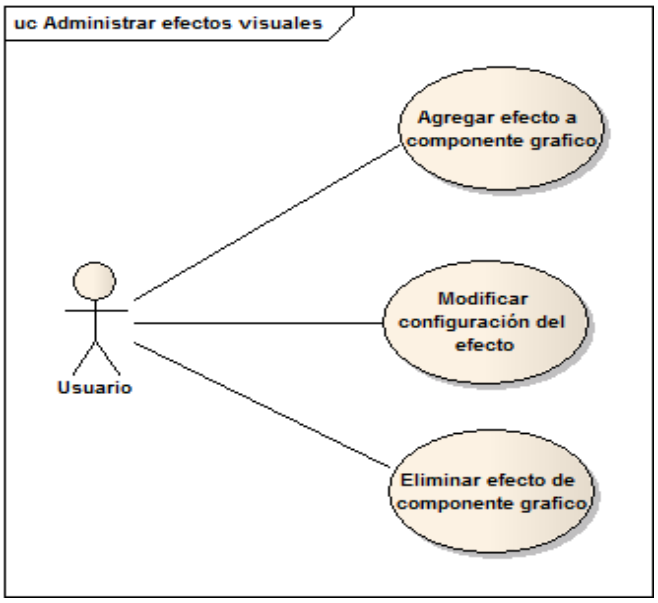


Figura 42 Diagrama del Caso de Uso Administrar Efectos Visuales

Caso de uso: Administrar Texturas.

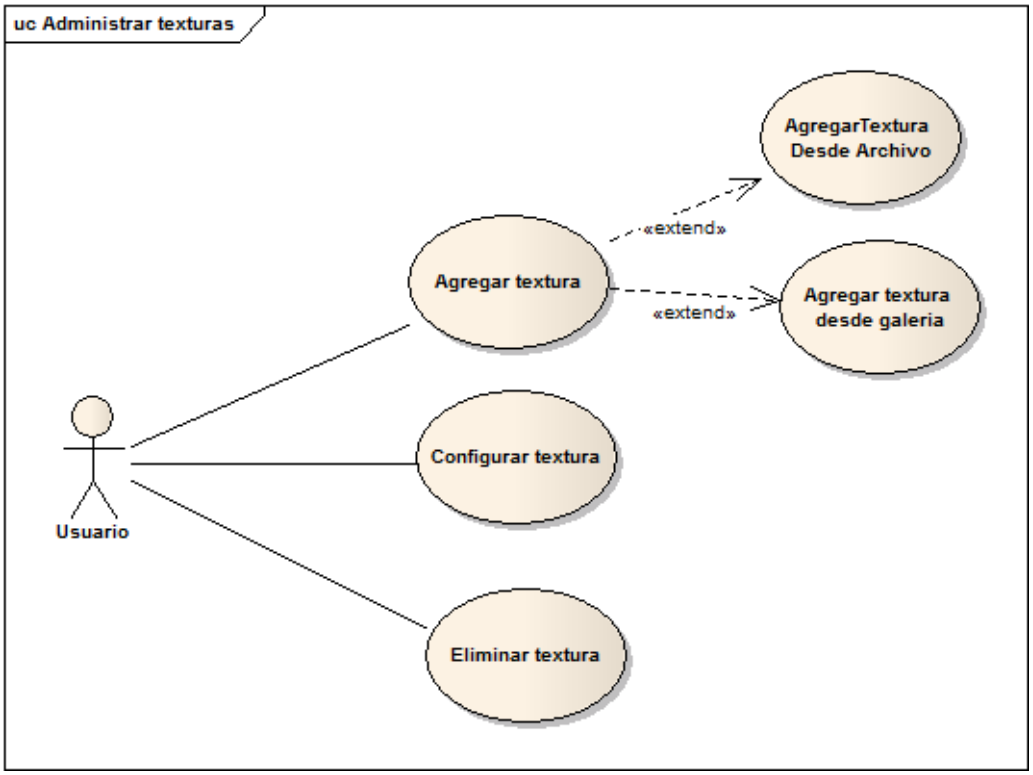


Figura 43 Diagrama del Caso de Uso Administrar Textura

8.3.3. PROTOTIPADO DE PANTALLAS

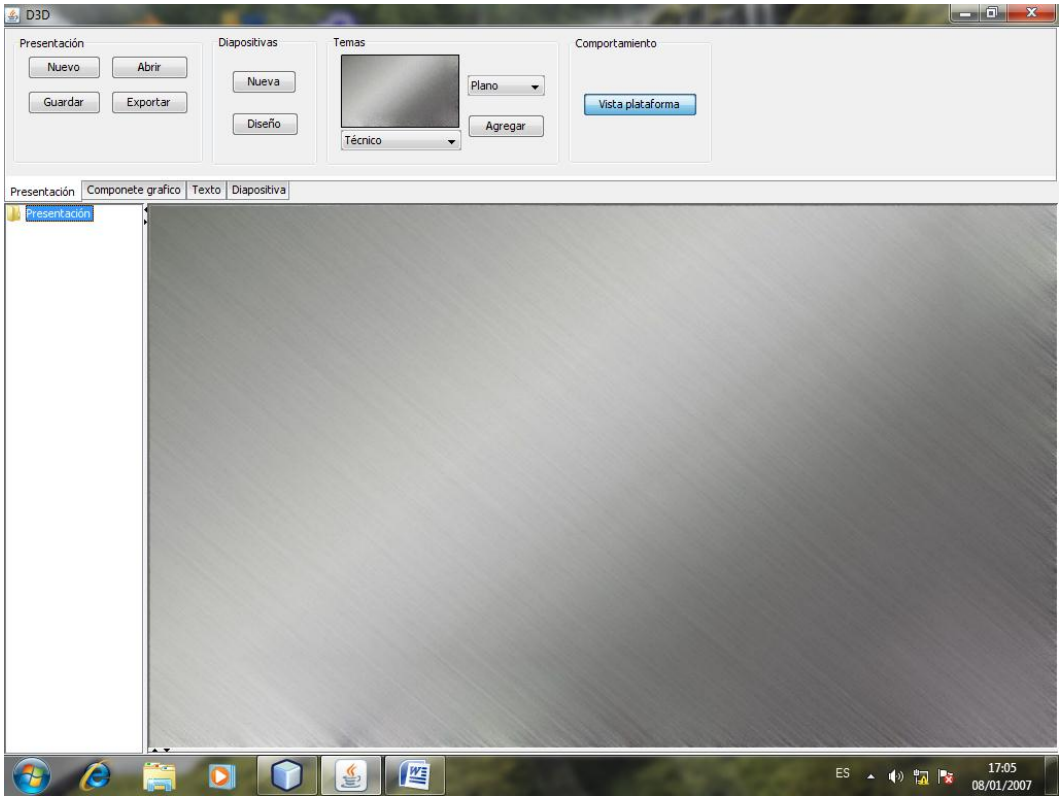
Nombre de la Pantalla: D3D-Presentación	**Código:** PP001
Tipo de Interfaz gráfica: JFrame	
Caso de Uso: Administrar Presentación	
	
Realizado por: Klever Daniel Macas Flores, Alfredo Rolando Macas Chocho.	Fecha: 15 de Enero de 2010

Tabla 13 Prototipo de la Pantalla – Administrar Presentación

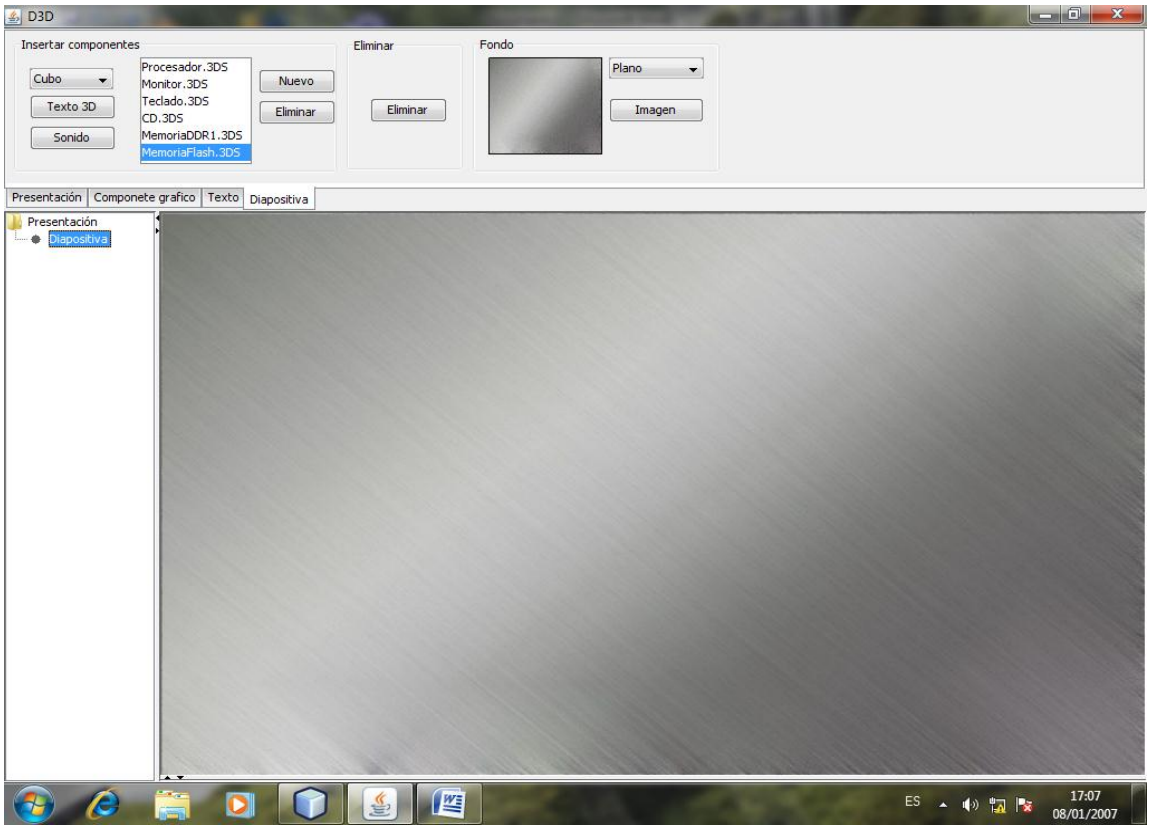
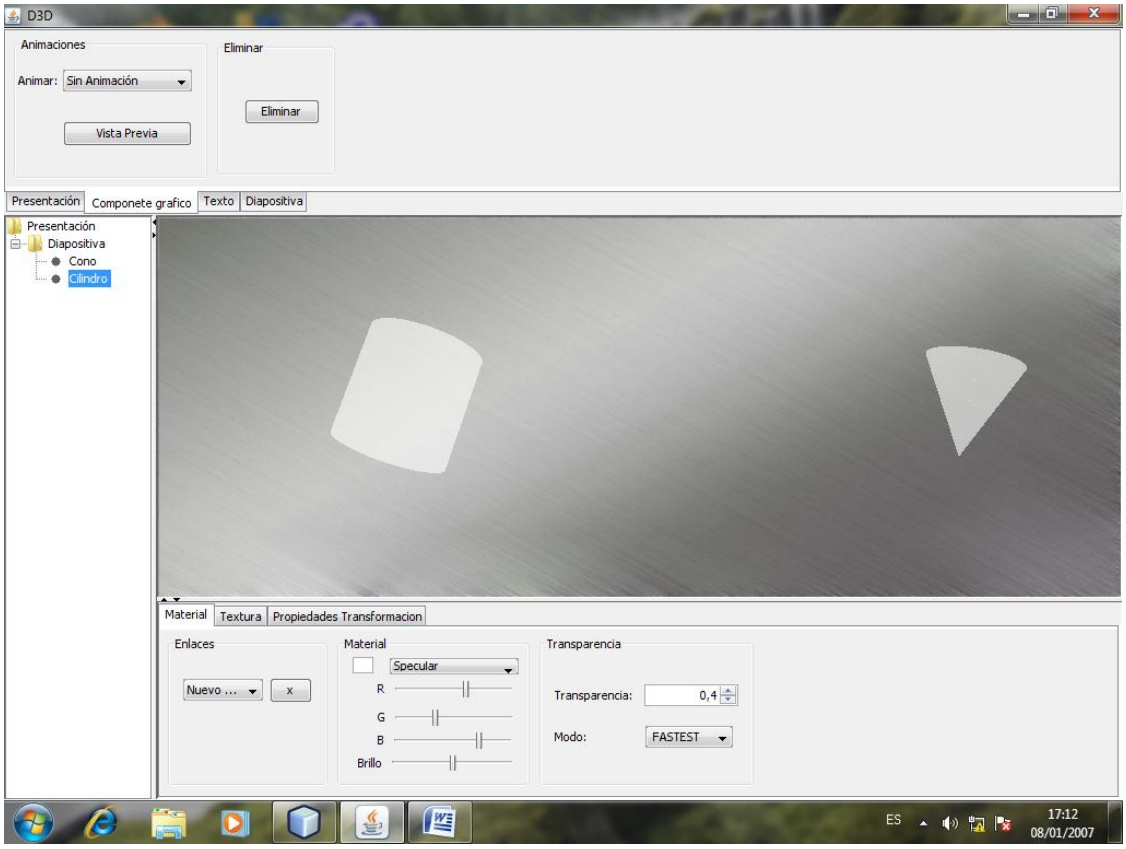
Nombre de la Pantalla: D3D- Diapositiva	**Código:** PP002
Tipo de Interfaz gráfica: JFrame	
Caso de Uso: Administrar Diapositiva	
	
Realizado por: Klever Daniel Macas Flores, Alfredo Rolando Macas Chocho.	Fecha: 15 de Enero de 2010

Tabla 14 Prototipo de la Pantalla - Administrar Diapositiva

Nombre de la Pantalla: Componentes Gráficos	**Código:** PP003
Tipo de Interfaz gráfica: JFrame	
Caso de Uso: Administrar Componentes Gráficos	



Realizado por: Klever Daniel Macas Flores, Alfredo Rolando Macas Chocho.	Fecha: 15 de Enero de 2010

Tabla 15 Prototipo de la Pantalla – Administrar Componentes Gráficos

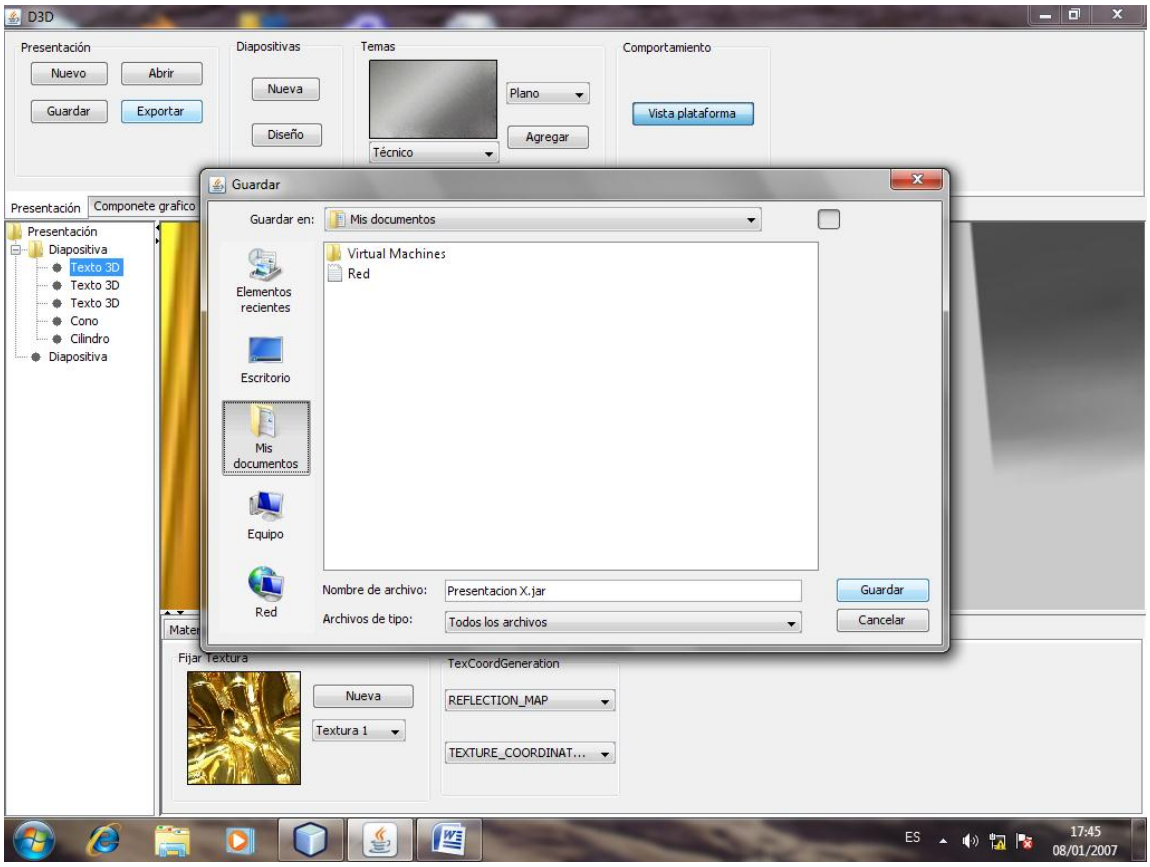
Nombre de la Pantalla: Exportar Presentación	**Código:** PP004
Tipo de Interfaz gráfica: JDialog	
Caso de Uso: Administrar Presentación	
	
Realizado por: Klever Daniel Macas Flores, Alfredo Rolando Macas Chocho.	Fecha: 15 de Enero de 2010

Tabla 16 Prototipo de la Pantalla – Exportar Presentación

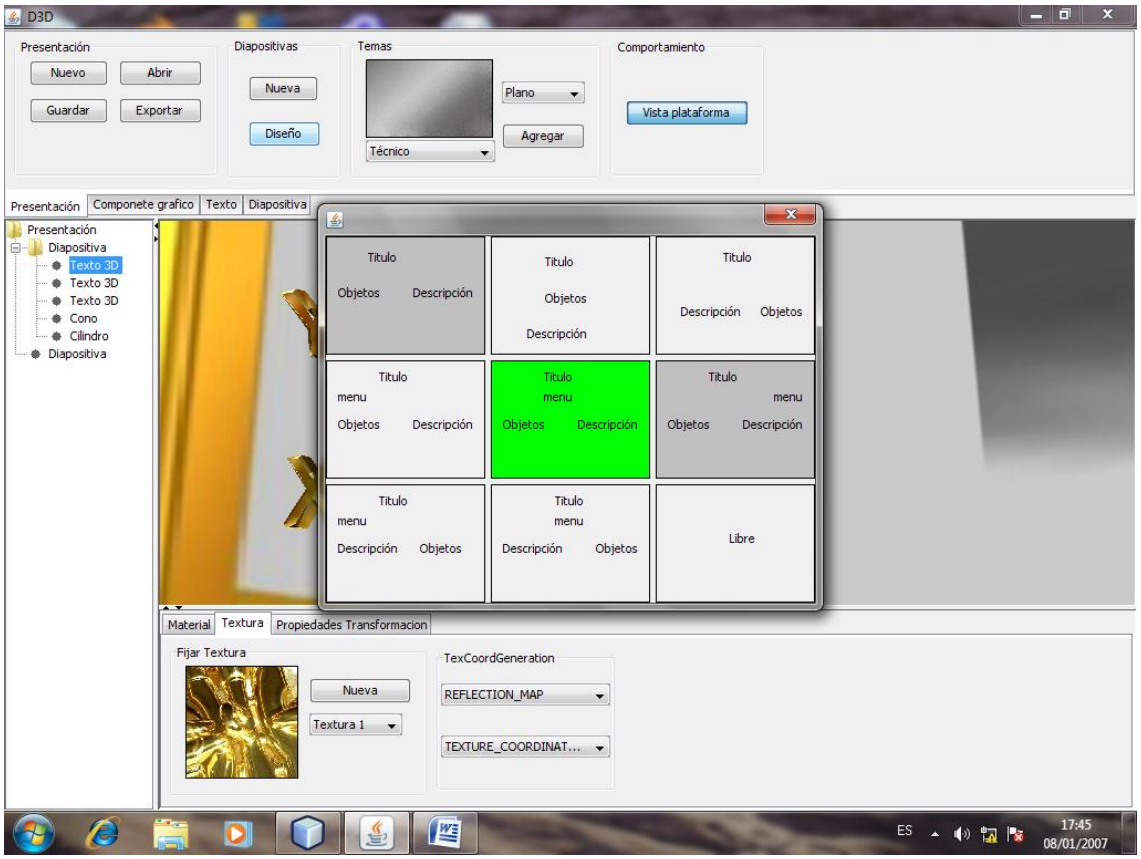
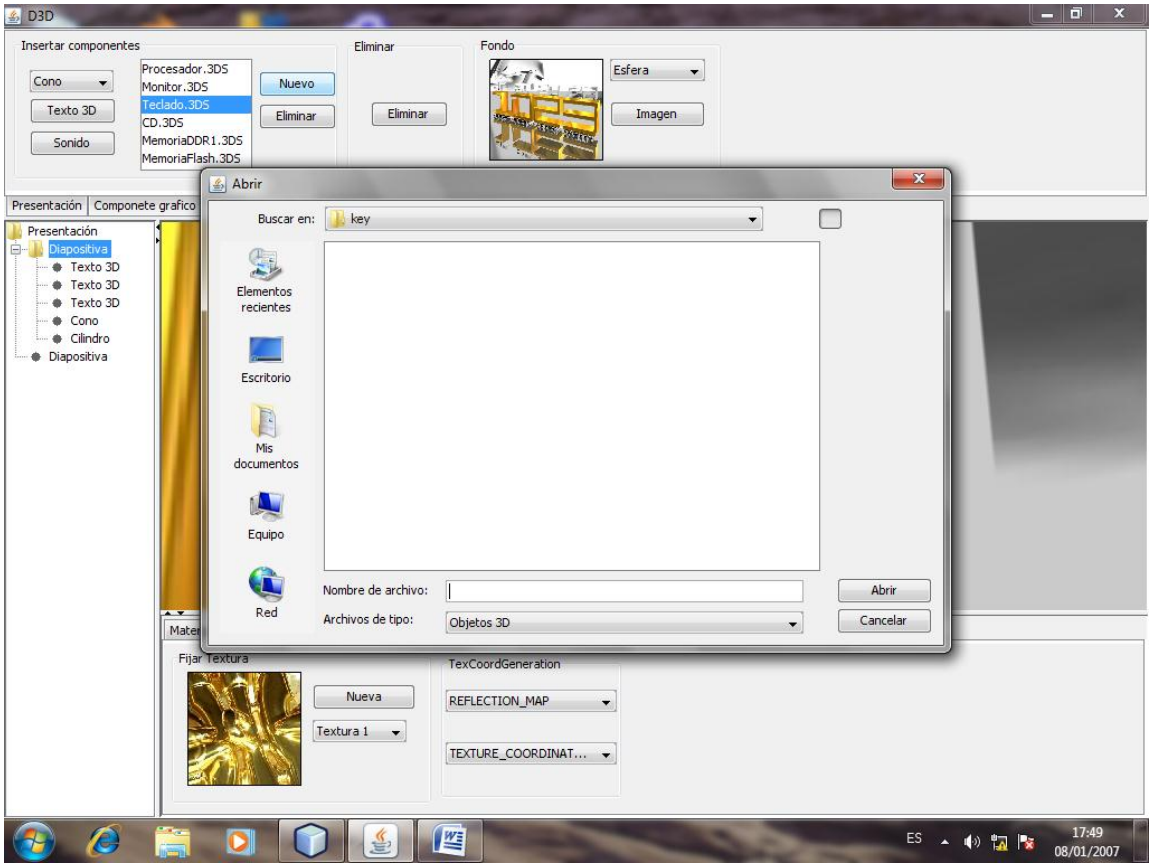
Nombre de la Pantalla: Agregar diapositiva con diseño	**Código:** PP005
Tipo de Interfaz gráfica: JFrame	
Caso de Uso: Administrar Diapositiva	
	
Realizado por: Klever Daniel Macas Flores, Alfredo Rolando Macas Chocho.	Fecha: 15 de Enero de 2010

Tabla 17 Prototipo de la Pantalla – Crear Presentación con diseño

Nombre de la Pantalla: Importar Recursos	**Código:**PP006
Tipo de Interfaz gráfica: JDialog	
Caso de Uso: Administrar Recursos	



Realizado por: Klever Daniel Macas Flores, Alfredo Rolando Macas Chocho.	Fecha: 15 de Enero de 2010

Tabla 18 Prototipo de la Pantalla – Importar Recursos

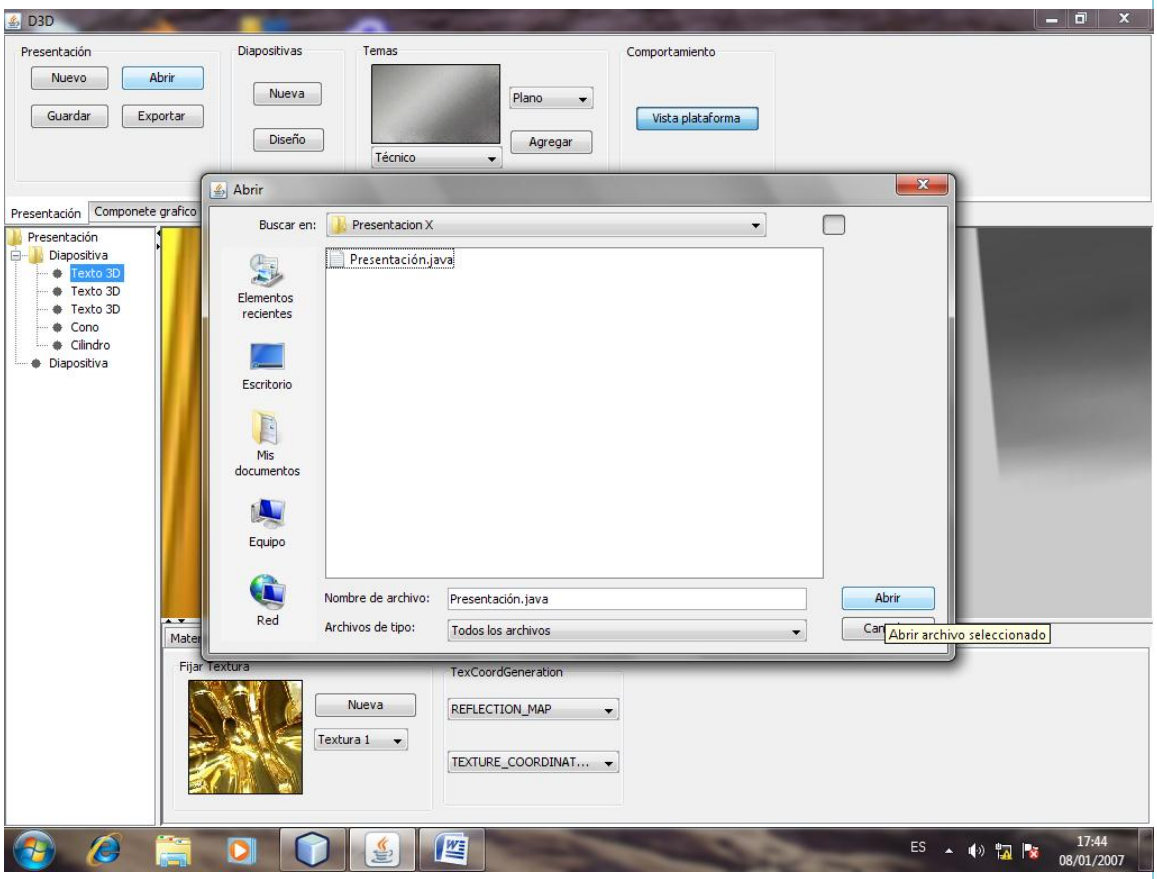
Nombre de la Pantalla: Abrir presentación	**Código:** PP007
Tipo de Interfaz gráfica: JDialog	
Caso de Uso: Administrar Presentación	
	
Realizado por: Klever Daniel Macas Flores, Alfredo Rolando Macas Chocho.	Fecha: 15 de Enero de 2010

Tabla 19 Prototipo de la Pantalla – Abrir Presentación

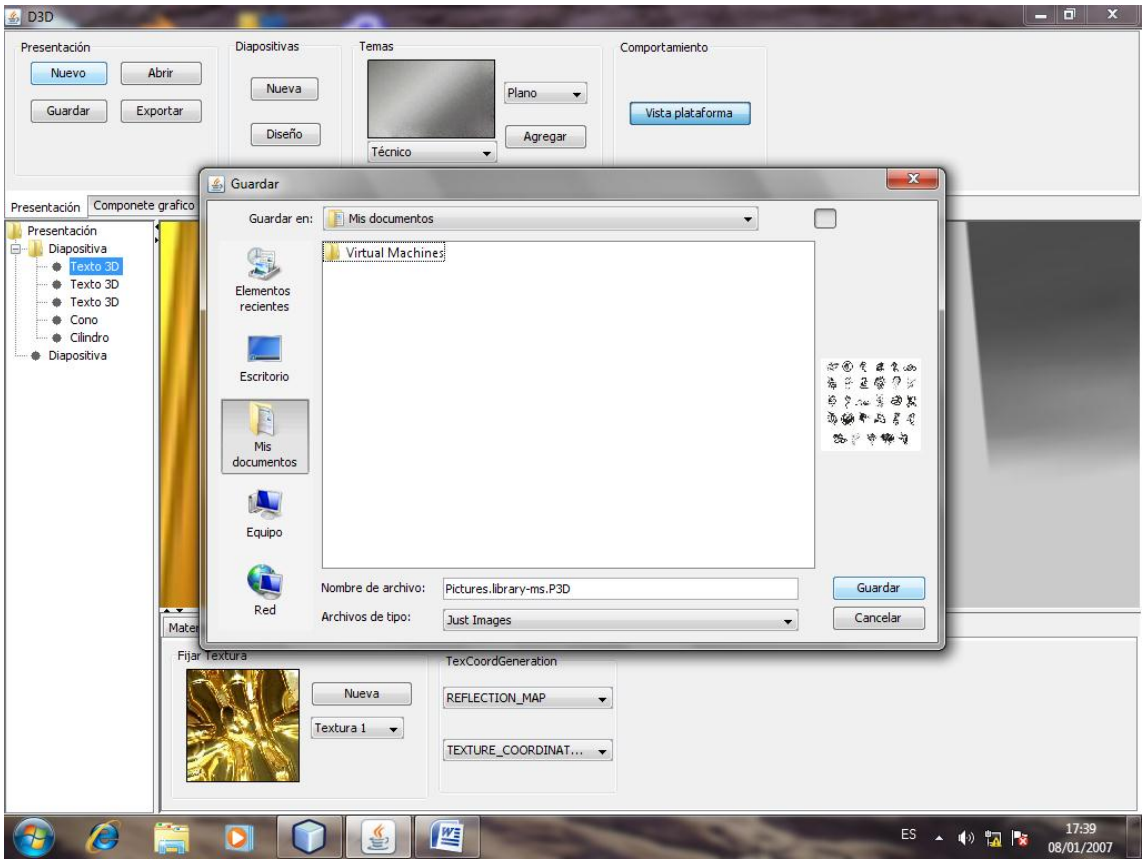
Nombre de la Pantalla: Guardar Presentación	**Código:** PP008
Tipo de Interfaz gráfica: JDialog	
Caso de Uso: Administrar Presentación	
	
Realizado por: Klever Daniel Macas Flores, Alfredo Rolando Macas Chocho.	Fecha: 15 de Enero de 2010

Tabla 20 Prototipo de la Pantalla – Guardar Presentación

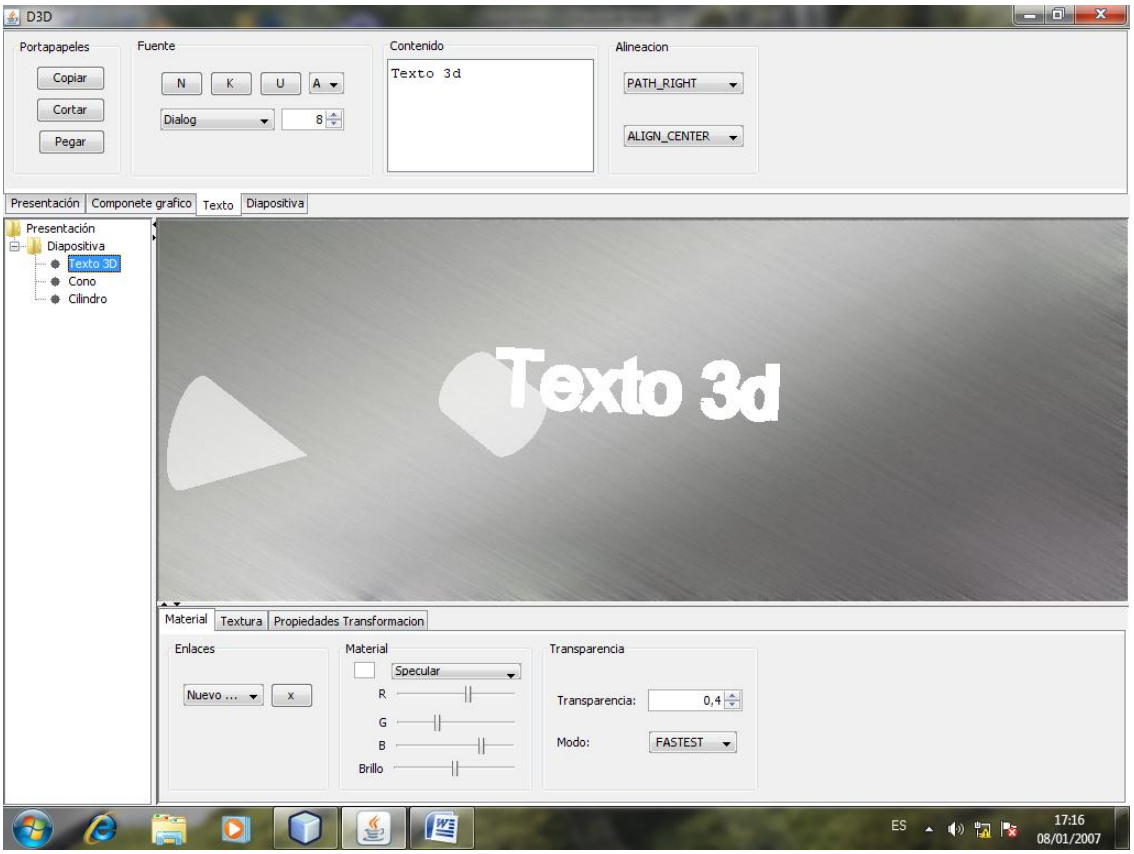
Nombre de la Pantalla: Agregar Componentes	**Código:** PP009
Tipo de Interfaz gráfica: JFrame	
Caso de Uso: Administrar Componentes Gráficos	
	
Realizado por: Klever Daniel Macas Flores, Alfredo Rolando Macas Chocho.	Fecha: 15 de Enero de 2010

Tabla 21 Prototipo de la Pantalla – Agregar Componentes

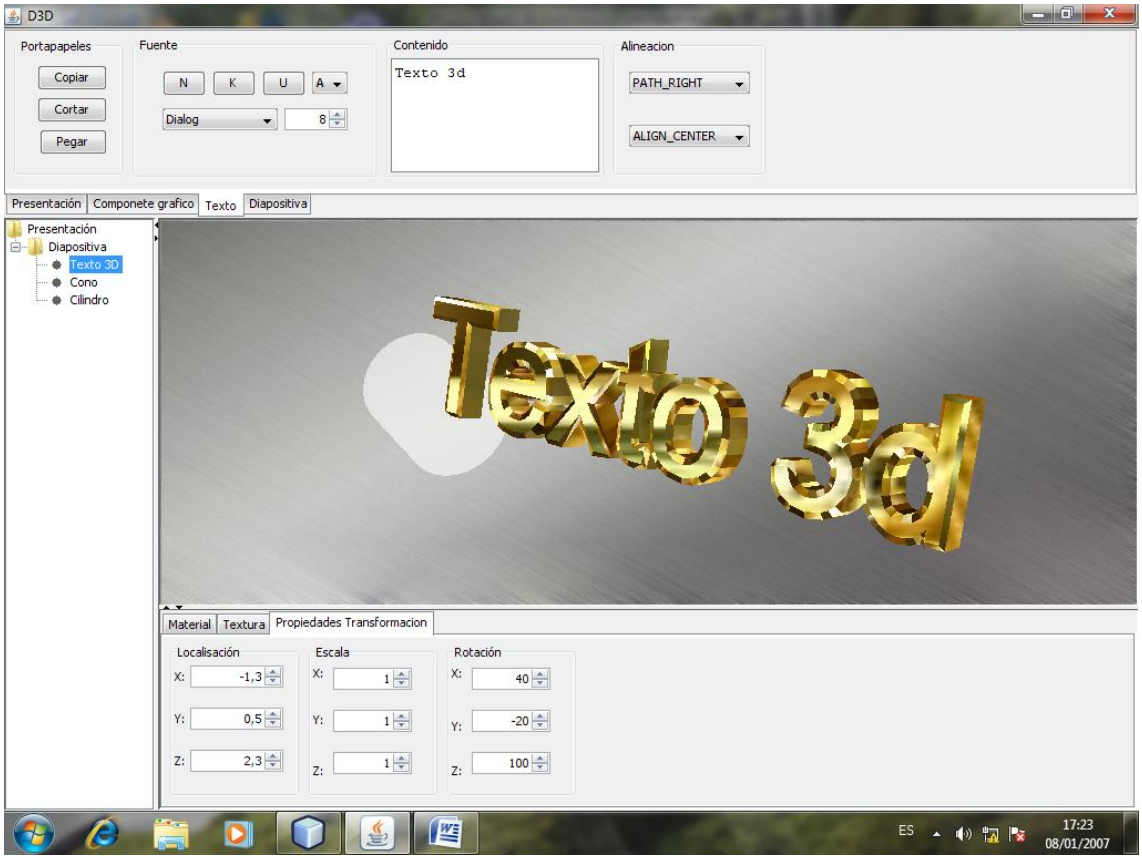
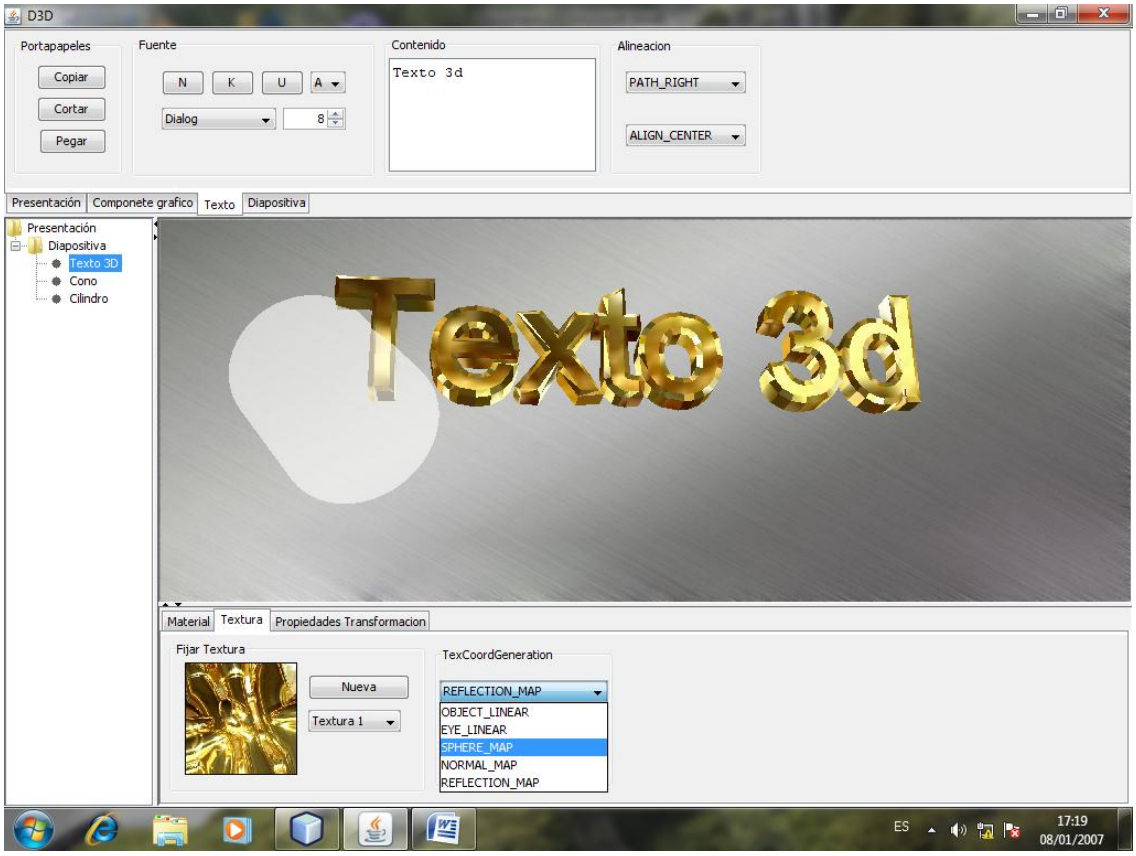
Nombre de la Pantalla: Configurar Componente	**Código:** PP010
Tipo de Interfaz gráfica: JFrame	
Caso de Uso: Administrar Componentes Gráficos	
	
Realizado por: Klever Daniel Macas Flores, Alfredo Rolando Macas Chocho.	Fecha: 15 de Enero de 2010

Tabla 22 Prototipo de la Pantalla – Configurar Componente

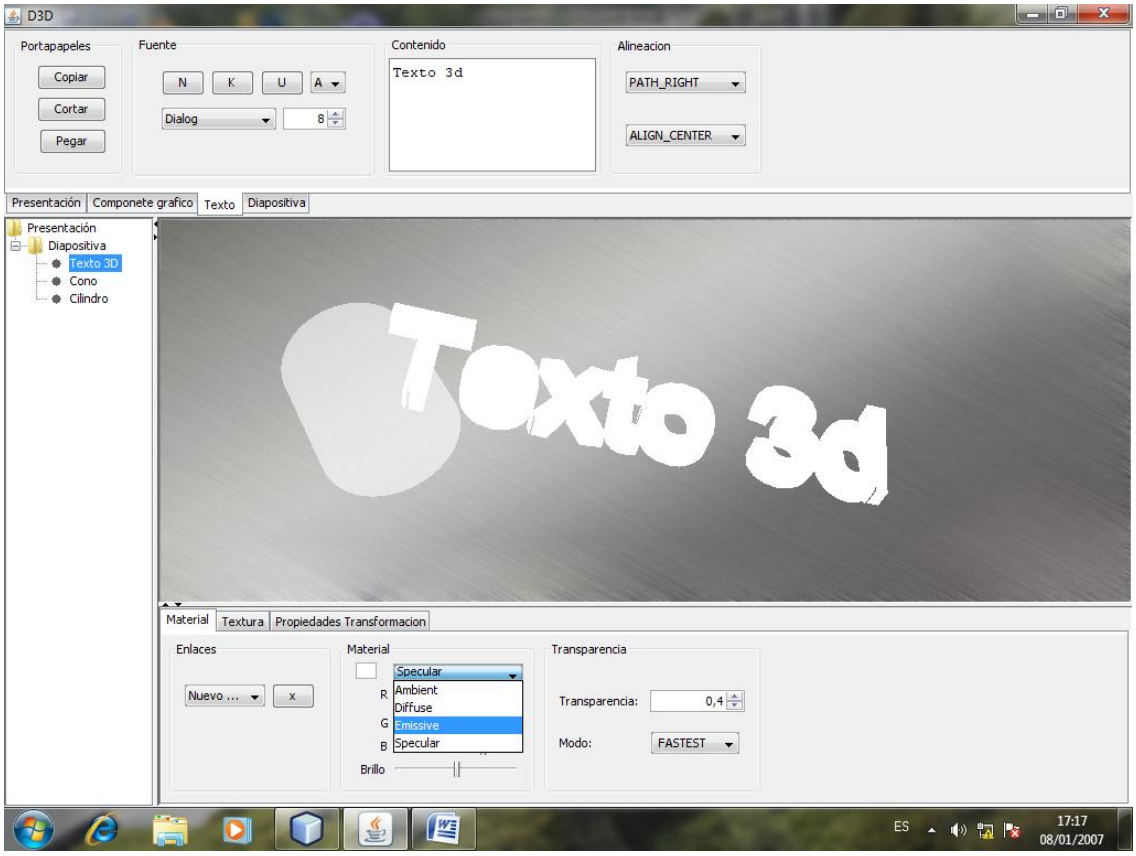
Nombre de la Pantalla: Agregar Texturas	**Código:** PP011
Tipo de Interfaz gráfica: JFrame	
Caso de Uso: Administrar Texturas	



Realizado por: Klever Daniel Macas Flores, Alfredo Rolando Macas Chocho.	Fecha: 15 de Enero de 2010

Tabla 23 Prototipo de la Pantalla – Agregar Textura

Nombre de la Pantalla: Agregar Material	**Código:** PP012
Tipo de Interfaz gráfica: JFrame	
Caso de Uso: Administrar Material	



Realizado por: Klever Daniel Macas Flores, Alfredo Rolando Macas Chocho.	**Fecha:** 15 de Enero de 2010

Tabla 24 Prototipo de la Pantalla – Agregar Material

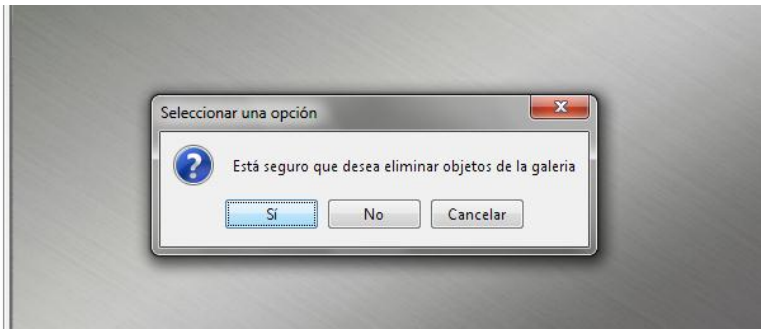
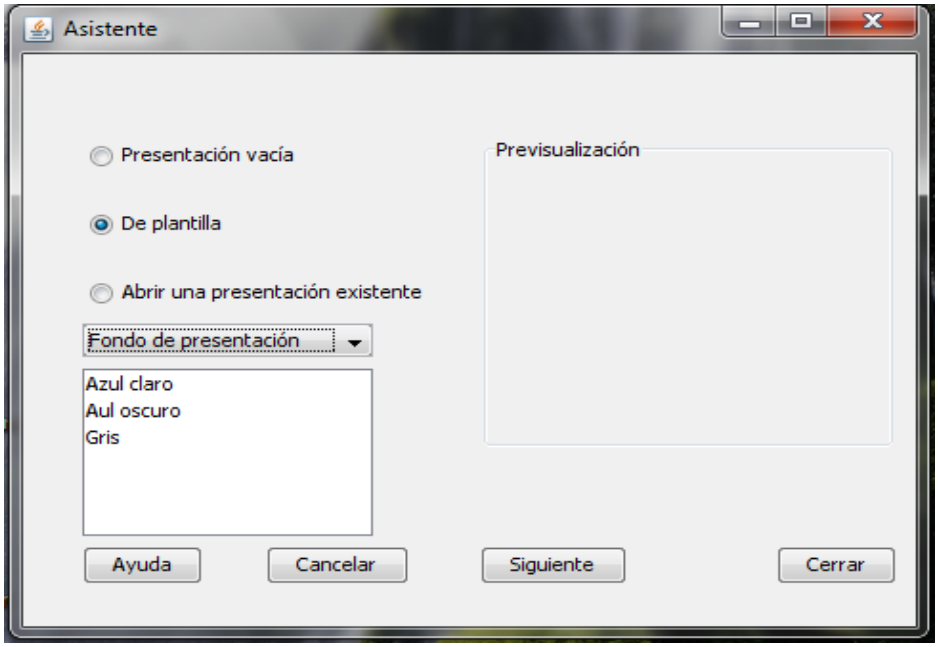
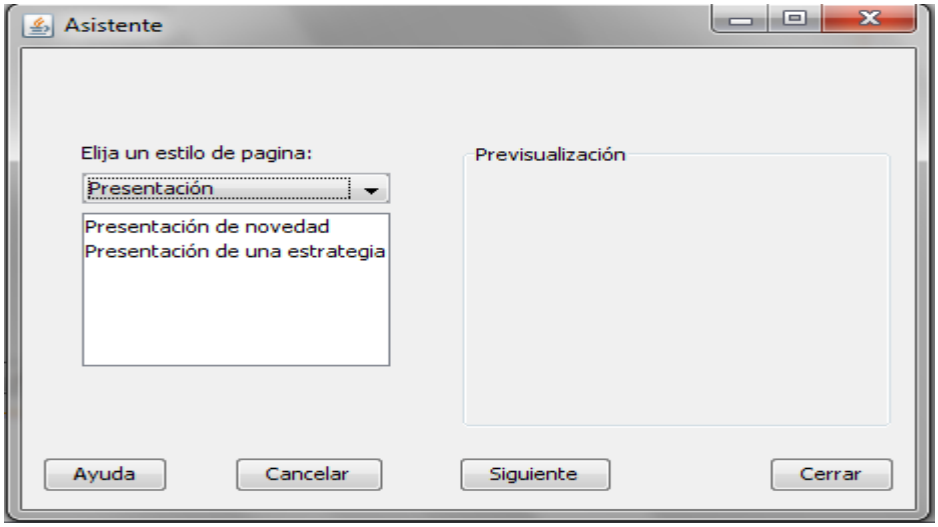
Nombre de la Pantalla: Eliminar Recurso de galería	**Código:**PP013
Tipo de Interfaz gráfica: JOptionPane	
Caso de Uso: Administrar Recursos	
	
Realizado por: Klever Daniel Macas Flores, Alfredo Rolando Macas Chocho.	**Fecha:** 15 de Enero de 2010

Tabla 25 Prototipo de la Pantalla – Dialogo Eliminar Recursos

Nombre de la Pantalla: Asistente	**Código:** PP014
Tipo de Interfaz gráfica: JDialog	
Caso de Uso: Administrar Presentación	





Realizado por: Klever Daniel Macas Flores, Alfredo Rolando Macas Chocho.	**Fecha:** 15 de Enero de 2010

Tabla 26 Prototipo de la Pantalla – Asistente de Software

8.4. DESCRIPCION DE CASOS DE USO

8.4.1. Caso de uso Crear presentación vacía.

<table>
<tr><td colspan="3" align="right">CU-001</td></tr>
<tr><td>Caso de Uso:</td><td colspan="2">CU-001 Crear presentación vacía.</td></tr>
<tr><td>Actores:</td><td colspan="2">Usuario(U), Software(S)</td></tr>
<tr><td>Resumen:</td><td colspan="2">El usuario del sistema podrá crear nuevas presentaciones de forma rápida y sencilla.</td></tr>
<tr><td>Referencias Cruzadas:</td><td colspan="2">RF-002</td></tr>
<tr><td colspan="3">Precondiciones: El asistente de software haya iniciado.</td></tr>
<tr><td colspan="3">Postcondiciones: Se cree la nueva presentación.</td></tr>
<tr><td colspan="3">Curso normal de eventos</td></tr>
<tr><td colspan="2">Acción del actor</td><td>Respuesta del sistema</td></tr>
<tr><td colspan="2">1. El usuario selecciona de asistente de software la opción presentación vacía.
2. El usuario presiona el botón siguiente.</td><td>3. El sistema presenta la pantalla para escribir el nombre del proyecto.</td></tr>
<tr><td colspan="2">4. El usuario escribe el nombre de la presentación y presiona el botón aceptar.</td><td>5. El sistema presenta ventana para seleccionar el destino para guardar.</td></tr>
<tr><td colspan="2">6. El usuario selecciona el destino.
7. El usuario presiona el botón guardar.</td><td>8. El sistema crea la presentación, presenta la pantalla principal y carga en el árbol de navegación la presentación con la configuración seleccionada.</td></tr>
<tr><td colspan="2">9. El caso de uso finaliza.</td><td></td></tr>
<tr><td colspan="3"></td></tr>
<tr><td colspan="3">Curso alterno de eventos</td></tr>
<tr><td colspan="3"></td></tr>
<tr><td colspan="3">2A. El usuario selecciona el botón cancelar.
3A. El sistema presenta la pantalla principal.</td></tr>
<tr><td colspan="3"></td></tr>
<tr><td colspan="3">2B. El usuario presiona el botón ayuda.
3B. El sistema presenta la ayuda de la aplicación.</td></tr>
</table>

Tabla 27 Descripción del caso de uso Crear Presentación vacía

8.4.2. Caso de uso Crear presentación con plantilla.

CU-002	
Caso de Uso:	**CU-002 Crear presentación con plantilla.**
Actores:	Usuario(U), Software(S)
Resumen:	El usuario del sistema podrá crear nuevas presentaciones de forma rápida y sencilla.
Referencias Cruzadas:	RF-002
Precondiciones: El asistente de software haya iniciado.	
Postcondiciones: Se cree la nueva presentación.	
Curso normal de eventos	
Acción del actor	**Respuesta del sistema**
1. El usuario selecciona de asistente de software la opción de plantilla.	2. El sistema carga las opciones de plantillas existentes ya sea fondo o presentación.
3. El usuario selecciona la plantilla.	4. El sistema presenta la vista previa de la diapositiva.
5. El usuario presiona el botón siguiente.	6. El sistema presenta la pantalla para escribir el nombre del proyecto.
7. El usuario escribe el nombre de la presentación. 8. El usuario presiona el botón aceptar.	9. El sistema presenta ventana para seleccionar el destino para guardar.
10. El usuario selecciona el destino. 11. El usuario presiona el botón guardar.	12. El sistema crea la presentación, presenta la pantalla principal y carga en el árbol de navegación la presentación con la configuración seleccionada.
13. El caso de uso finaliza.	
Curso alterno de eventos	
2A. El usuario selecciona el botón cancelar. 3A. El sistema presenta la pantalla principal.	
2B. El usuario presiona el botón ayuda. 3B. El sistema presenta la ayuda de la aplicación.	

Tabla 28 Descripción del caso de uso Crear Presentación con plantilla

8.4.3. Caso de uso Modificar Presentación.

CU-003	
Caso de Uso:	**CU-003 Modificar presentación.**
Actores:	Usuario(U), Software(S)
Resumen:	El usuario del sistema podrá crear presentaciones previamente creadas.
Referencias Cruzadas:	RF-002
Precondiciones: Se hayan creado presentaciones previamente.	
Postcondiciones: Se edite la presentación seleccionada.	
Curso normal de eventos	
Acción del actor	**Respuesta del sistema**
1. El usuario seleccione del asistente de software la opción abrir una presentación existente.	2. El sistema habilita el botón abrir en asistente de software.
3. El usuario presiona el botón abrir.	4. El presenta la pantalla de selección.
5. El usuario selecciona de la presentación existente el archivo config.xml. 6. El usuario presiona el botón abrir.	7. El sistema presenta la vista previa en el asistente de software.
8. El usuario presiona el botón siguiente.	9. El sistema crea la presentación, presenta la pantalla principal y carga en el árbol de navegación la presentación con la configuración seleccionada.
10. El caso de uso finaliza.	
Curso alterno de eventos	
1A. El usuario selecciona el botón cancelar. 2A. El sistema presenta la pantalla principal.	
1B. El usuario presiona el botón ayuda. 2B. El sistema presenta la ayuda de la aplicación.	
5C. El usuario selecciona el botón cancelar. 6C. El sistema cierra la pantalla de selección.	

Tabla 29 Descripción del caso de uso Modificar Presentación

8.4.4. Caso de uso Guardar Presentación.

<table>
<tr><td colspan="3" align="right">CU-004</td></tr>
<tr><td>Caso de Uso:</td><td colspan="2">CU-004 Guardar presentación.</td></tr>
<tr><td>Actores:</td><td colspan="2">Usuario(U), Software(S)</td></tr>
<tr><td>Resumen:</td><td colspan="2">El usuario del sistema podrá guardar los cambios realizados en la presentaciones creadas</td></tr>
<tr><td>Referencias Cruzadas:</td><td colspan="2">RF-002</td></tr>
<tr><td colspan="3">Precondiciones: El usuario cree la presentación previamente.</td></tr>
<tr><td colspan="3">Postcondiciones: Se guarde la presentación creada.</td></tr>
<tr><td colspan="3">Curso normal de eventos</td></tr>
<tr><td colspan="2">Acción del actor</td><td>Respuesta del sistema</td></tr>
<tr><td colspan="2">1. El usuario selecciona presentación del árbol de navegación de la pantalla Principal.</td><td>2. El sistema activa las opciones del panel presentación.</td></tr>
<tr><td colspan="2">3. El usuario selecciona el botón guardar del panel presentación.</td><td>4. El sistema guarda la presentación creada generando el archivo config.xml.
5. El sistema actualiza la GUI.</td></tr>
<tr><td colspan="2">6. El caso de uso finaliza.</td><td></td></tr>
<tr><td colspan="3"></td></tr>
<tr><td colspan="3">Curso alterno de eventos</td></tr>
<tr><td colspan="3">1A. El usuario selecciona un nodo diferente al de presentación del árbol de navegación.
2A. El sistema desbloquea las opciones del nodo seleccionado.
3A. El sistema bloquea la opción de guardar.</td></tr>
</table>

Tabla 30 Descripción del caso de uso Guardar Presentación

8.4.5. Caso de uso Exportar Presentación en formato png.

CU-005	
Caso de Uso:	**CU-005 Exportar Presentación en formato png.**
Actores:	Usuario(U), Software(S)
Resumen:	El usuario del sistema podrá exportar la presentación en formato png.
Referencias Cruzadas:	RF-005
Precondiciones: El usuario seleccione haya guardado la presentación previamente.	
Postcondiciones: Se exporte la presentación en el formato seleccionado.	
Curso normal de eventos	
Acción del actor	**Respuesta del sistema**
1. El usuario selecciona presentación del árbol de navegación en la Pantalla Principal.	2. El sistema activa las opciones del panel presentación.
3. El usuario presione el botón exporta del panel presentación.	4. El sistema presenta la opción para seleccionar el formato a exportar.
5. El usuario selecciona el formato png.	6. El sistema presenta la pantalla para exportar de acuerdo al formato seleccionado.
7. El usuario configura los parámetros deseados. 8. El usuario presiona botón generar.	9. El sistema carga las diferentes imágenes que componen la diapositiva.
10. El usuario selecciona la imagen deseada.	11. El sistema presenta una vista previa.
12. El usuario presiona el botón exportar.	13. El sistema presenta la pantalla para seleccionar el destino.
14. El usuario selecciona el destino y presiona el botón guardar.	15. El sistema guarda la imagen en el destino seleccionado.
16. El caso de uso finaliza.	
Curso alterno de eventos	
12A. El usuario presiona el botón exportar sin generar la captura de imágenes o sin seleccionar una imagen. 13A. El sistema presenta un mensaje de aviso que no existe imágenes generadas.	

Tabla 31 Descripción del caso de uso Exportar presentación en formato png

8.4.6. Caso de uso Exportar Presentación en formato scorm.

<table>
<tr><th colspan="3">CU-006</th></tr>
<tr><td>Caso de Uso:</td><td colspan="2">CU-006 Exportar presentación en formato scorm.</td></tr>
<tr><td>Actores:</td><td colspan="2">Usuario(U), Software(S)</td></tr>
<tr><td>Resumen:</td><td colspan="2">El usuario del sistema podrá exportar la presentación en el formato scorm para utilizarlo en un entorno virtual de aprendizaje.</td></tr>
<tr><td>Referencias Cruzadas:</td><td colspan="2">RF-005</td></tr>
<tr><td colspan="3">Precondiciones: El usuario cree y guarde la presentación.</td></tr>
<tr><td colspan="3">Postcondiciones: Se exporte la presentación en el formato seleccionado.</td></tr>
<tr><td colspan="3">Curso normal de eventos</td></tr>
<tr><td colspan="2">Acción del actor</td><td>Respuesta del sistema</td></tr>
<tr><td colspan="2">1. El usuario selecciona presentación del árbol de navegación en la Pantalla Principal.</td><td>2. El sistema activa las opciones del panel presentación.</td></tr>
<tr><td colspan="2">3. El usuario presione el botón exporta del panel presentación.</td><td>4. El sistema presenta la opción para seleccionar el formato scorm.</td></tr>
<tr><td colspan="2">5. El usuario selecciona el formato deseado.</td><td>6. El sistema presenta la pantalla para exportar la presentación en formato scorm.</td></tr>
<tr><td colspan="2">7. El usuario configura los parámetros deseados.
8. El usuario presiona botón exportar.</td><td>9. El sistema presenta la pantalla para seleccionar el destino.</td></tr>
<tr><td colspan="2">10. El usuario selecciona el destino.
11. El usuario presiona el botón guardar.</td><td>12. El sistema exporta la presentación al destino seleccionado.</td></tr>
<tr><td colspan="2">13. El caso de uso finaliza.</td><td></td></tr>
<tr><td colspan="3"></td></tr>
<tr><td colspan="3">Curso alterno de eventos</td></tr>
<tr><td colspan="3">1A. El usuario selecciona un nodo diferente al de presentación del árbol de navegación.
2A. El sistema desbloquea las opciones del nodo seleccionado.
3A. El sistema bloquea la opción de exportar.</td></tr>
<tr><td colspan="3">B11. El usuario presiona el botón cancelar.
B12. El sistema cierra la pantalla exportar.</td></tr>
</table>

Tabla 32 Descripción del caso de uso Exportar presentación en formato scorm

8.4.7. Caso de uso Seleccionar Fondo para la Presentación.

CU-007	
Caso de Uso:	**CU-007 Seleccionar fondo para la presentación.**
Actores:	Usuario(U), Software(S)
Resumen:	Permite al usuario preseleccionar un fondo predeterminado cada vez que agregue una nueva diapositiva a la presentación.
Referencias Cruzadas:	RF-002
Precondiciones: El usuario haya ingresado al sistema.	
Postcondiciones: El fondo seleccionado se cargara cada vez que se inserte una nueva diapositiva en la presentación.	
Curso normal de eventos	
Acción del actor	**Respuesta del sistema**
1. El usuario seleccione el nodo Presentación de la pantalla Principal.	2. El sistema habilita las opciones del panel Presentación.
3. El usuario despliega los fondos existentes.	4. El sistema carga la lista de fondos disponibles.
5. El usuario selecciona el deseado.	6. El sistema establece al nuevo fondo como predeterminado para la creación de nuevas diapositivas.
7. El caso de uso finaliza.	
Curso alterno de eventos	
1A. El usuario selecciona un nodo diferente al de presentación del árbol de navegación. 2A. El sistema desbloquea las opciones del nodo seleccionado. 3A. El sistema bloquea las opciones de presentación.	

Tabla 33 Descripción del caso de uso seleccionar fondo para la presentación

8.4.8. Caso de uso Generar vista Previa.

<table>
<tr><td colspan="3" align="right">CU-008</td></tr>
<tr><td>Caso de Uso:</td><td colspan="2">CU-008 Generar vista previa.</td></tr>
<tr><td>Actores:</td><td colspan="2">Usuario(U), Software(S)</td></tr>
<tr><td>Resumen:</td><td colspan="2">El usuario podrá visualizar la presentación en pantalla completa.</td></tr>
<tr><td>Referencias Cruzadas:</td><td colspan="2">RF-012</td></tr>
<tr><td colspan="3">Precondiciones: El usuario haya creado la presentación y agregado diapositivas.</td></tr>
<tr><td colspan="3">Postcondiciones: Se visualice de forma completa la presentación con sus diferentes componentes y configuraciones.</td></tr>
<tr><td colspan="3">Curso normal de eventos</td></tr>
<tr><td>Acción del actor</td><td colspan="2">Respuesta del sistema</td></tr>
<tr><td>1. El usuario selecciona el nodo presentación del árbol de navegación de la pantalla principal.</td><td colspan="2">2. El sistema activa las opciones de la pestaña presentación.</td></tr>
<tr><td>3. El usuario Guarda la presentación.
4. El usuario selecciona el botón de vista previa.</td><td colspan="2">5. El sistema ejecuta la visualización de la presentación en pantalla completa.</td></tr>
<tr><td>6. El caso de uso finaliza.</td><td colspan="2"></td></tr>
<tr><td colspan="3"></td></tr>
<tr><td colspan="3">Curso alterno de eventos</td></tr>
<tr><td colspan="3"></td></tr>
<tr><td colspan="3">1A. El usuario selecciona un nodo diferente al de presentación del árbol de navegación.
2A. El sistema desbloquea las opciones del nodo seleccionado.
3A. El sistema bloquea las opciones de presentación.</td></tr>
<tr><td colspan="3">3B El usuario no guarda la presentación.
4B El sistema presenta un mensaje de aviso.</td></tr>
</table>

Tabla 34 Descripción del caso de uso Generar Vista Previa

8.4.9. Caso de uso Imprimir presentación.

CU-009	
Caso de Uso:	**CU-009 Imprimir Presentación.**
Actores:	Usuario(U), Software(S)
Resumen:	El usuario podrá imprimir la presentación previamente realizada.
Referencias Cruzadas:	RF-018
Precondiciones: El usuario haya creado la presentación y agregado diapositivas.	
Postcondiciones: Se imprima la presentación con las diapositivas agregadas.	
Curso normal de eventos	
Acción del actor	**Respuesta del sistema**
1. El usuario selecciona el nodo presentación del árbol de navegación de la pantalla principal.	2. El sistema activa las opciones de la pestaña presentación.
3. El usuario selecciona el botón de imprimir.	4. El sistema presenta la pantalla para configurar impresión.
5. El usuario selecciona botón generar.	6. El sistema extrae las diapositivas a imprimir de la presentación.
7. El usuario presiona el botón imprimir.	8. El sistema realiza la impresión de la presentación de acuerdo a la configuración realizada.
Curso alterno de eventos	
1A. El usuario selecciona un nodo diferente al de presentación del árbol de navegación. 2A. El sistema desbloquea las opciones del nodo seleccionado. 3A. El sistema bloquea las opciones de presentación.	

Tabla 35 Descripción del caso de uso Imprimir Presentación

8.4.10. Caso de uso Agregar diapositiva en blanco.

<table>
<tr><td colspan="3" align="right">CU-010</td></tr>
<tr><td>Caso de Uso:</td><td colspan="2">CU-010 Agregar diapositiva en blanco.</td></tr>
<tr><td>Actores:</td><td colspan="2">Usuario(U), Software(S)</td></tr>
<tr><td>Resumen:</td><td colspan="2">El usuario del sistema podrá agregar nuevas diapositivas a la presentación de forma rápida y sencilla.</td></tr>
<tr><td>Referencias Cruzadas:</td><td colspan="2">RF-001</td></tr>
<tr><td colspan="3">Precondiciones: El usuario haya creado una presentación.</td></tr>
<tr><td colspan="3">Postcondiciones: Se agregue la nueva diapositiva a la presentación.</td></tr>
<tr><td colspan="3">Curso normal de eventos</td></tr>
<tr><td colspan="2">Acción del actor</td><td>Respuesta del sistema</td></tr>
<tr><td colspan="2">1. El usuario seleccione el nodo presentación del árbol de navegación en la pantalla principal.</td><td>2. El sistema habilita las opciones de la pestaña presentación.</td></tr>
<tr><td colspan="2">3. El usuario selecciona la opción nueva del panel diapositiva.</td><td>4. El sistema presenta el dialogo para seleccionar el tipo de diseño.</td></tr>
<tr><td colspan="2">5. El usuario selecciona la opción libre.</td><td>6. El sistema carga en el árbol de navegación la nueva diapositiva de la presentación.
7. El sistema actualiza la interfaz gráfica de usuario.</td></tr>
<tr><td colspan="2">8. El caso de uso finaliza.</td><td></td></tr>
<tr><td colspan="3"></td></tr>
<tr><td colspan="3">Curso alterno de eventos</td></tr>
<tr><td colspan="3">1A. El usuario selecciona un nodo diferente al de presentación del árbol de navegación.
2A. El sistema desbloquea las opciones del nodo seleccionado.
3A. El sistema bloquea las opciones de presentación.</td></tr>
<tr><td colspan="3"></td></tr>
<tr><td colspan="3"></td></tr>
</table>

Tabla 36 Descripción del caso de uso agregar diapositiva en blanco

8.4.11. Caso de uso Agregar diapositiva con diseño.

CU-011	
Caso de Uso:	**CU-011 Agregar diapositiva con diseño.**
Actores:	Usuario(U), Software(S)
Resumen:	El usuario del sistema podrá agregar nuevas diapositivas a la presentación de forma rápida y sencilla.
Referencias Cruzadas:	RF-001
Precondiciones: Se haya creada una presentación.	
Postcondiciones: Se agregue la nueva diapositiva a la presentación.	
Curso normal de eventos	
Acción del actor	**Respuesta del sistema**
1. El usuario seleccione el nodo presentación del árbol de navegación en la pantalla principal.	2. El sistema habilita las opciones de la pestaña presentación.
3. El usuario selecciona el botón nuevo del panel diapositiva.	4. El sistema presenta la pantalla con los diseños existentes.
5. El usuario selecciona el diseño deseado.	6. El sistema carga en el árbol de navegación la nueva diapositiva con el diseño establecido. 7. El sistema actualiza la interfaz gráfica de usuario.
8. El caso de uso finaliza.	
Curso alterno de eventos	
1A. El usuario selecciona un nodo diferente al de presentación del árbol de navegación. 2A. El sistema desbloquea las opciones del nodo seleccionado. 3A. El sistema bloquea las opciones de presentación.	

Tabla 37 Descripción del caso de uso Agregar diapositiva con diseño

8.4.12. Caso de uso Agregar color de fondo.

CU-012	
Caso de Uso:	**CU-012 Agregar color de fondo.**
Actores:	Usuario(U), Software(S)
Resumen:	El usuario podrá agregar un color como fondo a la diapositiva de forma rápida y sencilla.
Referencias Cruzadas:	RF-001
Precondiciones: El usuario seleccione la diapositiva deseada.	
Postcondiciones: Se establezca el fondo configurado por el usuario.	
Curso normal de eventos	
Acción del actor	**Respuesta del sistema**
1. El usuario selecciona el botón color de la pestaña fondo.	2. El sistema presenta la pantalla de selección de color.
3. El usuario selecciona el color deseado. 4. El usuario presiona el botón aceptar.	5. El sistema fija en la diapositiva el color seleccionado como fondo.
6. El caso de uso finaliza.	
Curso alterno de eventos	
4A. El usuario presiona el botón cancelar. 5A. El sistema cierra la pantalla de selección color.	

Tabla 38 Descripción del caso de uso agregar color de fondo

8.4.13. Caso de uso Agregar imagen de fondo.

CU-013	
Caso de Uso:	**CU-013 Agregar imagen de fondo.**
Actores:	Usuario(U), Software(S)
Resumen:	El usuario podrá agregar el fondo deseado a la diapositiva este puede ser una imagen ya sea de las que posee la aplicación o desde una dirección externa.
Referencias Cruzadas:	RF-001
Precondiciones: El usuario seleccione la diapositiva deseada.	
Postcondiciones: Se establezca el fondo configurado por el usuario.	
Curso normal de eventos	
Acción del actor	**Respuesta del sistema**
1. El usuario selecciona el botón Abrir de la pestaña Fondos.	2. El sistema presenta la pantalla de selección imagen.
3. El usuario selecciona la imagen desde una dirección externa. 4. El usuario presiona el botón Abrir.	5. El sistema verifica el peso de la imagen y presenta msj de aviso.
6. El usuario presiona aceptar.	7. El sistema fija la imagen seleccionada como fondo de la diapositiva.
8. El caso de uso finaliza.	
Curso alterno de eventos	
6A. El usuario presiona el botón cancelar. 7A. El sistema cierra la pantalla de selección imagen.	

Tabla 39 Descripción del caso de uso agregar imagen de fondo

8.4.14. Caso de uso Eliminar diapositiva.

CU-014	
Caso de Uso:	**CU-014 Eliminar diapositiva.**
Actores:	Usuario(U), Software(S)
Resumen:	El usuario podrá eliminar las diapositivas seleccionadas de la presentación.
Referencias Cruzadas:	RF-001
Precondiciones: El usuario haya creado diapositivas previamente.	
Postcondiciones: Se elimina la diapositiva seleccionada.	
Curso normal de eventos	
Acción del actor	**Respuesta del sistema**
1. El usuario seleccione la diapositiva que desee eliminar.	2. El sistema activa la pestaña diapositiva.
3. El usuario selecciona el botón eliminar.	4. El sistema elimina la diapositiva seleccionada de la presentación. 5. Se actualiza la GUI.
6. El caso de uso finaliza.	
Curso alterno de eventos	
1A. El usuario selecciona un nodo diferente al de diapositiva del árbol de navegación. 2A. El sistema desbloquea las opciones del nodo seleccionado. 3A. El sistema bloquea las opciones de diapositiva.	

Tabla 40 Descripción del caso de uso Eliminar diapositiva

8.4.15. Caso de uso Exportar diapositiva en formato png o gif.

CU-015	
Caso de Uso:	**CU-015 Exportar Diapositiva en formato png o gif.**
Actores:	Usuario(U), Software(S)
Resumen:	El usuario podrá generar imágenes de las diapositivas ya sean estas en formato png o gif.
Referencias Cruzadas:	RF-017
Precondiciones: El usuario guarde el proyecto y seleccione la diapositiva que desee exportar.	
Postcondiciones: Se cree la imagen de la diapositiva en el formato seleccionado.	
Curso normal de eventos	
Acción del actor	**Respuesta del sistema**
1. El usuario presiona el botón exportar de la pestaña diapositivas.	2. El sistema presenta la pantalla para seleccionar el formato png o gif.
3. El usuario selecciona formato deseado.	4. El sistema presenta la pantalla de configuración del formato de imagen.
5. El usuario selecciona el botón generar.	6. El sistema carga las imágenes que componen a la diapositiva.
7. El usuario selecciona el botón exportar.	8. El sistema presenta la pantalla de selección de destino a guardar.
9. El usuario selecciona destino y presiona el botón guardar.	10. El sistema guarda la imagen en el formato seleccionado.
11. El caso de uso finaliza.	
Curso alterno de eventos	
7A El usuario presiona exportar sin antes generar. 8A El sistema presenta mensaje de aviso	

Tabla 41 Descripción del caso de uso Exportar diapositiva en formato png o gif

8.4.16. Caso de uso Generar vista previa efecto.

CU-016	
Caso de Uso:	**CU-016 Generar vista previa de efecto.**
Actores:	Usuario(U), Software(S)
Resumen:	El usuario podrá visualizar previamente el efecto agregado.
Referencias Cruzadas:	RF-019
Precondiciones: El usuario haya agregado efectos.	
Postcondiciones: Se visualice el efecto seleccionado.	
Curso normal de eventos	
Acción del actor	**Respuesta del sistema**
1. El usuario seleccione la diapositiva del árbol de navegación.	2. El sistema activa la pestaña Animaciones.
3. El usuario selecciona el botón vista previa.	4. El sistema ejecuta la vista previa de los efectos contenidos en la diapositiva.
5. El usuario selecciona el botón detener de la pestaña diapositiva.	6. El sistema detiene el pre visualización de los efectos cargados.
7. El caso de uso finaliza.	
Curso alterno de eventos	
1A. El usuario selecciona el nodo presentación del árbol de navegación. 2A. El sistema desbloquea las opciones del nodo seleccionado. 3A. El sistema bloquea las opciones de animación.	
1B. El usuario selecciona el nodo audio del árbol de navegación. 2B. El sistema desbloquea las opciones del nodo seleccionado. 3B. El sistema bloquea las opciones de animación.	

Tabla 42 Descripción del caso de uso Generar vista previa de efecto

8.4.17. Caso de uso Agregar componente gráfico.

<table>
<tr><td colspan="3" align="right">CU-017</td></tr>
<tr><td>Caso de Uso:</td><td colspan="2">CU-017 Agregar componente gráfico.</td></tr>
<tr><td>Actores:</td><td colspan="2">Usuario(U), Software(S)</td></tr>
<tr><td>Resumen:</td><td colspan="2">El usuario podrá agregar componentes gráficos a las diapositivas ya sea objetos 3d, texto 3d o formas de manera rápida y sencilla.</td></tr>
<tr><td>Referencias Cruzadas:</td><td colspan="2">RF-013, RF-016</td></tr>
<tr><td colspan="3">Precondiciones: el usuario seleccione del árbol de navegación la diapositiva a la que desee agregar el componente gráfico.</td></tr>
<tr><td colspan="3">Postcondiciones: Se agregue el componente gráfico.</td></tr>
<tr><td colspan="3">Curso normal de eventos</td></tr>
<tr><td colspan="2">Acción del actor</td><td>Respuesta del sistema</td></tr>
<tr><td colspan="2">1. El usuario selecciona el componente grafico que desee agregar del panel insertar componentes.</td><td>2. El sistema muestra un dialogo para configurar el tamaño del componente.</td></tr>
<tr><td colspan="2">3. El usuario configura y presiona aceptar.</td><td>4. El sistema crea un nuevo componente.
5. El sistema agrega el componente grafico a la diapositiva.
6. El sistema inserta en el árbol de navegación un nuevo nodo del componente.
7. El sistema actualiza el árbol de navegación.</td></tr>
<tr><td colspan="2">8. El caso de uso finaliza.</td><td></td></tr>
<tr><td colspan="2"></td><td></td></tr>
<tr><td colspan="3">Curso alterno de eventos</td></tr>
<tr><td colspan="3">3A. El usuario presiona el botón cancelar.
4A. El sistema cierra el dialogo configurar componente.</td></tr>
<tr><td colspan="3"></td></tr>
<tr><td colspan="3"></td></tr>
</table>

Tabla 43 Descripción del caso de uso Agregar componente grafico

8.4.18. Caso de uso Modificar componente gráfico.

<table>
<tr><td colspan="3" align="right">CU-018</td></tr>
<tr><td>Caso de Uso:</td><td colspan="2">CU-018 Modificar componente gráfico.</td></tr>
<tr><td>Actores:</td><td colspan="2">Usuario(U), Software(S)</td></tr>
<tr><td>Resumen:</td><td colspan="2">El usuario del sistema podrá modificar la configuración del componente gráfico de forma sencilla.</td></tr>
<tr><td>Referencias Cruzadas:</td><td colspan="2">RF-013, RF-015, RF016</td></tr>
<tr><td colspan="3">Precondiciones: El componente grafico este creado.</td></tr>
<tr><td colspan="3">Postcondiciones: Se modifique el componente gráfico.</td></tr>
<tr><td colspan="3">Curso normal de eventos</td></tr>
<tr><td colspan="2">Acción del actor</td><td>Respuesta del sistema</td></tr>
<tr><td colspan="2">1. El usuario seleccione el componente grafico del árbol de navegación.</td><td>2. El sistema activa las opciones de configuración de componentes.</td></tr>
<tr><td colspan="2">3. El usuario modifica los parámetros de configuración.</td><td>4. El sistema fija los nuevos valores de configuración al componente.</td></tr>
<tr><td colspan="2">5. El caso de uso finaliza.</td><td></td></tr>
<tr><td colspan="3">Curso alterno de eventos</td></tr>
<tr><td colspan="3">1A. El usuario selecciona un nodo diferente al de los componentes gráficos del árbol de navegación.
2A. El sistema desbloquea las opciones del nodo seleccionado.
3A. El sistema bloquea las opciones de configuración del componente gráfico.</td></tr>
<tr><td colspan="3"></td></tr>
</table>

Tabla 44 Descripción del caso de uso Modificar componente grafico

8.4.19. Caso de uso Eliminar componente gráfico.

<table>
<tr><td colspan="2" align="right">CU-019</td></tr>
<tr><td>Caso de Uso:</td><td>CU-019 Eliminar componente gráfico.</td></tr>
<tr><td>Actores:</td><td>Usuario(U), Software(S)</td></tr>
<tr><td>Resumen:</td><td>El usuario del sistema podrá eliminar el componente gráfico de manera fácil.</td></tr>
<tr><td>Referencias Cruzadas:</td><td>RF-013, RF-016</td></tr>
<tr><td colspan="2">Precondiciones: El usuario haya seleccionado el componente grafico del árbol de navegación.</td></tr>
<tr><td colspan="2">Postcondiciones: Se elimine el componente gráfico.</td></tr>
<tr><td colspan="2">Curso normal de eventos</td></tr>
<tr><td>Acción del actor</td><td>Respuesta del sistema</td></tr>
<tr><td>1. El usuario selecciona un nodo componente grafico del árbol de navegación.</td><td>2. El sistema activa la opción de eliminar componente gráfico del panel configurar componente.</td></tr>
<tr><td>3. El usuario presiona el botón eliminar del panel eliminar.</td><td>4. El sistema elimina el componente de la diapositiva.
5. El sistema remueve el nodo componente grafico del árbol de navegación.
6. El sistema actualiza el árbol de navegación.</td></tr>
<tr><td>7. El caso de uso finaliza.</td><td></td></tr>
<tr><td colspan="2"></td></tr>
<tr><td colspan="2">Curso alterno de eventos</td></tr>
<tr><td colspan="2">1A. El usuario selecciona un nodo diferente al de los componentes gráficos del árbol de navegación.
2A. El sistema desbloquea las opciones del nodo seleccionado.
3A. El sistema bloquea las opciones de eliminar componente gráfico.</td></tr>
<tr><td colspan="2"></td></tr>
</table>

Tabla 45 Descripción del caso de uso Eliminar componente grafico

8.4.20. Caso de uso Importar Recursos.

CU-020	
Caso de Uso:	**CU-020 Importar recursos.**
Actores:	Usuario(U), Software(S)
Resumen:	El usuario del sistema podrá importar nuevos objetos a la galería de forma rápida y sencilla.
Referencias Cruzadas:	RF-003
Precondiciones: El usuario haya ingresado al sistema y haya activado la lista de recursos a la que desea importar.	
Postcondiciones: Se agregue el nuevo objeto a la galería.	
Curso normal de eventos	
Acción del actor	**Respuesta del sistema**
1. El usuario selecciona el nodo dependiendo del elemento a importar del árbol de navegación.	2. El sistema activa la pestaña del panel Galería de Recursos.
3. El usuario selecciona pestaña (Fondos, Textura, Objeto).	4. El sistema recupera el tipo de galería. 5. El sistema llama al controlador agregar nuevo objeto.
6. El usuario selecciona el botón importar.	7. El sistema presenta la pantalla de selección
8. El usuario selecciona el objeto y presiona abrir.	9. El sistema copia el objeto a la galería y actualiza la lista de objetos.
10. El caso de uso finaliza	
Curso alterno de eventos	
1A El usuario no selecciona el nodo objeto. 2A El usuario presiona botón importar. 3A El sistema presenta mensaje de aviso.	

Tabla 46 Descripción del caso de uso Importar Recursos

8.4.21. Caso de uso Eliminar Recursos.

<table>
<tr><td colspan="2" align="right">CU-021</td></tr>
<tr><td>Caso de Uso:</td><td>CU-021 Eliminar recursos.</td></tr>
<tr><td>Actores:</td><td>Usuario(U), Software(S)</td></tr>
<tr><td>Resumen:</td><td>El usuario del sistema podrá eliminar objetos de la galería de forma rápida y sencilla.</td></tr>
<tr><td>Referencias Cruzadas:</td><td>RF-004</td></tr>
<tr><td colspan="2">Precondiciones: El usuario haya seleccionado el objeto de la galería.</td></tr>
<tr><td colspan="2">Postcondiciones: Se elimine el objeto de la galería.</td></tr>
<tr><td colspan="2">Curso normal de eventos</td></tr>
<tr><td>Acción del actor</td><td>Respuesta del sistema</td></tr>
<tr><td>1. El usuario presiona el botón eliminar del panel galería de recursos.</td><td>2. El sistema llama al controlador de eliminar objeto.
3. El sistema muestra un dialogo de confirmación para eliminar.</td></tr>
<tr><td>4. El usuario presiona el botón SI.</td><td>5. El sistema obtiene el objeto seleccionado de la lista de objetos.
6. El sistema crea un archivo.
7. El sistema elimina el objeto de la galería.
8. El sistema recarga la lista de objetos.
9. El sistema actualiza la GUI.</td></tr>
<tr><td>10. El caso de uso finaliza.</td><td></td></tr>
<tr><td colspan="2"></td></tr>
<tr><td colspan="2">Curso alterno de eventos</td></tr>
<tr><td colspan="2">4A. El usuario presiona el botón NO.
5A. El sistema cierra el dialogo de confirmación</td></tr>
<tr><td colspan="2"></td></tr>
</table>

Tabla 47 Descripción del caso de uso Eliminar Recursos

8.4.22. Caso de uso Agregar Sonido.

<table>
<tr><td colspan="3" align="right">CU-022</td></tr>
<tr><td>Caso de Uso:</td><td colspan="2">CU-022 Agregar sonido.</td></tr>
<tr><td>Actores:</td><td colspan="2">Usuario(U), Software(S)</td></tr>
<tr><td>Resumen:</td><td colspan="2">El usuario podrá agregar audio a las diapositivas de forma fácil y sencilla.</td></tr>
<tr><td>Referencias Cruzadas:</td><td colspan="2">RF-010</td></tr>
<tr><td colspan="3">Precondiciones: El usuario haya seleccionado la diapositiva a la que desea agregar audio.</td></tr>
<tr><td colspan="3">Postcondiciones: Se agregue el sonido.</td></tr>
<tr><td colspan="3">Curso normal de eventos</td></tr>
<tr><td colspan="2">Acción del actor</td><td>Respuesta del sistema</td></tr>
<tr><td colspan="2">1. El usuario presiona el botón sonido del panel insertar componentes.</td><td>2. El sistema muestra la pantalla de selección de archivos.</td></tr>
<tr><td colspan="2">3. El usuario selecciona el archivo de audio que desee agregar.
4. El usuario presiona el botón abrir.</td><td>5. El sistema llama al controlador para agregar sonido.
6. El sistema obtiene el archivo seleccionado.
7. El sistema obtiene la dirección del archivo.
8. El sistema obtiene el nombre del archivo.
9. El sistema fija la dirección del sonido en la diapositiva.
10. El sistema inserta el nodo de sonido en el árbol de navegación.
11. El sistema actualiza el árbol de navegación.</td></tr>
<tr><td colspan="2">12. El caso de uso finaliza.</td><td></td></tr>
<tr><td colspan="3"></td></tr>
<tr><td colspan="3">Curso alterno de eventos</td></tr>
<tr><td colspan="3">4A. El usuario presiona el botón cancelar.
5A. El sistema cierra la pantalla de selección de archivos.</td></tr>
<tr><td colspan="3"></td></tr>
</table>

Tabla 48 Descripción del caso de uso Agregar Sonido

8.4.23. Caso de uso Eliminar Sonido.

CU-023	
Caso de Uso:	**CU-023 Eliminar sonido.**
Actores:	Usuario(U), Software(S)
Resumen:	El usuario del sistema podrá eliminar audio agregado a las diapositivas.
Referencias Cruzadas:	RF-010
Precondiciones: El archivo de sonido este agregado a la diapositiva.	
Postcondiciones: El archivo de sonido se elimine de la diapositiva.	
Curso normal de eventos	
Acción del actor	**Respuesta del sistema**
1. El usuario selecciona el nodo sonido del árbol de navegación.	2. El sistema activa las opciones de sonido.
3. El usuario selecciona el botón eliminar de la pestaña Sonido.	4. El sistema llama al controlador para eliminar sonido. 5. El sistema fija nulo al sonido de la diapositiva. 6. El sistema detiene la reproducción de audio si esta activada. 7. El sistema remueve el nodo sonido del árbol de navegación. 8. El sistema actualiza el árbol de navegación.
9. El caso de uso finaliza.	
Curso alterno de eventos	
1A. El usuario selecciona un nodo diferente al de sonido del árbol de navegación. 2A. El sistema desbloquea las opciones del nodo seleccionado. 3A. El sistema bloquea las opciones de eliminar sonido.	

Tabla 49 Descripción del caso de uso Eliminar Sonido

8.4.24. Caso de uso Agregar Material.

<table>
<tr><td colspan="3" align="right">CU-024</td></tr>
<tr><td>Caso de Uso:</td><td colspan="2">CU-024 Agregar material.</td></tr>
<tr><td>Actores:</td><td colspan="2">Usuario(U), Software(S)</td></tr>
<tr><td>Resumen:</td><td colspan="2">El usuario podrá agregar material al texto y formas de forma fácil y sencilla.</td></tr>
<tr><td>Referencias Cruzadas:</td><td colspan="2">RF-011</td></tr>
<tr><td colspan="3">Precondiciones: El componente grafico este agregado a la diapositiva.</td></tr>
<tr><td colspan="3">Postcondiciones: Se agregue el material.</td></tr>
<tr><td colspan="3">Curso normal de eventos</td></tr>
<tr><td colspan="2">Acción del actor</td><td>Respuesta del sistema</td></tr>
<tr><td colspan="2">1. El usuario selecciona el componente grafico del árbol de navegación.</td><td>2. El sistema activa el panel material.</td></tr>
<tr><td colspan="2">3. El usuario presione el botón nuevo del panel material.</td><td>4. El sistema llama al controlador agregar material.
5. El sistema crea un nuevo material.
6. El sistema fija el material en el componente gráfico.
7. El sistema actualiza la pantalla.</td></tr>
<tr><td colspan="2">8. El caso de uso finaliza.</td><td></td></tr>
<tr><td colspan="3"></td></tr>
<tr><td colspan="3">Curso alterno de eventos</td></tr>
<tr><td colspan="3">1A. El usuario selecciona un nodo diferente al de los componentes gráficos del árbol de navegación.
2A. El sistema desbloquea las opciones del nodo seleccionado.
3A. El sistema bloquea el panel material.</td></tr>
<tr><td colspan="3"></td></tr>
</table>

Tabla 50 Descripción del caso de uso Agregar Material

8.4.25. Caso de uso Modificar Configuración de material.

<table>
<tr><td colspan="3" align="right">CU-025</td></tr>
<tr><td>Caso de Uso:</td><td colspan="2">CU-025 Modificar configuración de material.</td></tr>
<tr><td>Actores:</td><td colspan="2">Usuario(U), Software(S)</td></tr>
<tr><td>Resumen:</td><td colspan="2">El usuario del sistema podrá modificar la configuración del material de una forma sencilla.</td></tr>
<tr><td>Referencias Cruzadas:</td><td colspan="2">RF-011</td></tr>
<tr><td colspan="3">Precondiciones: El usuario haya seleccione el componente gráfico del árbol de navegación.</td></tr>
<tr><td colspan="3">Postcondiciones: Se guarde la modificación del material.</td></tr>
<tr><td colspan="3">Curso normal de eventos</td></tr>
<tr><td>Acción del actor</td><td colspan="2">Respuesta del sistema</td></tr>
<tr><td>1. El usuario modifica la configuración del material.</td><td colspan="2">2. El sistema llama al controlador para modificar material.
3. El sistema obtiene el material del componente gráfico.
4. El sistema fija el ambientcolor.
5. El sistema fija el difusecolor.
6. El sistema fija el emissivecolor.
7. El sistema fija el specularcolor
8. El sistema fija el shininess
9. El sistema actualiza el panel material.</td></tr>
<tr><td>10. El caso de uso finaliza.</td><td colspan="2"></td></tr>
<tr><td colspan="3">Curso alterno de eventos</td></tr>
<tr><td colspan="3">3A. El componente gráfico no contiene ningún material.
4A. El sistema no aplica ningún cambio en la apariencia del componente gráfico.</td></tr>
<tr><td colspan="3"></td></tr>
</table>

Tabla 51 Descripción del caso de uso Modificar configuración de material

8.4.26. Caso de uso Eliminar Material.

<table>
<tr><td colspan="3" align="right">CU-026</td></tr>
<tr><td>Caso de Uso:</td><td colspan="2">CU-026 Eliminar material.</td></tr>
<tr><td>Actores:</td><td colspan="2">Usuario(U), Software(S)</td></tr>
<tr><td>Resumen:</td><td colspan="2">El usuario del sistema podrá eliminar el material de los componentes gráficos de forma fácil y sencilla.</td></tr>
<tr><td>Referencias Cruzadas:</td><td colspan="2">RF-011</td></tr>
<tr><td colspan="3">Precondiciones: El componente gráfico este agregado a la diapositiva.</td></tr>
<tr><td colspan="3">Postcondiciones: Se elimine el material.</td></tr>
<tr><td colspan="3">Curso normal de eventos</td></tr>
<tr><td colspan="2">Acción del actor</td><td>Respuesta del sistema</td></tr>
<tr><td colspan="2">1. El usuario selecciona el componente gráfico del árbol de navegación.</td><td>2. El sistema activa el panel material.</td></tr>
<tr><td colspan="2">3. El usuario presione el botón eliminar del panel material.</td><td>4. El sistema llama al controlador para eliminar material.
5. El sistema fija nulo al material en el componente gráfico.
6. El sistema actualiza el panel material.</td></tr>
<tr><td colspan="2">7. El caso de uso finaliza.</td><td></td></tr>
<tr><td colspan="3"></td></tr>
<tr><td colspan="3">Curso alterno de eventos</td></tr>
<tr><td colspan="3">1A. El usuario selecciona un nodo diferente al de los componentes gráficos del árbol de navegación.
2A. El sistema desbloquea las opciones del nodo seleccionado.
3A. El sistema bloquea el panel material.</td></tr>
<tr><td colspan="3"></td></tr>
</table>

Tabla 52 Descripción del caso de uso Eliminar Material

8.4.27. Caso de uso Agregar Efecto al componente gráfico.

<table>
<tr><td colspan="3" align="right">CU-027</td></tr>
<tr><td>Caso de Uso:</td><td colspan="2">CU-027 Agregar efecto al componente gráfico.</td></tr>
<tr><td>Actores:</td><td colspan="2">Usuario(U), Software(S)</td></tr>
<tr><td>Resumen:</td><td colspan="2">El usuario del sistema podrá agregar efectos a los componentes gráficos de una forma sencilla.</td></tr>
<tr><td>Referencias Cruzadas:</td><td colspan="2">RF-007, RF-008, RF-009</td></tr>
<tr><td colspan="3">Precondiciones: El objeto al que se desee agregar el efecto ya este agregado a la diapositiva.</td></tr>
<tr><td colspan="3">Postcondiciones: Se agregue el efecto.</td></tr>
<tr><td colspan="3">Curso normal de eventos</td></tr>
<tr><td colspan="2">Acción del actor</td><td>Respuesta del sistema</td></tr>
<tr><td colspan="2">1. El usuario selecciona el componente grafico del árbol de navegación.</td><td>2. El sistema verifica el tipo de componente gráfico.
3. El sistema carga los efectos soportados por el componente gráfico.</td></tr>
<tr><td colspan="2">4. El usuario selecciona el efecto.</td><td>5. El sistema agrega el efecto al objeto.
6. El sistema muestra el panel de configuración del efecto seleccionado.</td></tr>
<tr><td colspan="2">7. El caso de uso finaliza.</td><td></td></tr>
<tr><td colspan="3"></td></tr>
<tr><td colspan="3">Curso alterno de eventos</td></tr>
<tr><td colspan="3"></td></tr>
<tr><td colspan="3">1A. El usuario no selecciona componente gráfico.
2A. El sistema desactiva el panel de animaciones.</td></tr>
<tr><td colspan="3">3B. El componente no tiene material.
4B. El sistema muestra un mensaje “Para usar este efecto debe primero agregar un material”</td></tr>
<tr><td colspan="3">3C. El componente no tiene iniciado la transparencia.
4C. El sistema muestra un mensaje “Para usar este efecto debe primero inicializar la transparecia”.</td></tr>
</table>

Tabla 53 Descripción del caso de uso Agregar efecto al componente grafico

8.4.28. Caso de uso Modificar configuración del efecto.

CU-028	
Caso de Uso:	**CU-028 Modificar configuración del efecto.**
Actores:	Usuario(U), Software(S)
Resumen:	El usuario podrá modificar la configuración del objeto de forma sencilla.
Referencias Cruzadas:	RF-007, RF-008, RF-009
Precondiciones: El efecto ya este agregado.	
Postcondiciones: Se guarde la modificación del efecto.	
Curso normal de eventos	
Acción del actor	**Respuesta del sistema**
1. El usuario seleccione del efecto la opción modificar.	2. El sistema obtiene la configuración del efecto. 3. El sistema presenta la pantalla de configuración del efecto.
4. El usuario modifica los parámetros deseados. 5. El usuario presiona el botón aceptar.	6. El sistema actualiza los cambios realizados.
7. El caso de uso finaliza.	
Curso alterno de eventos	
5A. El usuario presiona el botón cancelar. 6A. El sistema cierra la pantalla de configuración efecto.	

Tabla 54 Descripción del caso de uso Modificar configuración del efecto

8.4.29. Caso de uso Eliminar efecto del componente gráfico.

CU-029	
Caso de Uso:	**CU-029 Eliminar efecto del componente gráfico.**
Actores:	Usuario(U), Software(S)
Resumen:	El usuario del sistema podrá eliminar los efectos de los componentes.
Referencias Cruzadas:	RF-007, RF-008, RF-009
Precondiciones: El componente gráfico esté agregado a la diapositiva.	
Postcondiciones: Se elimine el efecto del componente gráfico.	
Curso normal de eventos	
Acción del actor	**Respuesta del sistema**
1. El usuario selecciona el componente grafico del árbol de navegación.	2. El sistema activa las opciones de animación.
3. El usuario selecciona el efecto que desee quitar de la pestaña animaciones. **4.** El usuario selecciona la opción eliminar.	5. El sistema elimina el efecto del componente gráfico 6. El sistema actualiza la interfaz gráfica de usuario.
7. El caso de uso finaliza.	
Curso alterno de eventos	
1A. El usuario selecciona un nodo diferente al de los componentes gráficos del árbol de navegación. 2A. El sistema desbloquea las opciones del nodo seleccionado. 3A. El sistema bloquea el panel animación.	

Tabla 55 Descripción del caso de uso Eliminar efecto del componente grafico

8.4.30. Caso de uso Agregar textura desde archivo.

<table>
<tr><td colspan="2" align="right">CU-031</td></tr>
<tr><td>Caso de Uso:</td><td>CU-031 Agregar textura desde archivo.</td></tr>
<tr><td>Actores:</td><td>Usuario(U), Software(S)</td></tr>
<tr><td>Resumen:</td><td>El usuario podrá agregar texturas ya sea a los textos 3D o a las formas.</td></tr>
<tr><td>Referencias Cruzadas:</td><td>RF-008</td></tr>
<tr><td colspan="2">Precondiciones: El usuario haya seleccionado al componente gráfico.</td></tr>
<tr><td colspan="2">Postcondiciones: Se fije la textura al componente gráfico.</td></tr>
<tr><td colspan="2">Curso normal de eventos</td></tr>
<tr><td>Acción del actor</td><td>Respuesta del sistema</td></tr>
<tr><td>1. El usuario presiona el botón nuevo del panel texturas.</td><td>2. El sistema muestra la pantalla de selección de archivos.</td></tr>
<tr><td>3. El usuario selecciona la textura que desea agregar.
4. El usuario presiona agregar.</td><td>5. El sistema llama al controlador para agregar texturas.
6. El sistema obtiene el archivo seleccionado.
7. El sistema obtiene la dirección de la textura.
8. El sistema presenta la vista previa.
9. El sistema crea un lector de texturas.
10. El sistema fija la textura en el componente gráfico.</td></tr>
<tr><td>11. El caso de uso finaliza.</td><td></td></tr>
<tr><td colspan="2"></td></tr>
<tr><td colspan="2">Curso alterno de eventos</td></tr>
<tr><td colspan="2">4A. El usuario presiona el botón cancelar.
5A. El sistema cierra la pantalla de selección de archivos.</td></tr>
<tr><td colspan="2"></td></tr>
</table>

Tabla 56 Descripción del caso de uso Agregar textura desde archivo

8.4.31. Caso de uso Agregar textura desde galería.

CU-031	
Caso de Uso:	**CU-031 Agregar textura desde galería.**
Actores:	Usuario(U), Software(S)
Resumen:	El usuario podrá agregar texturas ya sea a los textos 3D o a las formas.
Referencias Cruzadas:	RF-008
Precondiciones: El usuario haya seleccionado el componente gráfico del árbol de navegación.	
Postcondiciones: Se fije la textura al componente gráfico.	
Curso normal de eventos	
Acción del actor	**Respuesta del sistema**
1. El usuario selecciona una textura de la lista texturas.	2. El sistema verifica el tipo de componente gráfico. 3. El sistema fija la textura en el panel vista previa. 4. El sistema llama al controlador para agregar textura. 5. El sistema lee la textura. 6. El sistema fija la textura en el componente gráfico.
7. El caso de uso finaliza.	
Curso alterno de eventos	
5A. La textura leída no tiene dimensiones aceptables. 6A. El sistema redimensiona la textura. 7A. el sistema fija la textura en el componente gráfico.	

Tabla 57 Descripción del caso de uso Agregar textura desde galería

8.4.32. Caso de uso Configurar textura.

<table>
<tr><td colspan="3" align="right">CU-032</td></tr>
<tr><td>Caso de Uso:</td><td colspan="2">CU-032 Configurar textura</td></tr>
<tr><td>Actores:</td><td colspan="2">Usuario(U), Software(S)</td></tr>
<tr><td>Resumen:</td><td colspan="2">El usuario del sistema podrá modificar la configuración de las texturas.</td></tr>
<tr><td>Referencias Cruzadas:</td><td colspan="2">RF-008</td></tr>
<tr><td colspan="3">Precondiciones: El componente gráfico esté agregado a la diapositiva.</td></tr>
<tr><td colspan="3">Postcondiciones: Se modifique la configuración de la textura.</td></tr>
<tr><td colspan="3">Curso normal de eventos</td></tr>
<tr><td colspan="2">Acción del actor</td><td>Respuesta del sistema</td></tr>
<tr><td colspan="2">1. El usuario selecciona el componente gráfico del árbol de navegación.</td><td>2. El sistema obtiene la textura del componente gráfico.
3. El sistema fija la textura en el panel de pre visualización de texturas.
4. El sistema fija las configuraciones de la textura.</td></tr>
<tr><td colspan="2">5. El usuario modifica los parámetros de configuración del panel texturas.</td><td>6. El sistema llama al controlador para modificar textura.
7. El sistema crea un nuevo generador de coordenadas para la textura.
8. El sistema obtiene la apariencia del componente gráfico.
9. El sistema fija la configuración en la apariencia del componente gráfico.</td></tr>
<tr><td colspan="2">10. El caso de uso finaliza.</td><td></td></tr>
<tr><td colspan="3"></td></tr>
<tr><td colspan="3">Curso alterno de eventos</td></tr>
<tr><td colspan="3">1A. El usuario selecciona un nodo diferente al de los componentes gráficos del árbol de navegación.
2A. El sistema desbloquea las opciones del nodo seleccionado.
3A. El sistema bloquea el Panel Textura.</td></tr>
<tr><td colspan="3"></td></tr>
</table>

Tabla 58 Descripción del caso de uso Configurar Textura

8.4.33. Caso de uso Eliminar textura.

CU-033	
Caso de Uso:	**CU-033 Eliminar textura.**
Actores:	Usuario(U), Software(S)
Resumen:	El usuario del sistema podrá eliminar la textura.
Referencias Cruzadas:	RF-008
Precondiciones: El componente gráfico esté agregado a la diapositiva.	
Postcondiciones: Se elimine la textura.	
Curso normal de eventos	
Acción del actor	**Respuesta del sistema**
1. El usuario selecciona el componente gráfico del árbol de navegación.	2. El sistema obtiene la textura del componente gráfico. 3. El sistema fija la textura en el panel de pre visualización de texturas. 4. El sistema fija las configuraciones de la textura.
5. El usuario presiona el botón eliminar del panel texturas.	6. El sistema elimina la textura del componente gráfico. 7. El sistema actualiza la interfaz gráfica de usuario.
8. El caso de uso finaliza.	
Curso alterno de eventos	
1A. El usuario selecciona un nodo diferente al de los componentes gráficos del árbol de navegación. 2A. El sistema desbloquea las opciones del nodo seleccionado. 3A. El sistema bloquea el Panel Textura.	

Tabla 59 Descripción del caso de uso Eliminar Textura

8.5. MODELADO DE SECUENCIA O MODELO DETALLADO

8.5.1. Modelado de secuencia del Caso de uso Crear presentación vacía.

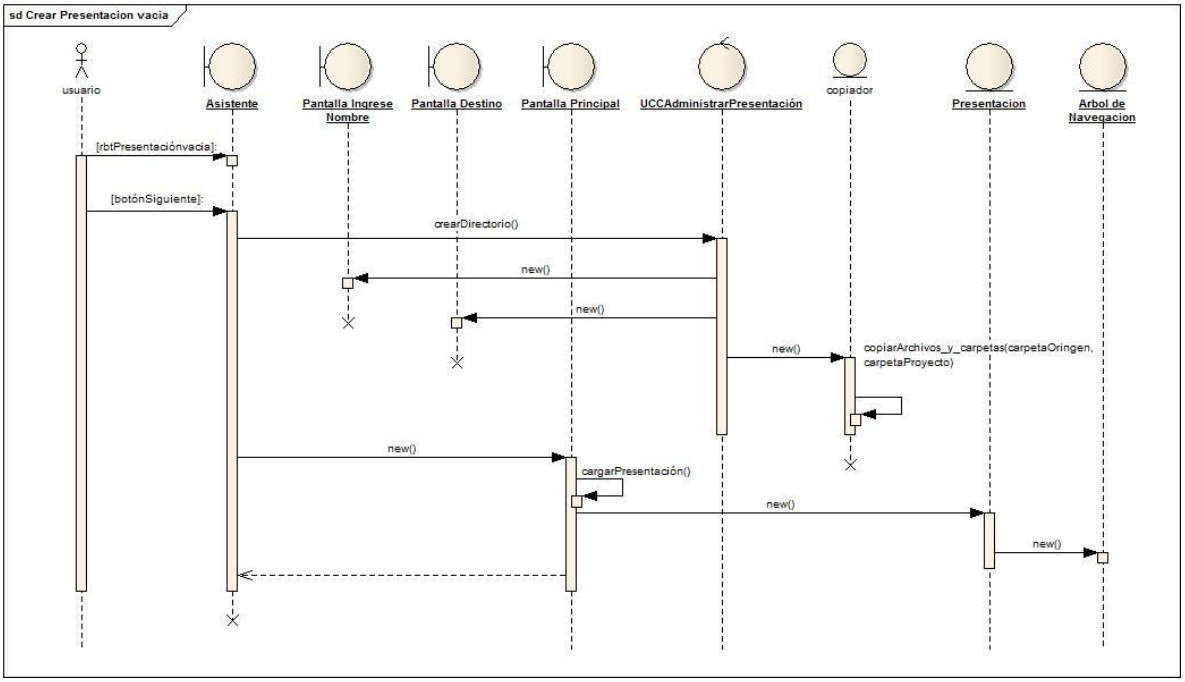


Figura 44 DS Flujo Normal Crear presentación vacía

Flujo Alterno

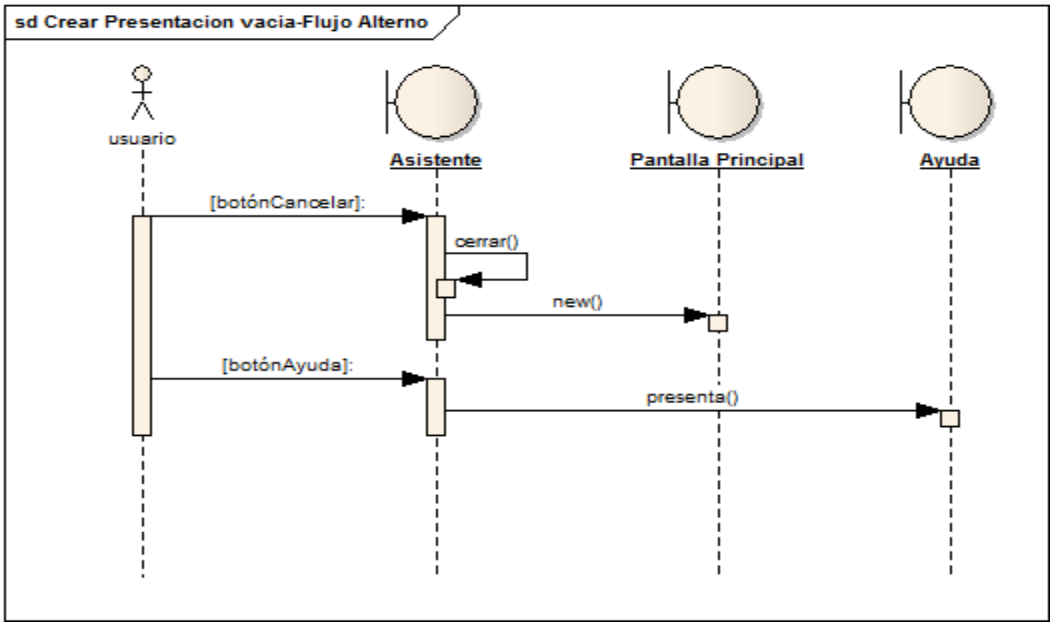


Figura 45 DS Flujo Alterno Crear presentación vacía

8.5.2. Modelado de secuencia del Caso de uso Crear presentación con plantilla.

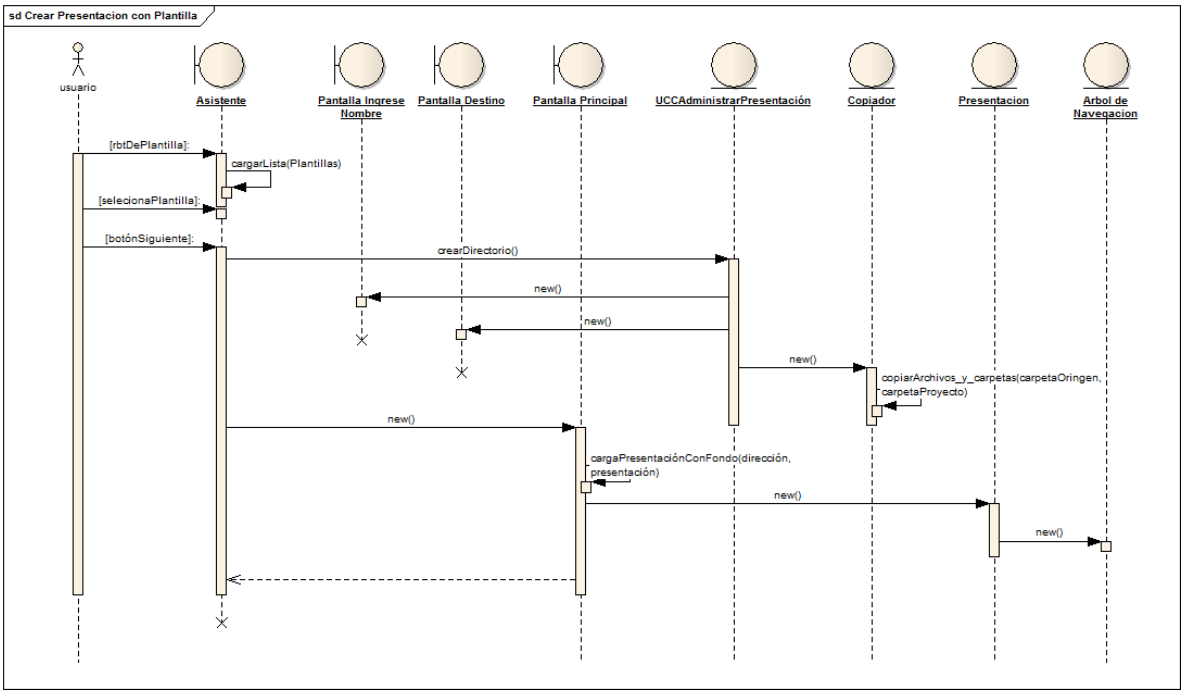


Figura 46 DS Flujo Normal Crear presentación con plantilla

Flujo Alterno

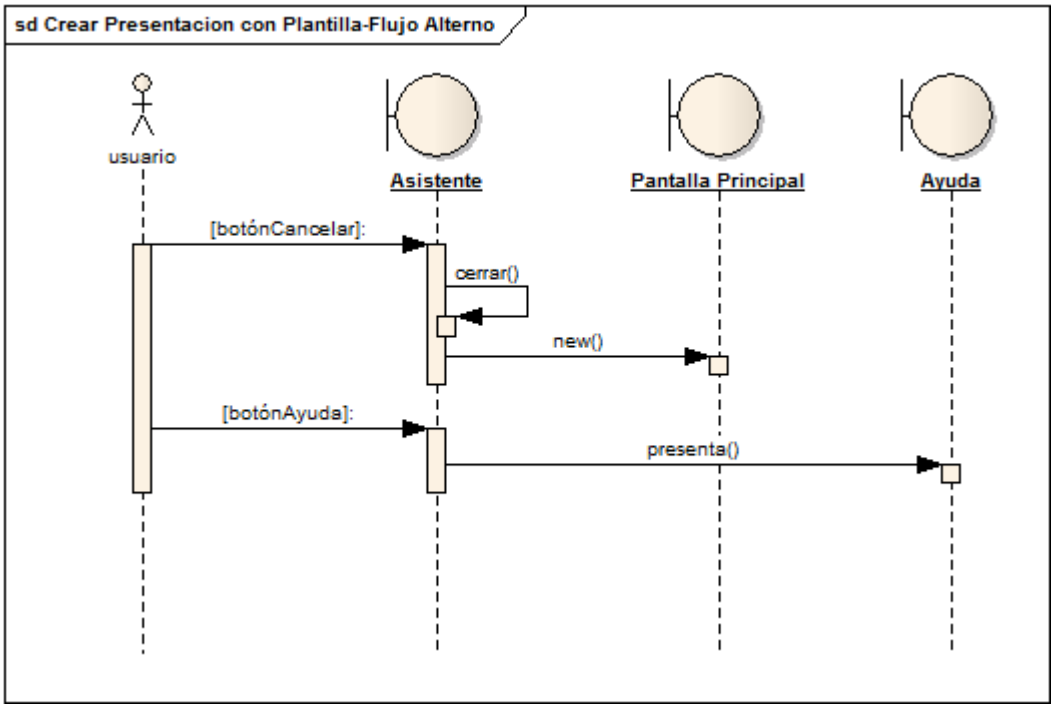


Figura 47 DS Flujo Alterno Crear presentación con plantilla

8.5.3. Modelado de secuencia del Caso de uso Modificar Presentación.

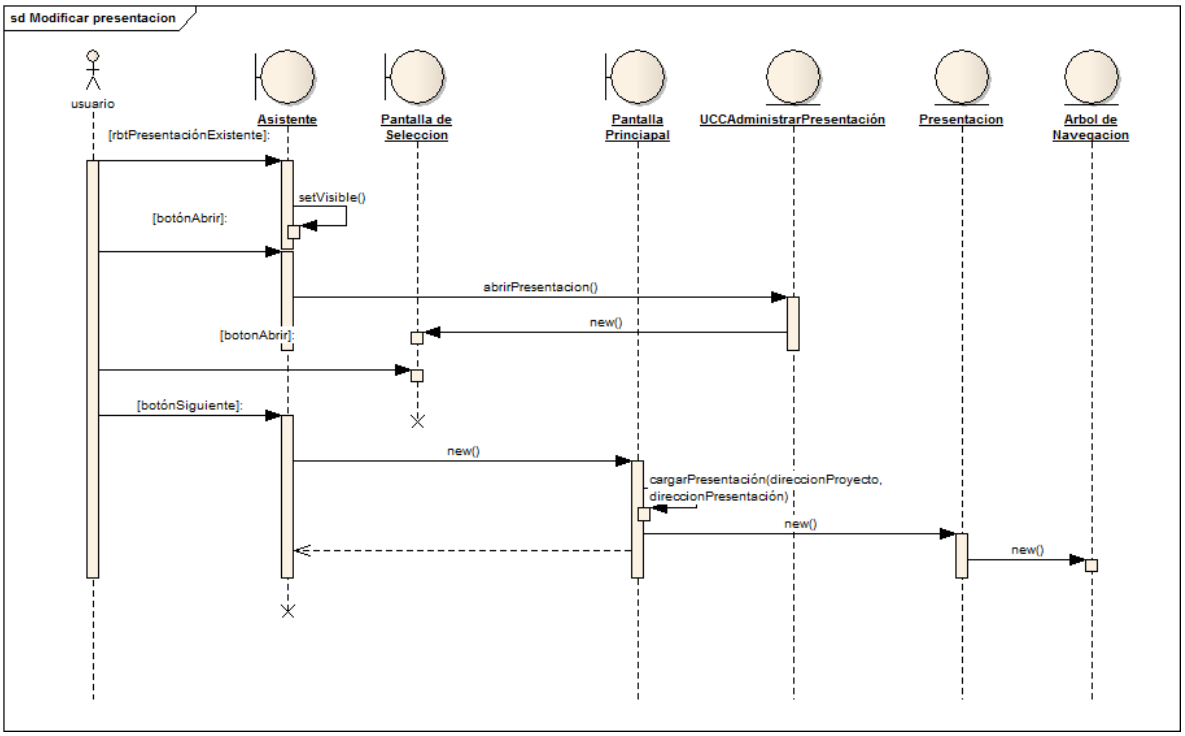


Figura 48 DS Flujo Normal Modificar presentación

Flujo Alterno

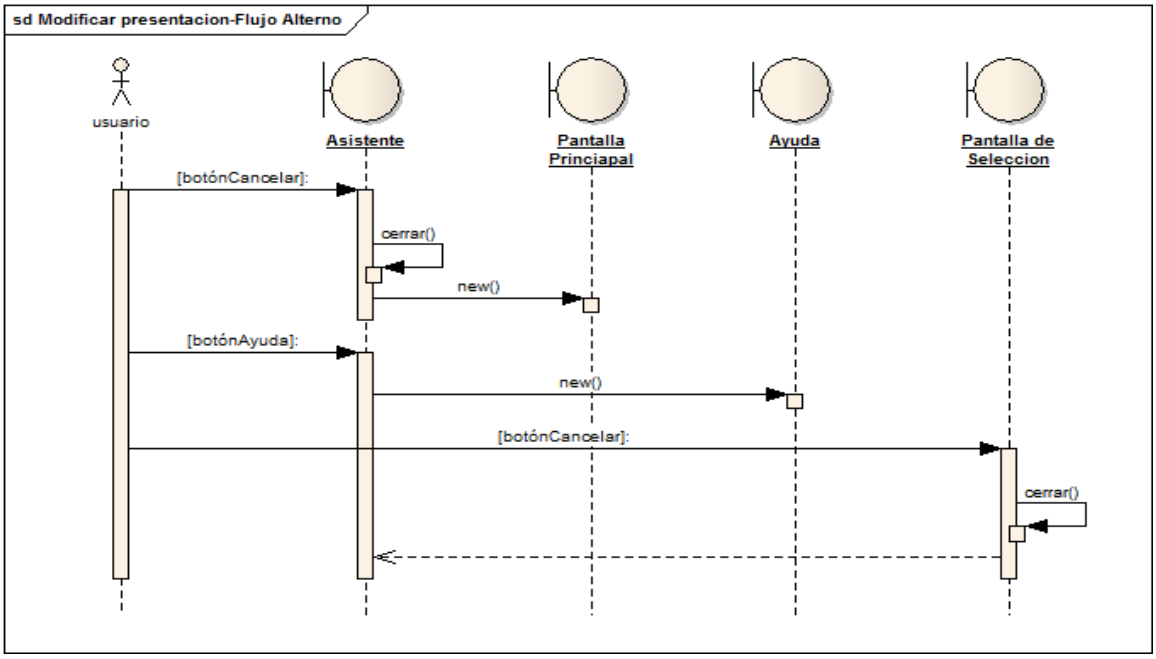


Figura 49 DS Flujo Alterno Modificar presentación

8.5.4. Modelado de secuencia del Caso de uso Guardar Presentación.

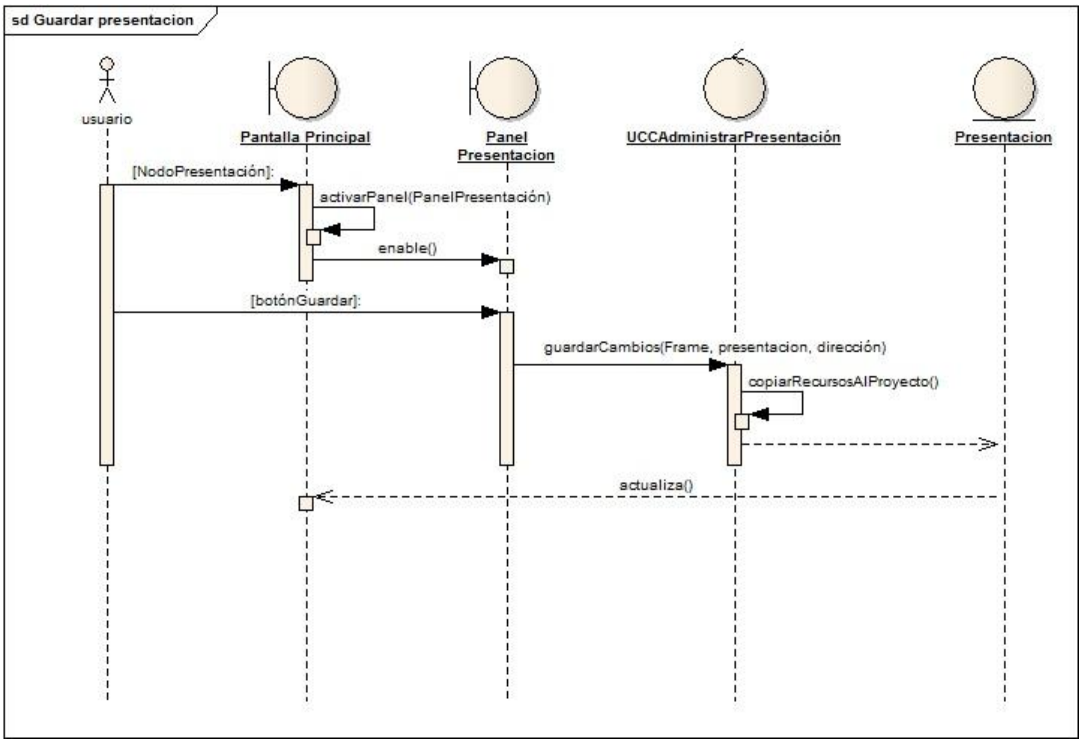


Figura 50 DS Flujo Normal Guardar presentación

Flujo Alterno

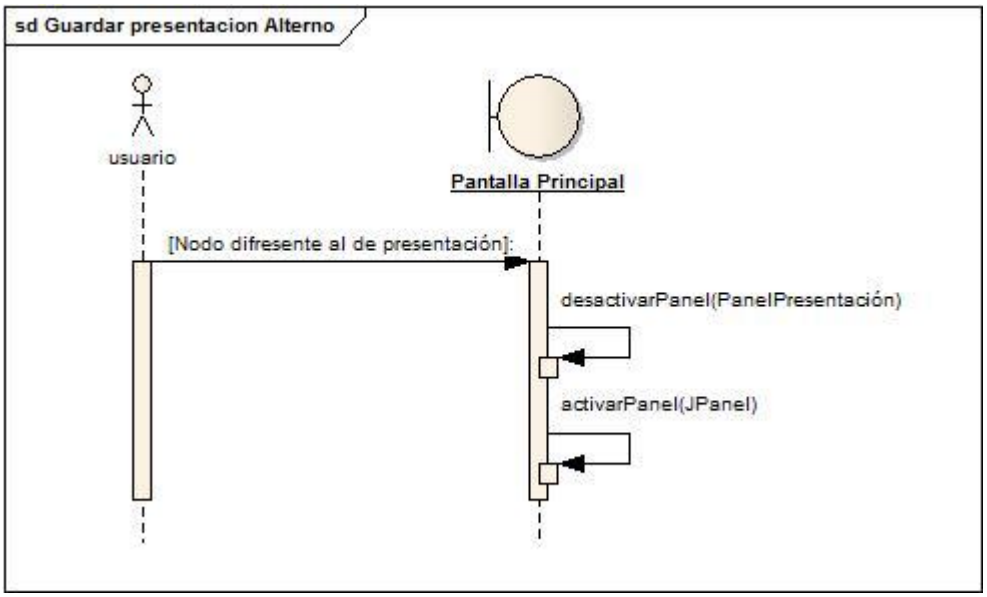


Figura 51 DS Flujo Alterno Guardar Presentación

8.5.5. Modelado de secuencia del Caso de uso Exportar Presentación en formato png.

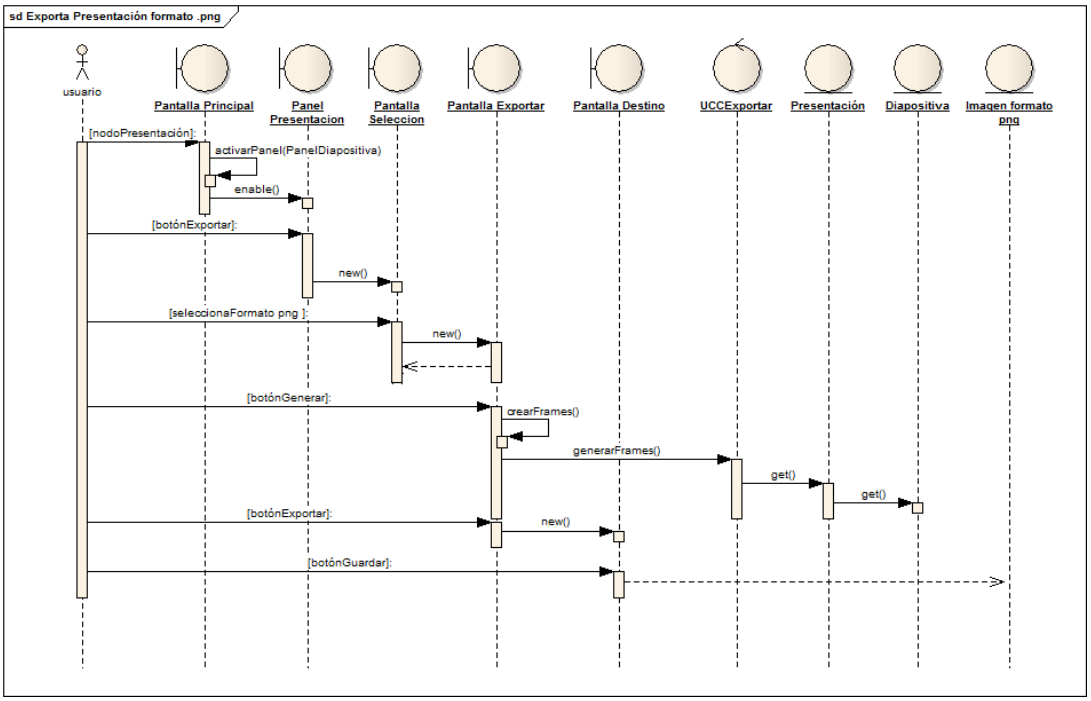


Figura 52 DS Flujo Normal Exportar presentación en formato png

Flujo Alterno

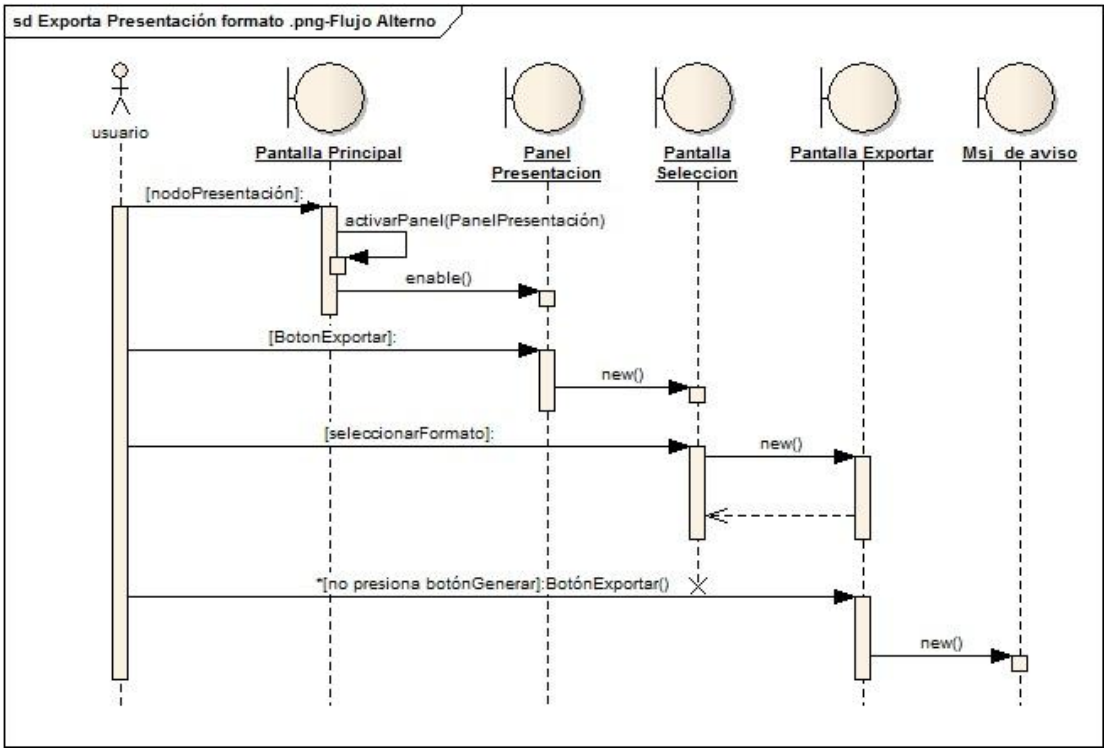


Figura 53 DS Flujo Alterno Exportar presentación en formato png

8.5.6. Modelado de secuencia del Caso de uso Exportar Presentación en formato scorm.

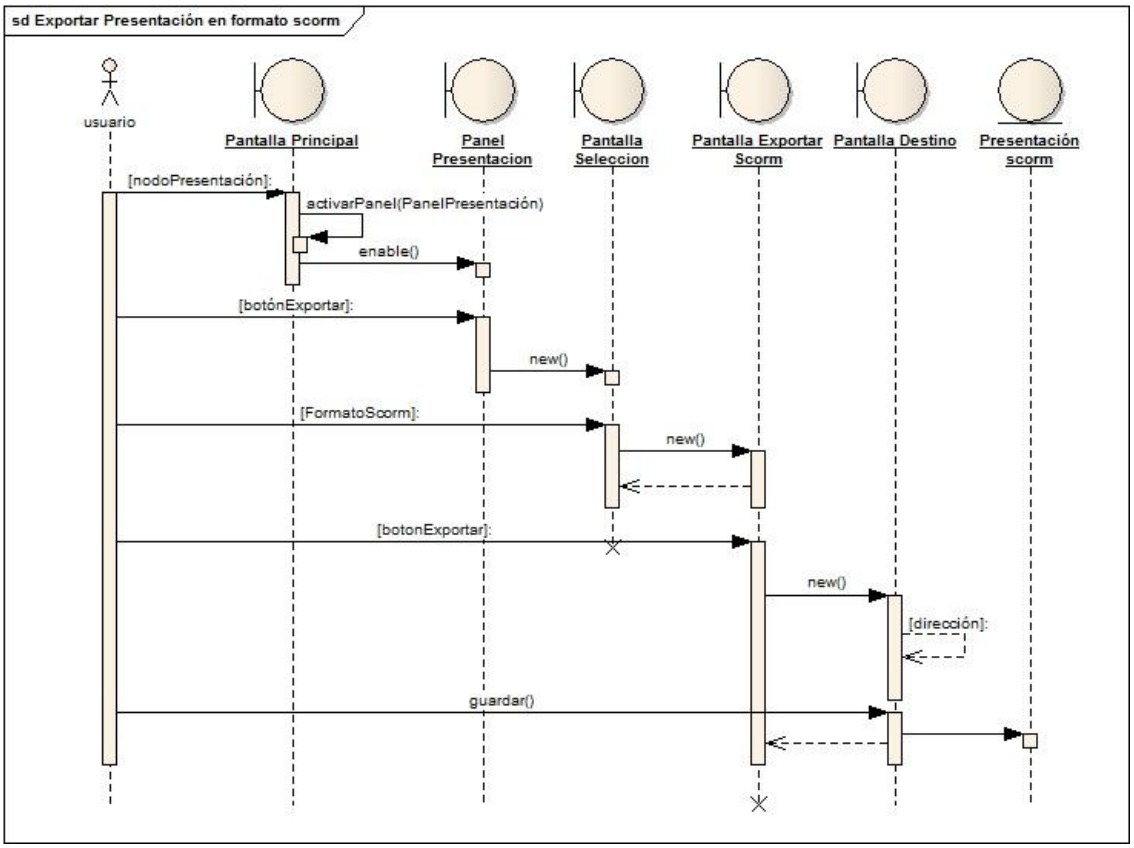


Figura 54 DS Flujo Normal Exportar presentación en formato scorm

Flujo Alterno

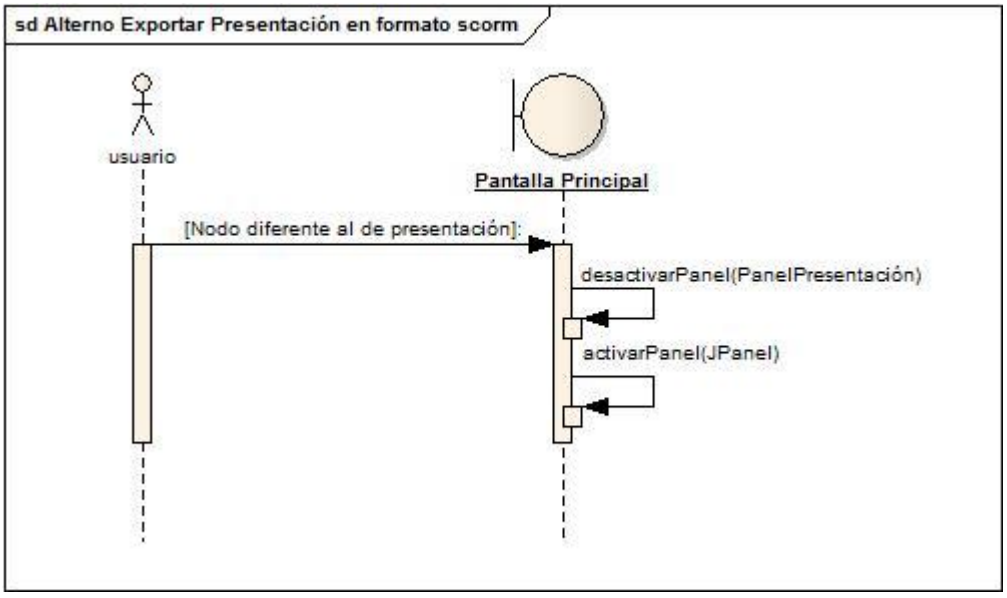


Figura 55 DS Flujo Alterno Exportar Presentación en formato scorm

8.5.7. Modelado de secuencia del Caso de uso Seleccionar Fondo para la Presentación.

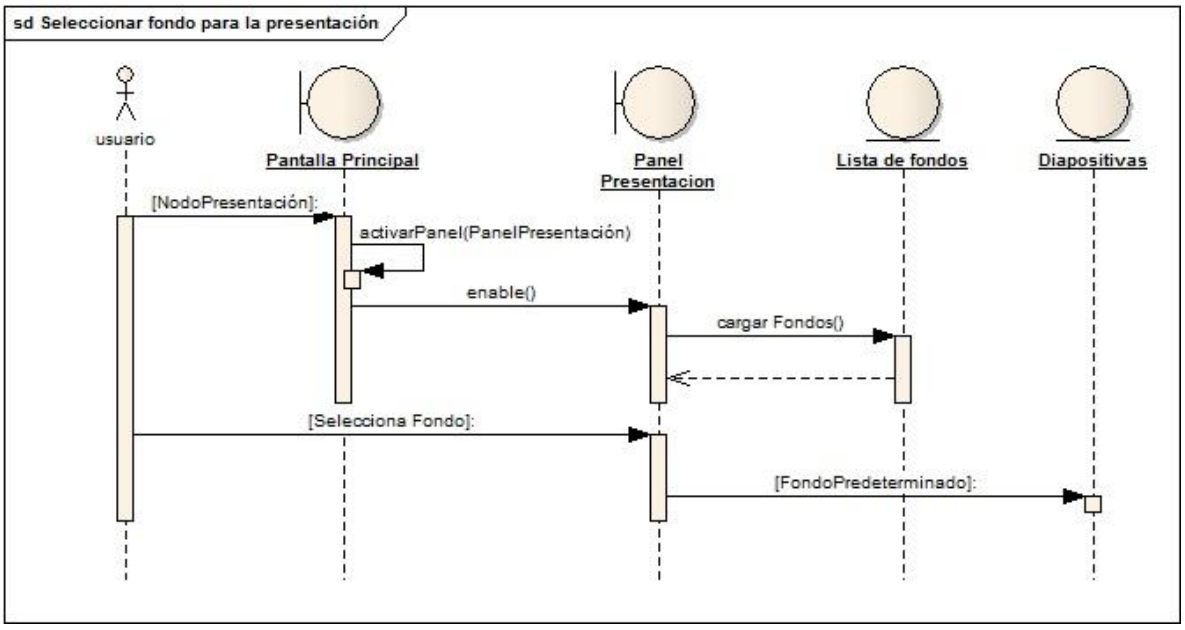


Figura 56 DS Flujo Normal Seleccionar fondo para presentación

Flujo Alterno

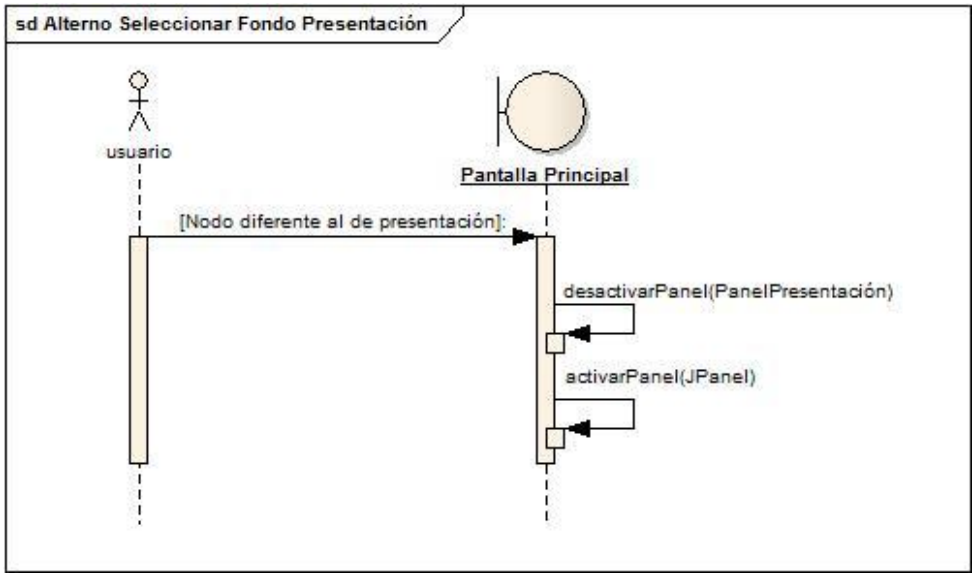


Figura 57 DS Flujo Alterno Seleccionar fondo para presentación.

8.5.8. Modelado de secuencia del Caso de uso Generar vista Previa.

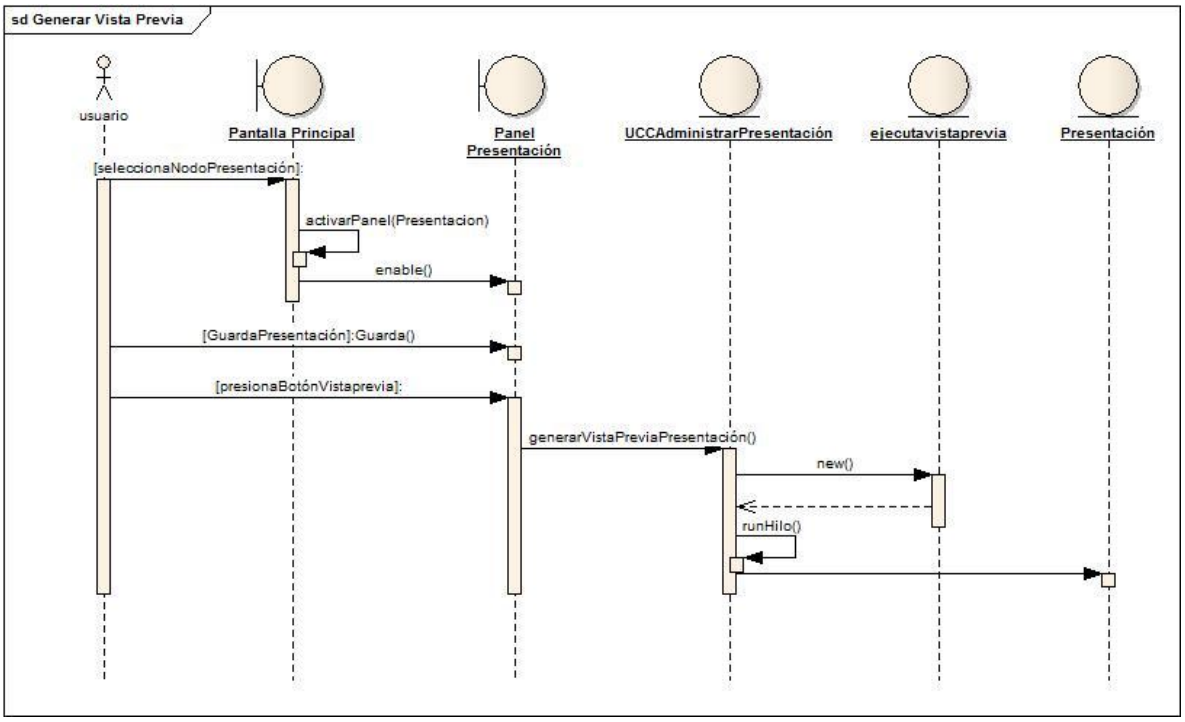


Figura 58 DS Flujo Normal Generar vista previa

Flujo Alterno

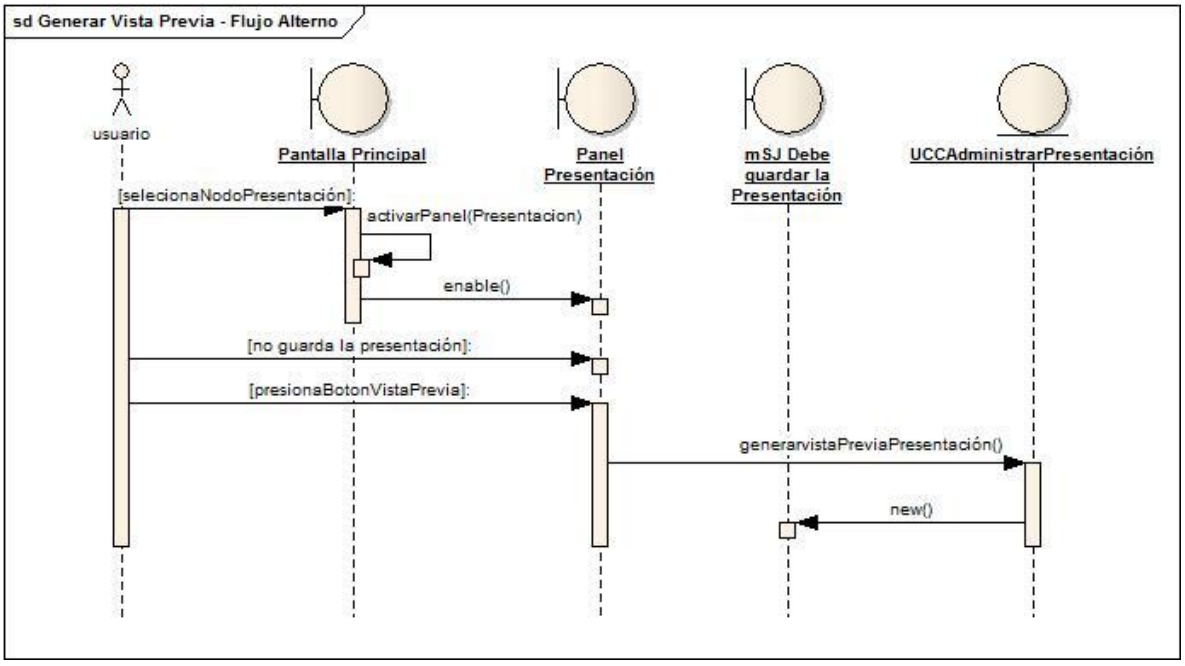


Figura 59 DS Flujo Alterno Generar Vista previa

8.5.9. Modelado de secuencia del Caso de uso Imprimir presentación.

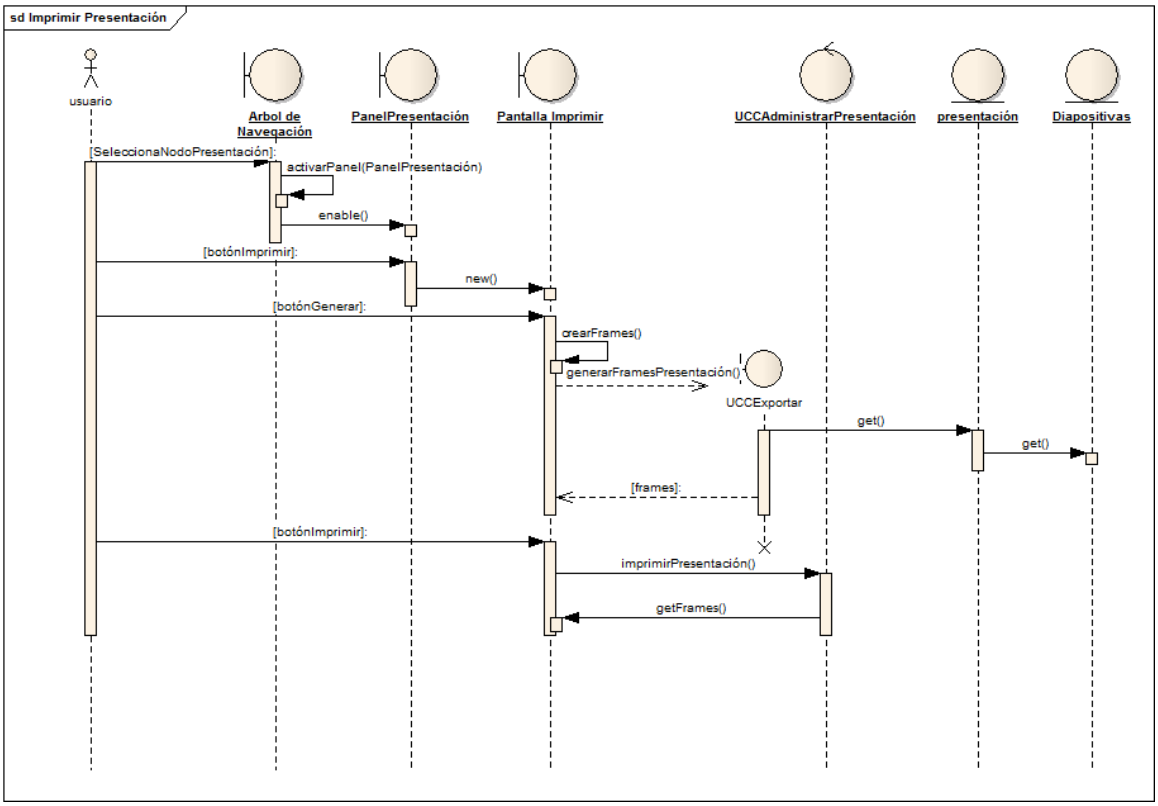


Figura 60 DS Flujo Normal Imprimir presentación

Flujo Alterno

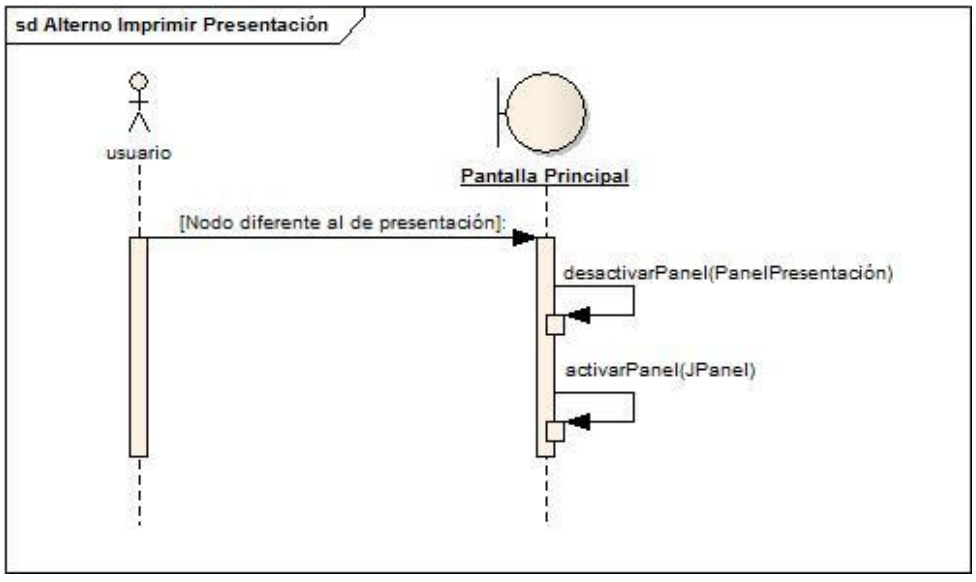


Figura 61 DS Flujo Alterno Imprimir Presentación

8.5.10. Modelado de secuencia del Caso de uso Agregar diapositiva en blanco.

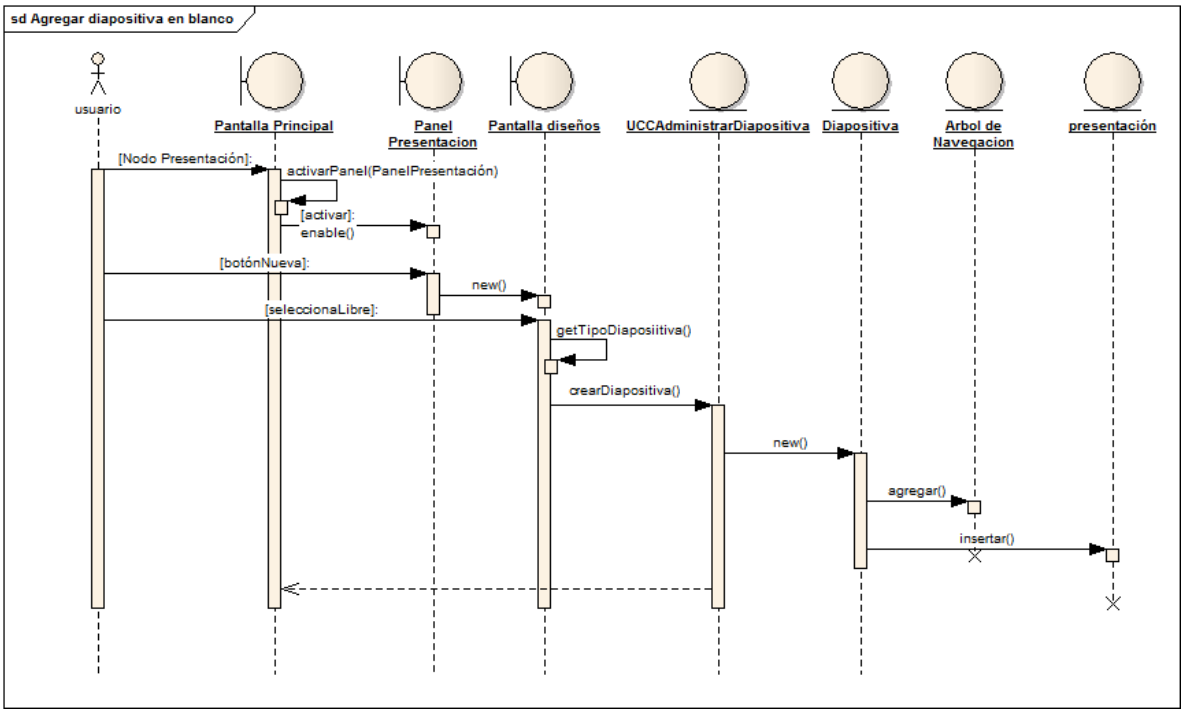


Figura 62 DS Flujo Normal Agregar Diapositiva en blanco

Flujo Alterno

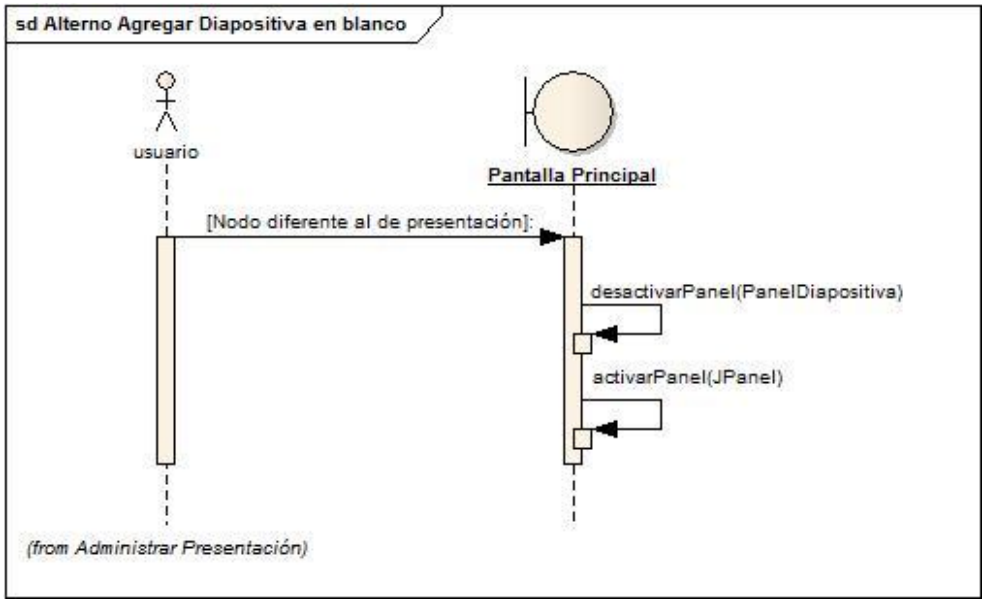


Figura 63 DS Flujo Alterno Agregar Diapositiva en Blanco

8.5.11. Modelado de secuencia del Caso de uso Agregar diapositiva con diseño.

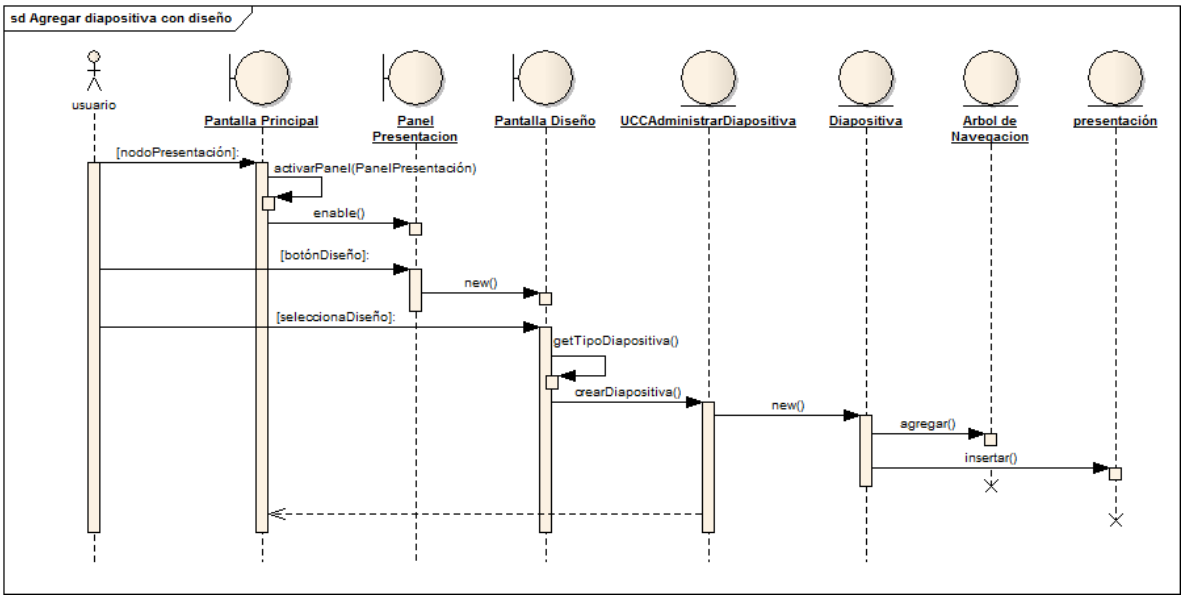


Figura 64 DS Flujo Normal Agregar Diapositiva con Diseño

Flujo Alterno

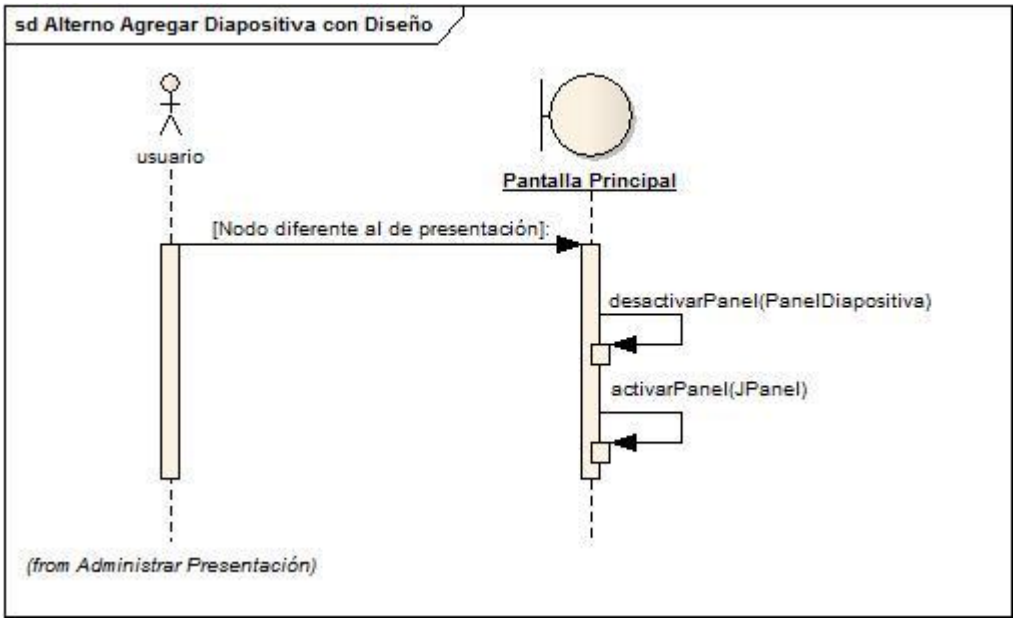


Figura 65 DS Flujo Alterno Agregar Diapositiva con Diseño

8.5.12. Modelado de secuencia del Caso de uso Agregar color de fondo.

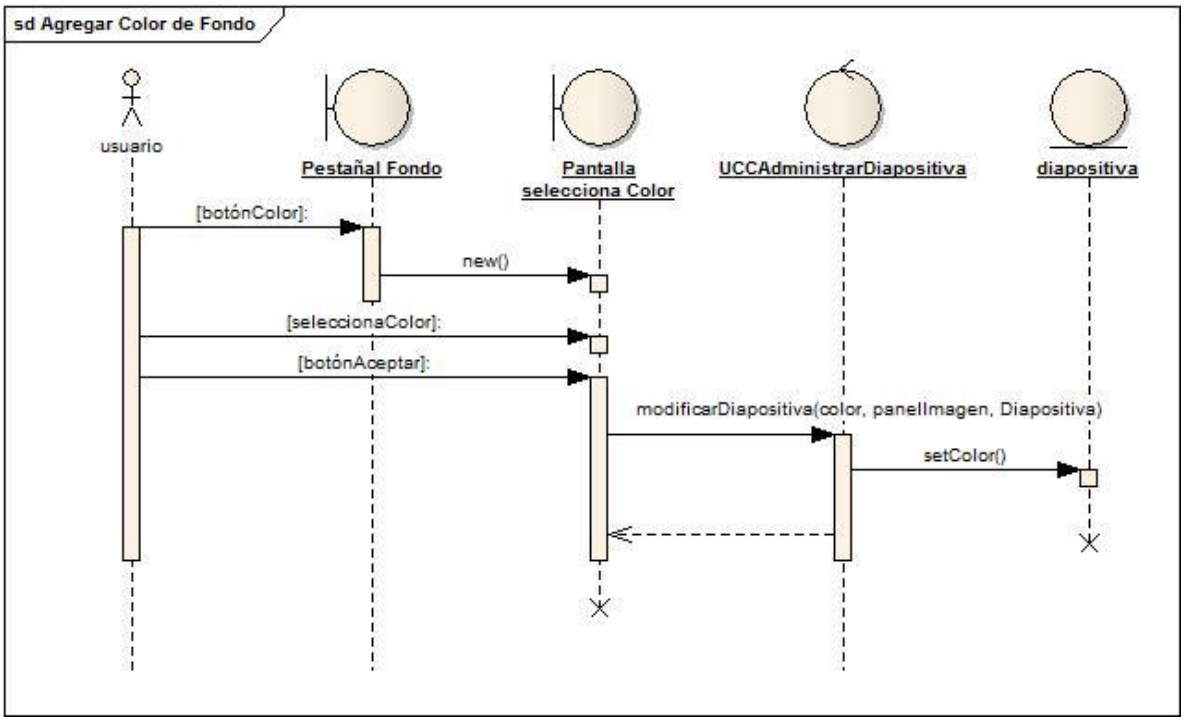


Figura 66 DS Flujo Normal Agregar color de fondo

Flujo Alterno

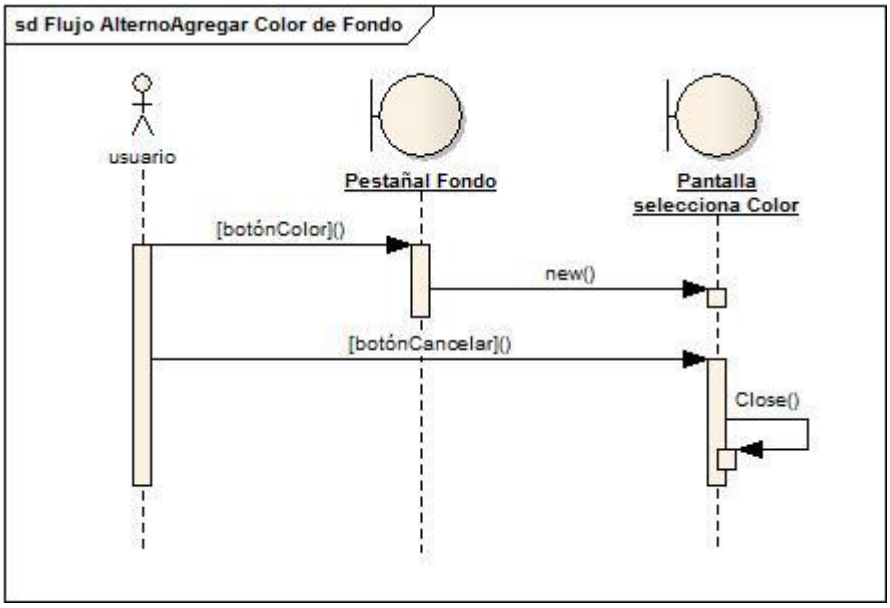


Figura 67 DS Flujo Alterno Agregar Color de Fondo

8.5.13. Modelado de secuencia del Caso de uso Agregar imagen de fondo.

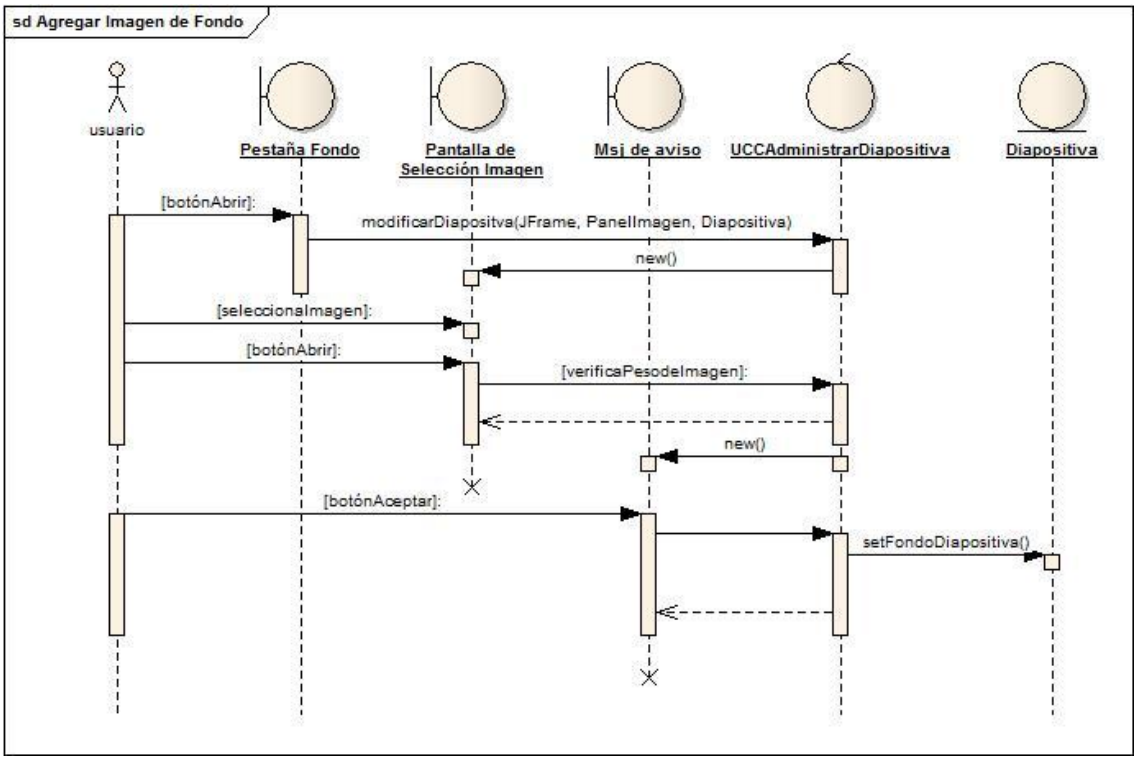


Figura 68 DS Flujo Normal Agregar imagen de fondo

Flujo Alterno

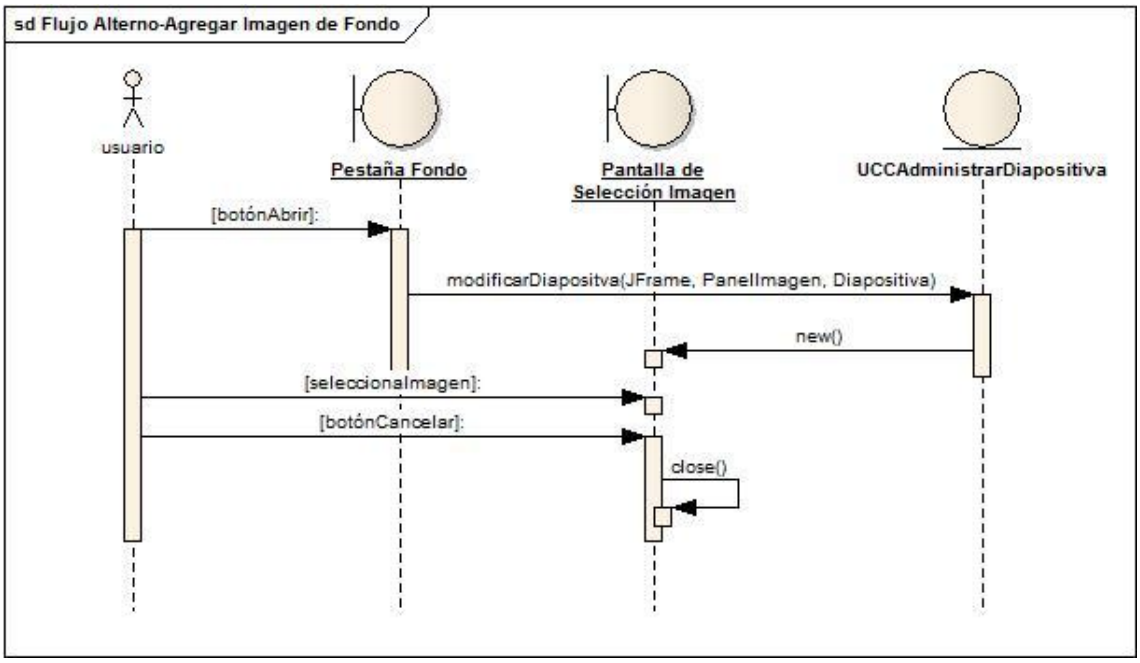


Figura 69 DS Flujo Alterno Agregar Imagen de Fondo

8.5.14. Modelado de secuencia del Caso de uso Eliminar diapositiva.

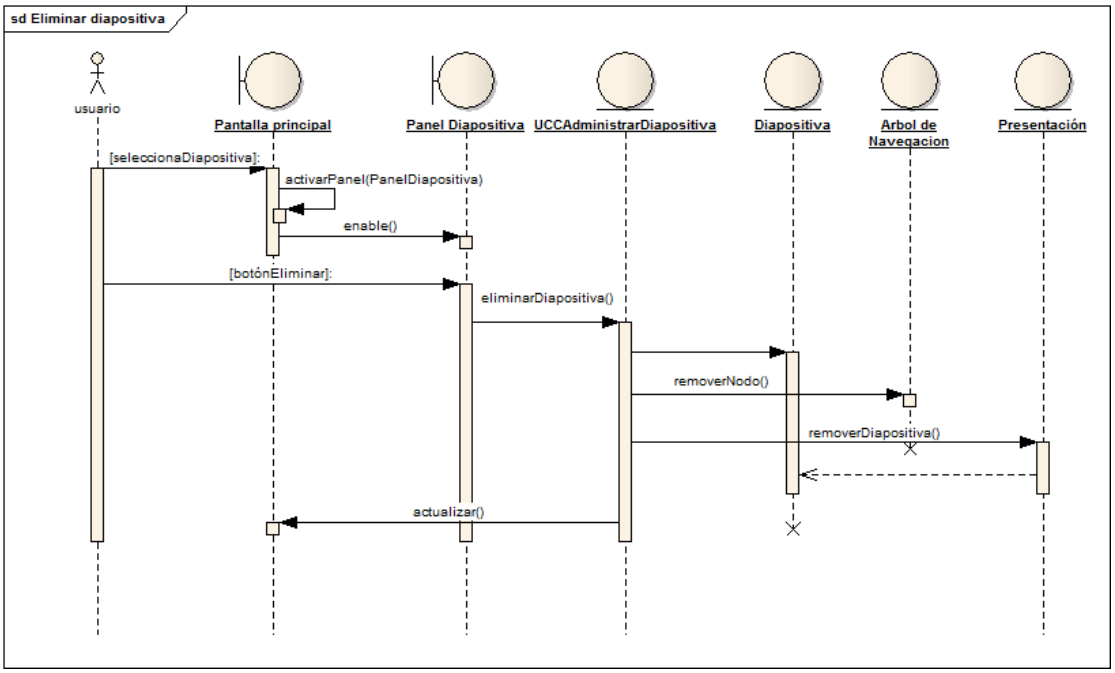


Figura 70 DS Flujo Normal Eliminar Diapositiva

Flujo Alterno

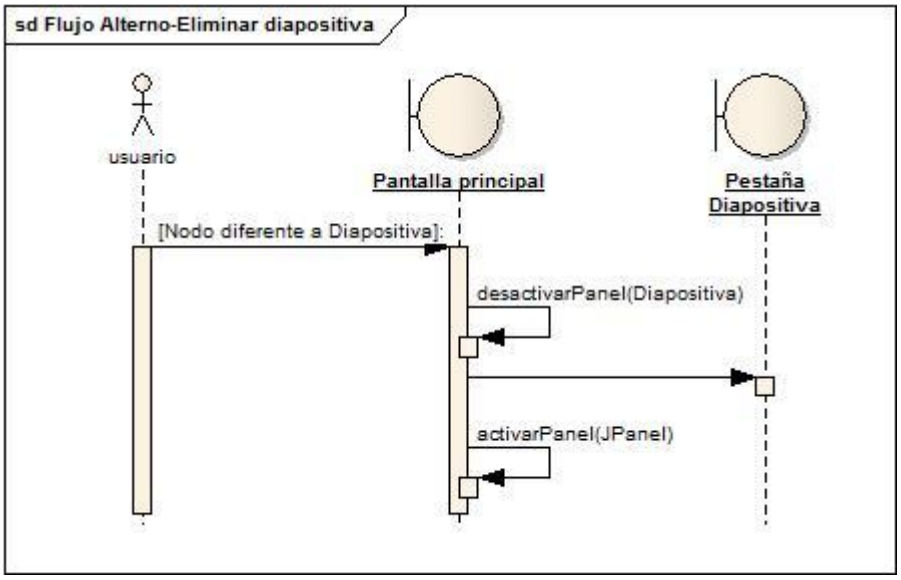


Figura 71 DS Flujo Alterno Eliminar Diapositiva

8.5.15. Modelado de secuencia del Caso de uso Exportar diapositiva en formato png o gif.

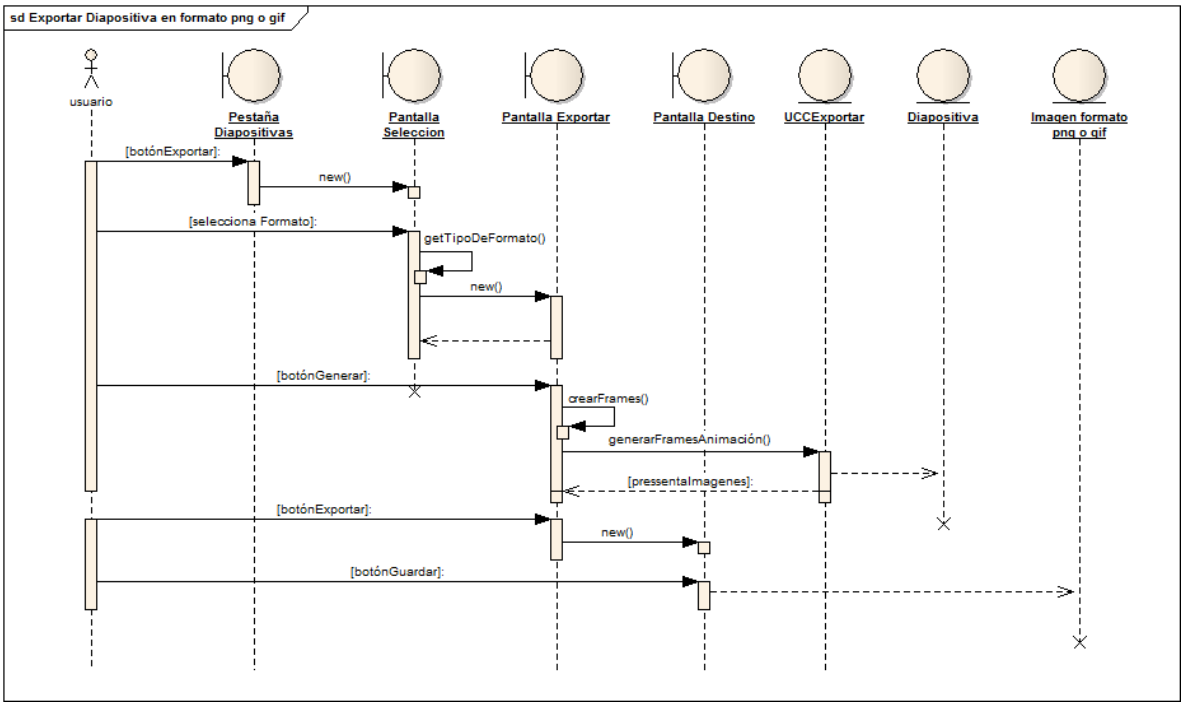


Figura 72 DS Flujo Normal Exportar diapositiva en formato png o gif

Flujo Alterno

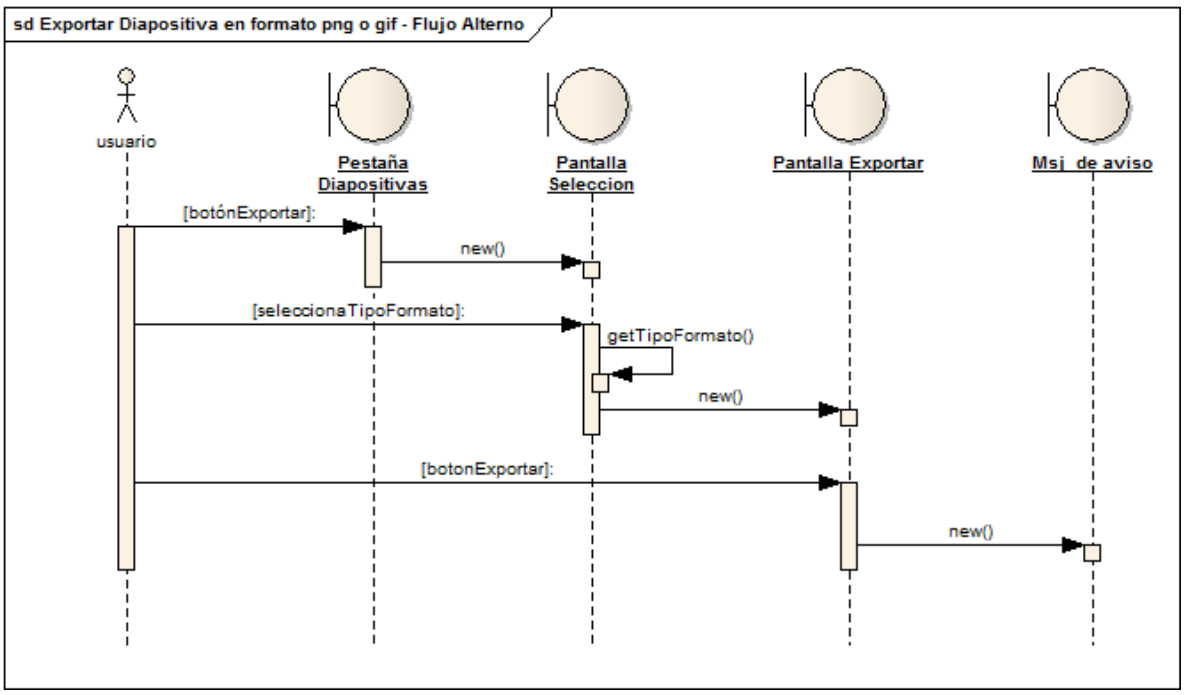


Figura 73 DS Flujo Alterno Exportar diapositiva en formato png o gif

8.5.16. Modelado de secuencia del Caso de uso Generar vista previa efecto.

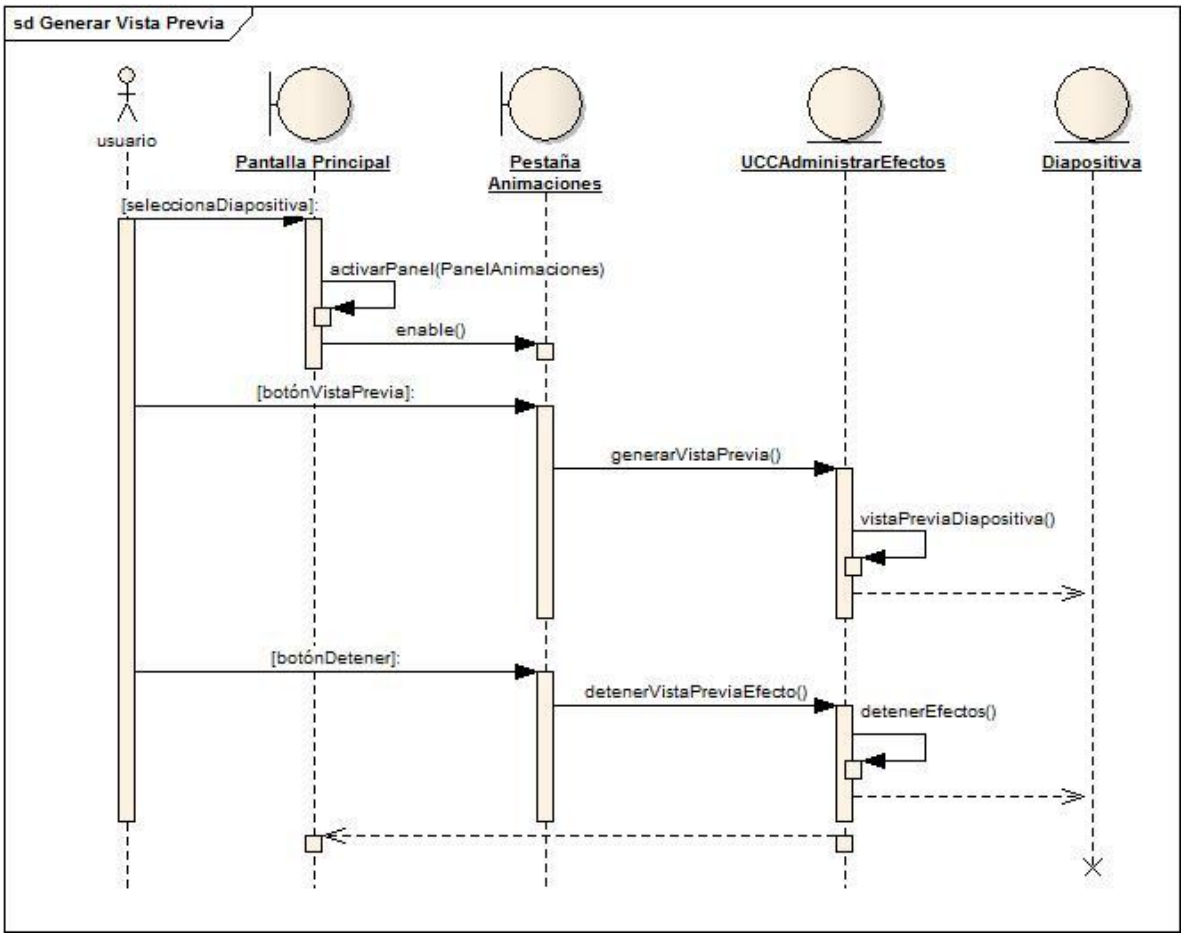


Figura 74 DS Flujo Normal Generar vista previa efecto

Flujo Alterno

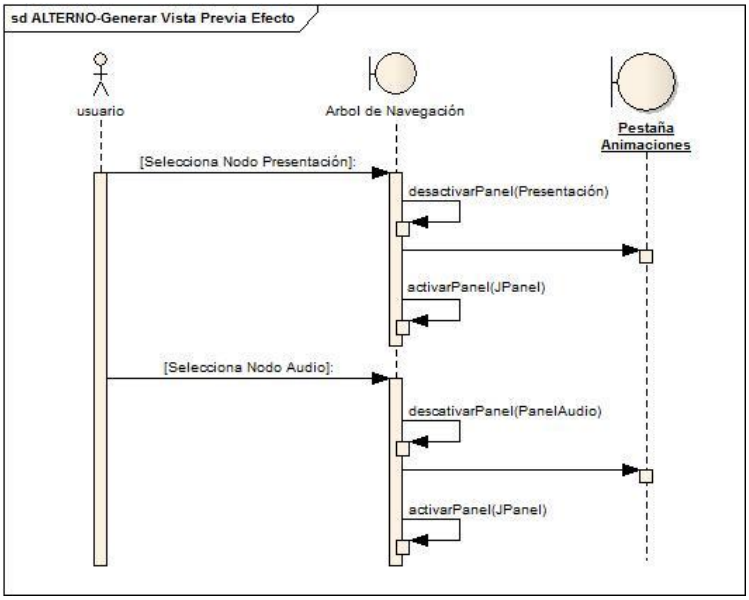


Figura 75 DS Flujo Alterno Generar Vista Previa Efecto

8.5.17. Modelado de secuencia del Caso de uso Agregar componente gráfico.

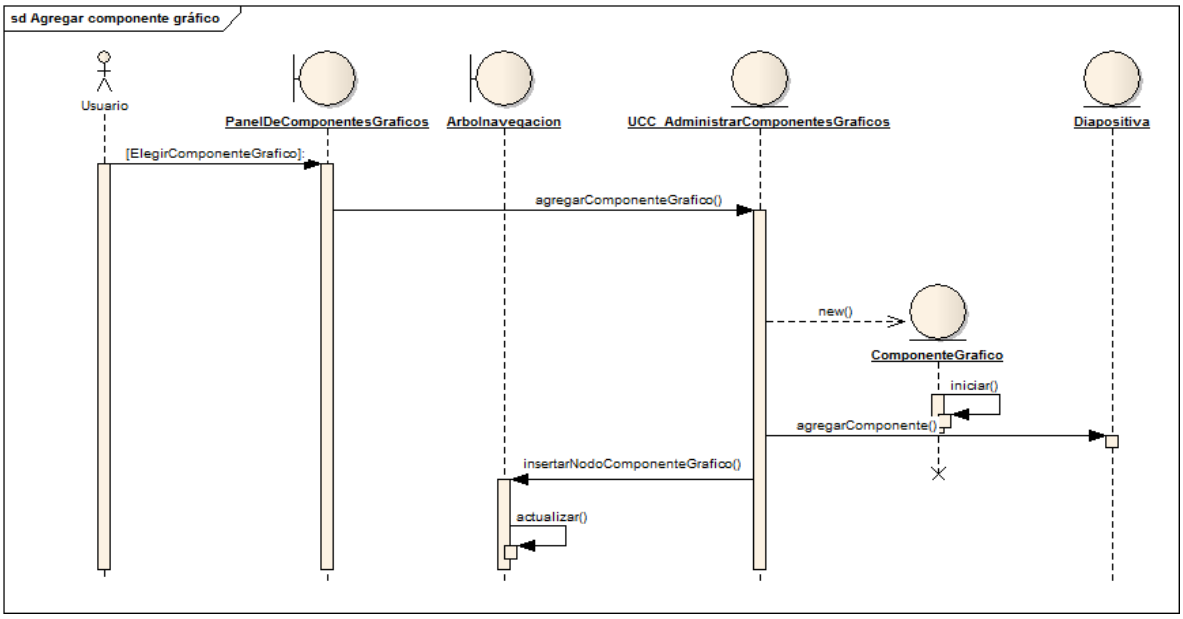


Figura 76 DS Flujo Normal Agregar componente gráfico

Flujo Alterno

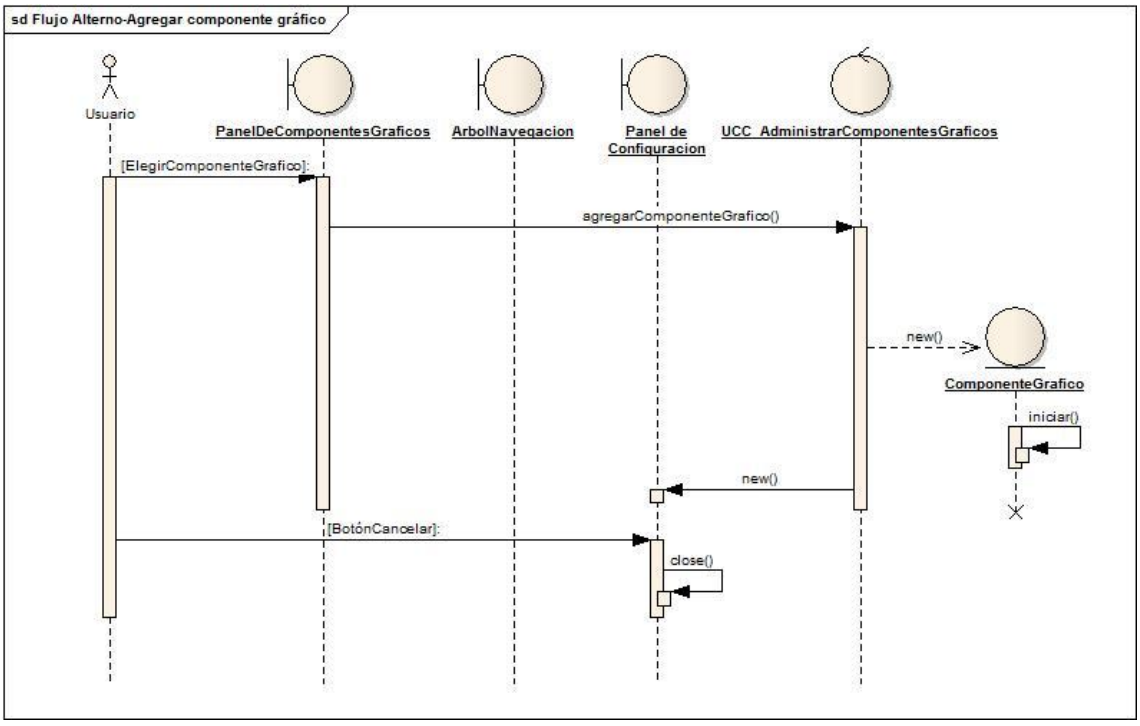


Figura 77 DS Flujo Alterno Agregar Componente Grafico

8.5.18. Modelado de secuencia del Caso de uso Modificar componente gráfico.

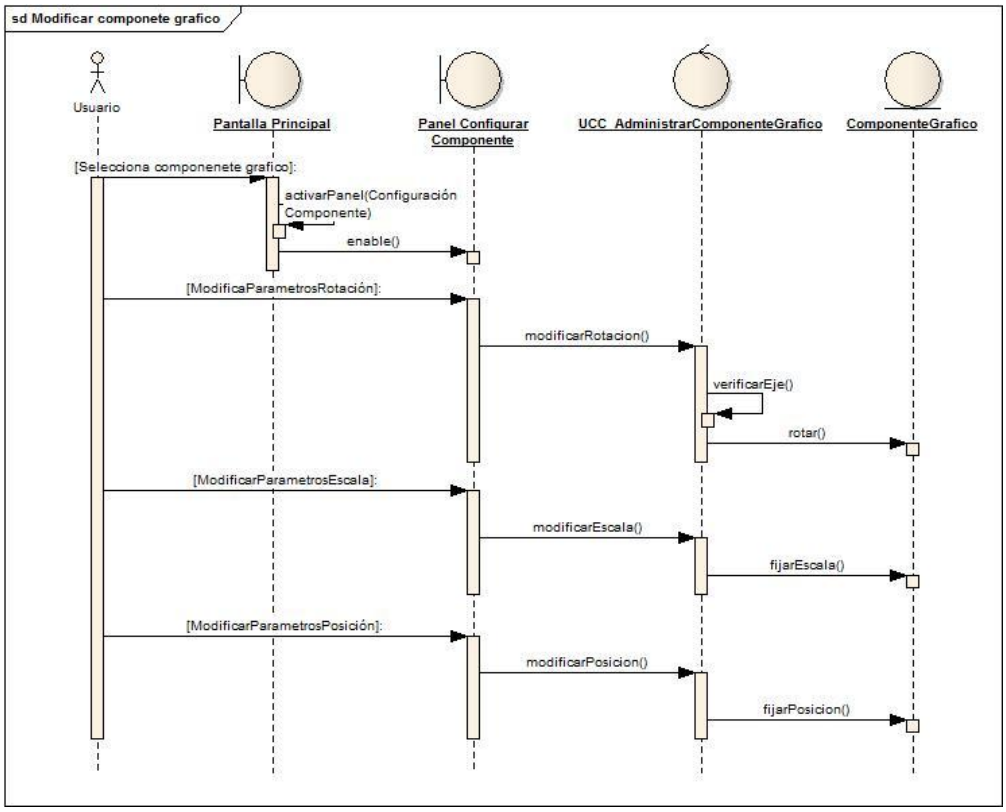


Figura 78 DS Flujo Normal Modificar componente grafico

Flujo Alterno

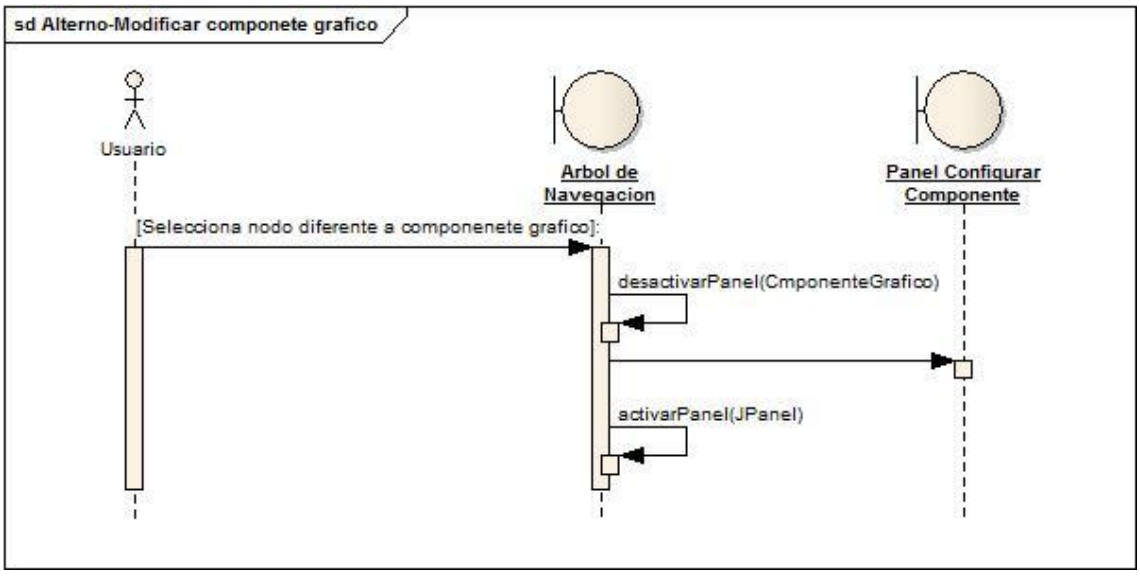


Figura 79 DS Flujo Alterno Modificar Componente Grafico

8.5.19. Modelado de secuencia del Caso de uso Eliminar componente gráfico.

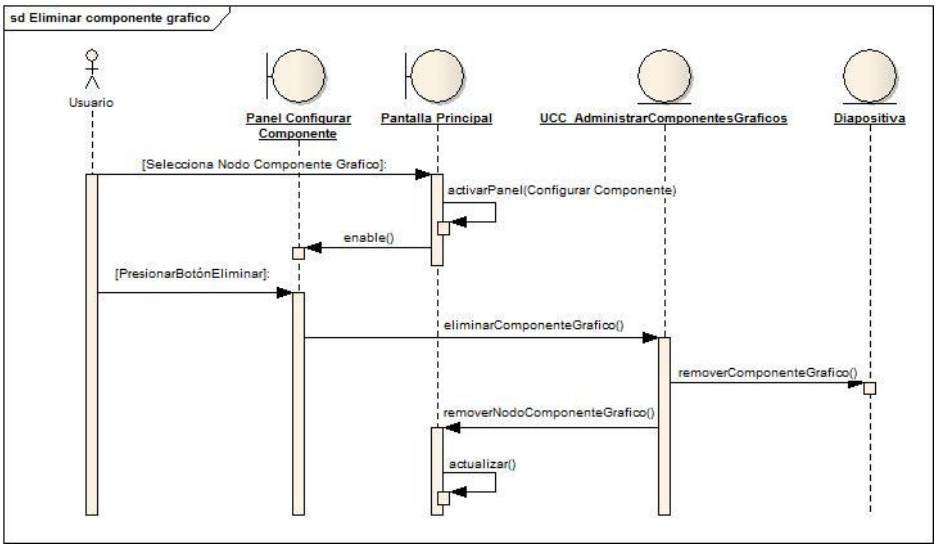


Figura 80 DS Flujo Normal Eliminar componente grafico

Flujo Alterno

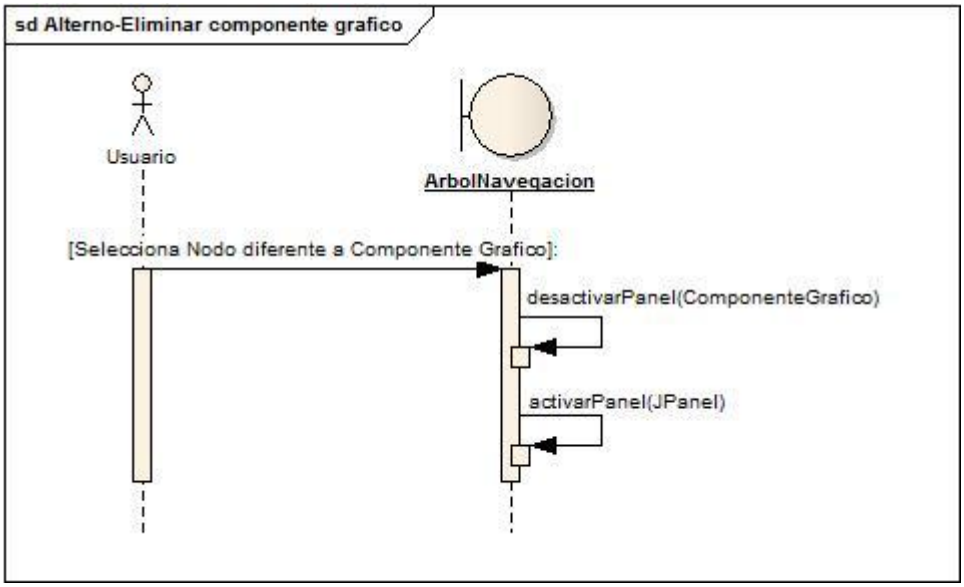


Figura 81 DS Flujo Alterno Eliminar Componente Grafico

8.5.20. Modelado de secuencia del Caso de uso Importar Recursos.

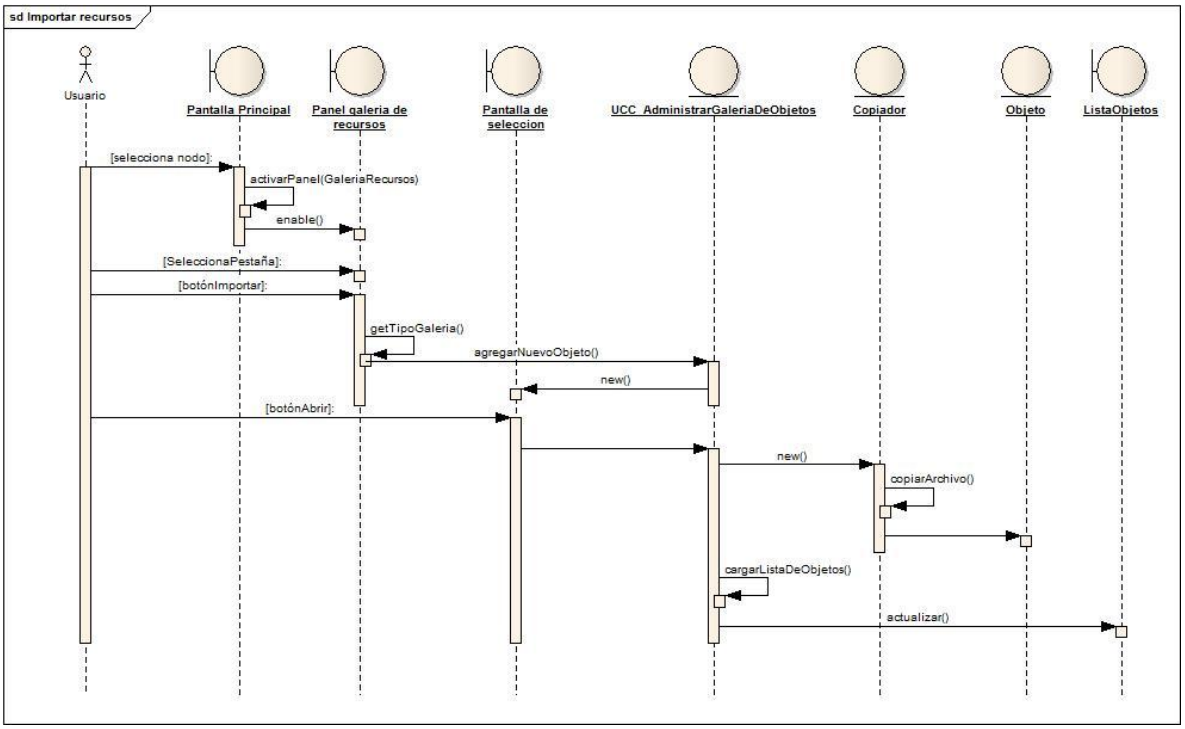


Figura 82 DS Flujo Normal Importar recursos

Flujo Alterno

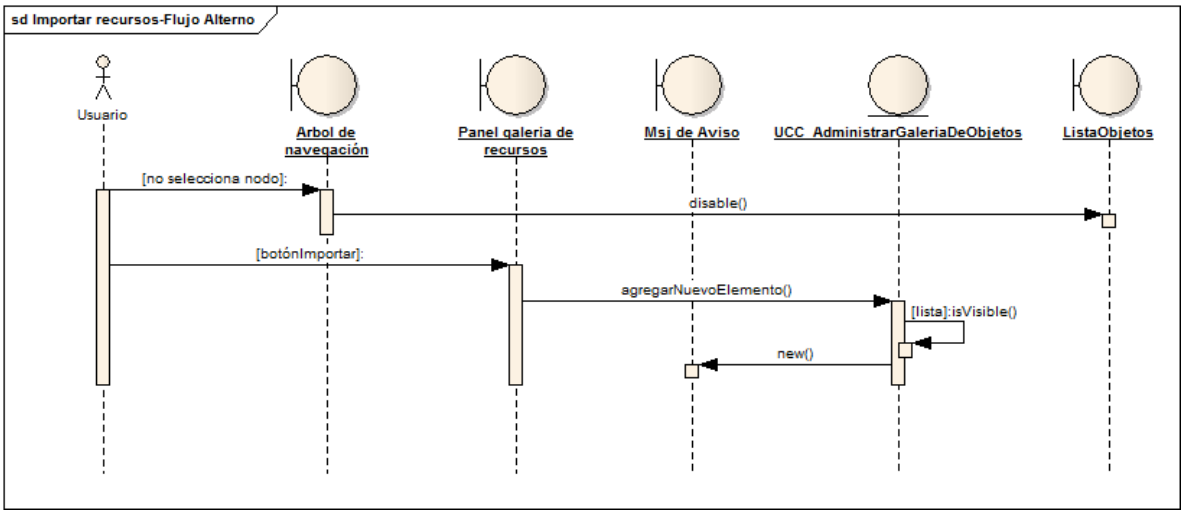


Figura 83 DS Flujo Alterno Importar recursos

8.5.21. Modelado de secuencia del Caso de uso Eliminar Recursos.

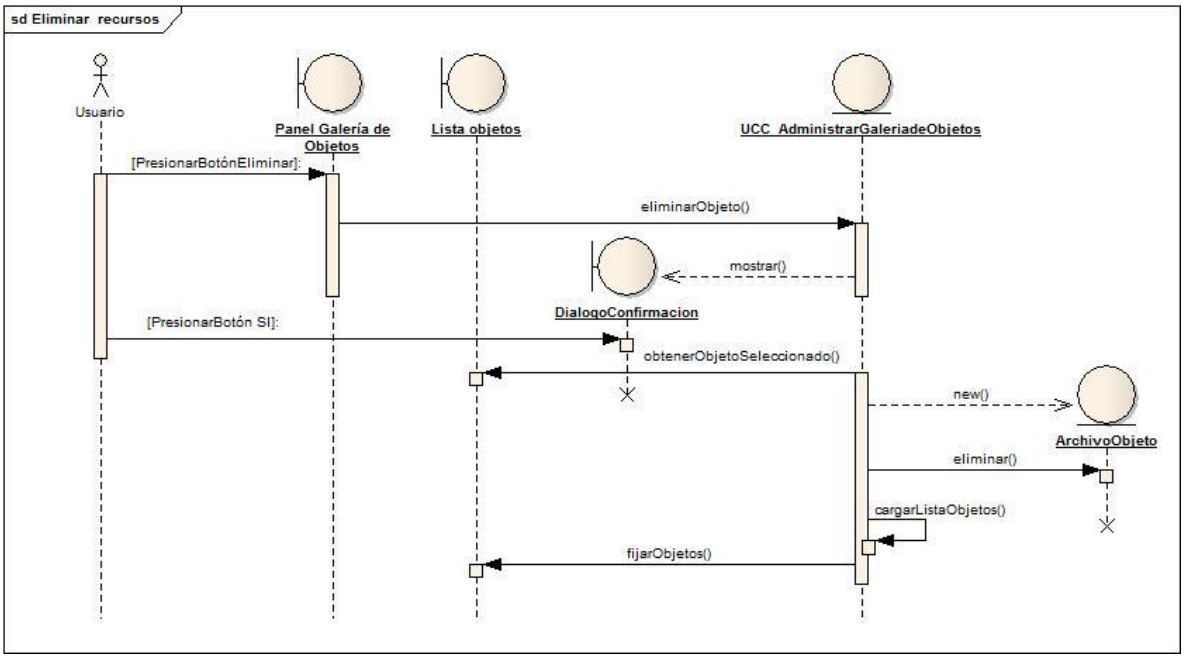


Figura 84 DS Flujo Normal Eliminar Recursos

Flujo Alterno

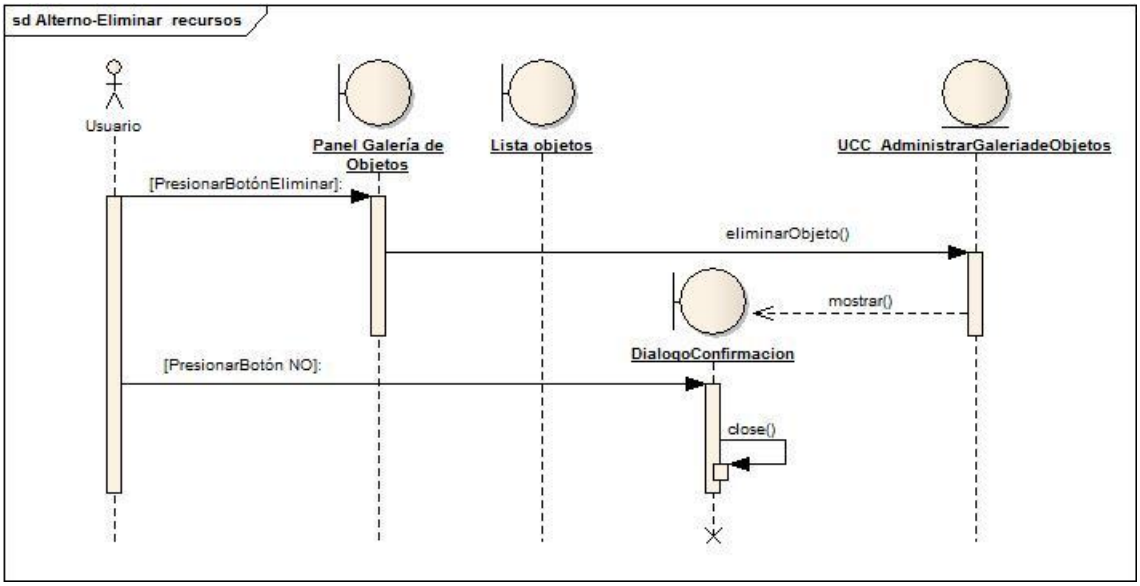


Figura 85 DS Flujo Alterno Eliminar Recursos

8.5.22. Modelado de secuencia del Caso de uso Agregar Sonido.

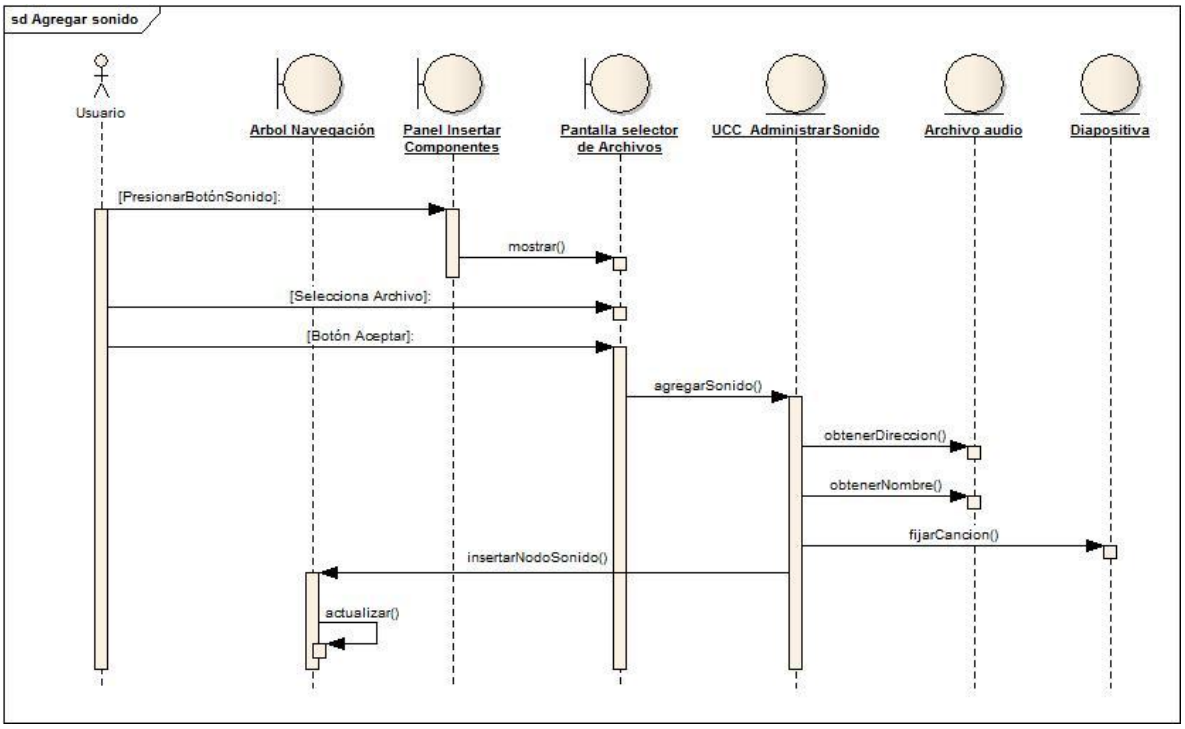


Figura 86 DS Flujo Normal Agregar Sonido

Flujo Alterno

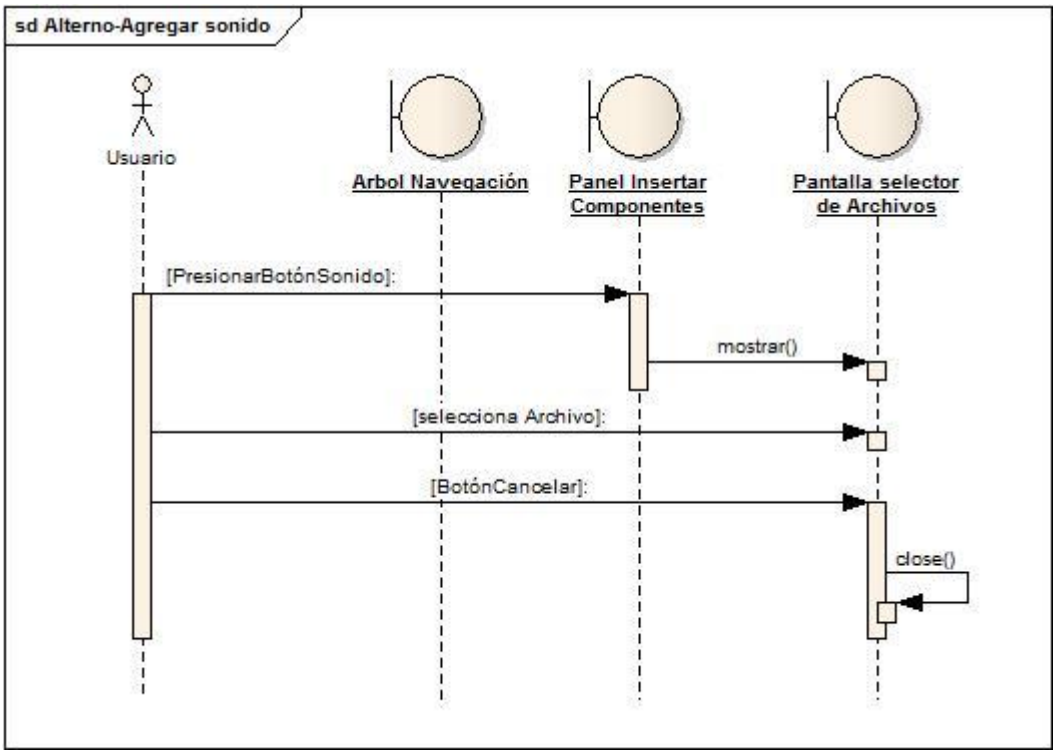


Figura 87 DS Flujo Alterno Agregar Sonido

8.5.23. Modelado de secuencia del Caso de uso Eliminar Sonido.

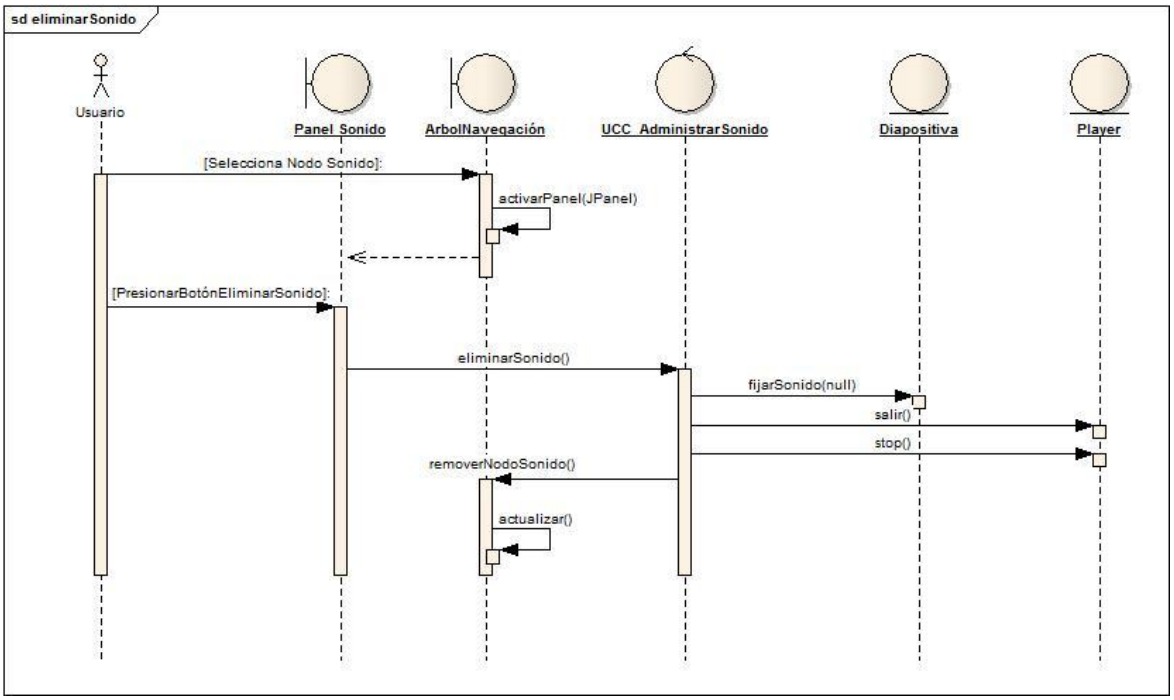


Figura 88 DS Flujo Normal Eliminar Sonido

Flujo Alterno

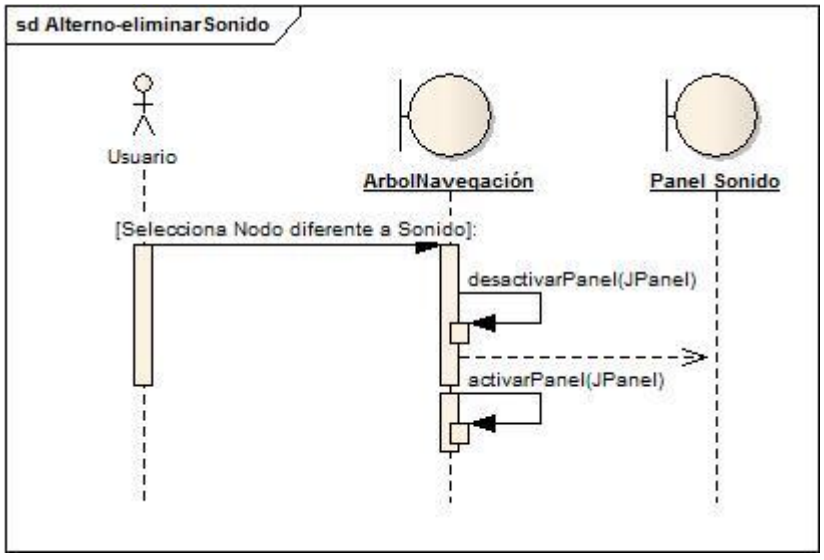


Figura 89 DS Flujo Alterno Eliminar Sonido

8.5.24. Modelado de secuencia del Caso de uso Agregar Material.

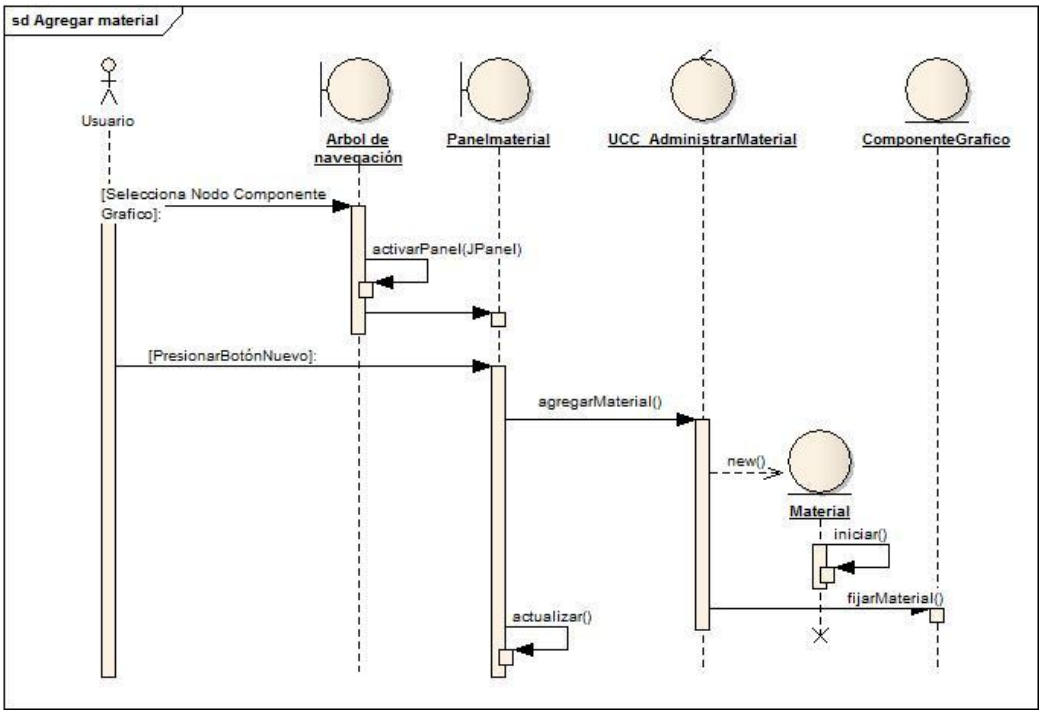


Figura 90 DS Flujo Normal Agregar Material

Flujo Alterno

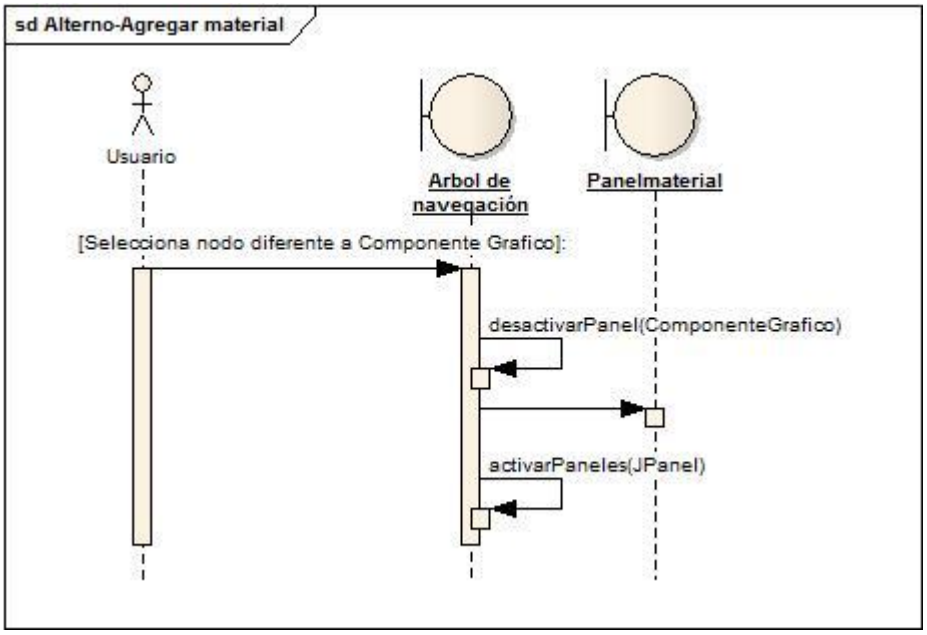


Figura 91 DS Flujo Alterno Agregar Material

8.5.25. Modelado de secuencia del Caso de uso Modificar Configuración de material.

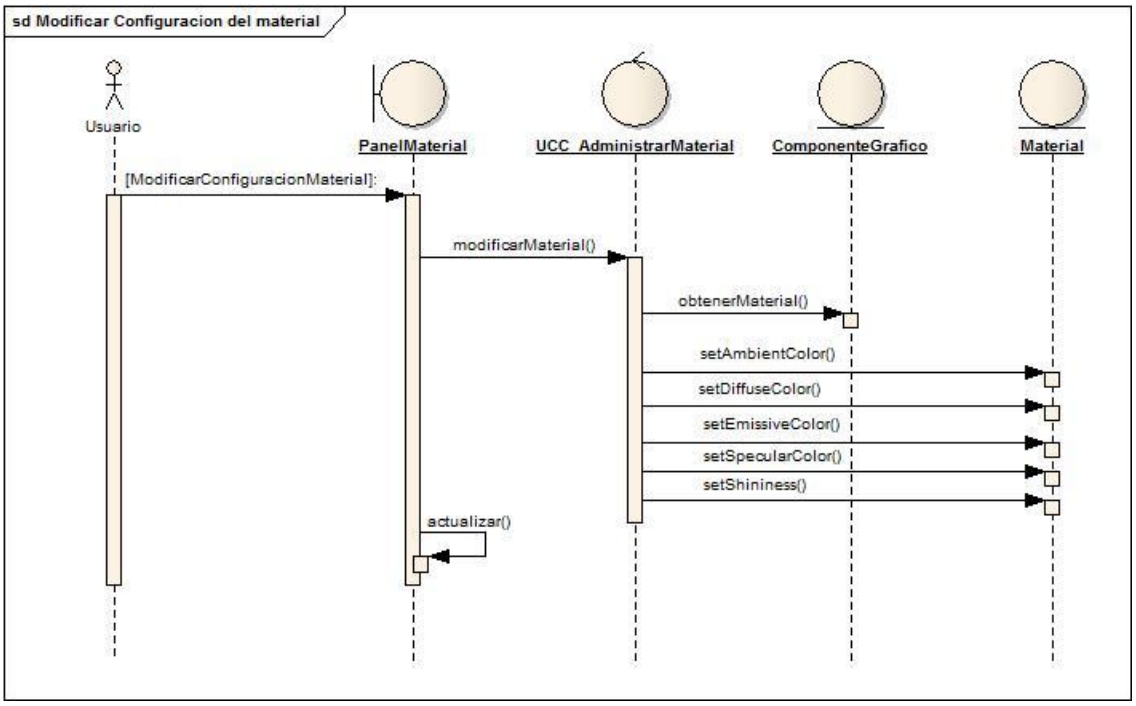


Figura 92 DS Flujo Normal Modificar Configuración de Material

Flujo Alterno

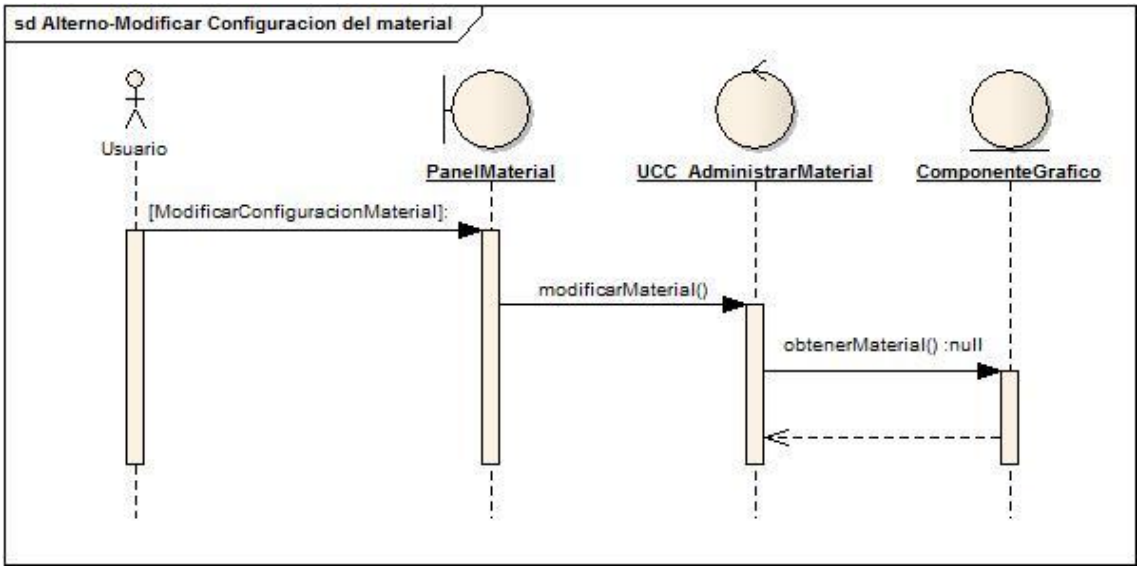


Figura 93 DS Flujo Alterno Modificar Configuración de Material

8.5.26. Modelado de secuencia del Caso de uso Eliminar Material.

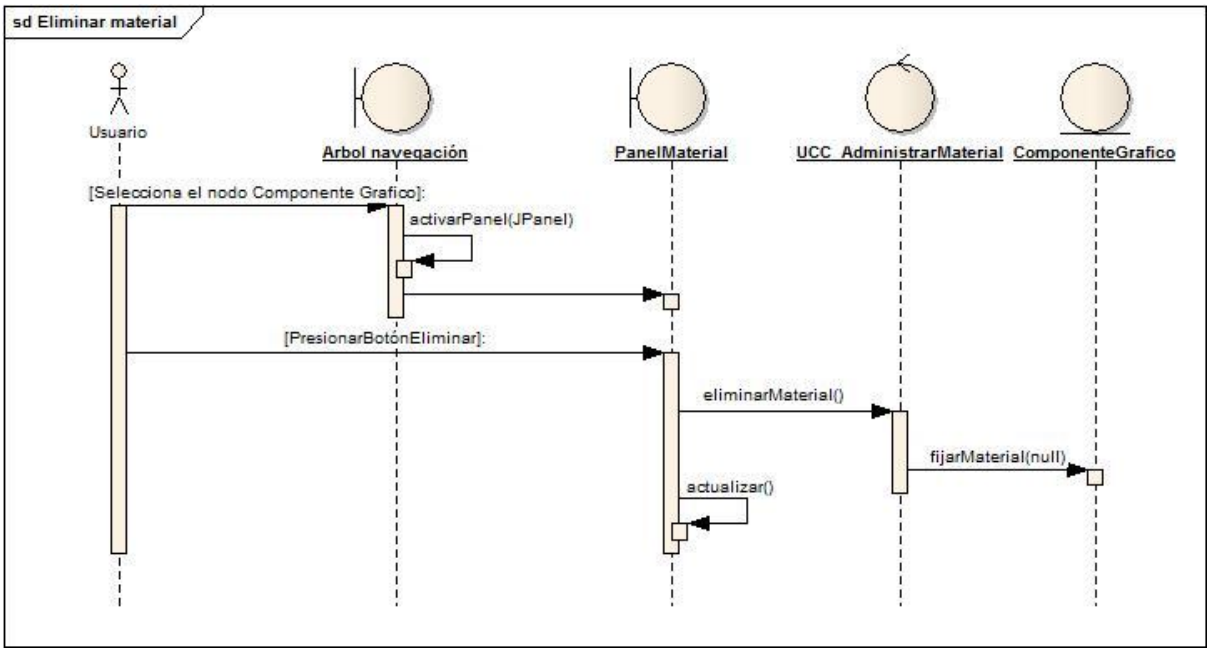


Figura 94 DS Flujo Normal Eliminar Material

Flujo Alterno

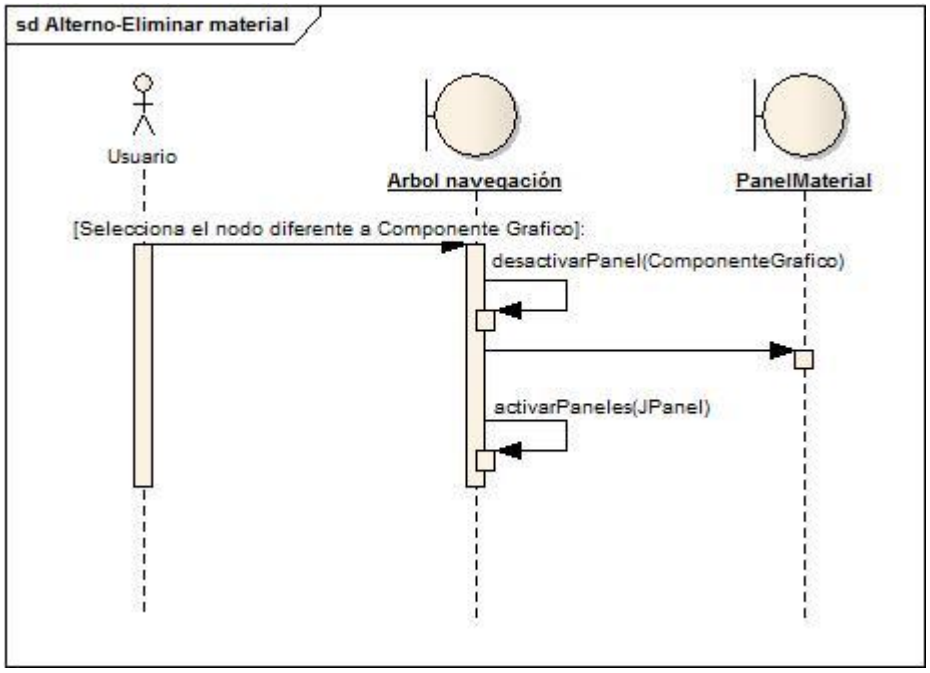


Figura 95 DS Flujo Alterno Eliminar Material

8.5.27. Modelado de secuencia del Caso de uso Agregar Efecto al componente gráfico.

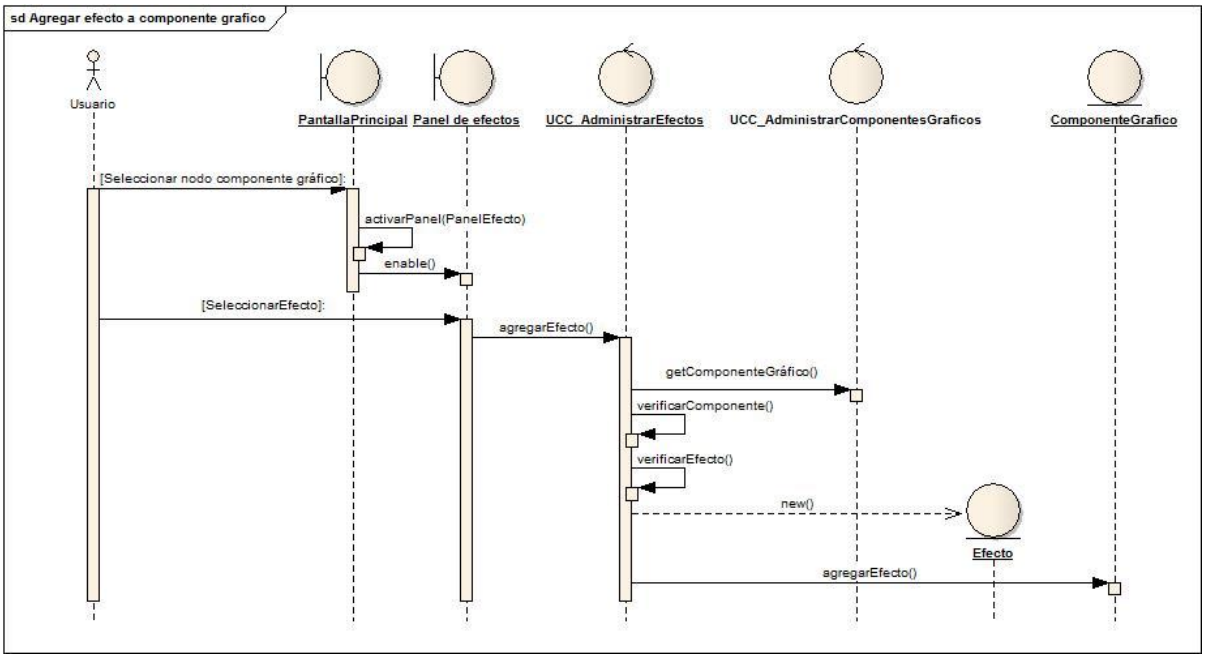


Figura 96 DS Flujo Normal Agregar efecto al componente grafico

Flujo Alterno

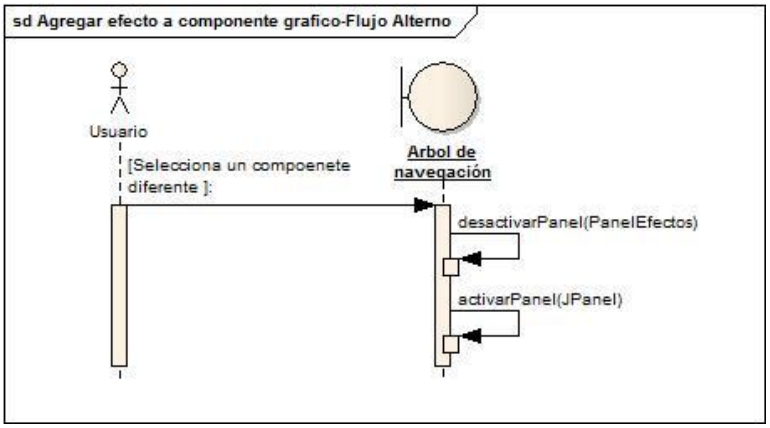


Figura 97 DS Flujo Alterno Agregar efecto al componente grafico

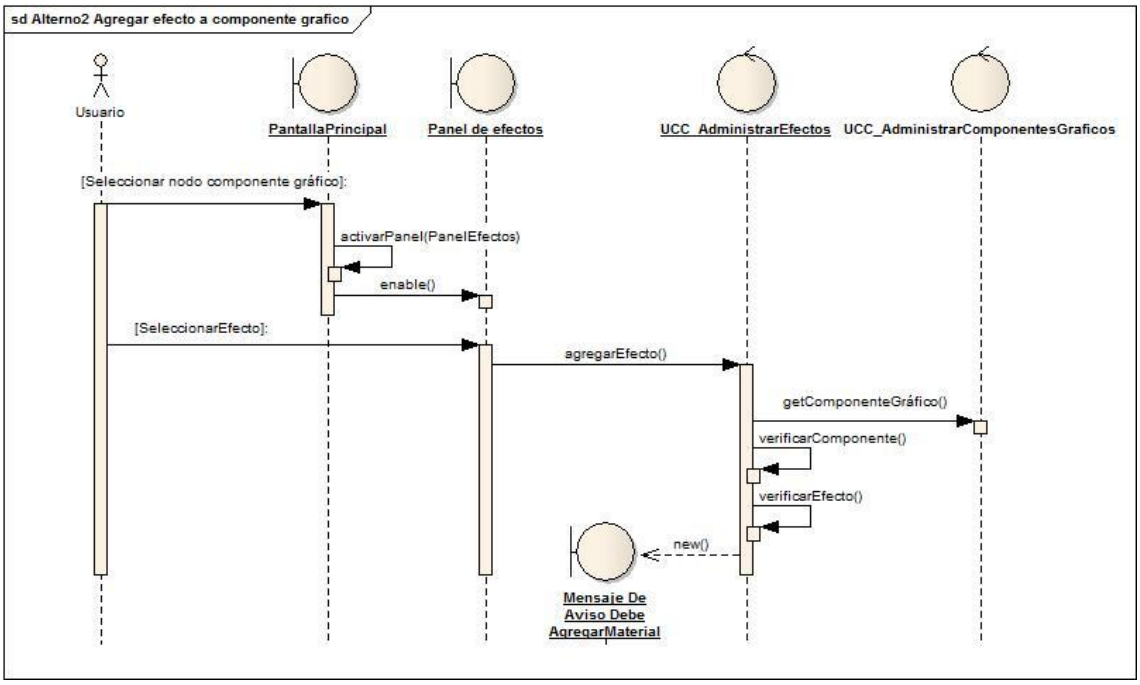


Figura 98 DS Flujo Alterno Agregar Efecto a Componente Grafico

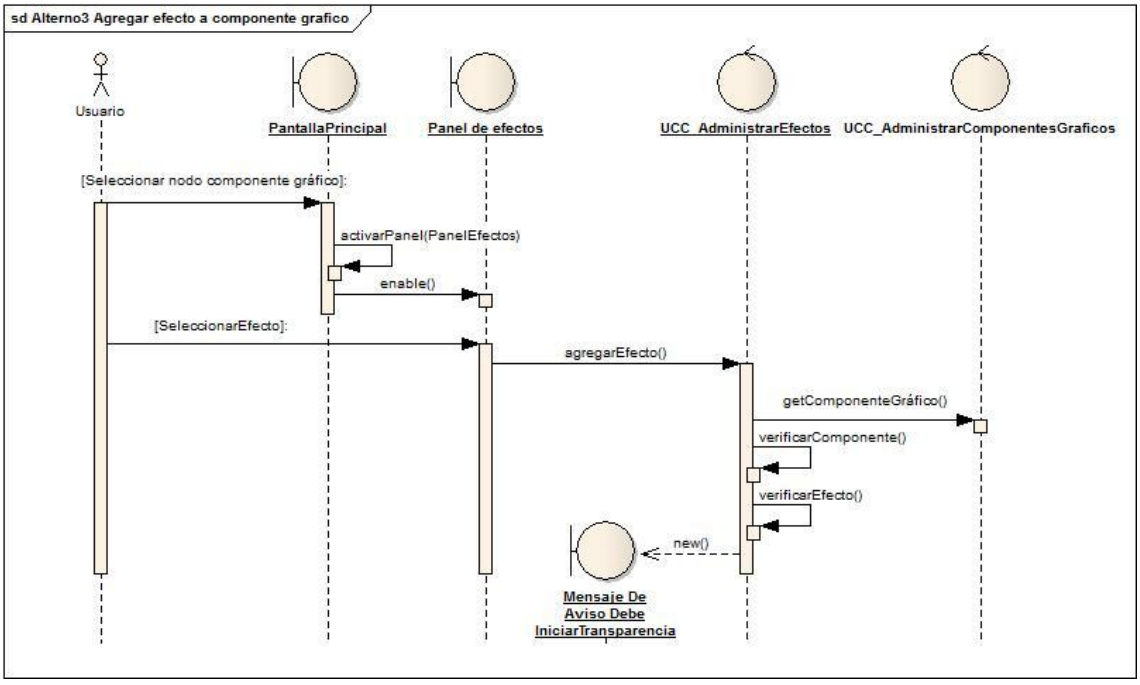


Figura 99 DS Flujo Alterno Agregar Efecto a Componente Grafico

8.5.28. Modelado de secuencia del Caso de uso Modificar configuración del efecto.

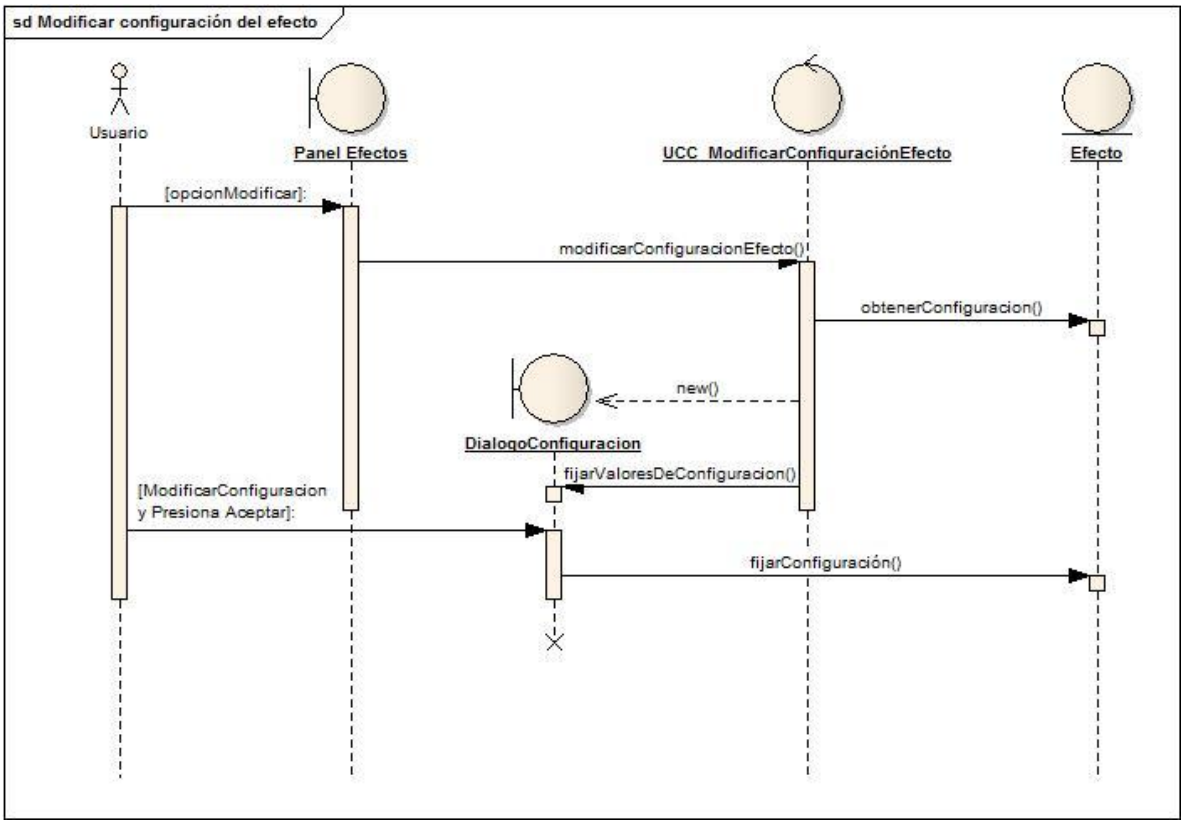


Figura 100 DS Flujo Normal Modificar configuración del efecto

Flujo Alterno

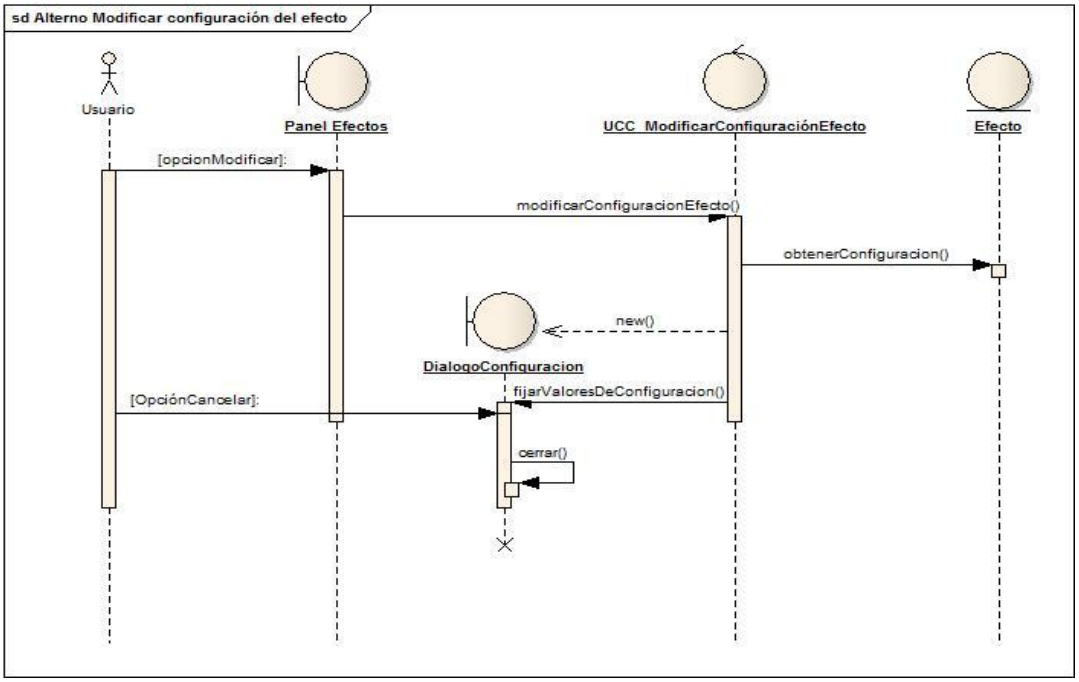


Figura 101 DS Flujo Alterno Modificar configuración del efecto

8.5.29. Modelado de secuencia del Caso de uso Eliminar efecto del componente gráfico.

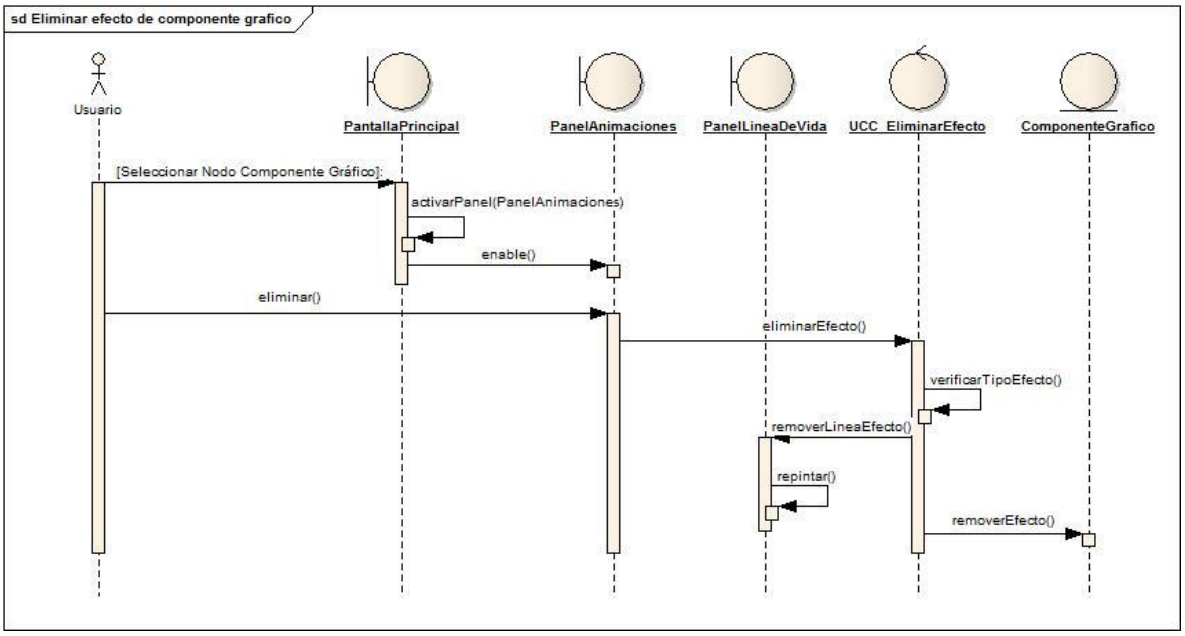


Figura 102 DS Flujo Normal Eliminar efecto del Componente Grafico

Flujo Alterno

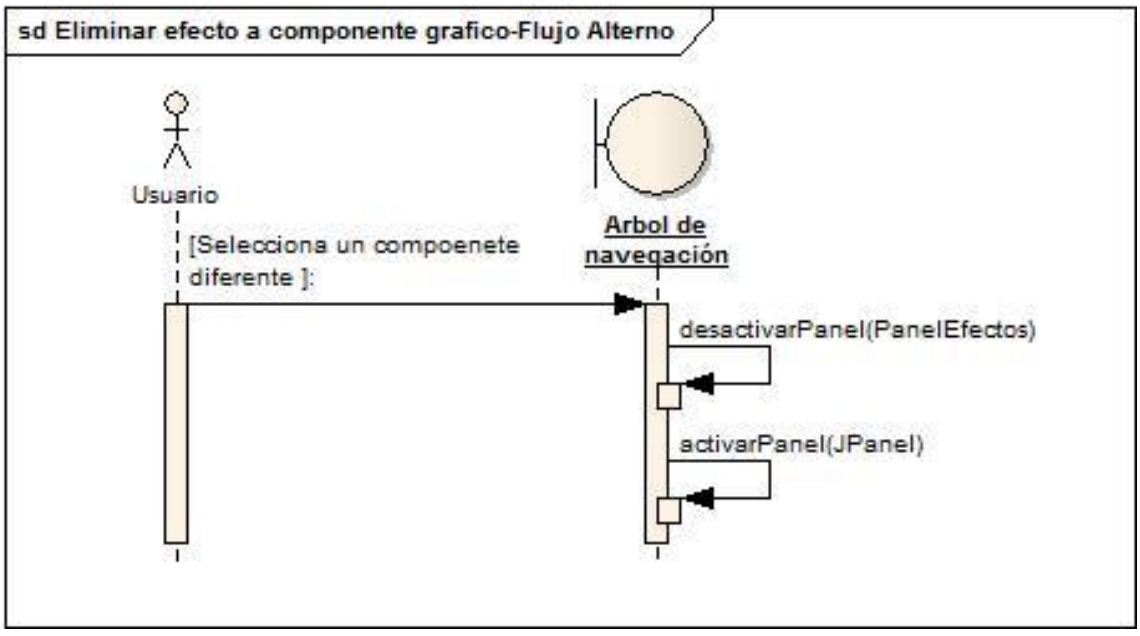


Figura 103 DS Flujo Alterno Eliminar Efecto del Componente Grafico

8.5.30. Modelado de secuencia del Caso de uso Agregar textura desde archivo.

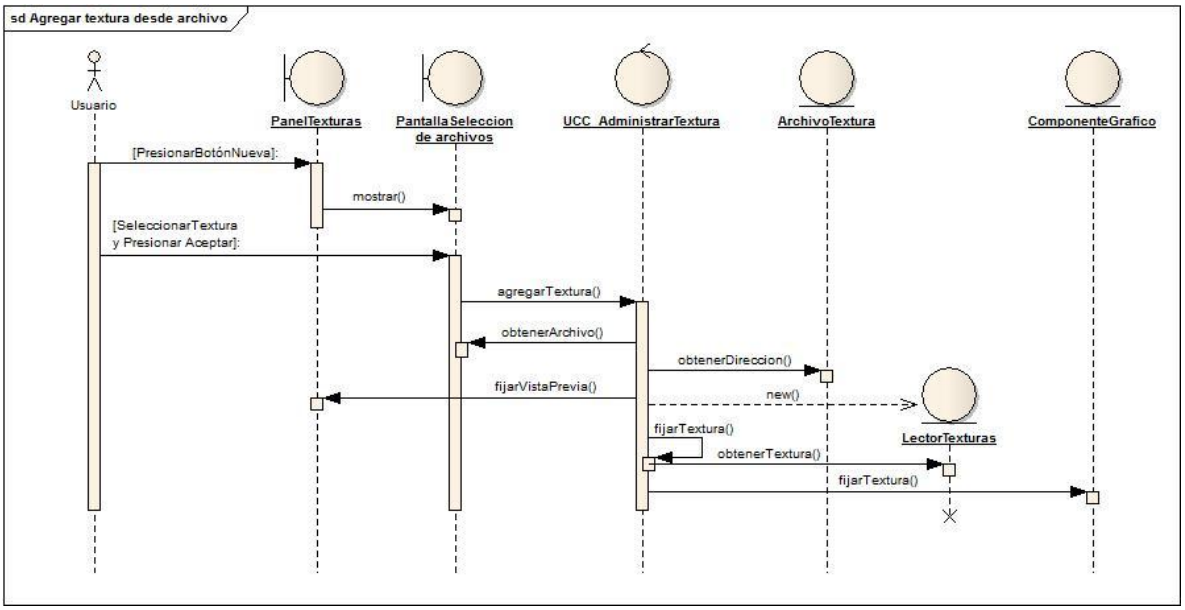


Figura 104 DS Flujo Normal Agregar textura desde archivo

Flujo Alterno

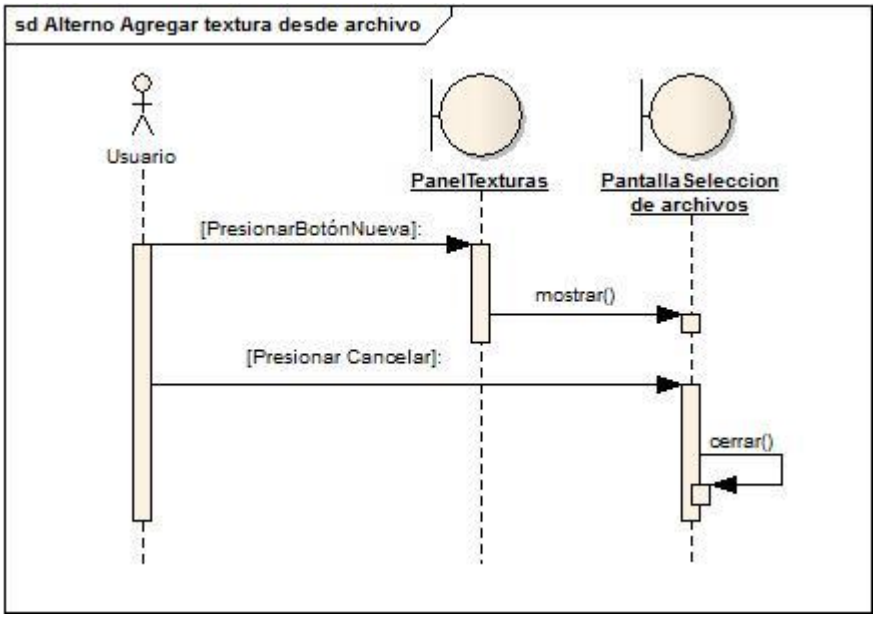


Figura 105 DS Flujo Alterno Agregar Textura desde Archivo

8.5.31. Modelado de secuencia del Caso de uso Agregar textura desde galería.

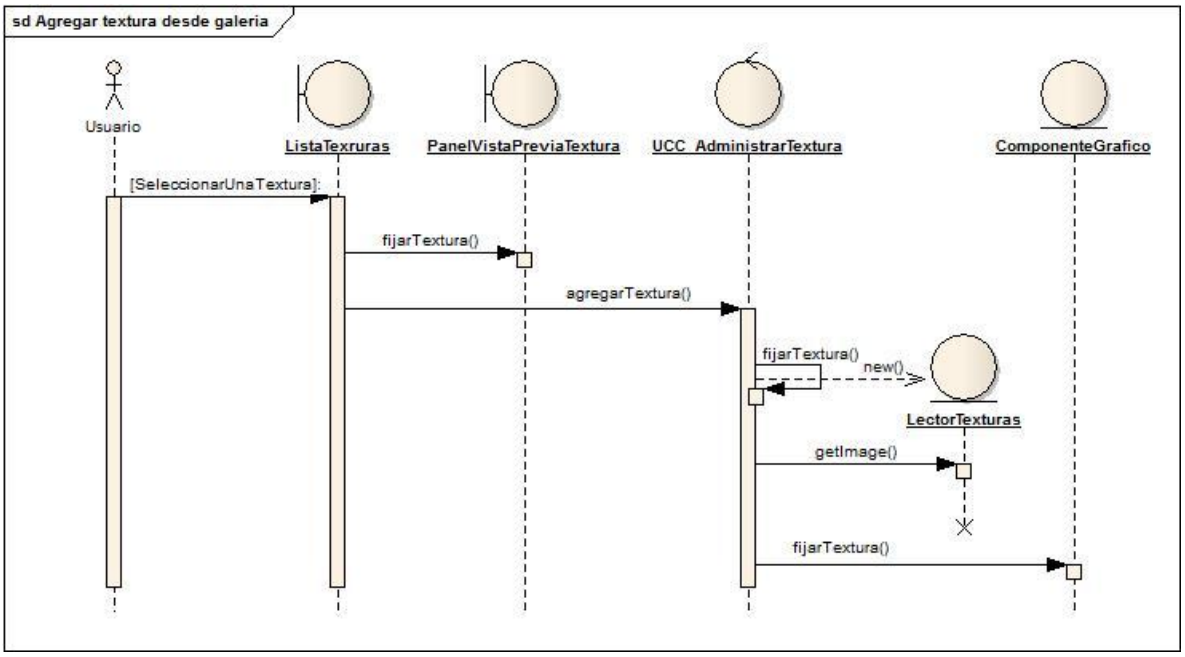


Figura 106 DS Flujo Normal Agregar Textura desde Galería

Flujo Alterno

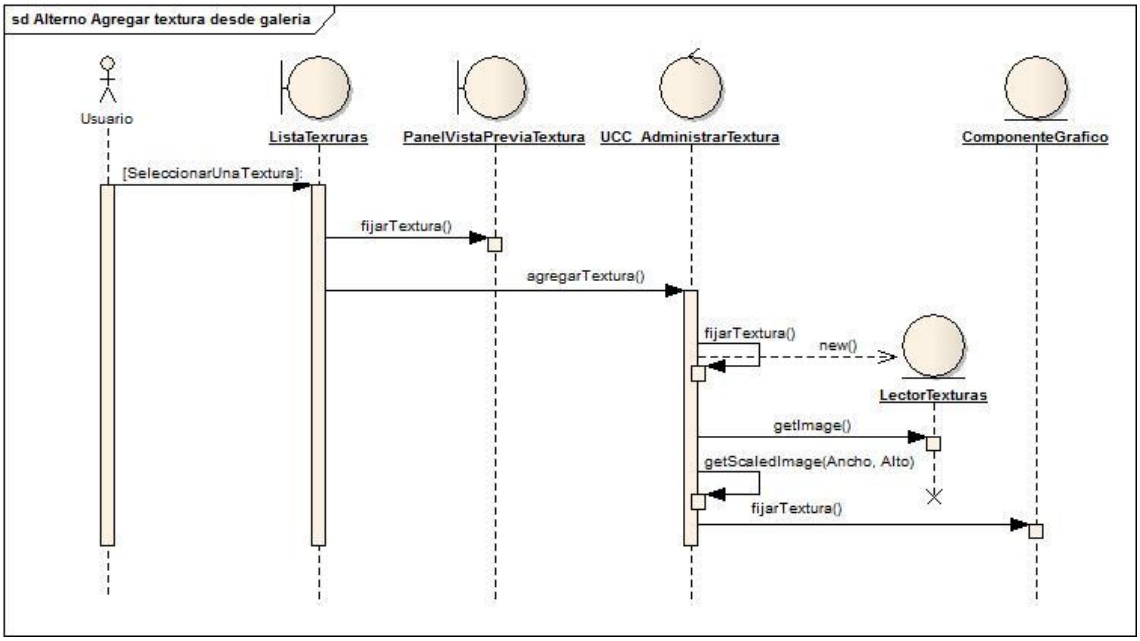


Figura 107 DS Flujo Alterno Agregar Textura desde Galería

8.5.32. Modelado de secuencia del Caso de uso Configurar textura.

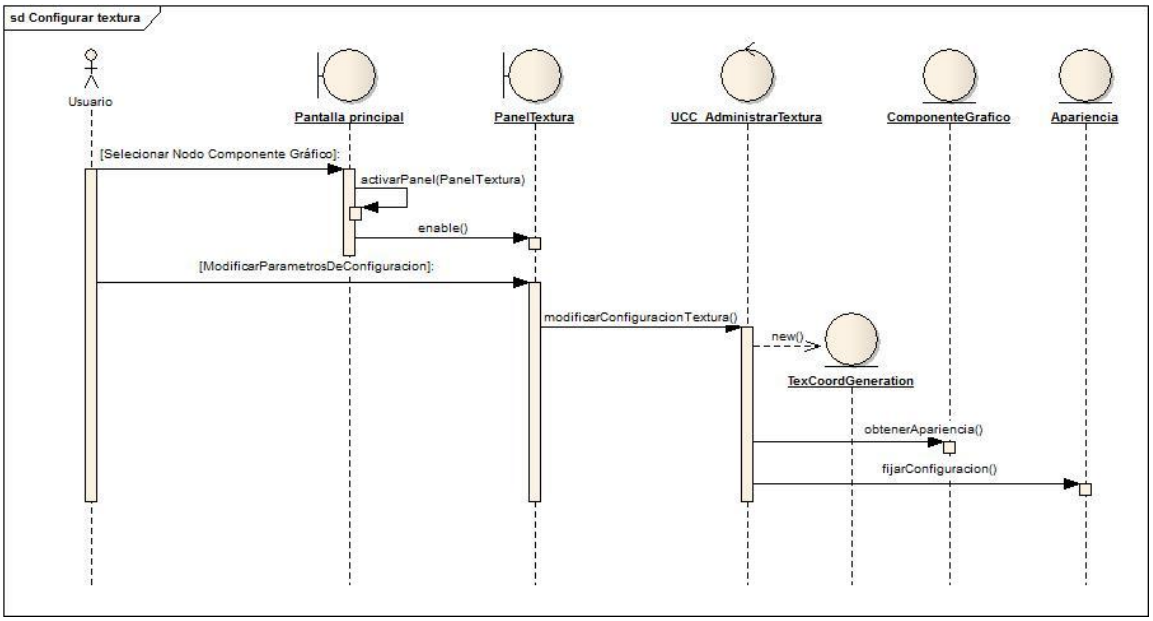


Figura 108 DS Flujo Normal Configurar Textura

Flujo Alterno

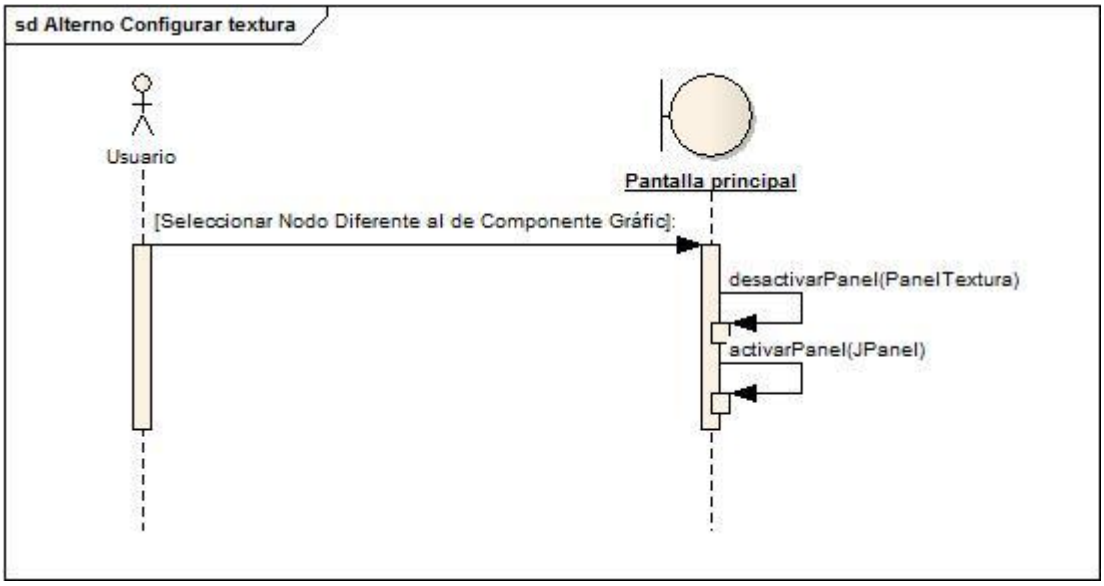


Figura 109 DS Flujo Alterno Configurar Textura

8.5.33. Modelado de secuencia del Caso de uso Eliminar textura.

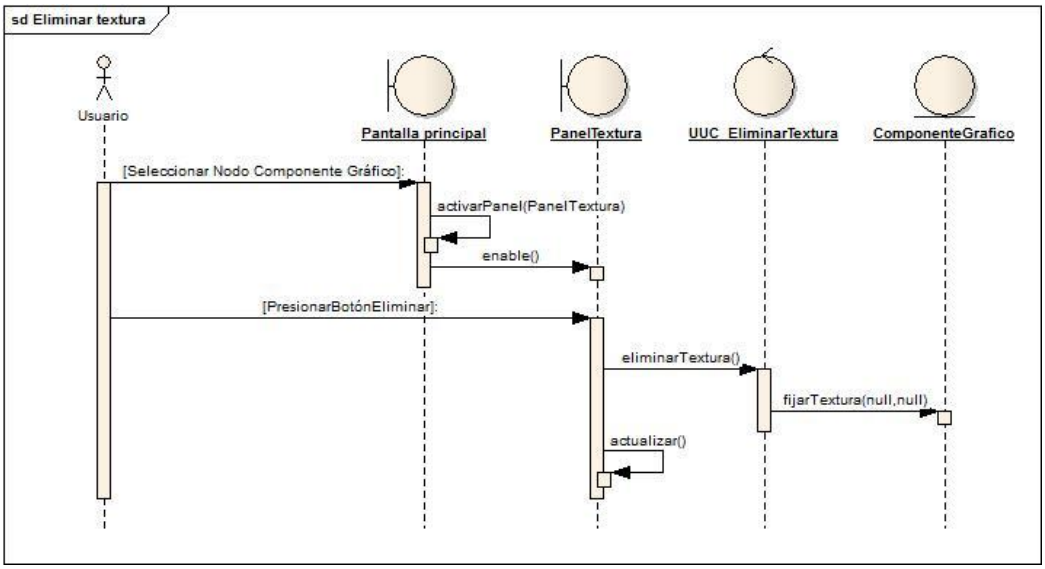


Figura 110 DS Flujo Normal Eliminar textura

Flujo Alterno

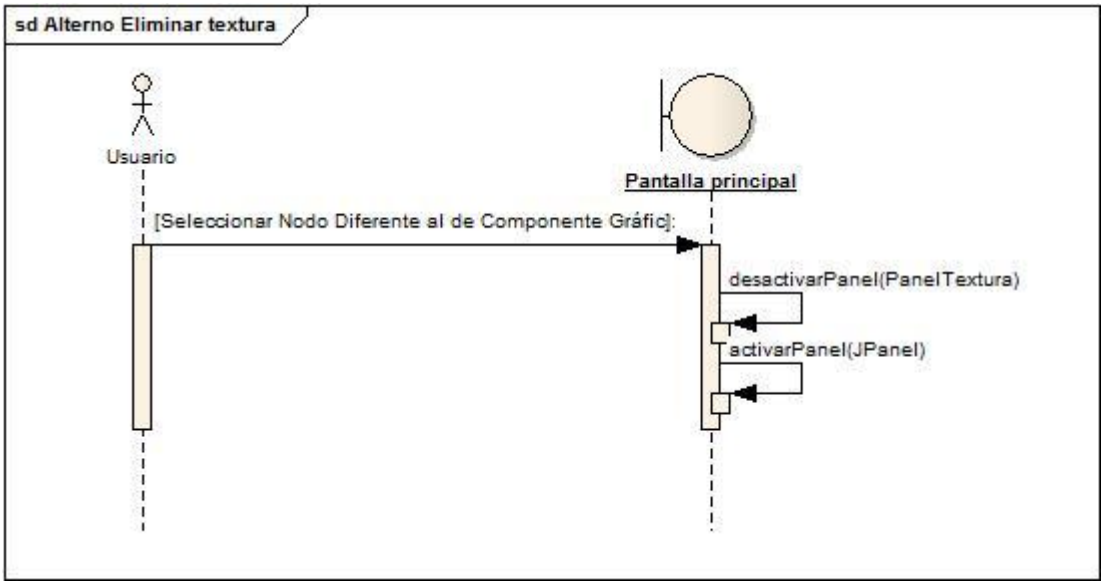


Figura 111 DS Flujo Alterno Eliminar Textura

8.6. Diagrama de Clases Final o Modelo del Dominio

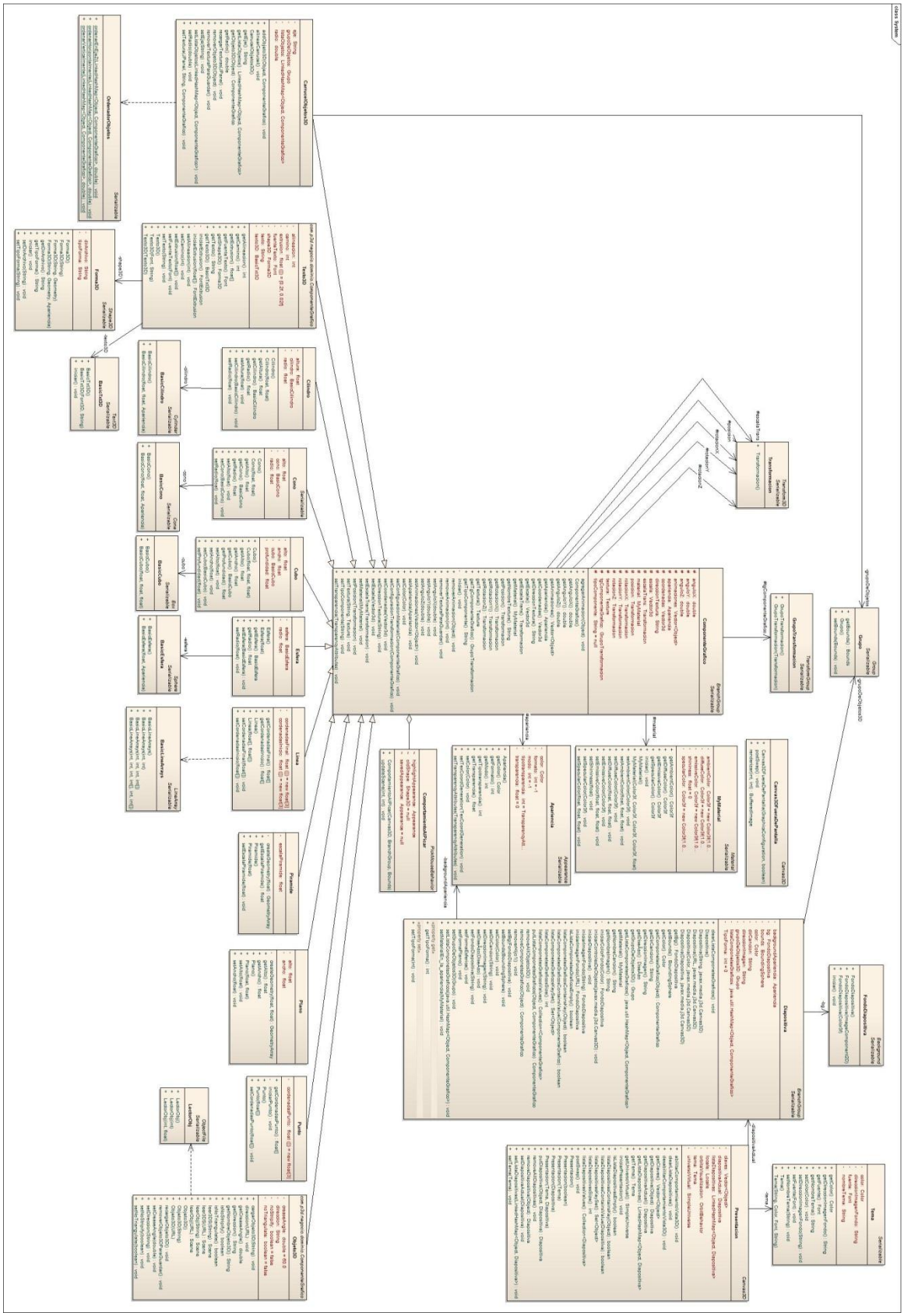


Figura 112 Modelo del Dominio – System

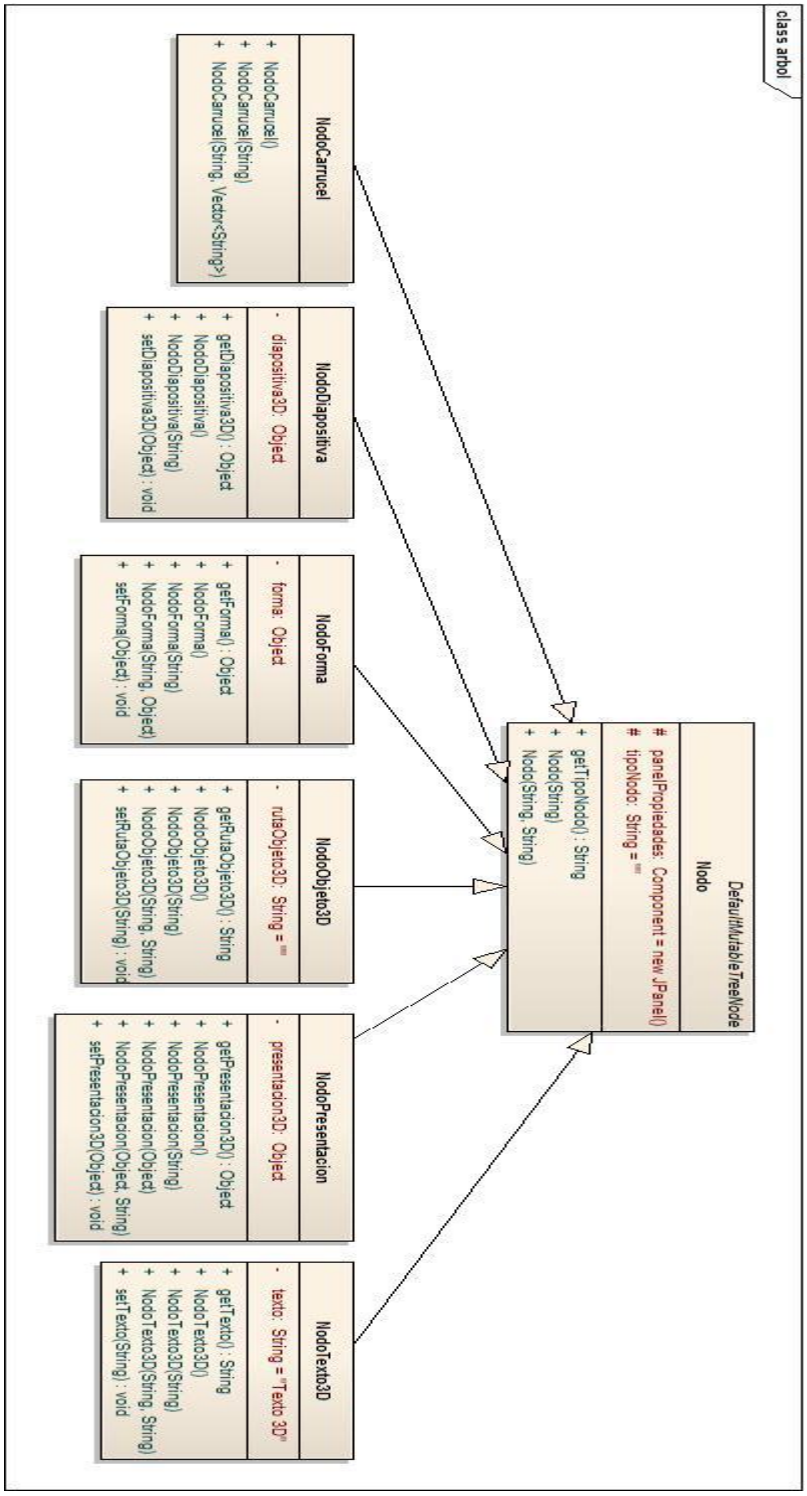


Figura 113 Modelo del Dominio- Árbol de Navegación

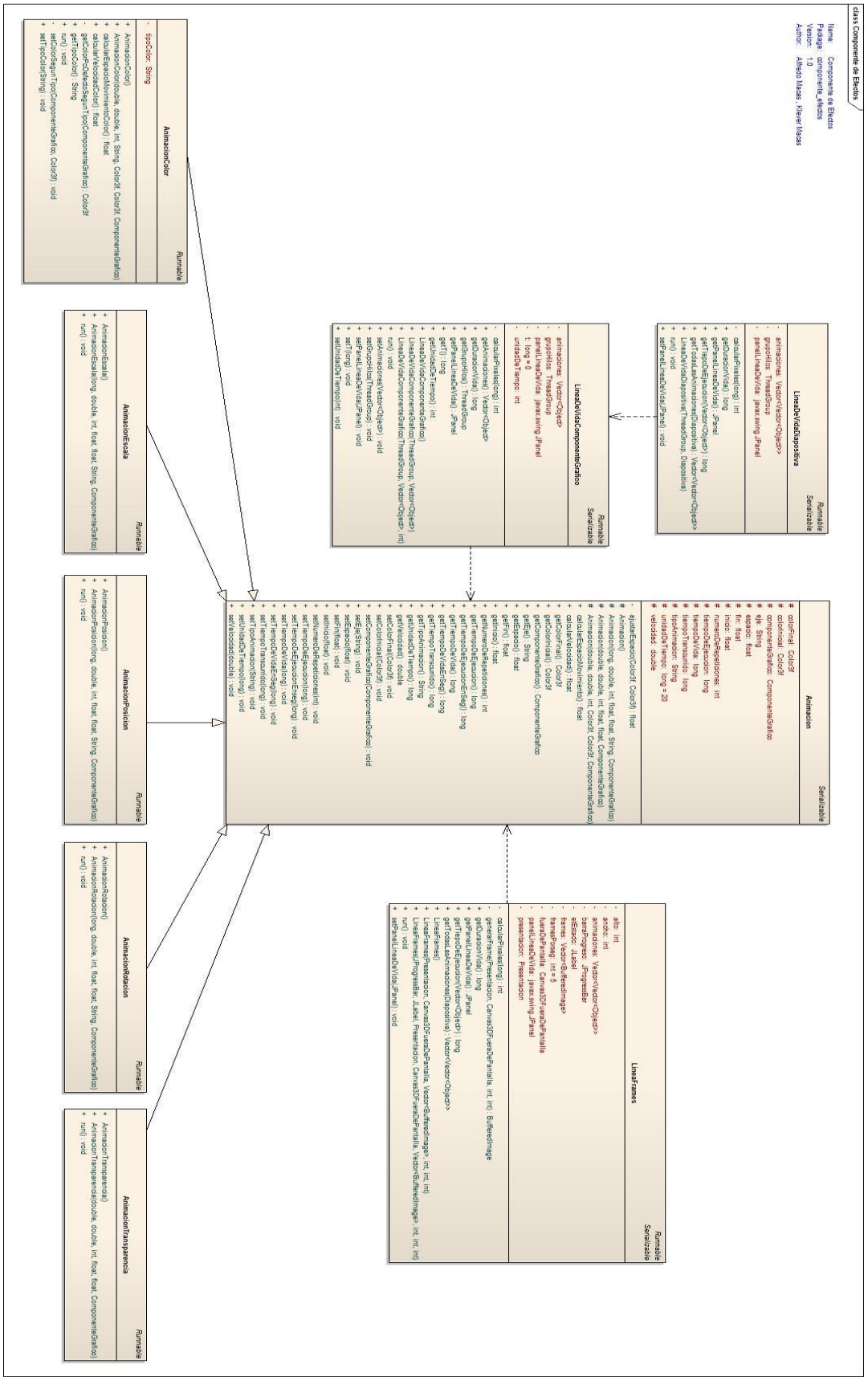


Figura 114 Modelo del Dominio – Componente de Efectos

8.7. MODELADO DE ARQUITECTURA

8.7.1. Modelo de Arquitectura

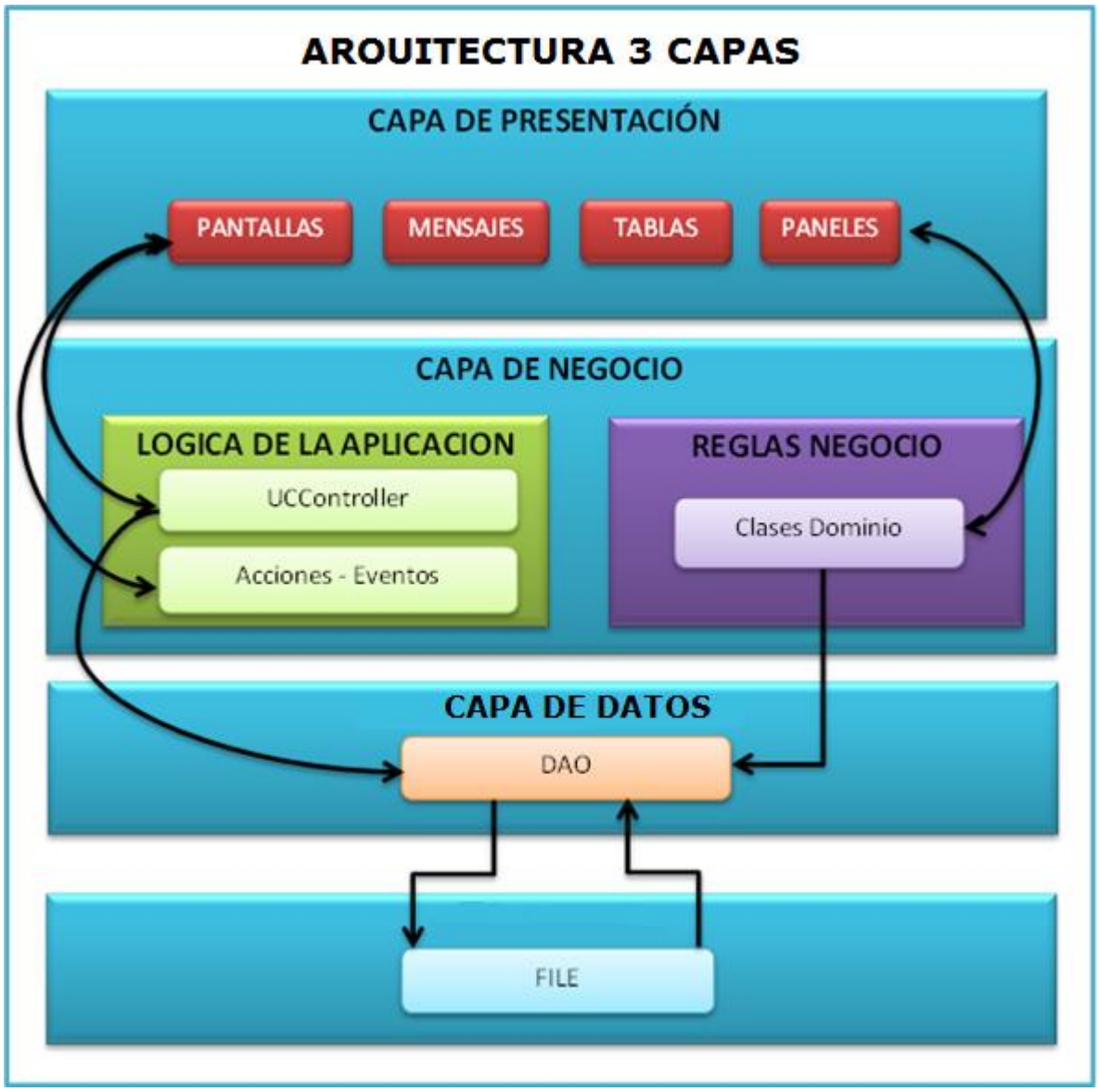


Figura 115 Arquitectura del Sistema

8.7.2. Diagrama de paquetes

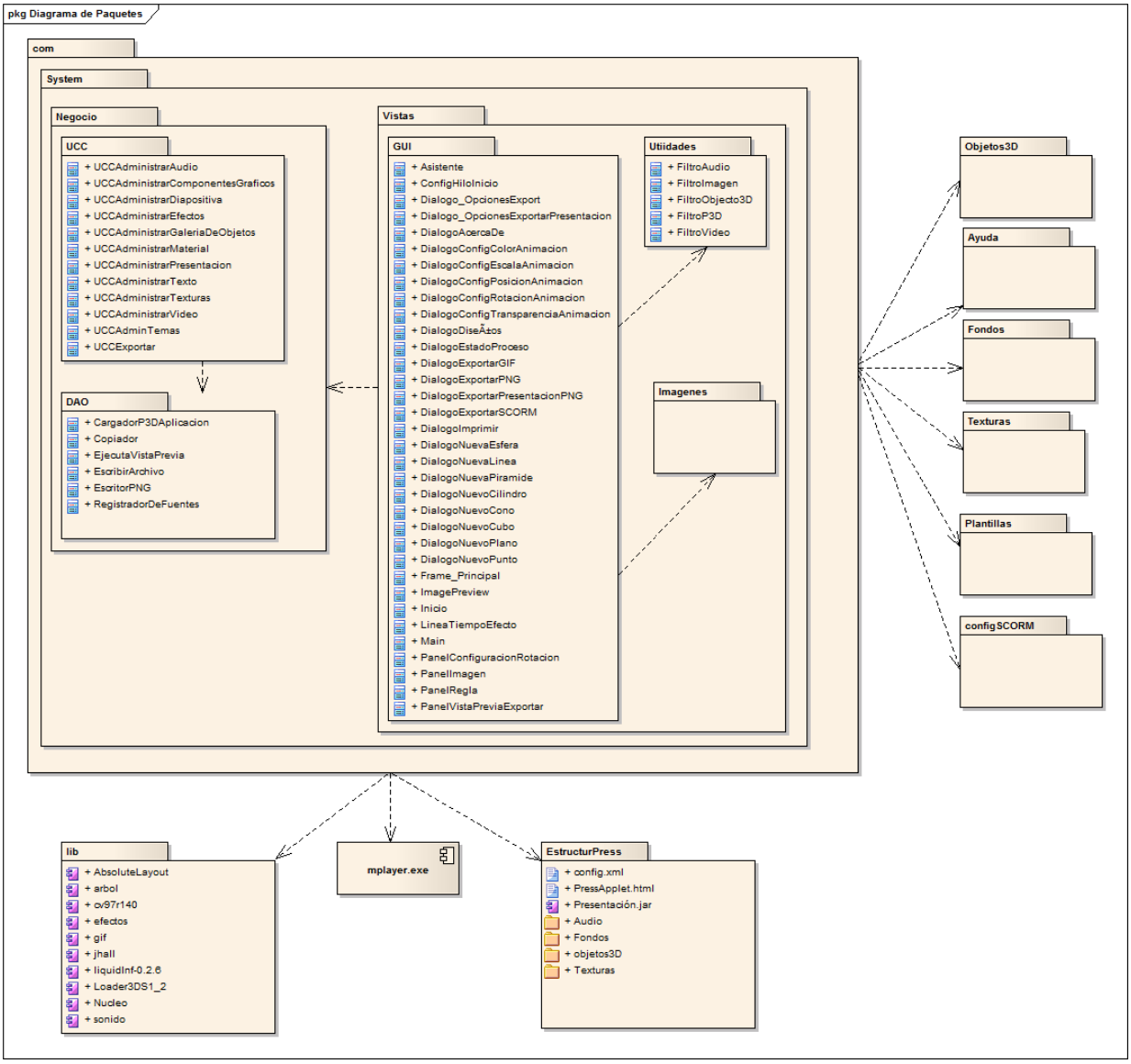


Figura 116 Diagrama de Paquetes

8.7.3. Diagrama de clases por paquete

Paquete com.System.negocio.UCC

Figura 117 Diagrama de Clases del Paquete UCC

Paquete com.System.negocio.DAO

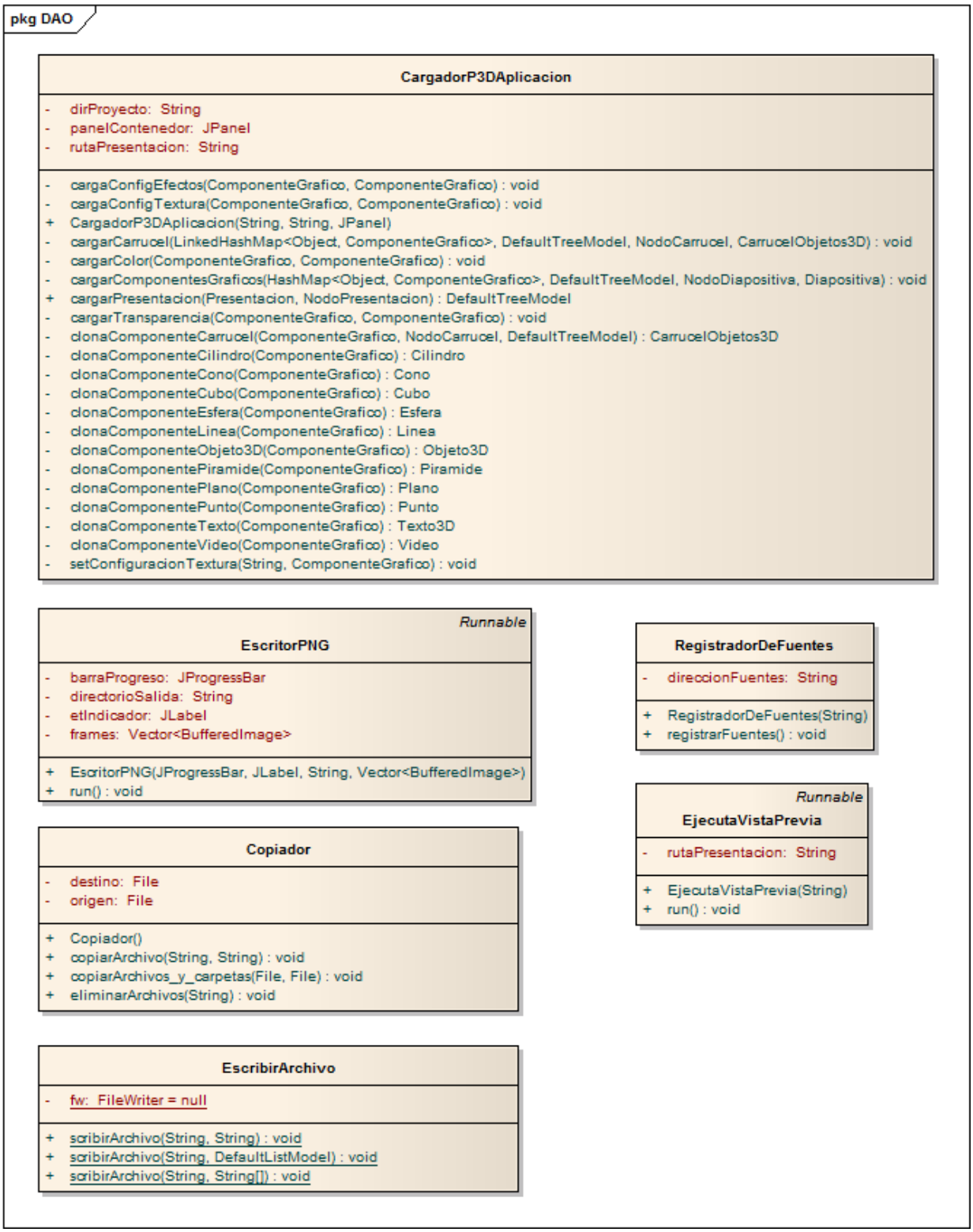


Figura 118 Diagrama de Clases del Paquete DAO

Paquete com.System.Vistas.GUI

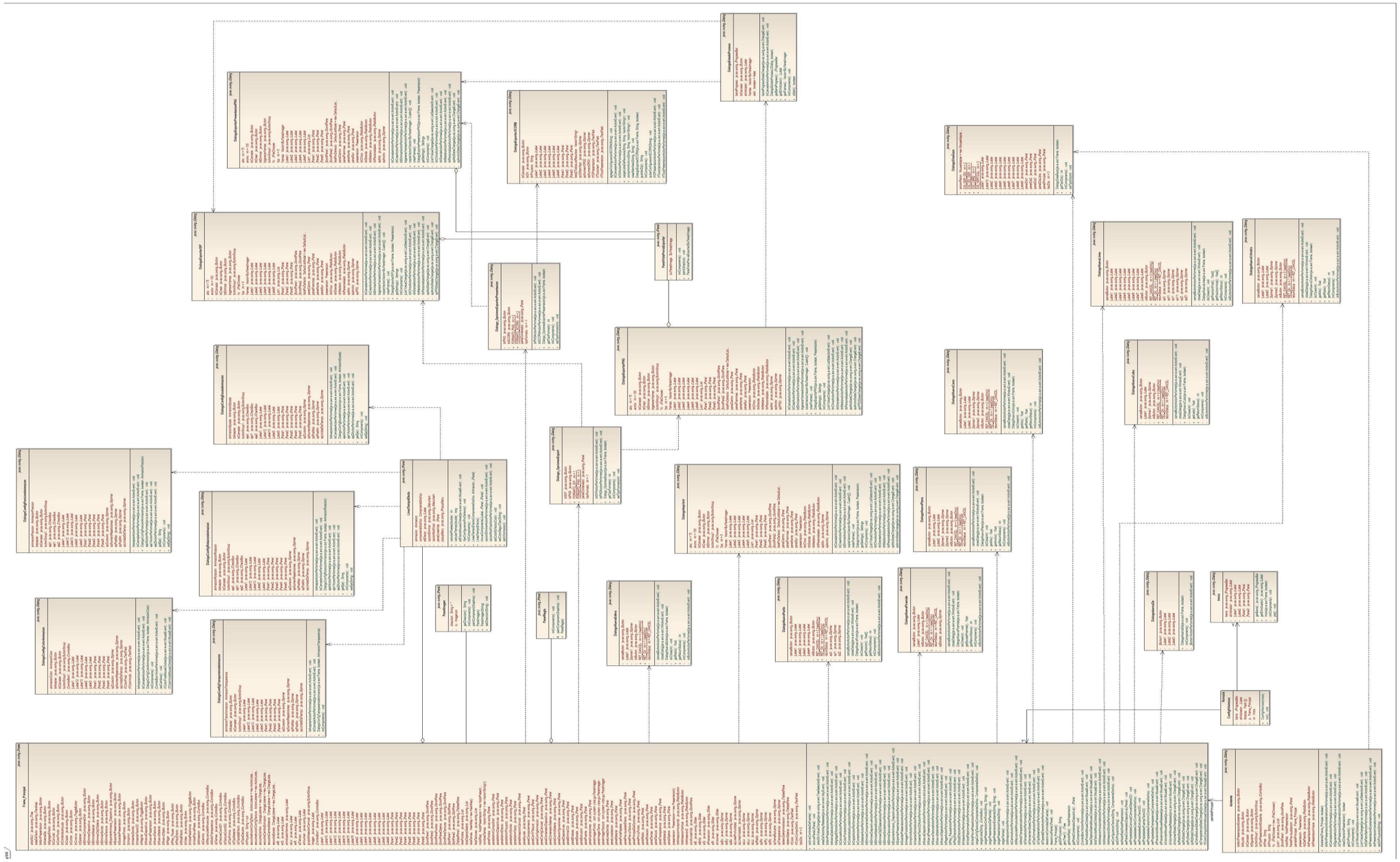


Figura 119 Diagrama de Clases del Paquete GUI

Paquete com.System.Vistas.Filtros

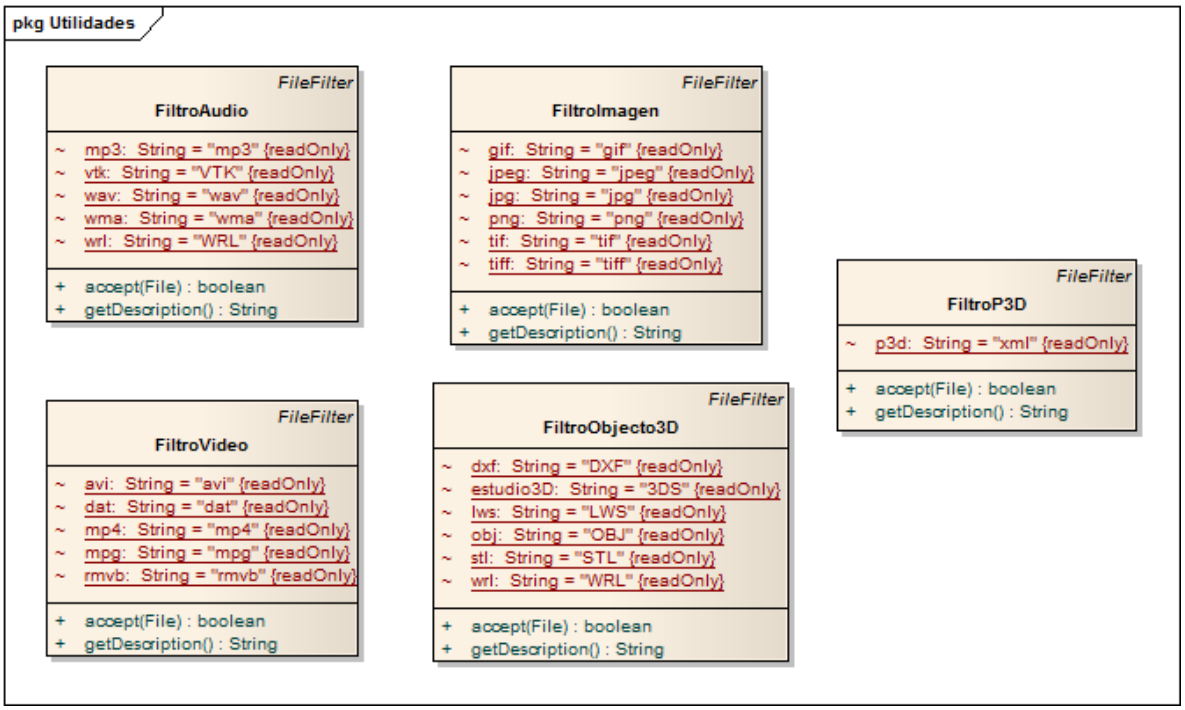


Figura 120 Diagrama de Clases del Paquete Vista - Utilidades

8.7.4. Diagrama de clases por cada caso de uso

8.7.4.1. Diagrama de clases para el caso de uso Crear presentación vacía.

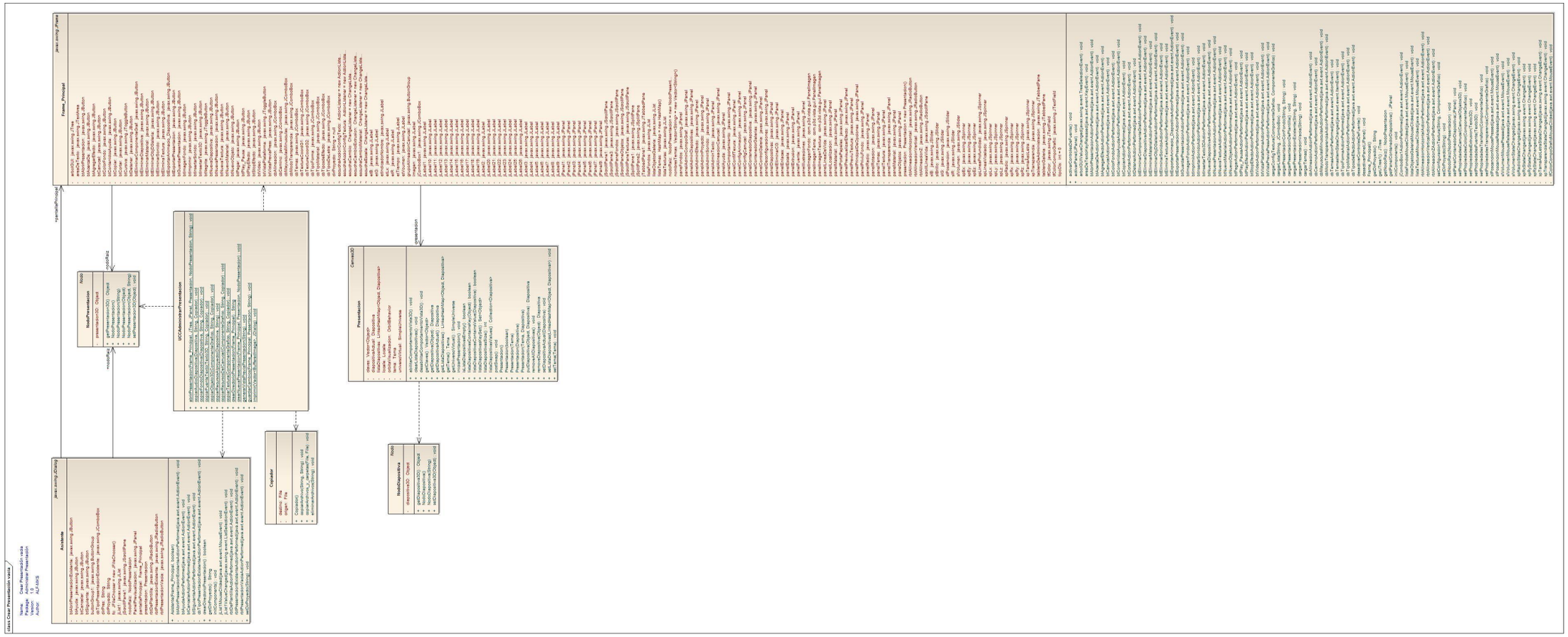


Figura 121 DC Crear Presentación Vacía

8.7.4.2. Diagrama de clases para el caso de uso Crear presentación con plantilla.

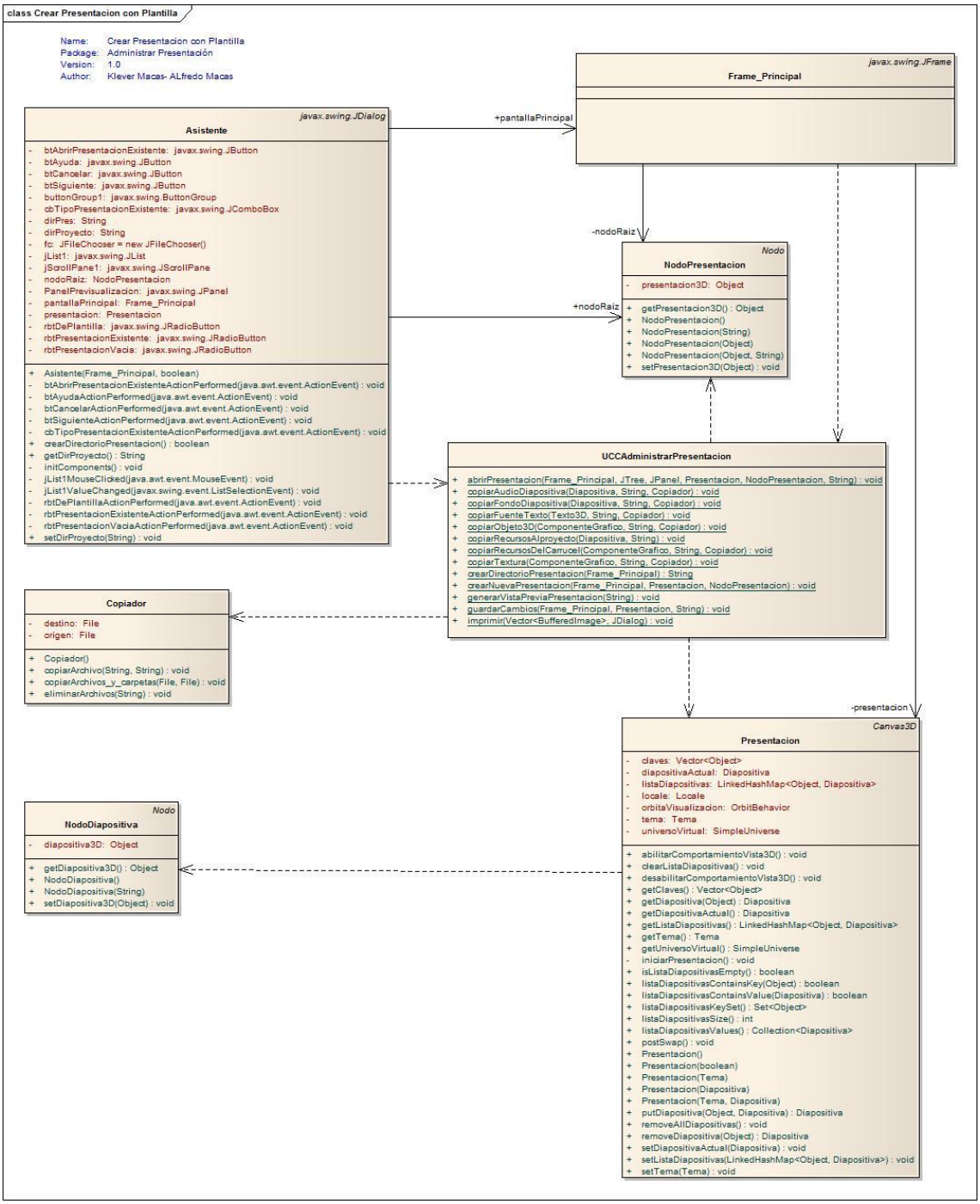


Figura 122 DC Crear Presentación con Plantilla

8.7.4.3. Diagrama de clases para el caso de uso Modificar Presentación.

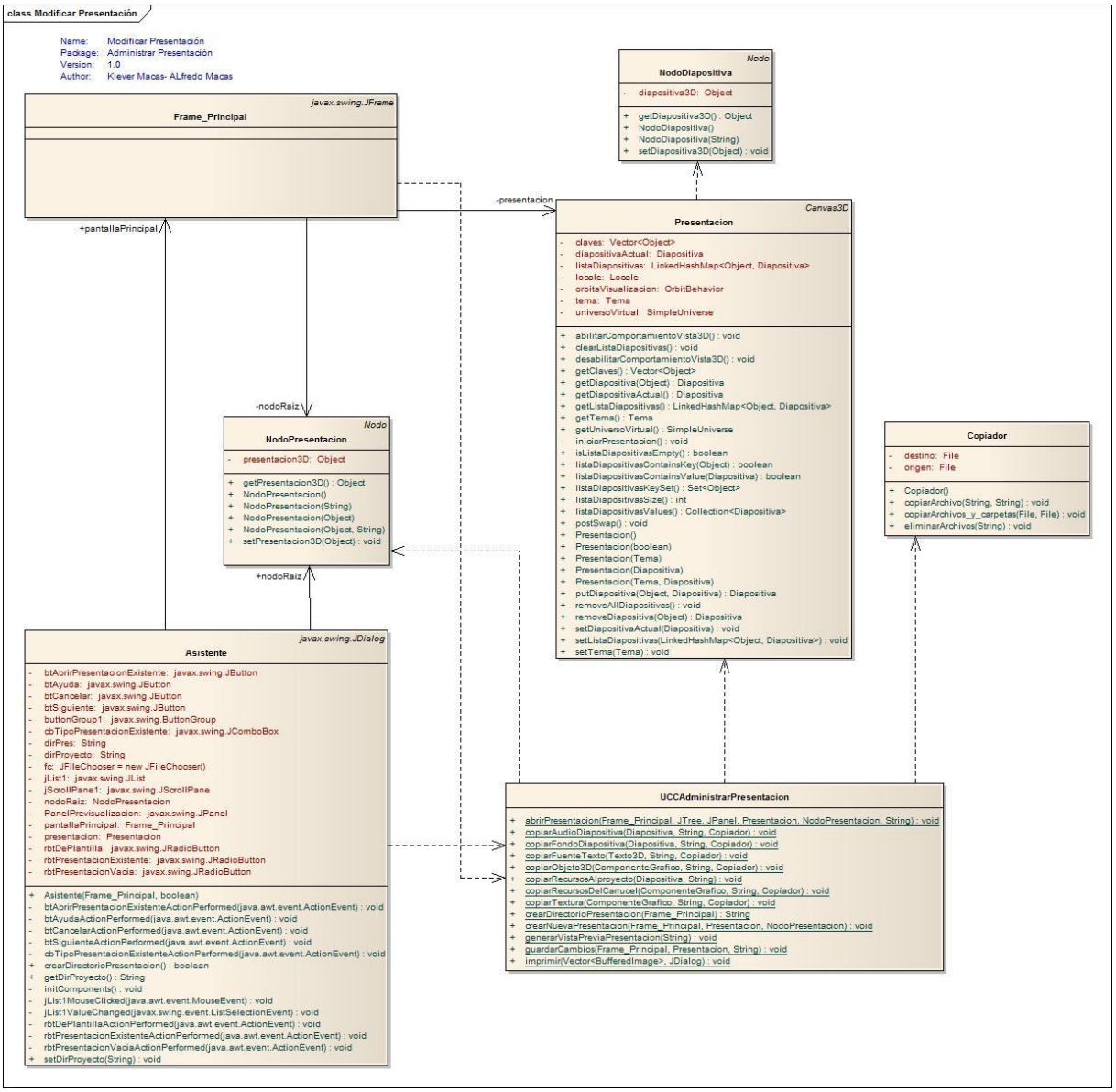


Figura 123 DC Modificar Presentación

8.7.4.4. Diagrama de clases para el caso de uso Guardar Presentación.

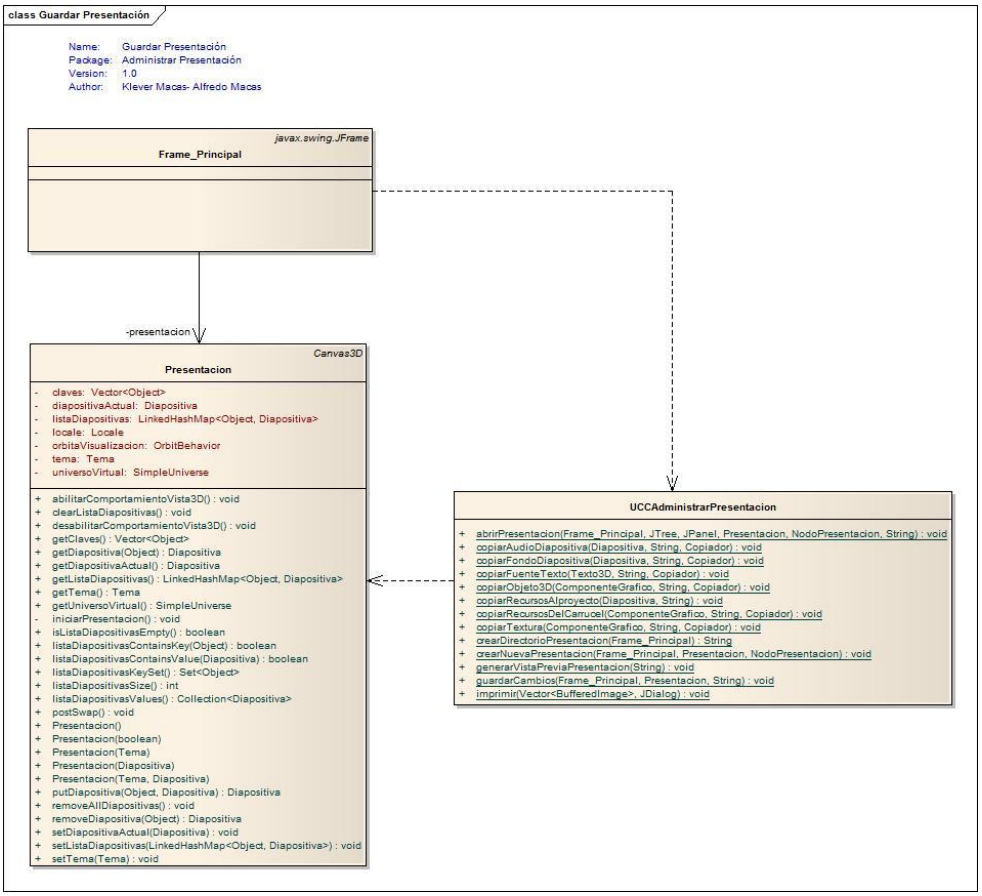


Figura 124 DC Guardar Presentación

8.7.4.5. Diagrama de clases para el caso de uso Exportar Presentación en formato png.

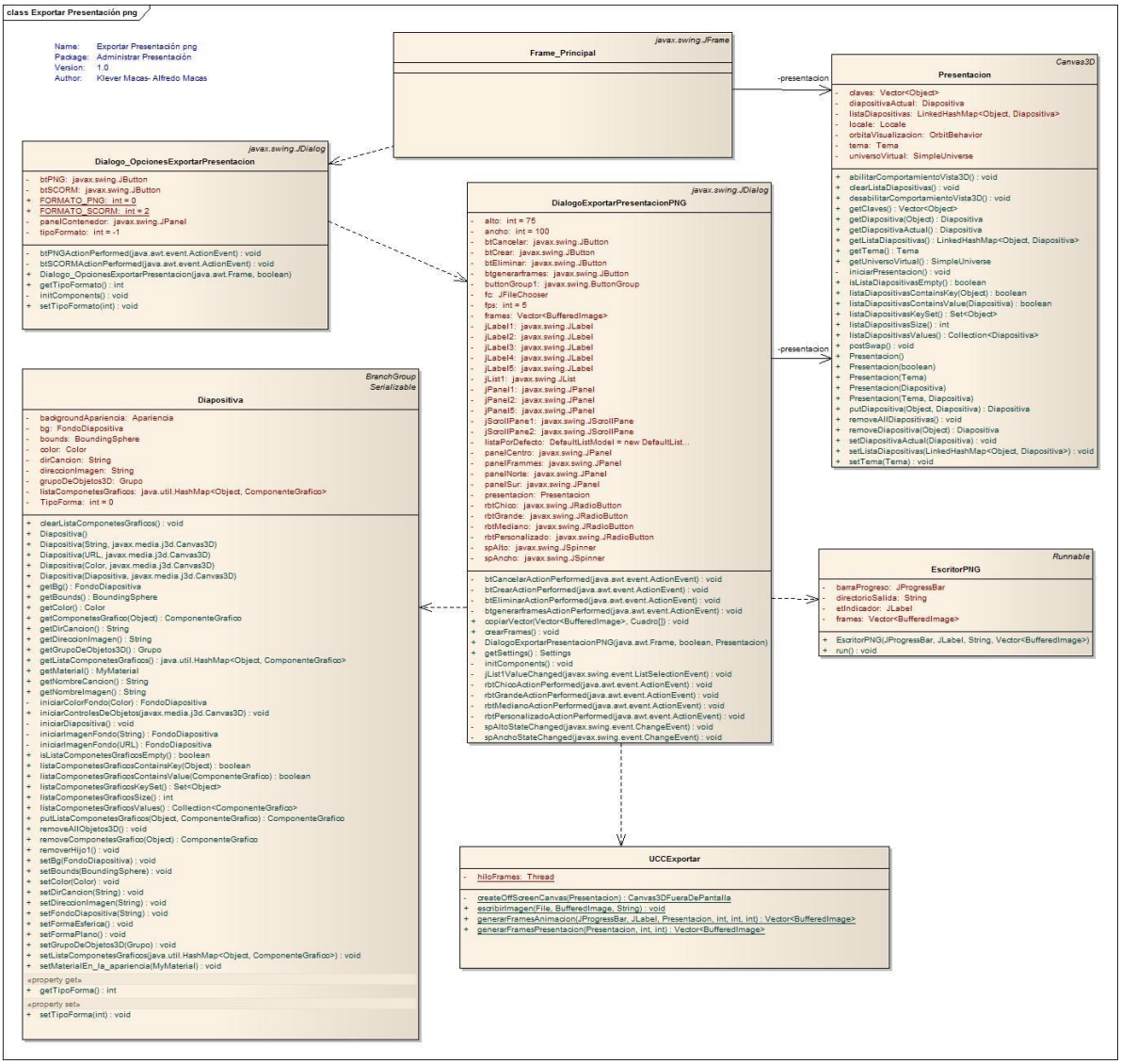


Figura 125 DC Exportar Presentación en formato png

8.7.4.6. Diagrama de clases para el caso de uso Exportar Presentación en formato scorm.

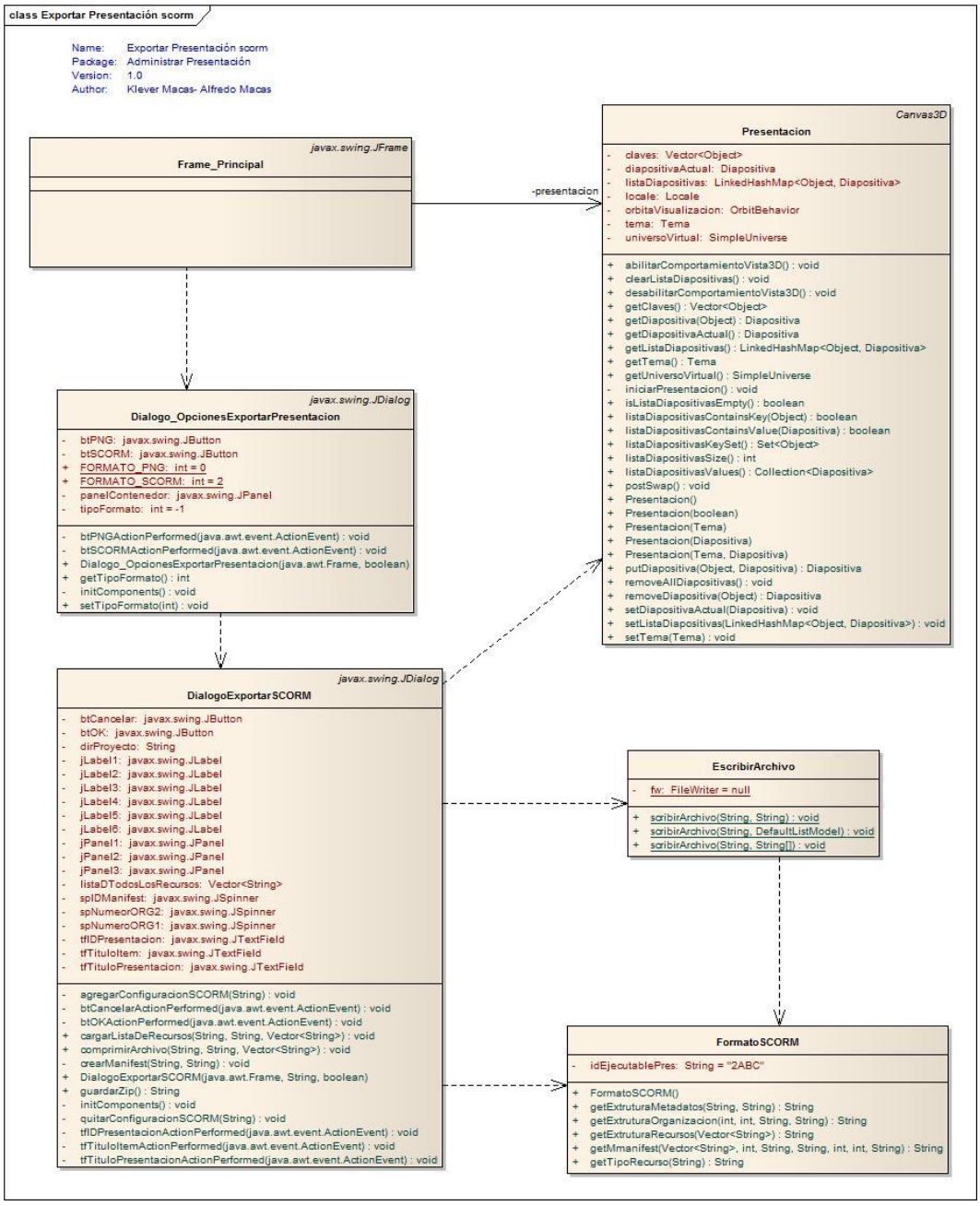


Figura 126 DC Exportar Presentación en formato scorm

8.7.4.7. Diagrama de clases para el caso de uso Seleccionar Fondo para la Presentación.

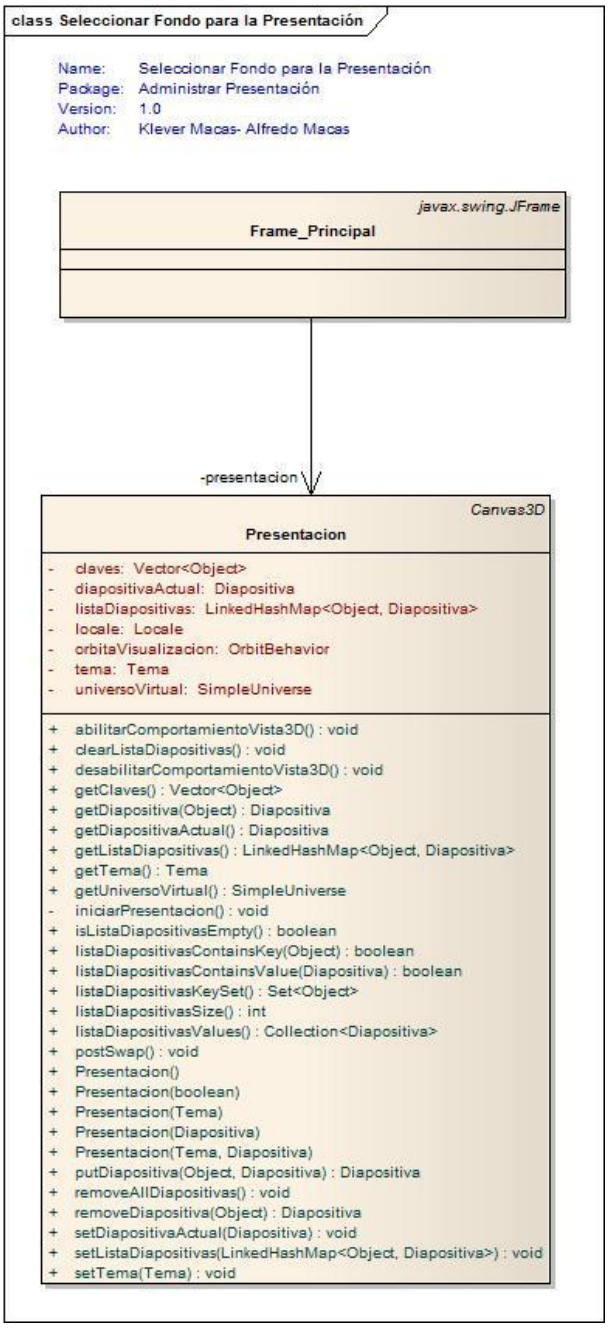


Figura 127 DC Seleccionar Fondo para Presentación

8.7.4.8. Diagrama de clases para el caso de uso Generar vista Previa.

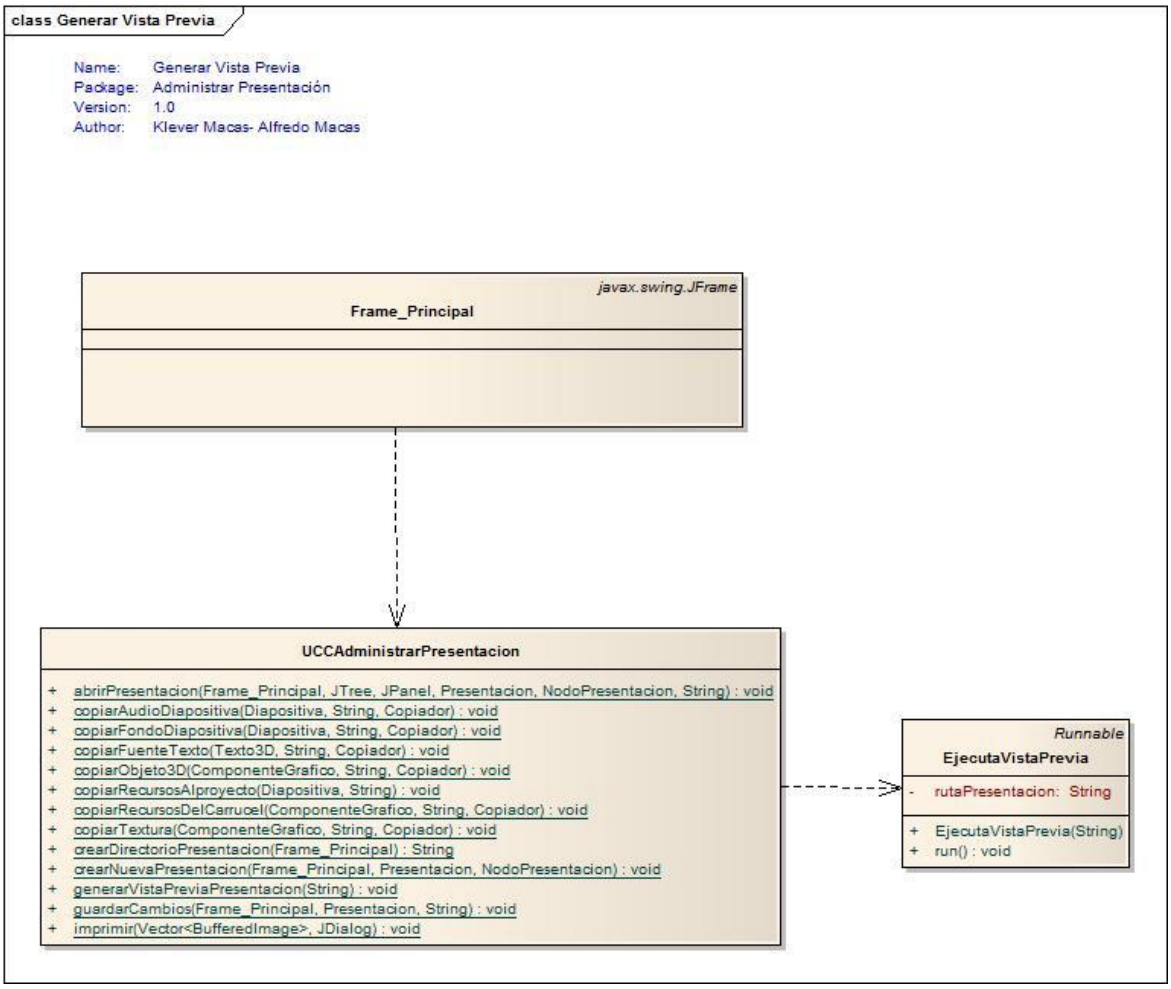


Figura 128 DC Generar Vista Previa

8.7.4.9. Diagrama de clases para el caso de uso Imprimir presentación.

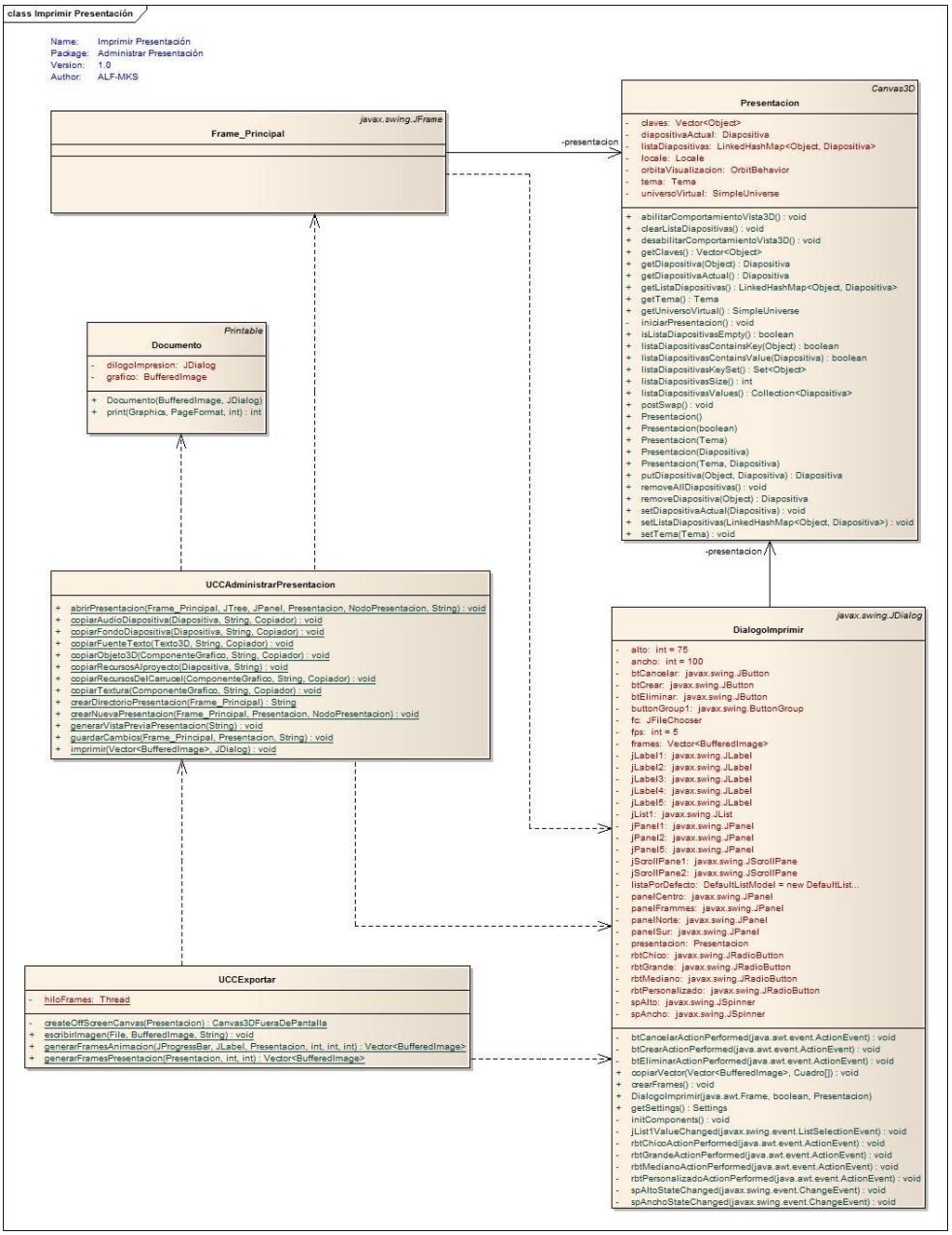


Figura 129 DC Imprimir Presentación

8.7.4.10. Diagrama de clases para el caso de uso Agregar diapositiva en blanco.

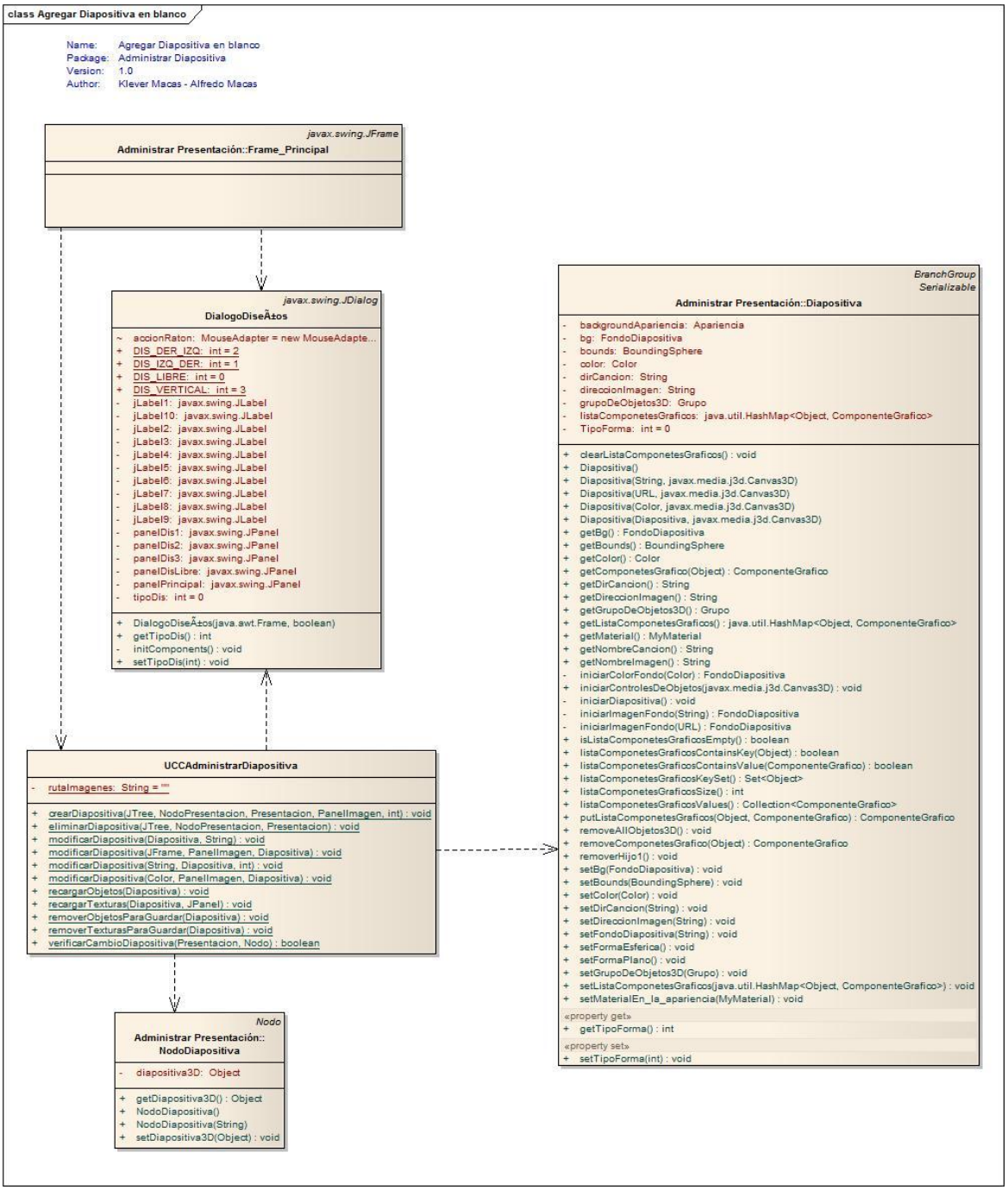


Figura 130 DC Agregar Diapositiva en Blanco

8.7.4.11. Diagrama de clases para el caso de uso Agregar diapositiva con diseño.

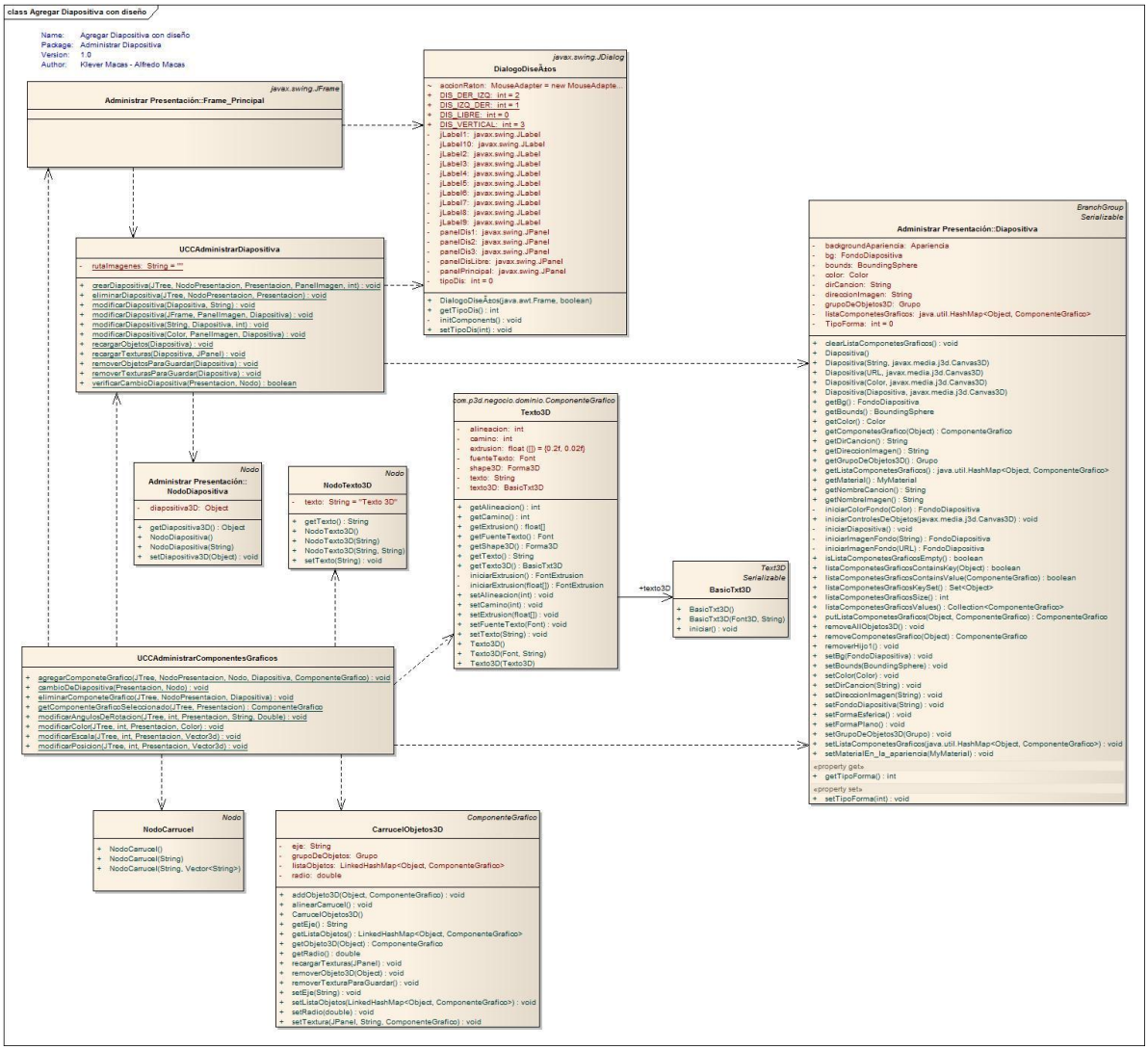


Figura 131 DC Agregar Diapositiva con Diseño

8.7.4.12. Diagrama de clases para el caso de uso Agregar color de fondo.

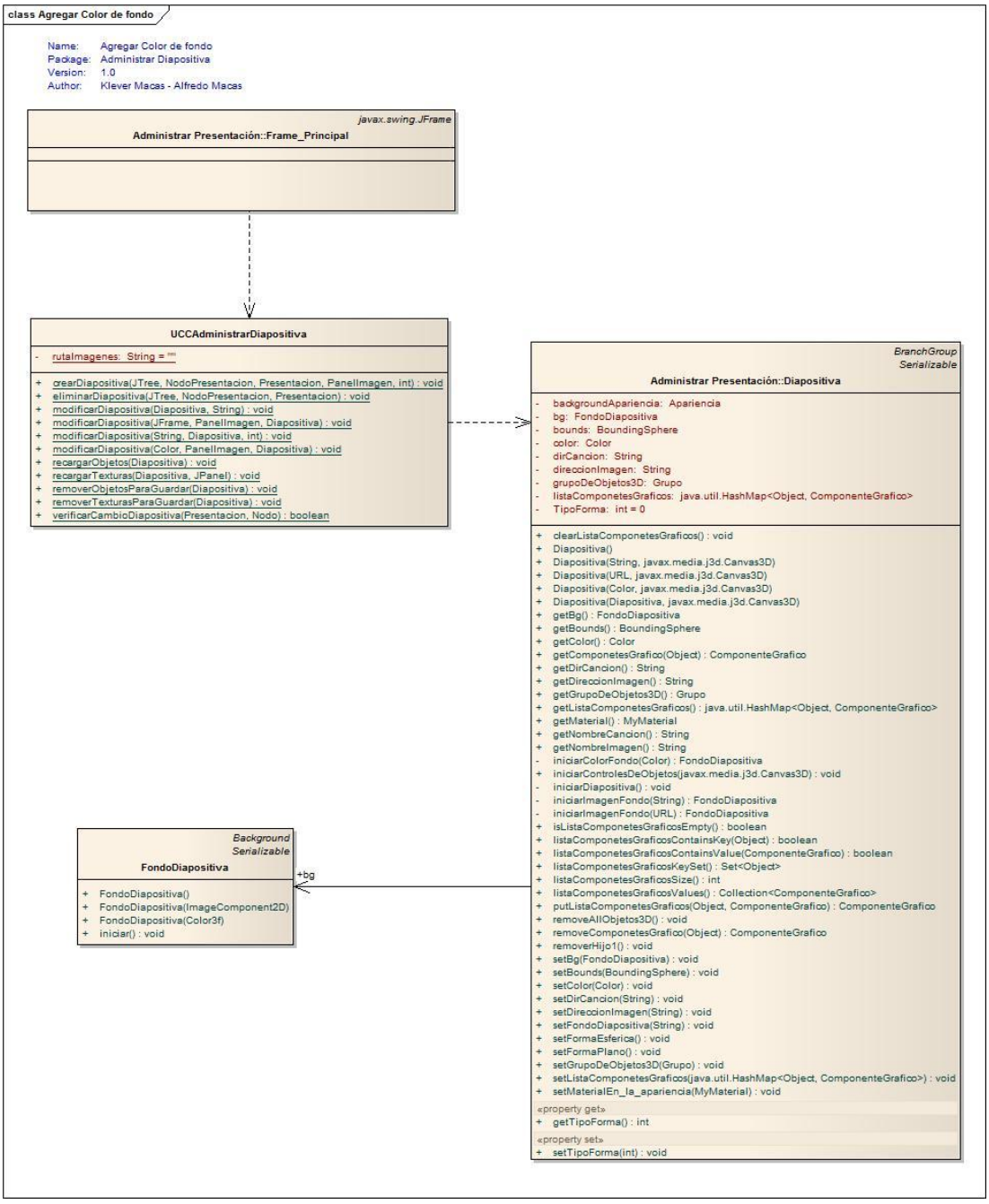


Figura 132 DC Agregar Color de Fondo

8.7.4.13. Diagrama de clases para el caso de uso Agregar imagen de fondo.

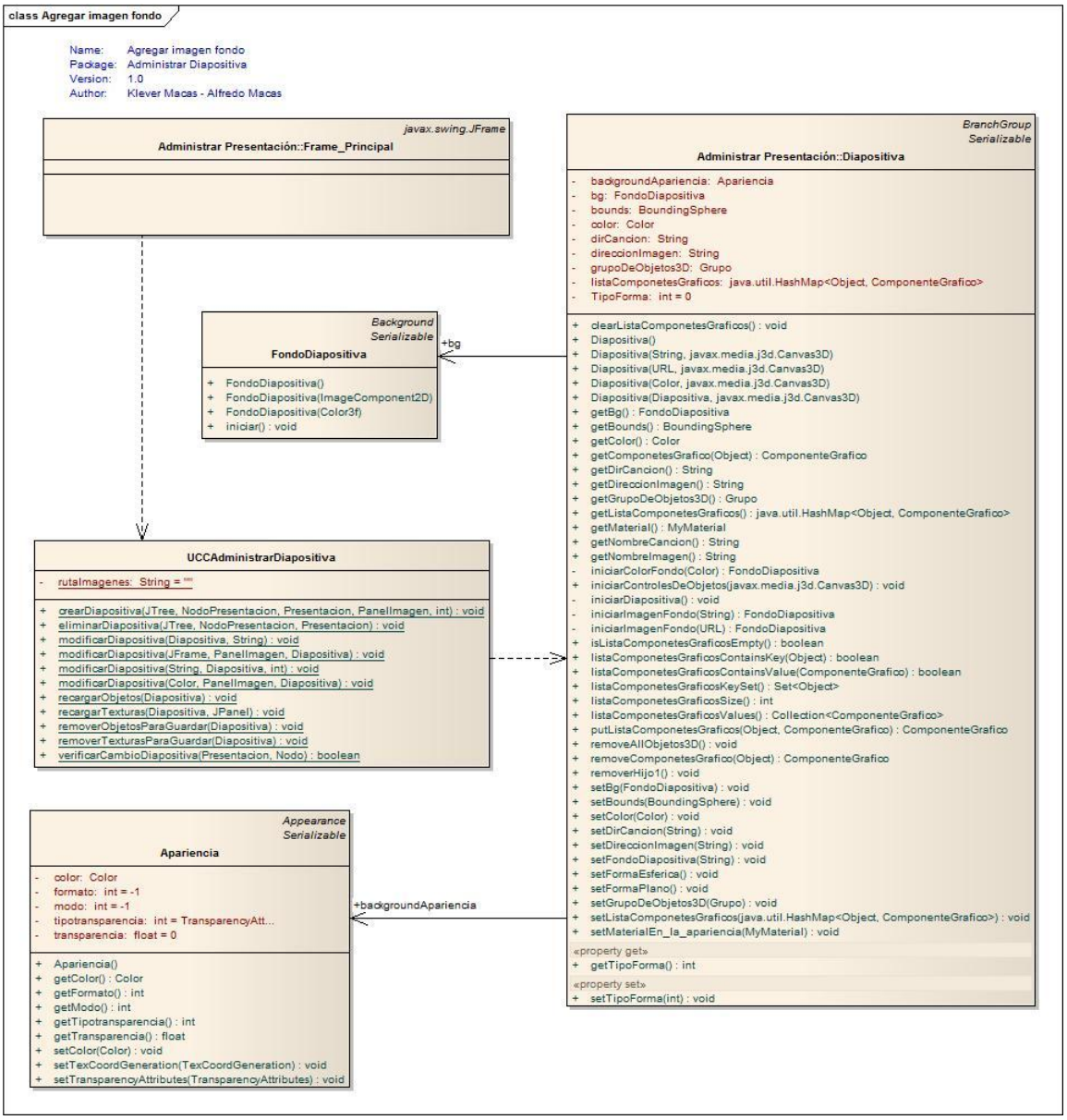


Figura 133 DC Agregar Imagen de Fondo

8.7.4.14. Diagrama de clases para el caso de uso Eliminar diapositiva.

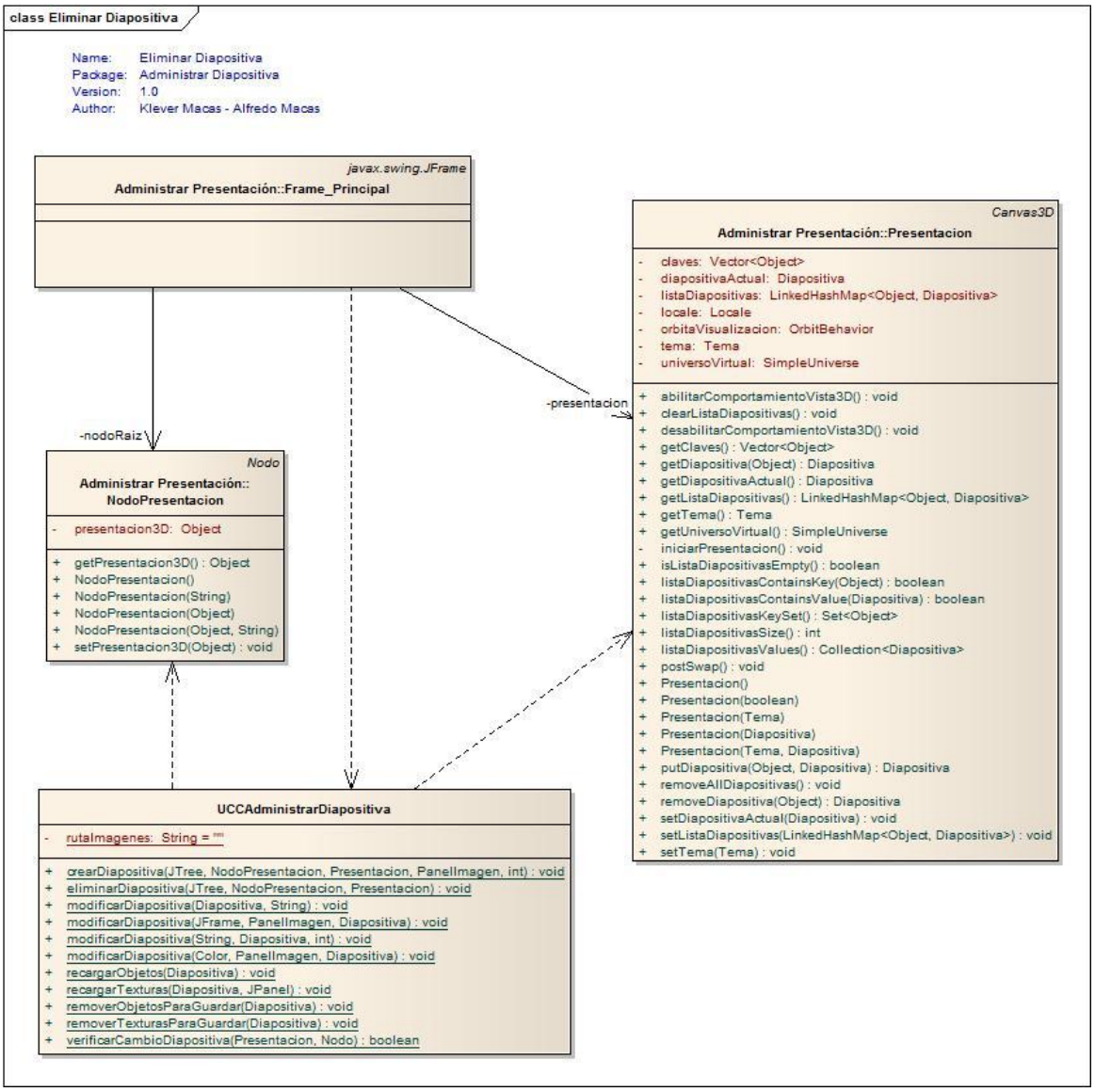


Figura 134 DC Eliminar Diapositiva

8.7.4.15. Diagrama de clases para el caso de uso Exportar diapositiva en formato png o gif.

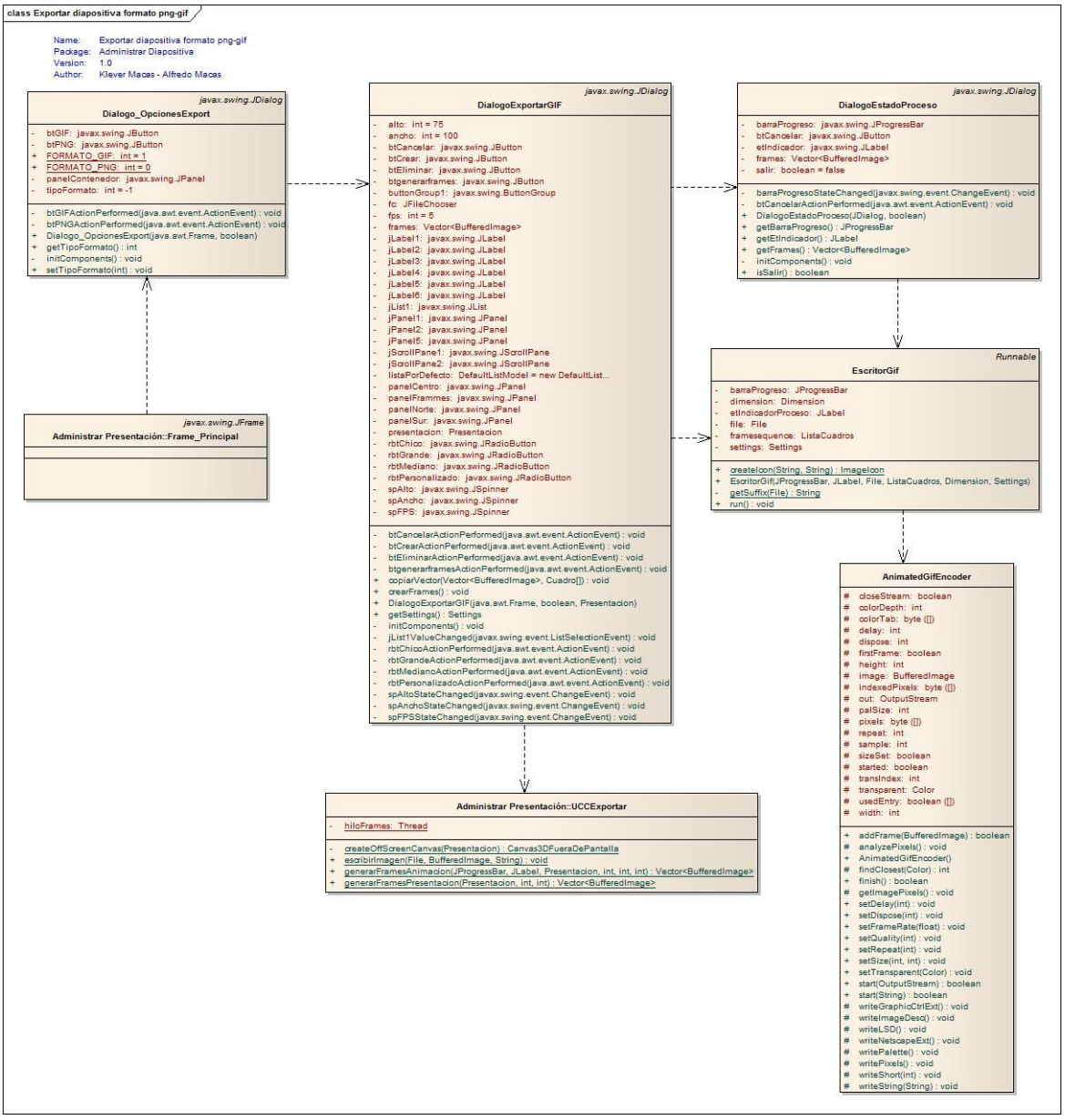


Figura 135 DC Exportar diapositiva en formato png o gif

8.7.4.16. Diagrama de clases para el caso de uso Generar vista previa efecto.

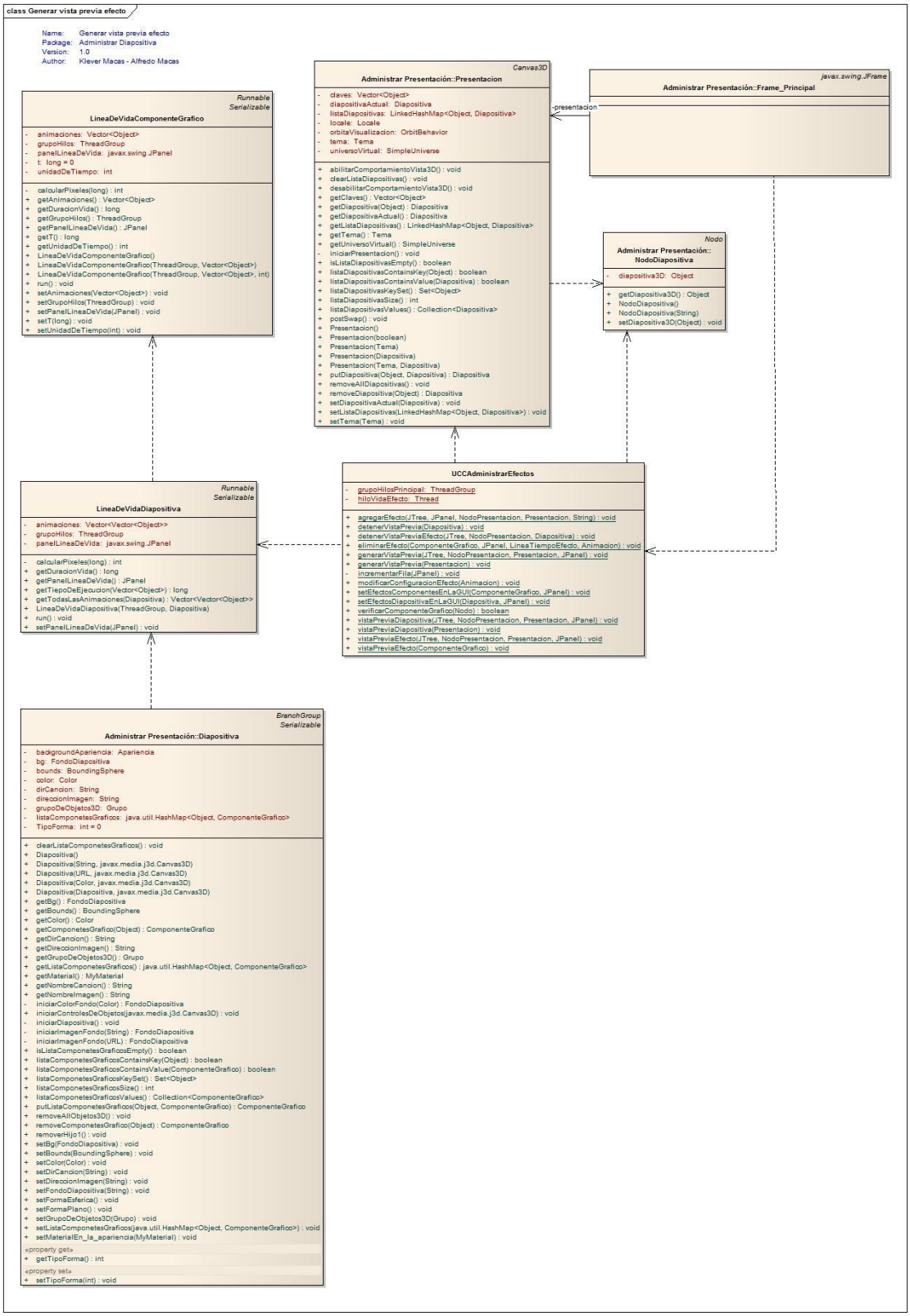


Figura 136 DC Generar Vista Previa de Efecto

8.7.4.17. Diagrama de clases para el caso de uso Agregar componente gráfico.

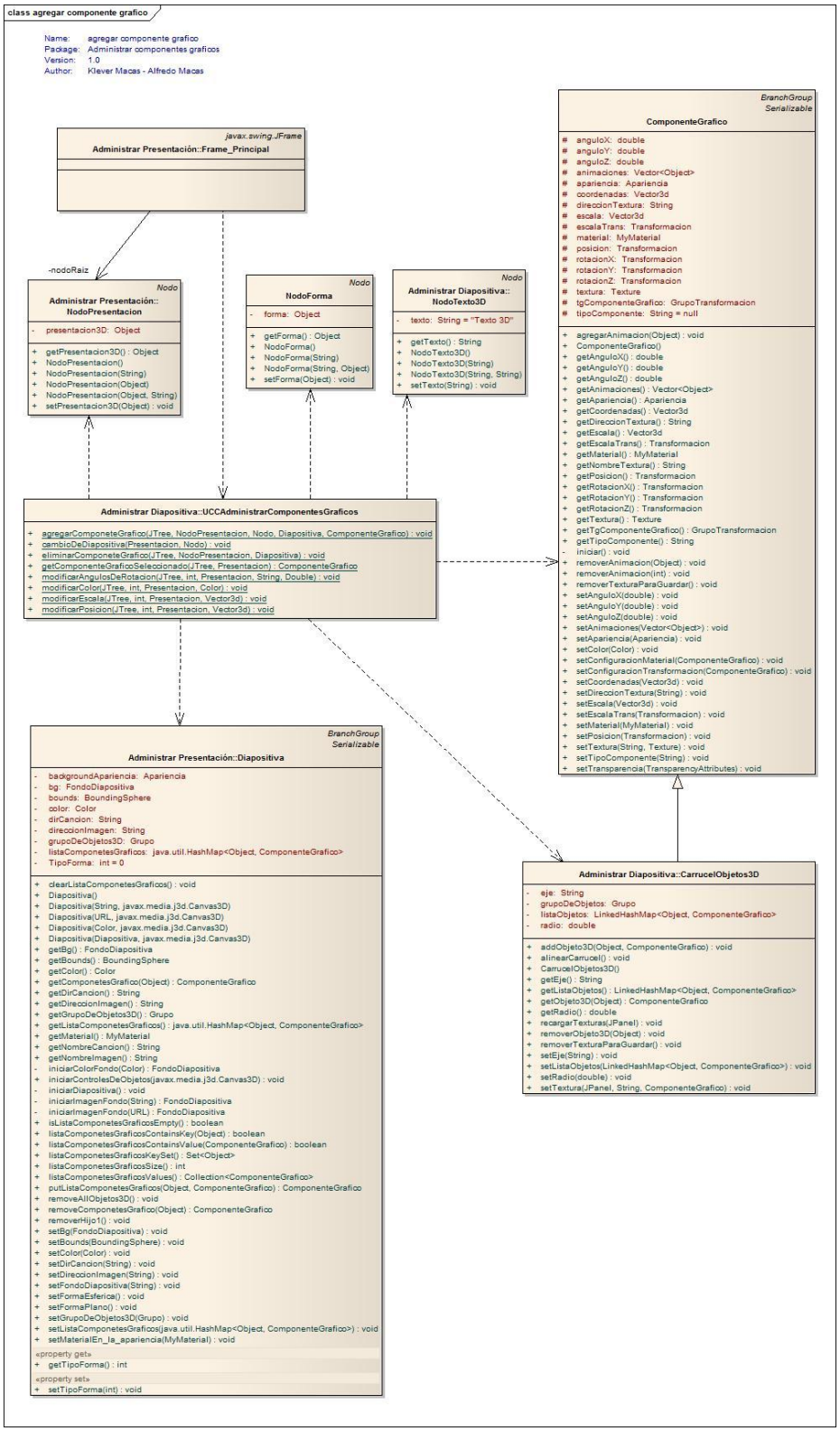


Figura 137 DC agregar Componente Grafico

8.7.4.18. Diagrama de clases para el caso de uso Modificar componente gráfico.

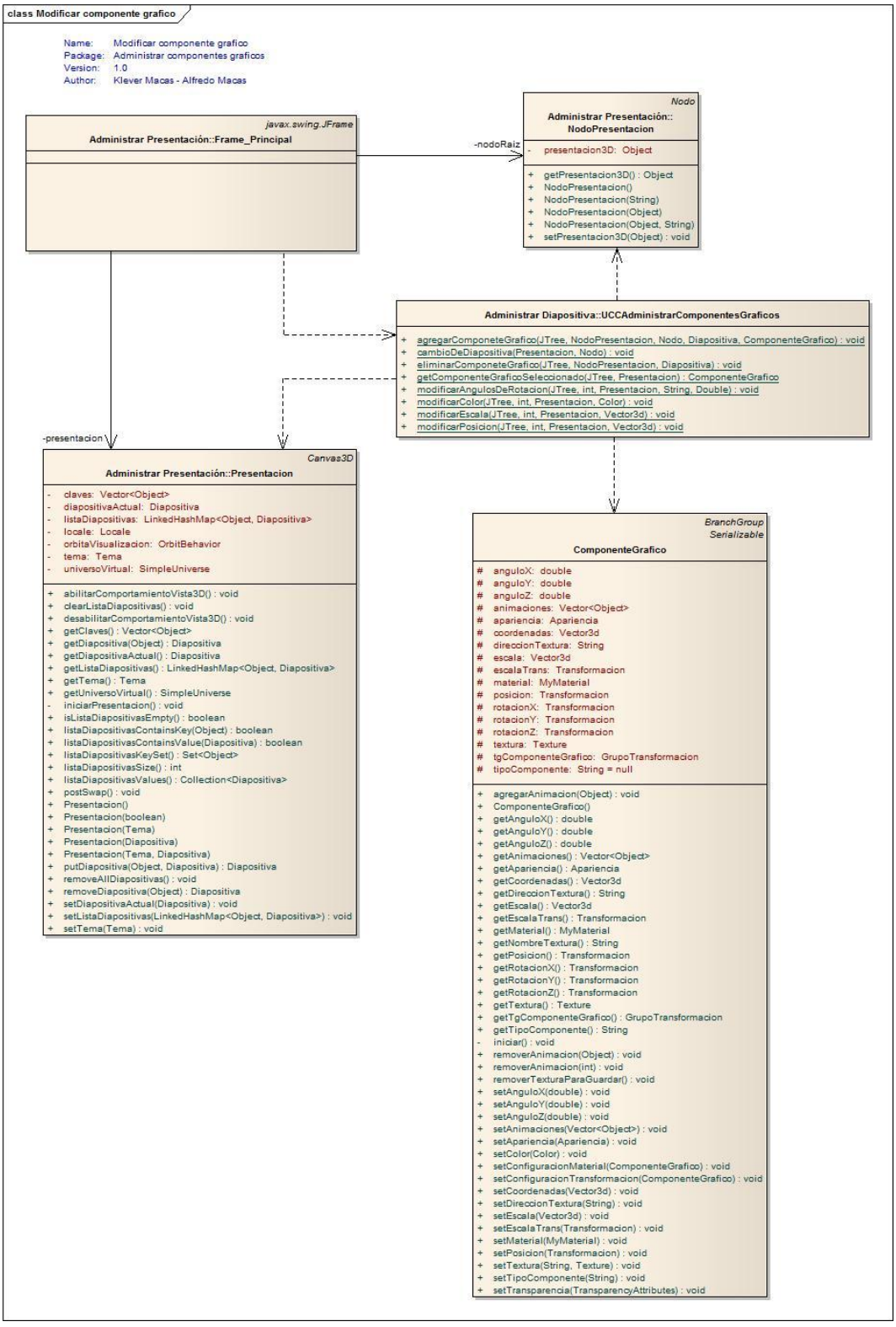


Figura 138 DC Modificar Componente Grafico

8.7.4.19. Diagrama de clases para el caso de uso Eliminar componente gráfico.

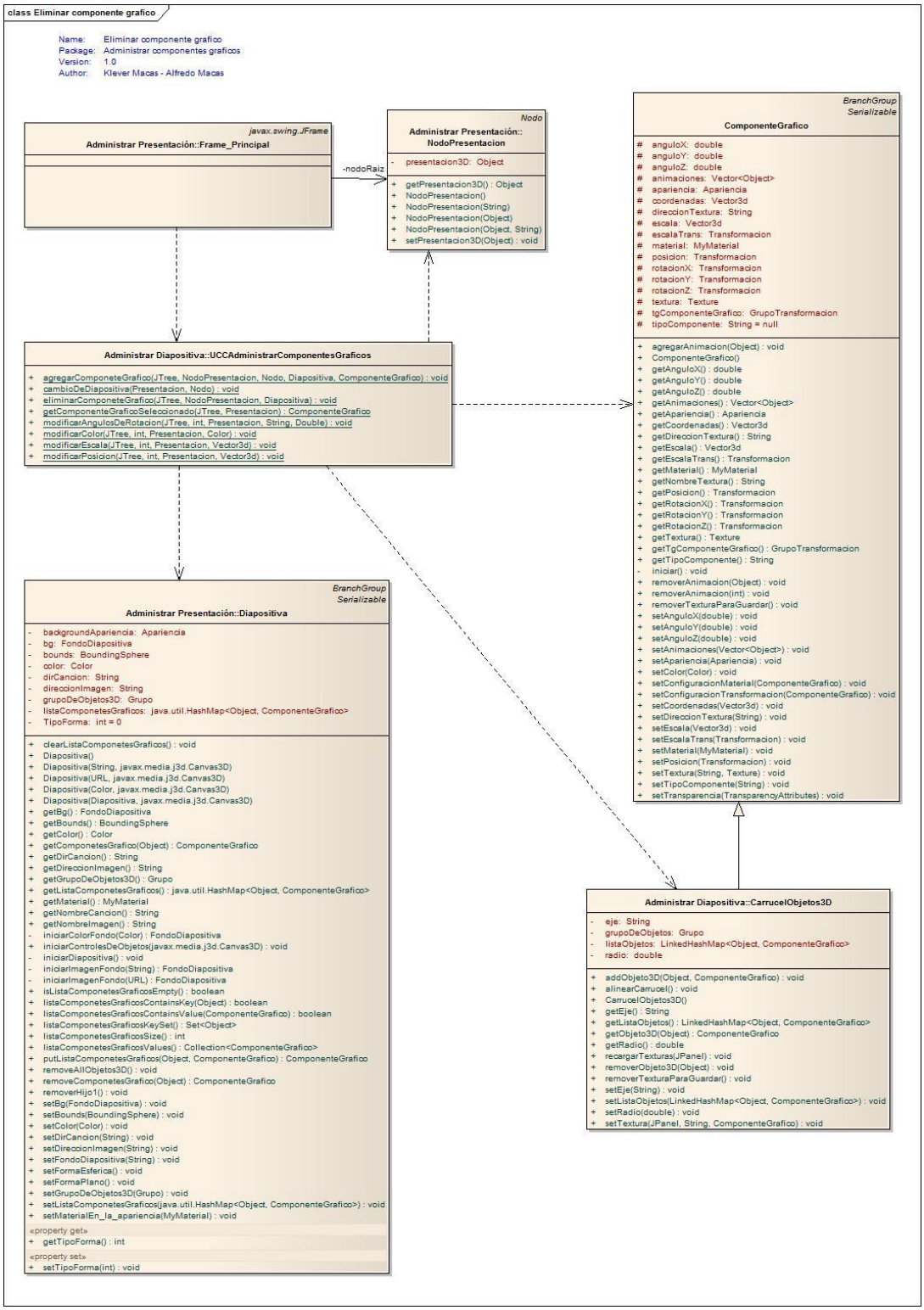


Figura 139 DC eliminar Componente grafico

8.7.4.20. Diagrama de clases para el caso de uso Importar Recursos.

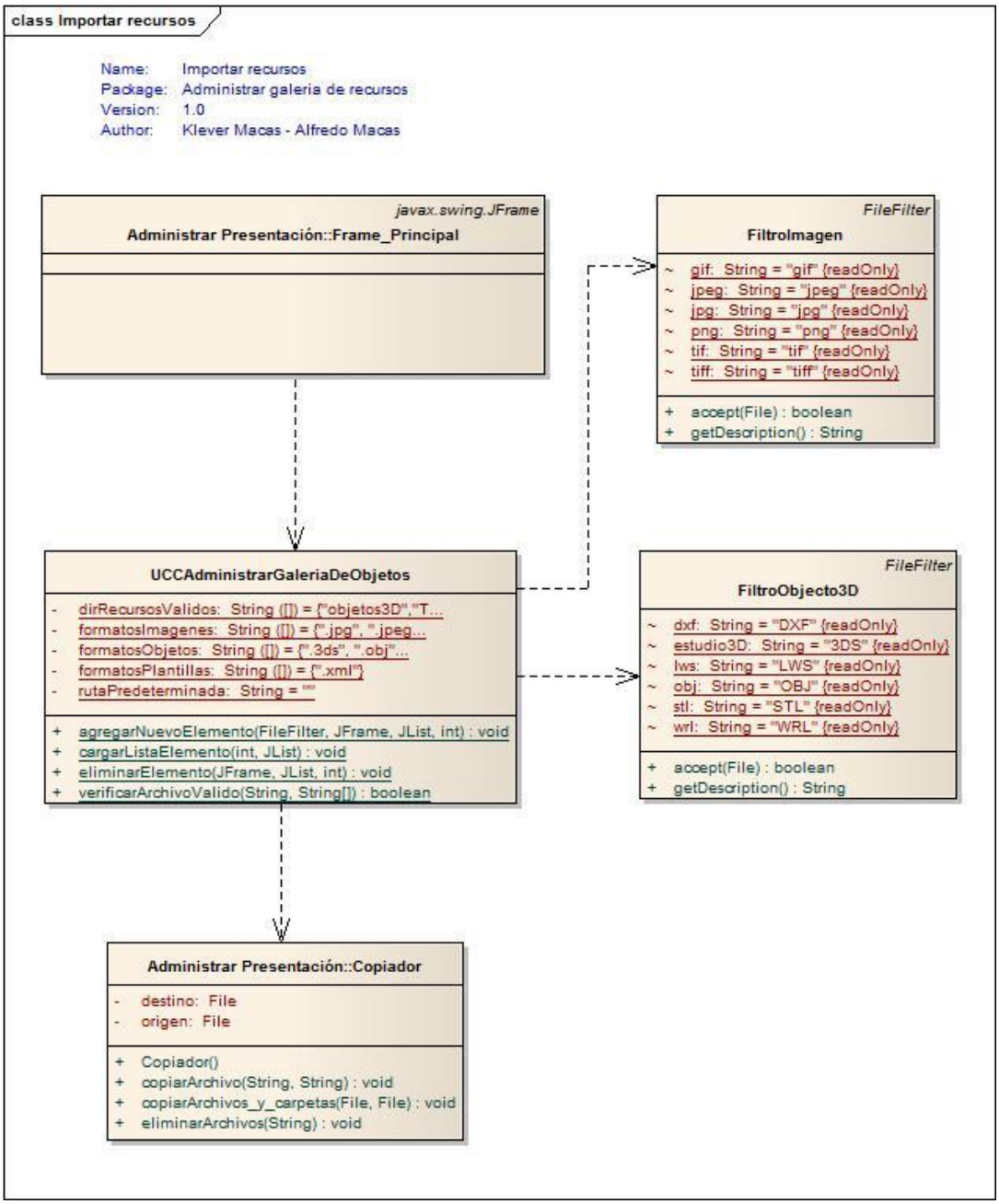


Figura 140 DC Importar Recursos

8.7.4.21. Diagrama de clases para el caso de uso Eliminar Recursos.

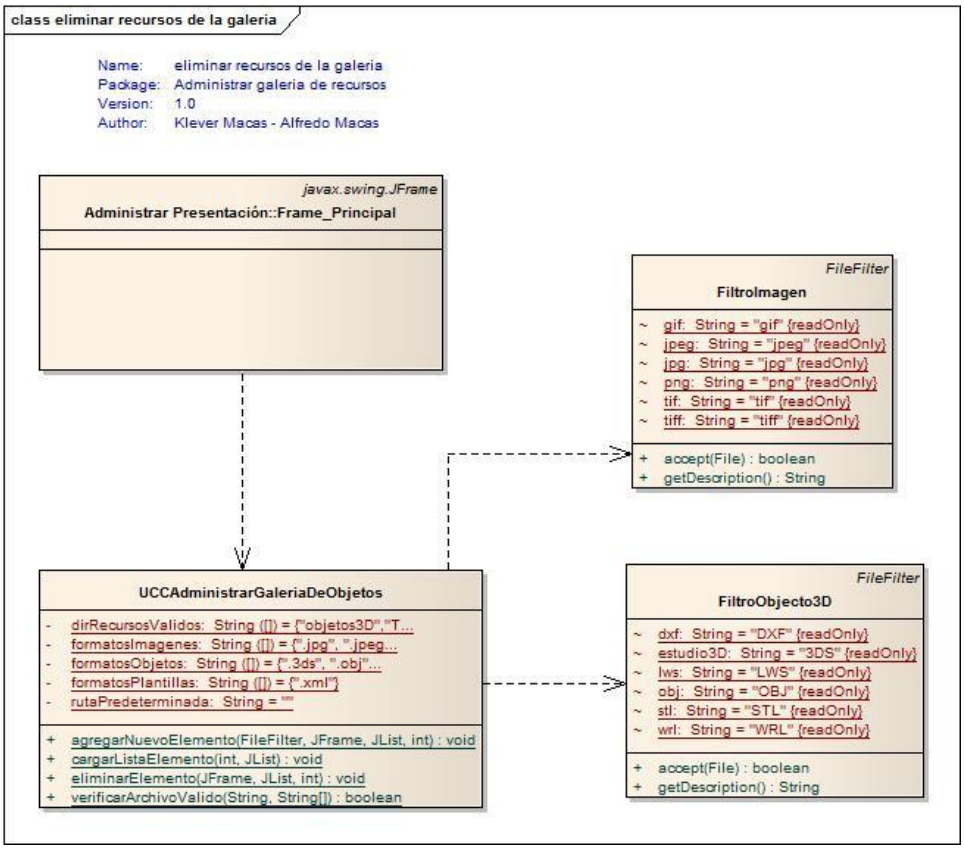


Figura 141 DC Eliminar Recursos

8.7.4.22. Diagrama de clases para el caso de uso Agregar Sonido.

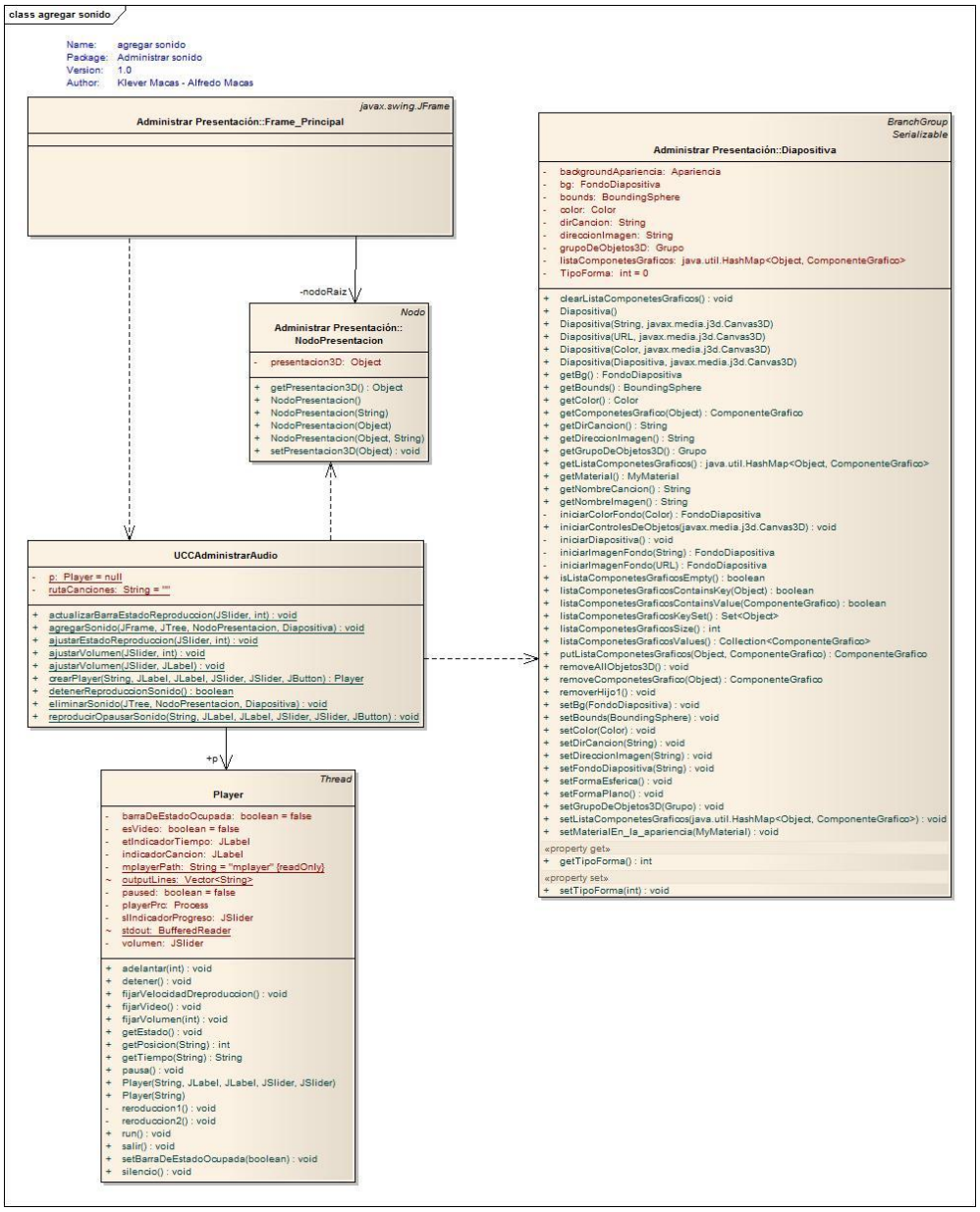


Figura 142 DC Agregar Sonido

8.7.4.23. Diagrama de clases para el caso de uso Eliminar Sonido.

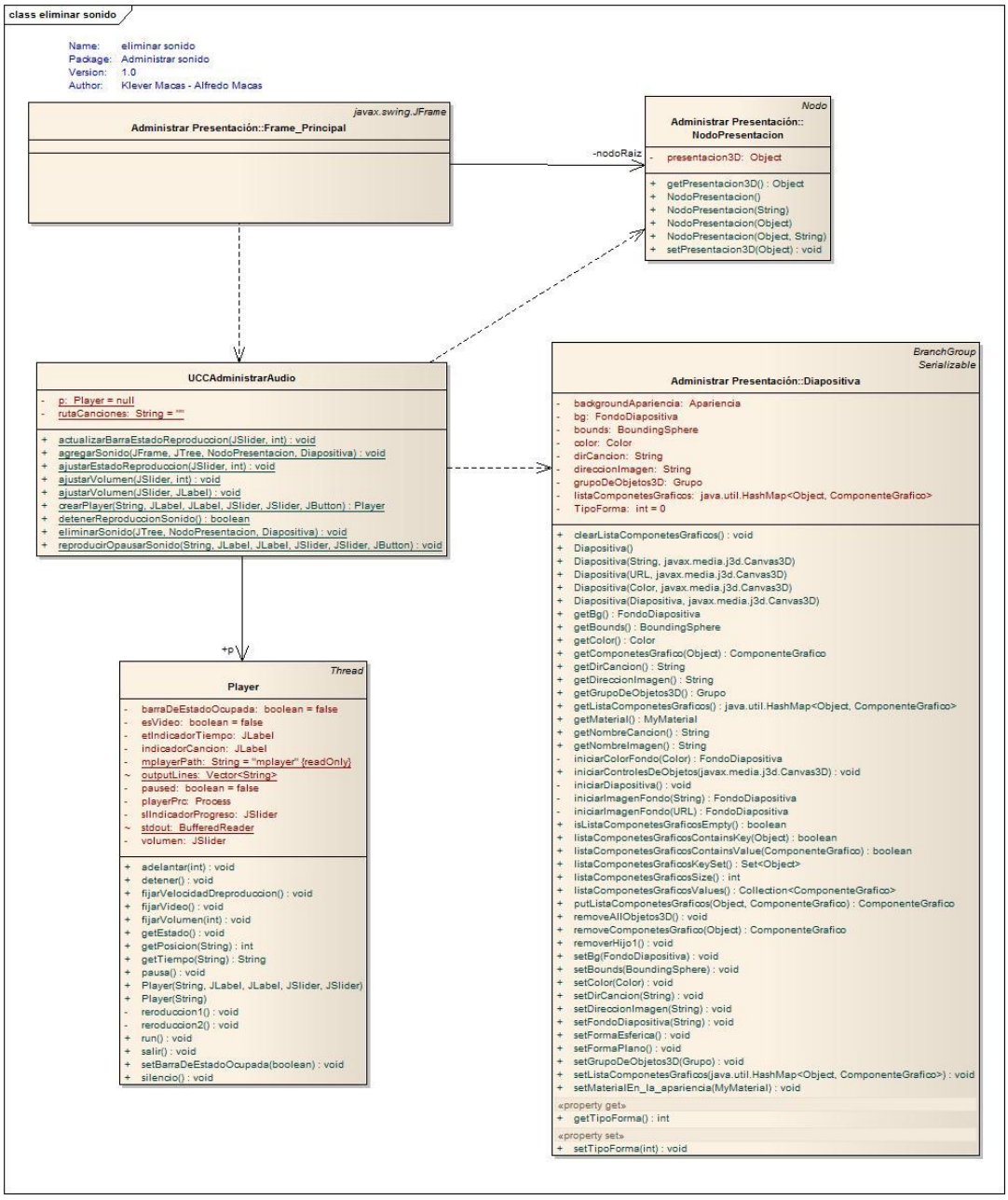


Figura 143 DC Eliminar Sonido

8.7.4.24. Diagrama de clases para el caso de uso Agregar Material.

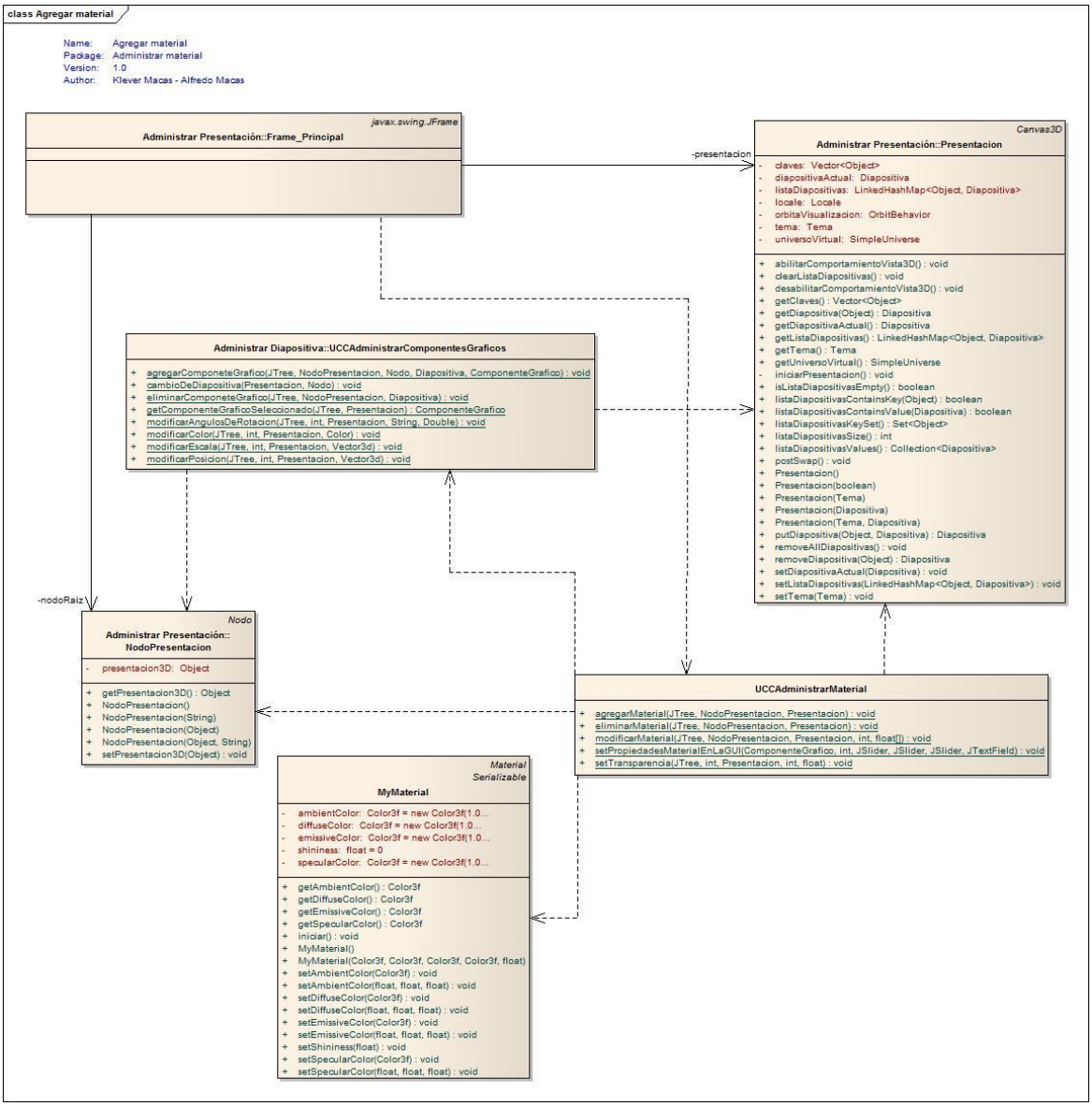


Figura 144 DC Agregar Material

8.7.4.25. Diagrama de clases para el caso de uso Modificar Configuración de material.

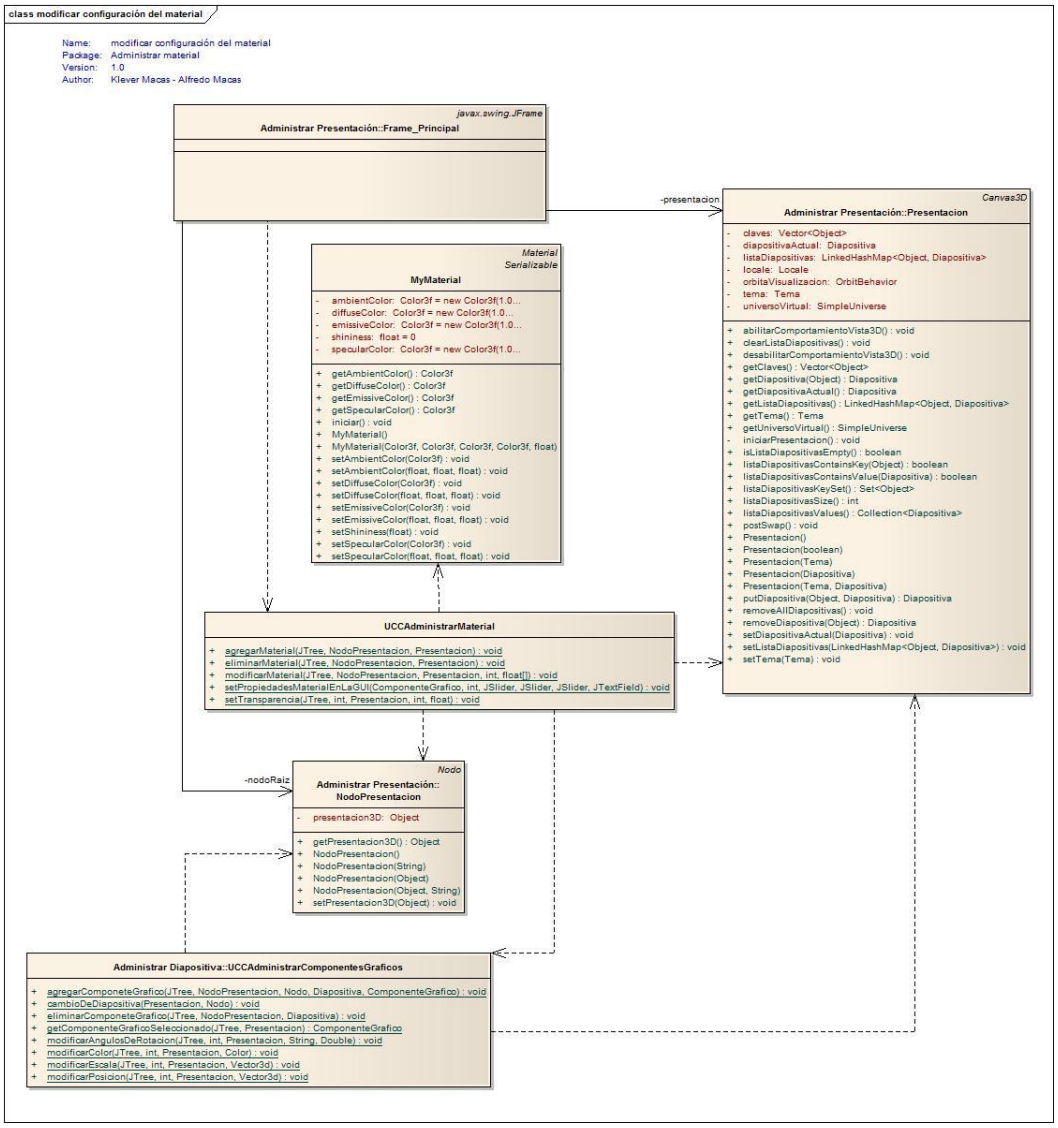


Figura 145 DC Modificar Configuración de Material

8.7.4.26. Diagrama de clases para el caso de uso Eliminar Material.

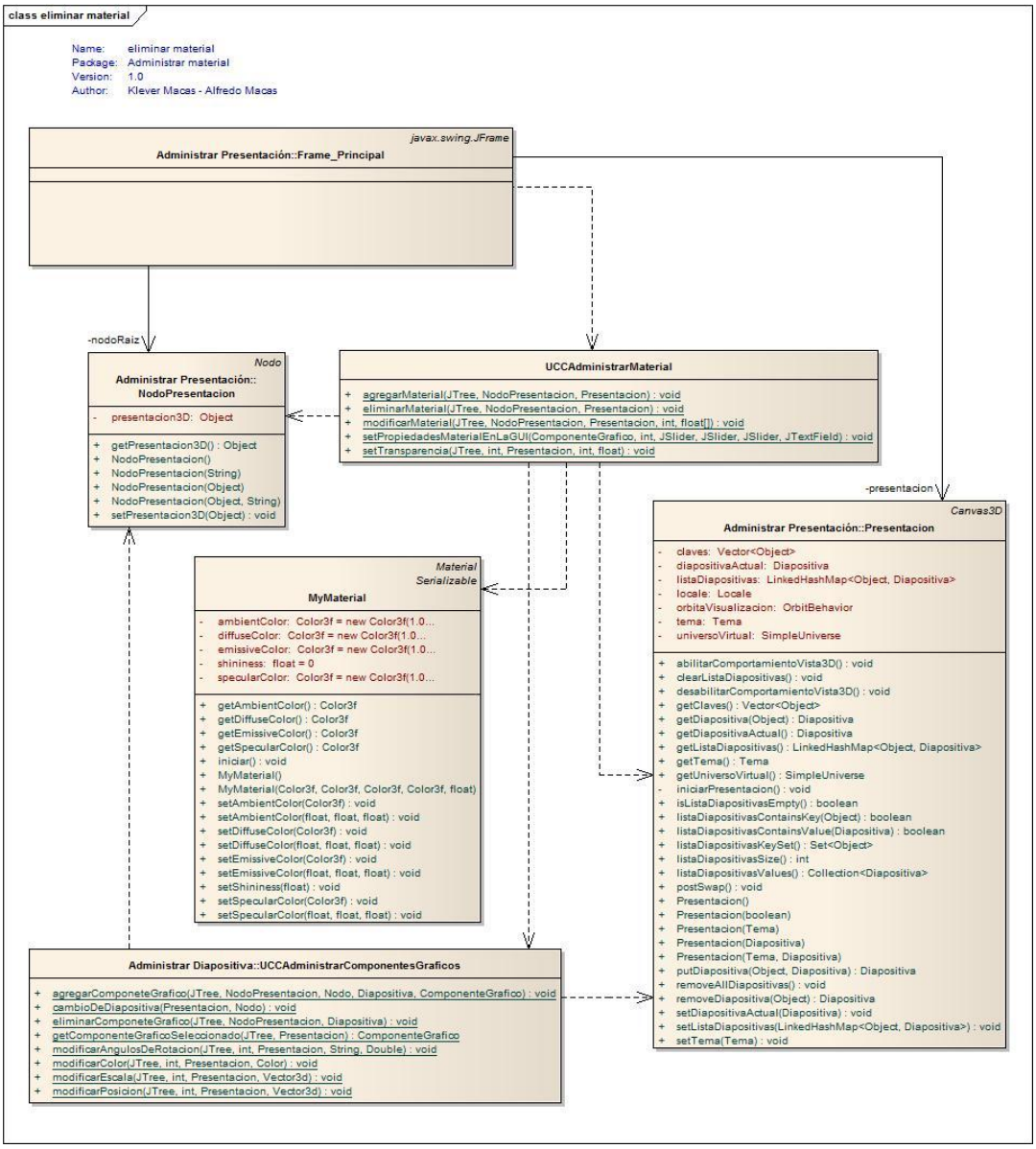


Figura 146 DC Eliminar Material

8.7.4.27. Diagrama de clases para el caso de uso Agregar Efecto al componente gráfico.

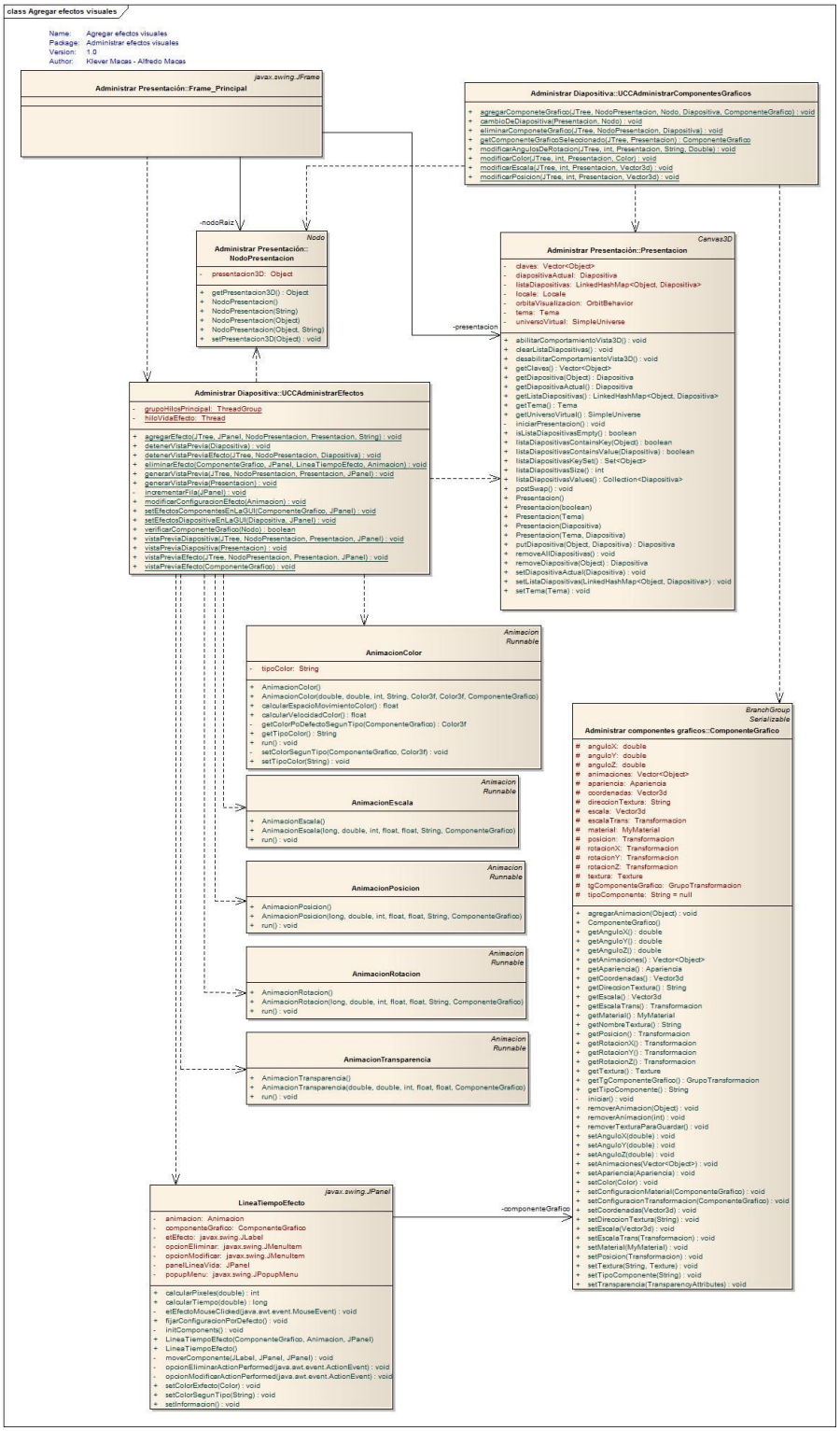


Figura 147 DC Agregar Efecto al Componente Grafico

8.7.4.28. Diagrama de clases para el caso de uso Modificar configuración del efecto.

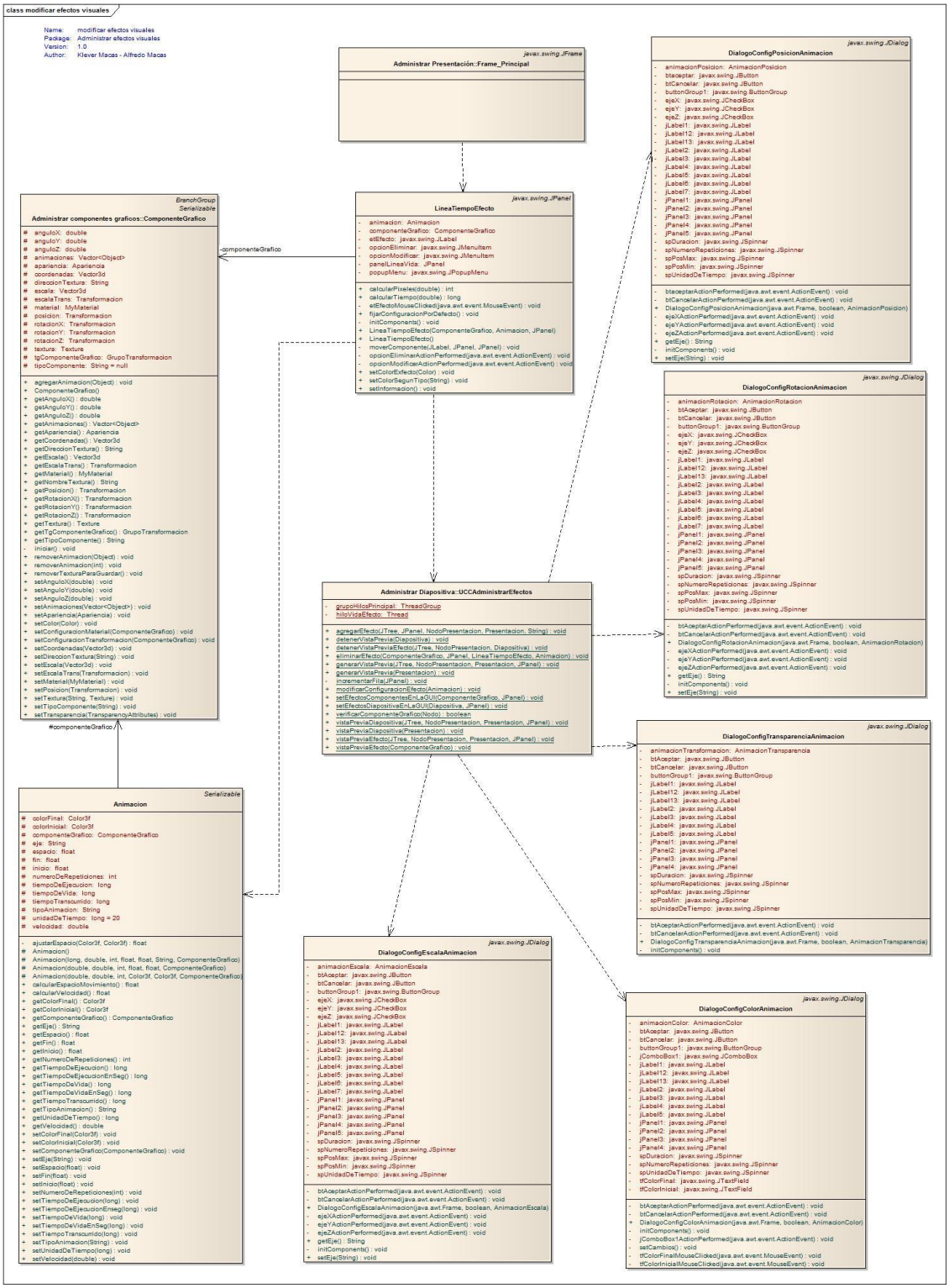


Figura 148 DC Modificar Configuración del Efecto

8.7.4.29. Diagrama de clases para el caso de uso Eliminar efecto del componente gráfico.

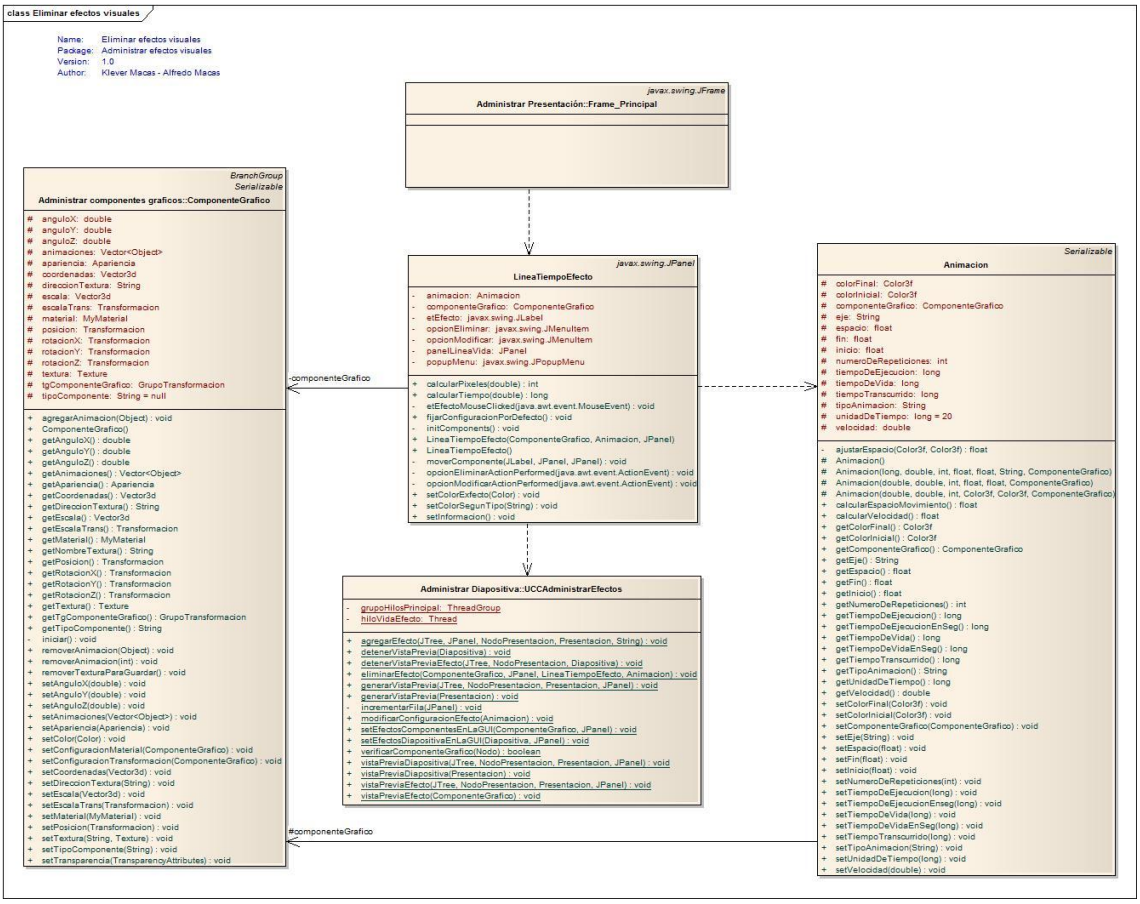


Figura 149 DC Eliminar efecto del componente grafico

8.7.4.30. Diagrama de clases para el caso de uso Agregar textura desde archivo.

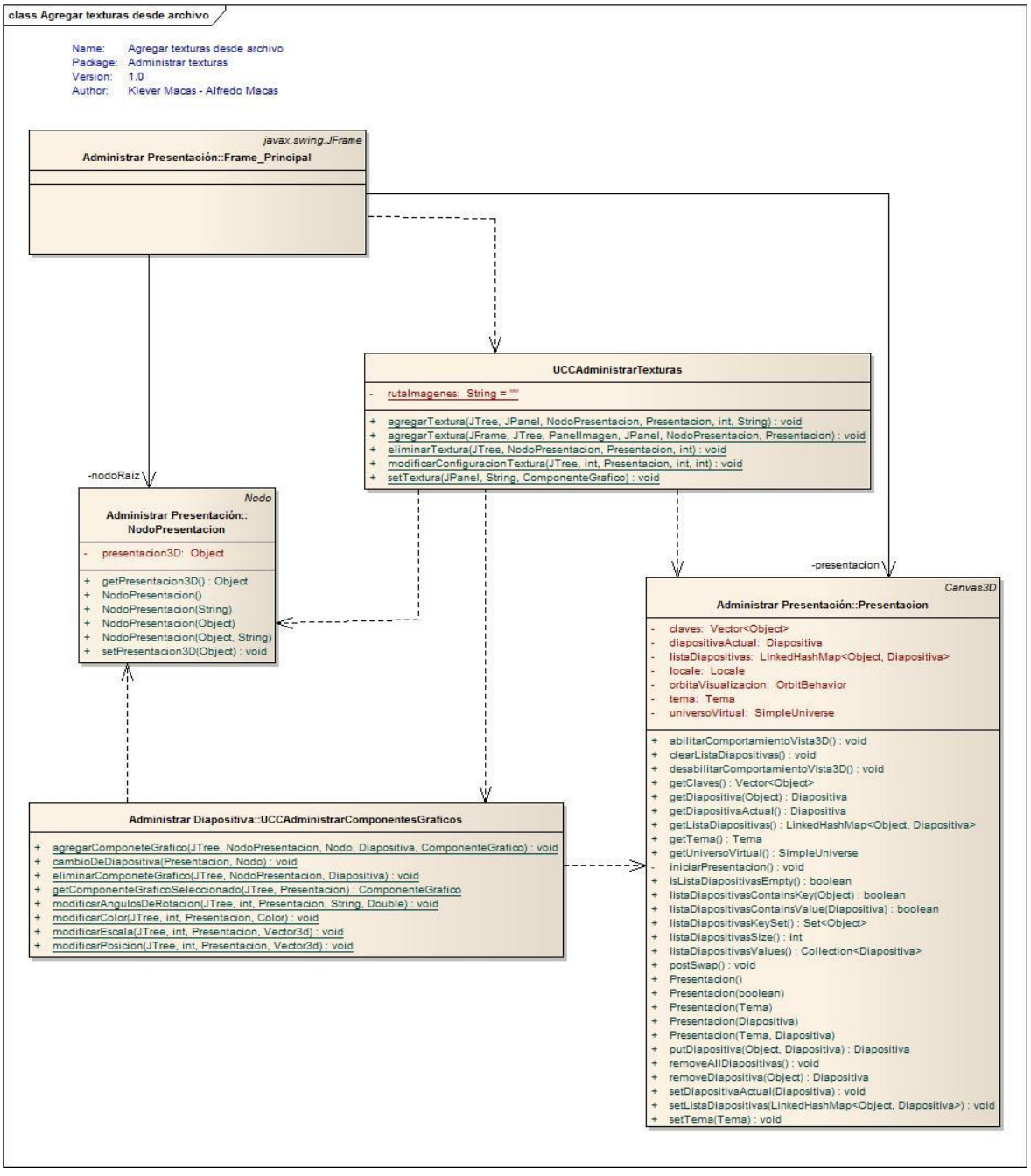


Figura 150 DC Agregar Textura desde Archivo

8.7.4.31. Diagrama de clases para el caso de uso Agregar textura desde galería.

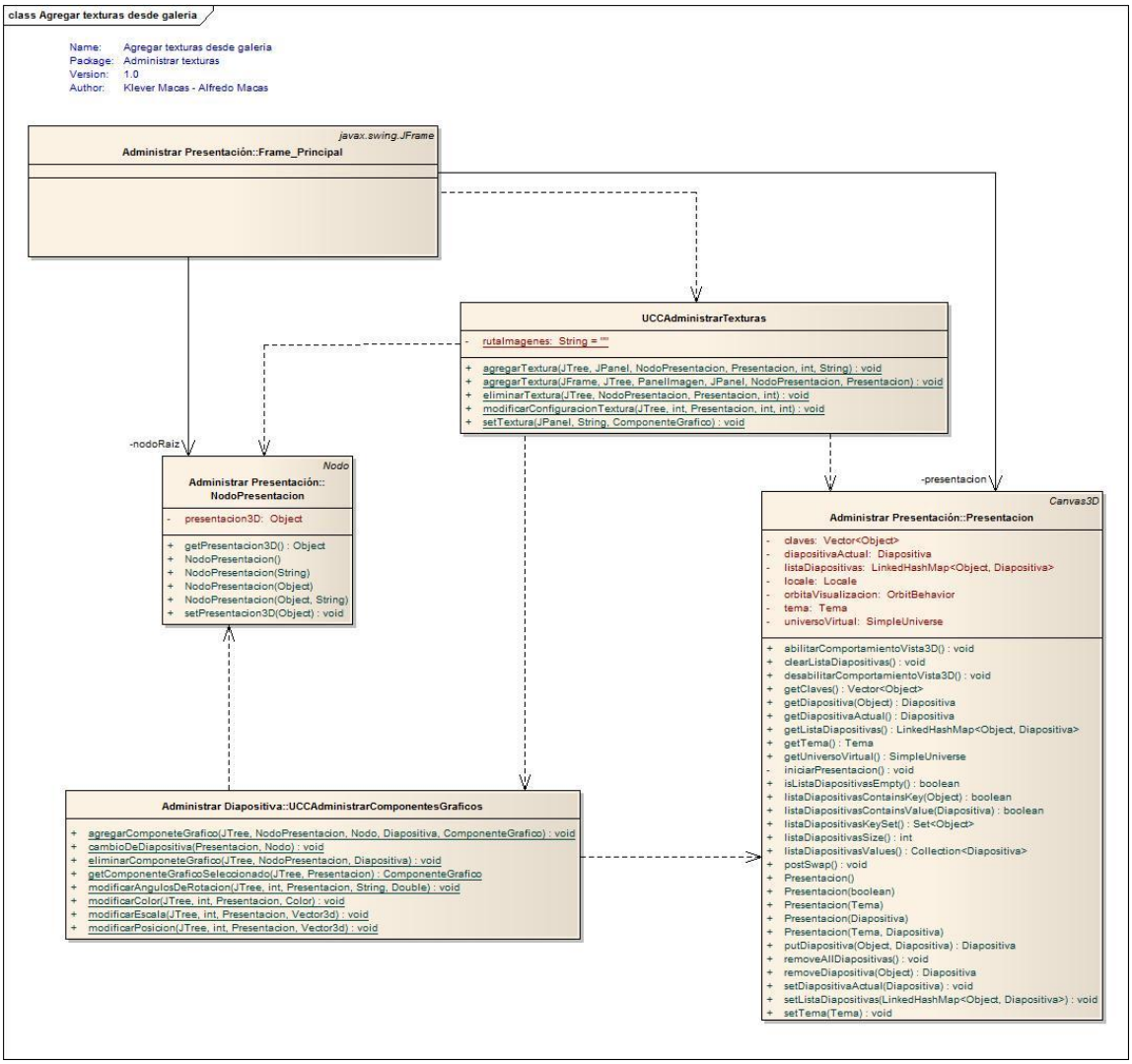


Figura 151 DC Agregar textura desde archivo

8.7.4.32. Diagrama de clases para el caso de uso Configurar textura.

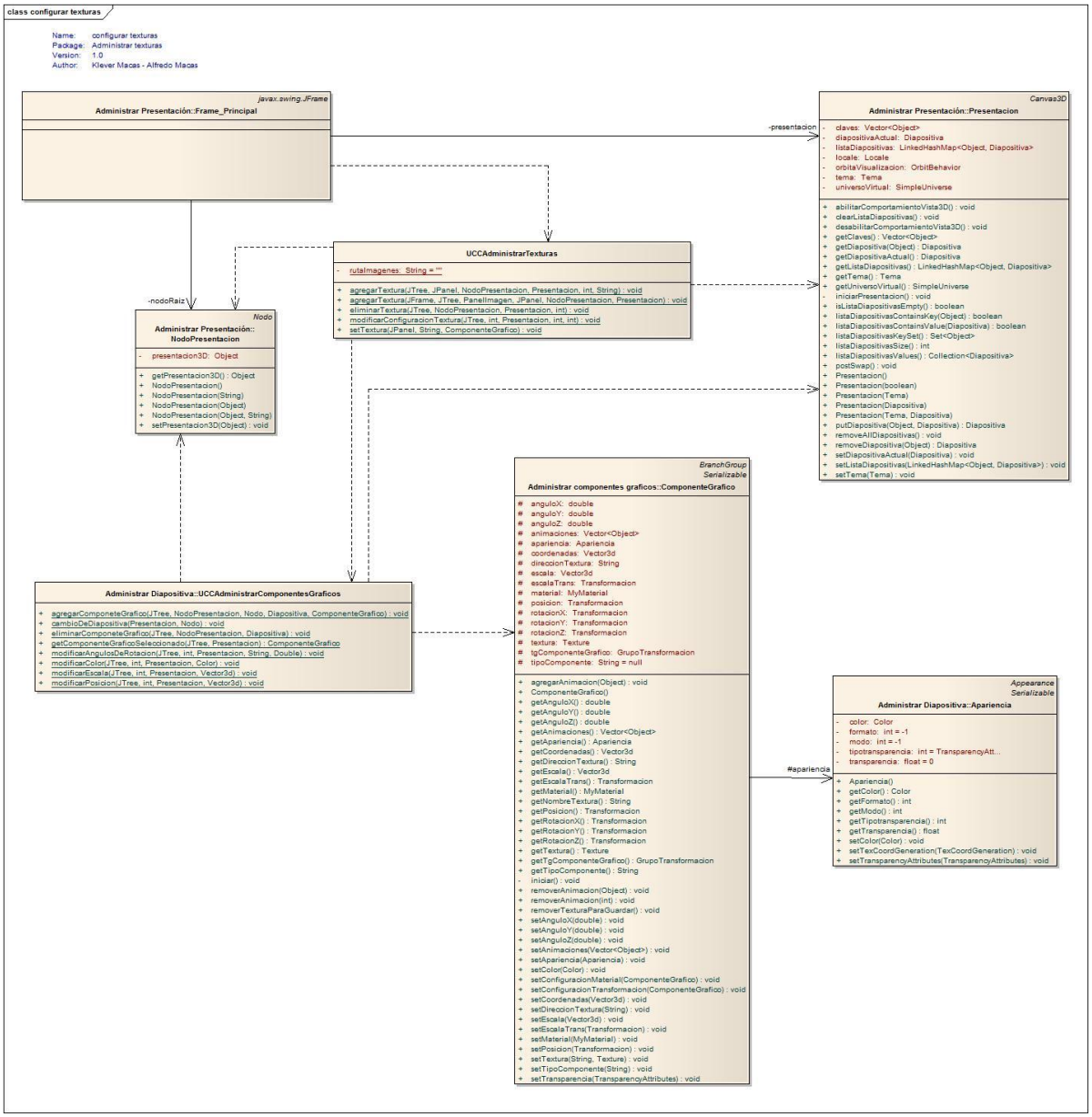


Figura 152 DC Configurar Textura

8.7.4.33. Diagrama de clases para el caso de uso Eliminar textura.

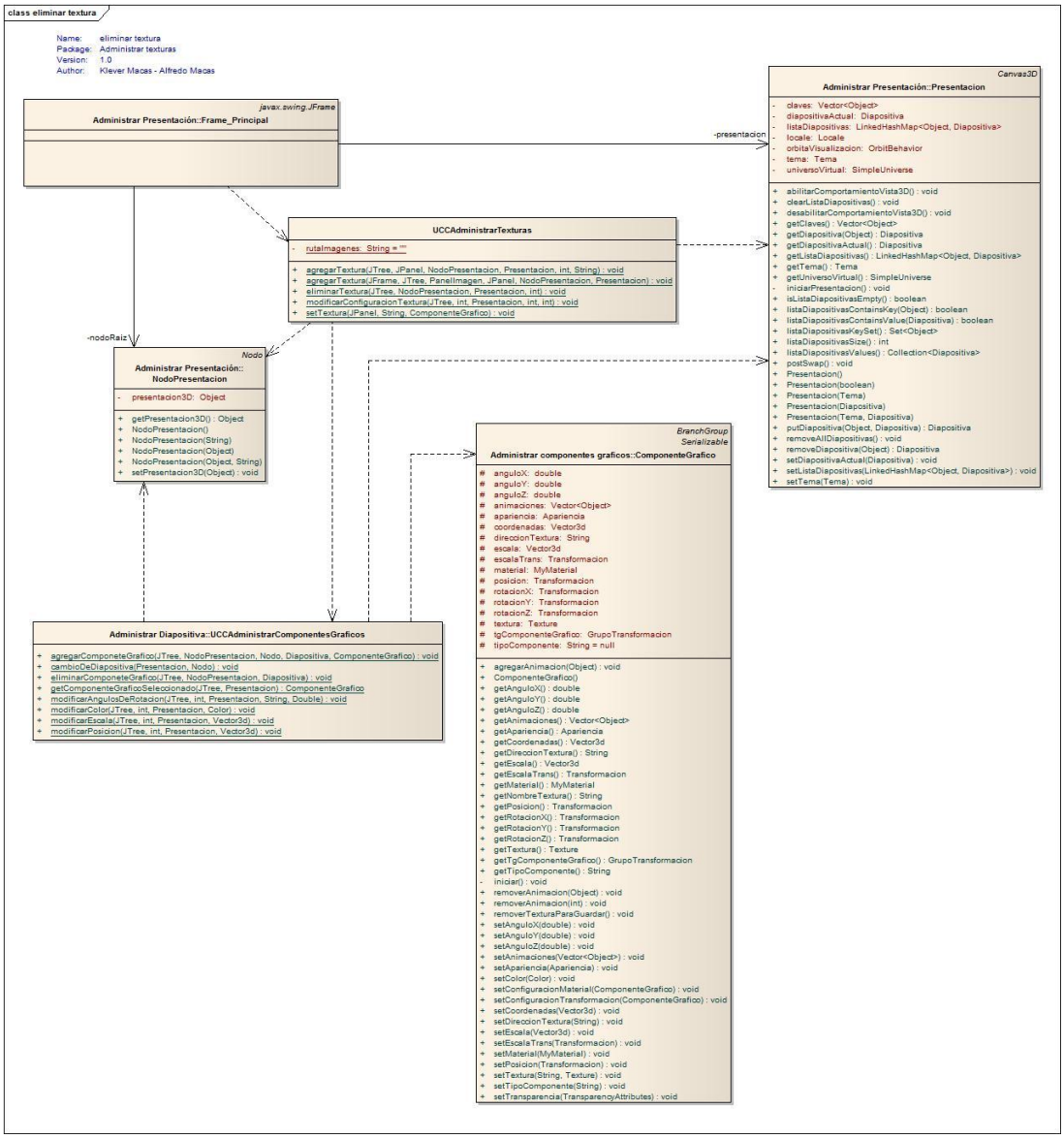


Figura 153 DC Eliminar Textura

8.7.5. Diagrama de componentes

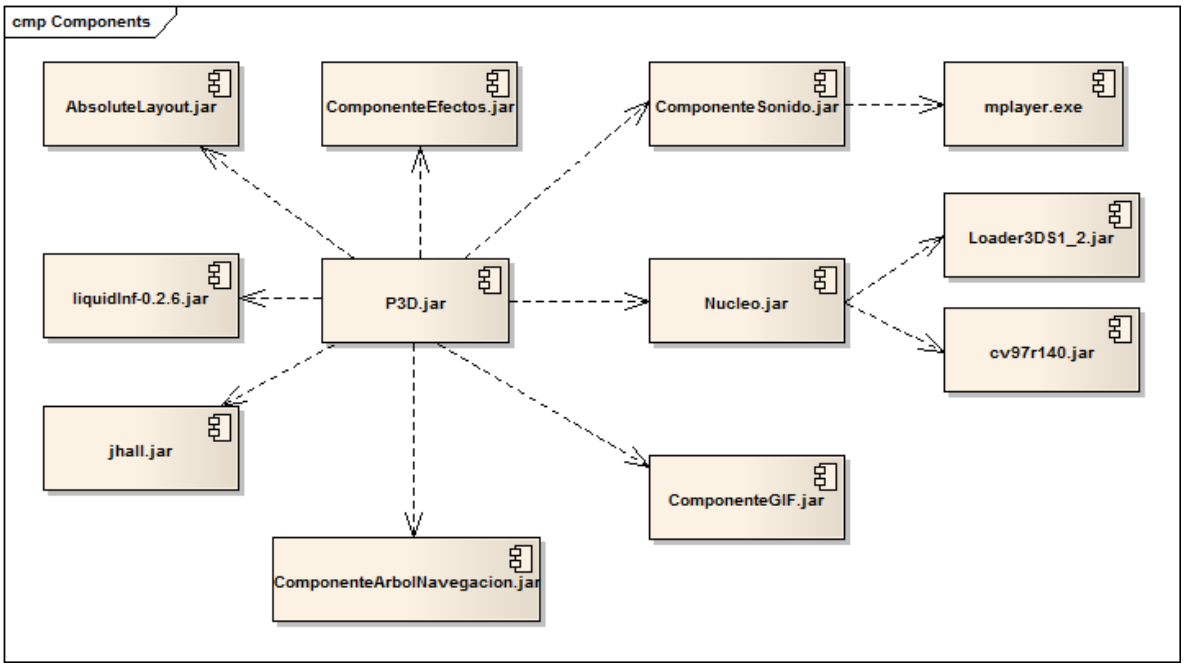


Figura 154 Diagrama de Componentes

8.7.6. Diagrama de Despliegue

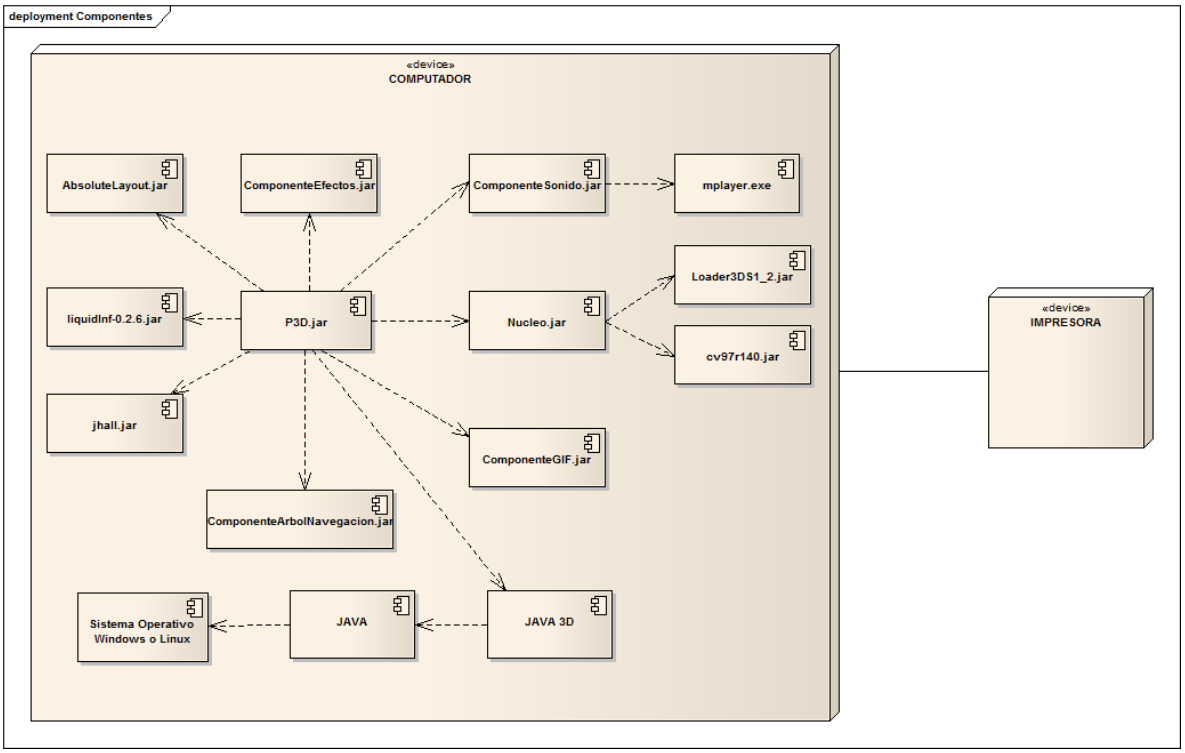


Figura 155 Diagrama de Paquetes

8.8. Algoritmo para ordenar varios objetos dentro de la presentación.

Para poder ordenar múltiples objetos en una diapositiva se diseñó el siguiente algoritmo basándonos en la ecuación de la circunferencia.

ALGORITMO PARA ORDENAR HORIZONTALMENTE

1. Obtener todos los objetos que se desea ordenar.
2. Dividir 360 grados para el número de objetos obtenidos lo cual nos devuelve un valor al cual denotaremos con la letra "k" y será el ángulo de separación entre dos objetos.
3. Inicializar el radio (r) de la circunferencia.
4. Calcular las coordenadas del objeto en base a las funciones trigonométricas x=cosβ.r y z=sinβ.r
5. Fijar las coordenadas al componente gráfico.
6. Fijamos los ángulos de rotación del componente, x=0; y=90-angulo de separación; z=0.
7. Incrementamos el ángulo de separación.
8. Repetir los pasos del 4 al 7 hasta terminar con todos los objetos.

Seudocódigo.

```
INICIO
    Lista=array de objetos;
    Real k=360/número de elementos de la lista;
    Real  radio=valor;
    Real x=0;
    Real y=0;
    Real z=0;
    Real ángulo=k;
    Para i=0 hasta i<número de elementos de la lista
        x=coseno (ángulo).radio;
        z=seno (ángulo).radio;
        Componente grafico cg=Lista[i];
        cg.fijarcoordenadas(x, y, z);
        cg.fijarangulox (0);
        cg.fijaranguloy (90-angulo);
```

```
            cg.fijaranguloz (0);
            ángulo=ángulo+k;
        fin para;
FIN
```

ALGORITMO PARA ORDENAR VERTICALMENTE

1. Obtener todos los objetos que se desea ordenar.
2. Dividir 360 grados para el número de objetos obtenidos lo cual nos devuelve un valor al cual denotaremos con la letra "k" y será el ángulo de separación entre dos objetos.
3. Inicializar el radio (r) de la circunferencia.
4. Calcular las coordenadas del objeto en base a las funciones trigonométricas y=cosβ.r y z=sinβ.r
5. Fijar las coordenadas al componente gráfico.
6. Fijamos los ángulos de rotación del componente, x=ángulo de separación-90; y=0; z=0.
7. Incrementamos el ángulo de separación.
8. Repetir los pasos del 4 al 7 hasta terminar con todos los objetos.

Seudocódigo.

```
INICIO
        Lista=array de objetos;
        Real k=360/número de elementos de la lista;
        Real  radio=valor;
        Real x=0;
        Real y=0;
        Real z=0;
        Real ángulo=k;
        Para i=0 hasta i<número de elementos de la lista
                y=coseno (ángulo).radio;
                z=seno (ángulo).radio;
                Componente grafico cg=Lista[i];
                cg.fijarcoordenadas(x, y, z);
```

```
            cg.fijarangulox (ángulo-90);
            cg.fijaranguloy (0);
            cg.fijaranguloz (0);
            ángulo=ángulo+k;
        fin para;
FIN
```

ALGORITMO PARA ORDENAR EN EJE Z

1. Obtener todos los objetos que se desea ordenar.
2. Dividir 360 grados para el número de objetos obtenidos lo cual nos devuelve un valor al cual denotaremos con la letra "k" y será el ángulo de separación entre dos objetos.
3. Inicializar el radio (r) de la circunferencia.
4. Calcular las coordenadas del objeto en base a las funciones trigonométricas x=cosβ.r y Y=sinβ.r
5. Fijar las coordenadas al componente gráfico.
6. Fijamos los ángulos de rotación del componente, x=0; y=0.
7. Incrementamos el ángulo de separación.
8. Repetir los pasos del 4 al 7 hasta terminar con todos los objetos.

Seudocódigo.

```
INICIO
        Lista=array de objetos;
        Real k=360/número de elementos de la lista;
        Real  radio=valor;
        Real x=0;
        Real y=0;
        Real z=0;
        Real ángulo=k;
        Para i=0 hasta i<número de elementos de la lista
            x=coseno (ángulo).radio;
            Y=seno (ángulo).radio;
            Componente grafico cg=Lista[i];
```

```
            cg.fijarcoordenadas(x, y, z);
            cg.fijarangulox (0);
            cg.fijaranguloy (0);
            ángulo=ángulo+k;
        fin para;
FIN
```

8.9. Plan de pruebas

La ejecución del plan de pruebas en el desarrollo de un sistema informatizado es una etapa que no se puede excluir de su proceso de desarrollo. Esta fase permite evaluar el rendimiento del sistema construido así como detectar, identificar y corregir errores e incrementar sugerencias adicionales. También se puede realizar pruebas al sistema durante todo el proceso de implementación lo que facilita la corrección de errores con mayor pertinencia.

La fase de pruebas implica dos partes fundamentales que son la verificación y la validación.

8.9.1. Verificación

Parte de la fase de ejecución de pruebas que consiste en la evaluación del sistema desarrollado o de uno de sus componentes en escenarios simulados, para comprobar si este satisface las especificaciones inicialmente definidas en la fase de análisis y diseño de la aplicación. Así mismo permite comprobar si el sistema se está construyendo correctamente y que no contiene errores de implementación.

8.9.2. Validación

Consiste en demostrar que un sistema sistematizado es apropiado para el uso previsto. Esto se logra mediante la realización de un proceso, durante las etapas de desarrollo o al final, que consiste en evaluar en el sistema todos los módulos con el fin de comprobar si estos cumplen con las necesidades y los requisitos del usuario y si el sistema produce las salidas esperadas. Su propósito es suministrar una valoración sobre todos los módulos que permiten tanto al usuario y al programador identificar fallas, falta de controles y sus consecuencias.

El plan a utilizar consta de las siguientes partes:

1. Identificador del plan. Nombre representativo que identifique al plan y que permita relacionarlo con su alcance. Debe distinguirse la versión y la fecha del plan.

2. Alcance. Indica el tipo de prueba y las propiedades/elementos del software a ser probado.

3. Ítems a probar. Indica la configuración a probar y las condiciones mínimas que debe cumplir para comenzar a aplicarse el plan. Se debe tener cuidado en el momento de probar una aplicación que aún tiene fallas pero si esperamos que la aplicación este completamente terminada se pueden detectar fallas graves muy tarde.

4. Estrategia. Describe la técnica, patrón y/o herramientas a utilizarse en el diseño de los casos de prueba.

5. Criterios de suspensión y requisitos de reanudación. Señala las circunstancias en las cuales, el plan debe ser suspendido, repetido o culminado.

6. Documentos a entregar. Especifica los documentos que se generan al culminar el plan de pruebas, esto puede ser una especificación de pruebas, casos de prueba, resumen del proceso y bitácora de pruebas.

7. Recursos. Explica los medios físicos e intangibles necesarios y deseables del ambiente de prueba, incluyendo las características del hardware, el software, el sistema operativo o cualquier otro software necesario para llevar a cabo las pruebas, así como la distribución específica de los componentes del sistema a probar así como también la configuración del software de apoyo. También se incluye los recursos humanos necesarios para la ejecución de las pruebas.

8. Calendario. Señala las fechas establecidas como hitos en la realización de las pruebas y las dependencias en el tiempo de las tareas a realizar.

9. Responsables. Establece el responsable de las tareas programadas en el plan.

En cualquier tipo de validación se debe considerar el personal involucrado en este proceso, las partes del sistema a ser validado y los métodos de validación que se establecerán para ejecutar la validación. El personal involucrado en la validación incluye las personas que harán uso del sistema y que se encuentran inmersos en el contexto de la problemática que se pretende solucionar. En el sistema se validará los resultados presumiblemente útiles para el usuario final, esto nos permitirá obtener como resultado el rendimiento general del sistema y corregir errores que se presenten en la implementación.

8.9.3. Validación orientada al uso: Usabilidad

Usabilidad es la eficacia, eficiencia y satisfacción con la que un producto permite alcanzar objetivos específicos a usuarios específicos en un contexto de uso específico.

Su objetivo fue estudiar la usabilidad del programa en su entorno real con usuarios reales. Para ello se evaluaron aspectos tales como la facilidad de uso del sistema, su robustez, su interfaz gráfica, etc.

La validación orientada al uso se llevó a cabo durante el mes de julio, con un grupo de docentes del Área de la Energía, las Industrias y los Recursos Naturales no Renovables, los cuales mediante la utilización de la herramienta pudieron, verificar las diferentes funciones que posee la aplicación.

En base a la experiencia adquirida en la etapa de prueba por parte de los docentes, se aplicó una encuesta para poder evaluar tanto la funcionalidad, usabilidad e interfaz gráfica, lo que ayudo a detectar problemas que se presentaban en la aplicación.

El número total de docentes del Área de la Energía, las Industrias y los Recursos Naturales no Renovables es de 86, los cuales se encuentran distribuidos en las diferentes carreras existentes del Área, para aplicar las pruebas se tomó un porcentaje tomando en cuenta la técnica de muestreo.

8.9.3.1. Selección de la muestra

Ya que la población general de los docentes del Área es extensa se creyó conveniente aplicar la encuesta a una muestra, para lo cual se procedió a calcularla de la siguiente manera.

Para la seleccionar la muestra tenemos que tener en cuenta los siguientes parámetros:
N: población general.
k= Es una constante que depende del nivel de confianza que asignemos.

k	1,15	1,28	1,44	1,65	1,96	2	2,58
Nivel de confianza	75%	80%	85%	90%	95%	95,5%	99%

Tabla 60 Constante k relacionada de acuerdo al nivel de confianza

e: es el error muestral deseado. El error muestral es la diferencia que puede haber entre el resultado que obtenemos preguntando a una muestra de la población y el que obtendríamos si preguntáramos al total de ella.

q: es la proporción de individuos que poseen en la población la característica de estudio. Los valores a considerar constan en la siguiente tabla:

- Para $3 \leq N \leq 19$ ------- Se asume q = 0,01 (un 1 %).
- Para $20 \leq N \leq 29$ ------ Se asume q = 0,01 hasta 0,02 (del 1 al 2 %).
- Para $30 \leq N \leq 79$ ----- Se asume q = 0,02 hasta 0,05 (del 2 al 5 %).
- Para $80 \leq N \leq 159$ ---- Se asume q = 0,05 hasta 0,10 (del 5 al 10 %).
- Para $N \geq 160$ --------- Se asume q = 0,05 hasta 0,20 (del 5 al 20 %).

p: es la proporción de individuos que no poseen esa característica, es decir, es 1-q.
n: es el tamaño de la muestra (número de encuestas que vamos a hacer).

Para calcular la muestra para la validación de la aplicación hemos tomado como datos los siguientes:
N: el número total de docentes existentes en el área es 86.
k: 99% para caculos es igual a 2.58.

El nivel de confianza indica la probabilidad de que los resultados de nuestra investigación sean ciertos. Un nivel de confianza de 99% es lo mismo que decir que nos podemos equivocar un 1%.

e: 10%.

Significa que si de la encuesta realizada con el error muestral de 10% y el 80% de los docentes se encuentran satisfechos significa que entre el 70% y 90% lo estarán también.

q: Basándonos en la tabla anterior podemos tomar los valores correspondientes de acuerdo a la población con lo que sería el valor de 0.05 hasta 0.10. Para lo cual hemos creído conveniente el valor de 0.05, ya que esto significa que del total de la población un 95 % ha utilizado por lo menos un software para creación de diapositivas.

p: Es igual al valor de 1-q, dando como resultado 0.95

Para calcular el tamaño de la muestra aplicamos la siguiente formula:

$$n = \frac{k^2 * p * q * N}{\left(\left(e^2 * (N-1)\right) + k^2 * p * q\right)}$$

$$n = \frac{2.58^2 * 0.95 * 0.05 * 86}{\left(\left(0.1^2 * (86-1)\right) + 2.58^2 * 0.95 * 0.05\right)}$$

$$n = \frac{27.191394}{1.166179}$$

$$n = 23.3166555$$

Luego de realizar los cálculos correspondientes determinamos la muestra real la cual es de 23 docentes los cuales se puede tomar de forma aleatoria de entre los existentes del Área de Energía las Industrias y los Recursos Naturales no Renovables.

8.9.3.2. Técnicas de validación

La técnica a utilizar para el proceso de validación consiste en una Ficha de Validación que recoge un conjunto de preguntas a aplicársele al usuario basadas en la técnica del cuestionario. Se utilizará una ficha de validación, que será para los docentes del Área de Energía las Industrias y los Recursos Naturales no Renovables. (Ver ficha de validación en el Anexo 1).

8.9.4. Plan de validación

Fecha: 20 de julio de 2010

Versión: 1.0

Indicador	Usabilidad, navegación e ingreso en el sistema de presentaciones de diapositivas en 3d.
Alcance	Se probara los controles de ingreso a datos, confiabilidad, requerimientos de usuario, integración de datos entre el software de soporte y el archivo de configuración de la presentación generada.
Ítems a probar	Componente de navegación, animación y sonido.
Estrategia	Análisis de entrada y salida.
Criterios de suspensión y reanudación	Se suspenderá el proceso de pruebas en caso de que no exista las condiciones necesarias tales como: el computador para realizar la instalación del sistema, disponibilidad del tiempo de usuario, detección de errores que no permitirán culminar las pruebas de la aplicación. Se reanudara el proceso de pruebas cuando los criterios de suspensión sean superados. Se culminara las pruebas de validación una vez que se ha verificado el cumplimiento de los requerimientos impuestos por el usuario y los errores encontrados no requieran de una nueva revisión por parte del usuario.
Documentación	Se realizara un informe de pruebas y las correcciones realizadas, adjunto como respaldo las fichas entregadas al usuario.
Recursos	Computador de características aceptables Sistema de presentaciones de diapositivas tridimensionales. Ficha elaborada para la validación de la herramienta FV01 Usuario Datos reales del software de actualizaciones
Calendario	Del 12 de julio al 30 de julio
Responsable	Grupo de trabajo: Klever Macas y Alfredo Macas

Tabla 61 Plan de Validación

8.9.5. Ejecución del plan de pruebas

8.9.5.1. Pruebas de funcionalidad y aceptación

Siguiendo la metodología adoptada para el desarrollo de software y tomando en cuenta las diferentes etapas que posee, nuestro grupo de trabajo realizo pruebas a lo largo de todo el proceso de implementación realizando cambios en base a los nuevos errores encontrados lo cual se cumple con una característica de iconix que es de ser iterativo e incremental, estas pruebas comprenden el proceso de codificación, la creación de los diferentes componentes y la integración total de los mismos.

Culminada la fase de implementación y ya teniendo una versión estable de la aplicación se presentó al Director de nuestra tesis Ing. Luis Antonio Chamba Eras, quien aporto con nuevas ideas que ampliaría la funcionalidad de la aplicación las mismas que fueron agregadas.

Con las nuevas correcciones realizadas la aplicación estaba lista para someterse a pruebas y validar su funcionalidad y usabilidad para lo cual tomamos una muestra de la población de los docentes del Área de Energía las Industrias y los Recursos Naturales No Renovables (AEIRNR) lo cual dio como resultado un total de 26 docentes los cuales fueron seleccionados tomando en cuenta la disponibilidad de tiempo que poseían. La aplicación fue presentada en julio del 2010 a los diferentes docentes del AEIRNR encargados de validar la aplicación los mismos que brindaron sugerencias y opiniones que ayudarían a mejorarla las mismas que se realizaron en su totalidad y de la mejor manera posible.

8.9.5.2. Pruebas de usabilidad(funcionalidad, diseño y presentación)

La aplicación se proba durante toda la fase de implementación por parte de los desarrolladores, validando su funcionalidad, diseño y la presentación los cuales deben adaptarse en lo mejor posible a las necesidades de los usuarios, ya que es una aplicación multiplataforma se desarrolló en el sistema operativo Windows y Linux.

Con la finalidad de realizar las pruebas definitivas a los usuarios finales y tomando en cuenta que el tamaño de la muestra era considerable y el tiempo que demandaba encuestar sería de 15 a 25 minutos, se descartó la idea de utilizar un centro de cómputo puesto que resultaría complicado reunir a los docentes ya que el tiempo y los horarios

de clases que poseían hacían que esto se torne difícil, es así que se optó por la encuesta individual procediendo a la instalación en un pc portátil de manera que se realizaría en un momento en que se encuentren totalmente disponibles y puedan realizar la prueba de la aplicación de la manera más tranquila lo cual permitió que aporten ideas y sugerencias para mejorarla.

Culminadas las encuestas se procedió a realizar los análisis de los resultados y de esta forma encontrar los errores que se hayan pasado por alto en el proceso de implementación.

8.9.5.2.1. Análisis de resultados de las pruebas de validación

Luego del análisis realizado en base a la información recolectada en la etapa de pruebas los resultados obtenidos fueron los siguientes:

Ficha FV01 Ficha de validación en el "Construcción De Una Herramienta Multimedia Para El Desarrollo De Diapositivas Tridimensionales Bajo Licencia Gpl, Haciendo Uso De La Tecnología Java 3d, Orientada Al Sector Académico Y Profesional Del Área De La Energía Las Industrias Y Los Recursos Naturales No Renovables".

Luego de terminar las encuestas realizadas a la muestra de docentes de la población del Área de Energía, las Industrias y los Recursos Naturales No Renovables obtuvimos los siguientes datos:

1 Accesibilidad

1.1 ¿Se presentó algún problema al ingresar a la aplicación?

SI ( ) **NO ( )**

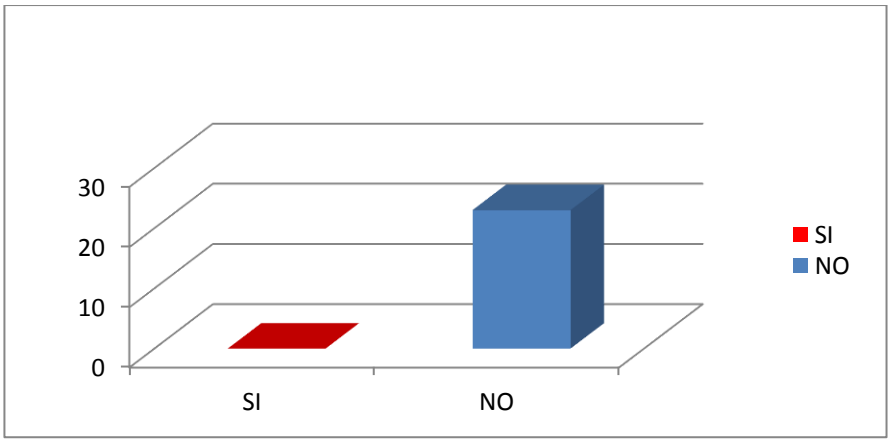


Figura 156 Resultados del Ingreso al Sistema

El 100% de los encuestados considera que no existe problema al ingresar a la aplicación. Esto se debe a que el ingreso a la misma resulta familiar a los programas utilizados por la mayor parte de usuarios.

1.2 ¿El tiempo de respuesta del sistema al ejecutar una acción oscila entre?

- **a) 0-2 segundos ( )**
- **b) 3-5 segundos ( )**
- **c) 6-10 segundos ( )**
- **d) 11 o más segundos ( )**

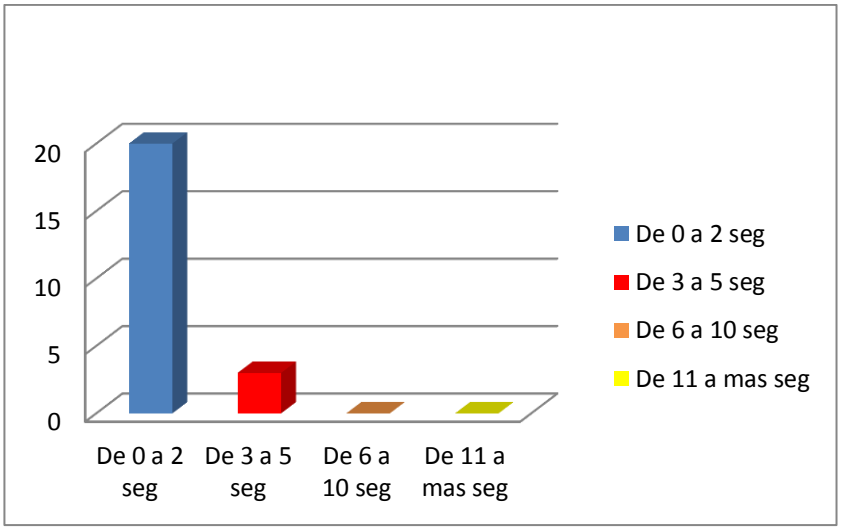


Figura 157 Resultados del tiempo de respuesta

El 87% de la población encuestada considera que al ejecutar una acción en el sistema el tiempo de respuesta oscila entre 0 y 2 segundos. En cambio el 13% restante considera que el tiempo de respuesta oscila entre 3 a 5 segundos. En base a los resultados obtenidos podemos afirmar que el tiempo de respuesta se encuentra dentro de los parámetros aceptables por parte de los desarrolladores debido a que esto depende de las características de hardware del computador (RAM, tarjeta de video y procesador) en el cual se ejecute.

2. Funcionalidad.

2.1 ¿Tuvo una falla al utilizar las diferentes funciones del sistema?

Administrar presentación	**SI ( ) NO ( )**
Administrar Diapositiva	**SI ( ) NO ( )**
Administrar componentes gráficos	**SI ( ) NO ( )**
Administrar galería de objetos	**SI ( ) NO ( )**
Administrar sonido	**SI ( ) NO ( )**
Administrar material	**SI ( ) NO ( )**
Administrar efectos visuales	**SI ( ) NO ( )**
Administrar texturas	**SI ( ) NO ( )**

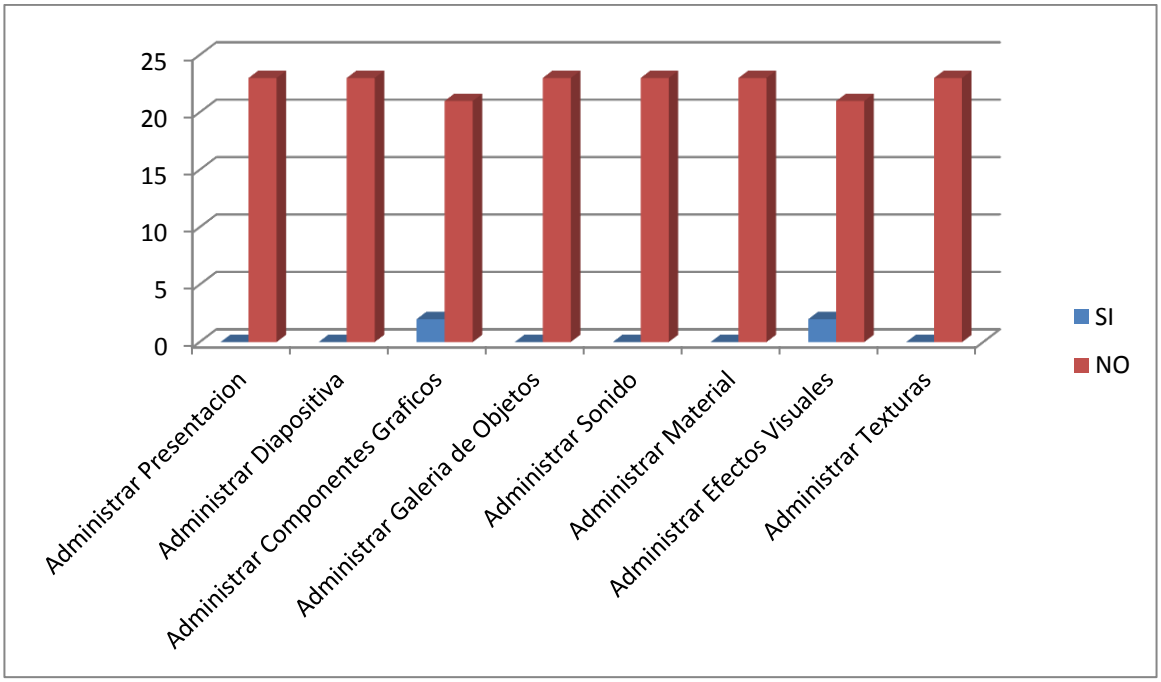


Figura 158 Resultados de la funcionalidad del sistema

Lo que se refiere a la funcionalidad del sistema 2 de los 23 encuestados tuvieron problemas con la administración de componentes gráficos, así mismo 2 de los 23 encuestados al momento de administrar los efectos visuales se les presentaron problemas. Esto se debe a que se produjeron pequeños fallos con el manejo de múltiples threads que fueron solucionados en su totalidad gracias a los resultados obtenidos.

2.2 ¿La información ingresada en la presentación se guardó de forma correcta?

Administrar presentación	**SI ( ) NO ( )**
Administrar Diapositiva	**SI ( ) NO ( )**
Administrar componentes gráficos	**SI ( ) NO ( )**
Administrar galería de objetos	**SI ( ) NO ( )**
Administrar sonido	**SI ( ) NO ( )**
Administrar material	**SI ( ) NO ( )**
Administrar efectos visuales	**SI ( ) NO ( )**
Administrar texturas	**SI ( ) NO ( )**

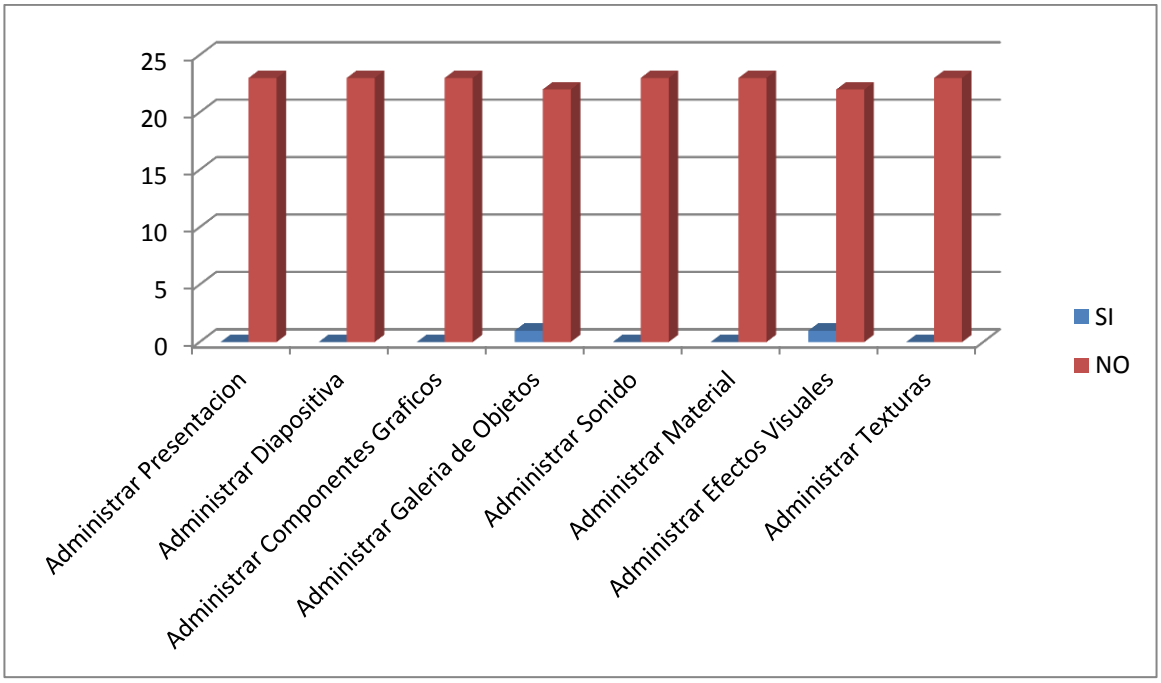


Figura 159 Resultados relacionados con el almacenamiento de la información

El 99% de los encuestados aseguro que la información suministrada en cada componente se almaceno de forma correcta, el 1 % en cambio aseguro que la información suministrada no se almaceno de forma correcta.

En base a los resultados obtenidos podemos asegurar que los problemas encontrados se debieron a incompatibilidad con los formatos de imagen y el manejo de múltiples hilos en el componente de animaciones.

2.3 Según su criterio: ¿El sistema cubre todos los controles, al momento de agregar imágenes, objetos y audio?

SI ( ) NO ( )

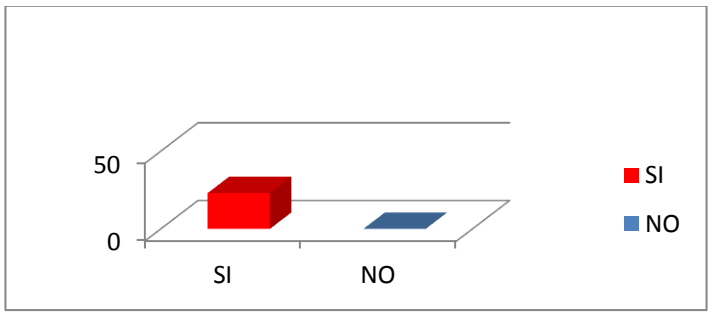


Figura 160 Resultados del control de información ingresada

El cien por ciento de los encuestados considera que el sistema cumple con todos los controles adecuados al momento agregar imágenes, objetos y audio.

3. Diseño y Presentación.

3.1 ¿Considera amigable la interfaz gráfica de usuario?

SI ( ) NO ( )

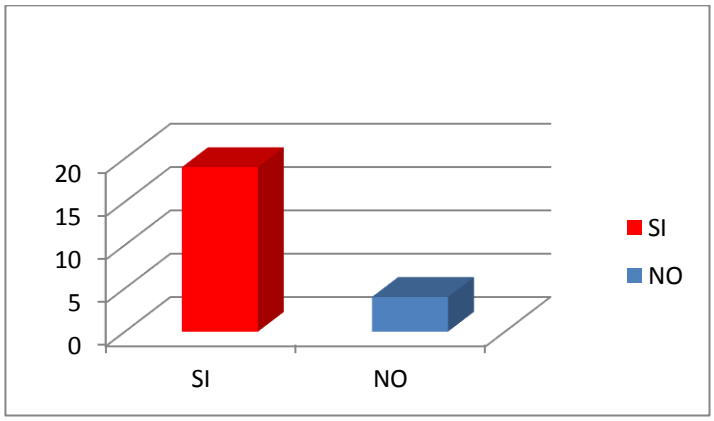


Figura 161 Resultados relacionados con la interfaz grafica

El 83% de la población encuestada considera que la interfaz gráfica del sistema es amigable, el 17% restante considera que se debe realizar algunos cambios para hacerla mucho más amigable. En base a los resultados obtenidos concluimos a que esto se debe a que algunos de los usuarios no están familiarizados con el manejo de aplicaciones relacionadas con la tecnología 3d.

3.2 ¿Considera usted que el diseño de la aplicación es apropiada para facilitar la ejecución de sus actividades?

SI ( ) NO ( )

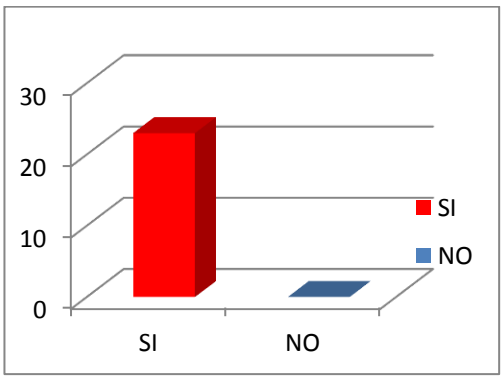


Figura 162 Resultados relacionados con la usabilidad del sistema

El cien por ciento de la población encuestada considera que el diseño del sistema es apropiado para facilitar el manejo de sus actividades.

3.2 ¿Cómo considera usted la comprensión y utilización del sistema?

a) Fácil ( )
b) Aceptable ( )
c) Complicado ( )

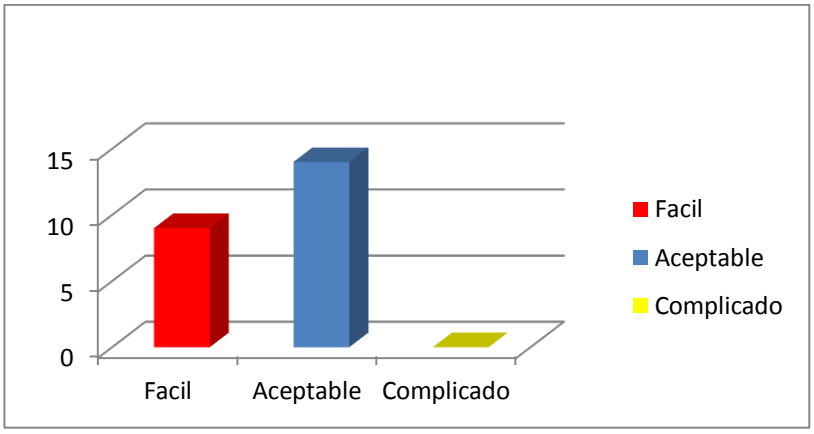


Figura 163 Resultados relacionados con la comprensión del sistema

El 39% de la población considera la comprensión y utilización del sistema fácil, el 61% restante aseguro que la comprensión y utilización del sistema es aceptable.

En base a los resultados obtenidos podemos decir que al porcentaje que le resulta fácil de manejar la aplicación es debido a que ellos utilizan aplicaciones de modelado 3d cuyas interfaces son complejas, en cambio el porcentaje que considera aceptable el manejo de la aplicación no hace mucho uso de aplicaciones de este tipo.

8.9.5.3. Informe de resultados

Informe de: "Docentes del Área de Recursos Naturales no Renovables"

Identificador	Ficha FV01 usabilidad, navegación y manejo de los diferentes componentes de la aplicación para crear presentaciones tridimensionales.
Resumen	El plan de validación fue ejecutado a la muestra tomada a partir de la población general de todos los docentes del AERNNR
Variaciones	Se explicó previamente el funcionamiento del sistema cuales eran sus funciones más importantes para posterior a eso el encuestado realice un análisis de la aplicación y así pueda dar su opinión acerca de la misma
Resumen de resultados	Los resultados estadísticos de las pruebas realizadas sobre la administración de la información se encuentran en la sección Análisis de resultados de las pruebas de validación.
Resumen de actividades	Se comenzó por explicar los componentes que la aplicación tenia, para luego comenzar a probar cada uno de ellos comenzando primero por la opción de crear una presentación nueva, a la cual se le agrego diferentes diapositivas. A las diapositivas el encuestado le agrego los objetos deseados y le configuro los efectos de acuerdo a su elección.
Aprobación	La aplicación por parte de los docentes fue aprobado en su totalidad.

Tabla 62 Informe de Resultados

Una vez finalizadas las pruebas al personal docente, se obtuvo varias sugerencias tales como mejor la interfaz de usuario ya que necesita ser lo más amigable posible, así mismo colocar una ayuda que servirá de mucho a usuarios nuevos y de este modo facilitar su uso. Las opiniones brindadas por los encuestados sirvieron de mucha ayuda ya que así se conseguirá un producto de calidad, puesto que ellos son los usuarios finales de aplicación.

8.9.6. Pruebas de usuario

COMPONENTES	ERRORES	SOLUCION	FECHA
Animaciones	Al agregar varios efectos en un mismo tiempo de ejecución se distorsionan los movimientos.	Sincronizar los hilos y funciones de los efectos.	21-07-2010
	Al cargar tiempos de ejecución mayor a cero se cambia los milisegundos por segundos.	Crear dos funciones diferentes una para cargar en milisegundos y otra en segundos.	22-07-2010
	Al detener vista previa de las animaciones el hilo que genera la vista previa pierde el control sobre los hilos que generan los efectos.	Crear un grupo de hilos y registrarlos a todos los hilos que se derivan de generar vista previa de efectos.	22-07-2010
	Al configurar un efecto no se actualiza la etiqueta de información del efecto.	Llamar al método setInformacion() en el método fijarConfiguracionPorDefecto()	25-07-2010
Núcleo	Agregar material a las formas básicas cono, cubo, cilindro, línea, punto y esfera.	Agregar en el constructor la propiedad GENERATE_NORMALS.	21-07-2010
	Al ordenar carrusel de forma horizontal los objetos no se alinean de forma que al girar el carrusel se los observa de posiciones diferentes.	Al componente grafico fijar el ángulo de rotación Y en: **Angulo:** Angulo de separación entre componentes. **90º:** Angulo complementario. setAnguloY (90-angulo):	25-05-2010
	Al ordenar carrusel de forma vertical los objetos no se alinean de forma que al girar el carrusel se los observa de posiciones diferentes.	Al componente gráfico fijar el ángulo de rotación X en: **Angulo:** Angulo de separación entre componentes. **90º:** Angulo complementario. setAnguloX (angulo-90):	25-05-2010
GIF	Al generar el archivo.gif queda abierto el stream de escritura del archivo.	En la clase “AnimatedGifEncoder “ en el método “finish()” cerrar la salida con instrucción “out.close();”	18-06-2010

SCORM	Al cargar el SCORM no se reconoce los recursos del proyecto.	Eliminar los espacios en el código de generación manifest.xml. Por ejemplo si tenemos: < resource identifier="RES-22" type="Imagen" href="Texturas/gold.jpg"> < file href="Texturas/gold.jpg" /> </resource> Entonces la solución sería: <resource identifier="RES-22" type="Imagen" href="Texturas/gold.jpg"> <file href="Texturas/gold.jpg" /> </resource>	01-07-2010
	Al cargar SCORM NO se reconoce el formato.	Fijar la codificación de caracteres de la cadena que contiene el manifiesto a "UTF-8" de la siguiente forma: String manifest = new String(cadenaManifest.getBytes("UTF-8"));	02-07-2010
Presentación	Al ejecutar la presentación hecha en Linux en Windows las fuentes se cambian por que los dos sistemas no tienen las mismas fuentes de texto.	Guardar las fuentes del sistema que se ocupan en la presentación y al cargar la presentación registrarlas en el sistema.	30-07-2010
	Al cargar la presentación en el Applet se producen errores de direccionamiento.	Crear un cargador web para la presentación.	08-07-2010
	En algunas Laptops nuevas no funciona el FullScreen que se ordena desde java.	Deshabilitar FullScreen y en vez de ello quitar la decoración de la ventana con "setUndecorated(true); " y ponerla en el siguiente estado con "setExtendedState(JFrame.MAXIMIZED_BOTH);".	12-07-2010

	Al ejecutar una presentación en Linux no carga los recursos directamente por lo que hay que escribir un script darle permisos de ejecución y escribir el siguiente comando en el "java –jar FinalPresentacion.jar", guardar y luego ejecutar el script.	Obtener la ruta da la presentación dentro del código de la presentación de la siguiente forma: String ruta= getClass().getResource("").getPath(); Luego separar solo la ruta hasta la carpeta del proyecto.	15-07-2010
	Al cargar una presentación hecha en Linux en Windows se produce un error de direccionamiento por la codificación URL de la dirección principal de la presentación.	Decodificar la dirección del proyecto con URLDecoder. Ejemplo: String ruta=URLDecoder.decode(getClass().getResource("").getPath(), "UTF-8");	15-07-2010
	Al reproducir sonido en Windows no se reconoce la dirección del archivo de audio.	Remover el primer carácter de direccionamiento de la dirección principal ya que Windows no lo utiliza. Por ejemplo si tenemos: **\C:\Documents and Settings\ALF-MKS** Por lo tanto quedaría: **C:\Documents and Settings\ALF-MKS**	16-07-2010
	Al corregir el problema del direccionamiento de audio en Windows se produce errores de direccionamiento de archivos en Linux.	Preguntar si es un sistema Windows remover el primer carácter, caso contrario no remover nada.	17-07-2010

Tabla 63 Pruebas de Usuario

8.10. Instalación final y explotación del sistema

La presente aplicación está orientada al Área de la Energía las Industrias y los Recursos Naturales No Renovables y siendo de código abierto estará disponible en su totalidad en la biblioteca del Área ya sea para docentes o estudiantes que deseen hacer uso de ella.

El personal docente se beneficiara en gran parte con la presente aplicación ya que le permitirá hacer sus exposiciones de manera más interactiva, manipulando los objetos de manera que el estudiante capte mejor los detalles y tenga una mejor percepción acerca del objeto de estudio, además le permitirá exportar las presentaciones en formato scorm el mismo que es utilizado en los entornos de aprendizaje virtual desarrollados con moodle.

Por parte de los estudiantes la presente herramienta le permitirá realizar sus presentaciones de manera más llamativa e interactiva ya que una de las características de la presentación con diapositivas es captar la atención de las personas que se asisten a la presentación, así mismo la utilización de la aplicación les permitirá adquirir habilidades para el manejo de software relacionado con la tecnología 3D.

Adema la aplicación se encuentra bajo la licencia GPL, la cual permite hacer uso total del código fuente, esto con la finalidad de incentivar a nuevos desarrolladores a mejor la aplicación, incrementar funcionalidades o en base a la misma desarrollar nuevas aplicaciones que brinden soluciones a diferentes problemas que se presenten en la colectividad.

9. VALORACION TECNICA-ECONOMICA –AMBIENTAL

El sistema se desarrolló de la mejor manera posible ya que se contó con todos los recursos humanos, económicos y tecnológicos como hardware y software lo cual hizo posible culminar de forma exitosa el presente proyecto.

Las herramientas empleadas para la implantación del proyecto son de libre distribución lo cual permitió la fácil adquisición de las mismas ya que en su mayoría se encuentran disponibles en sus sitios web. El lenguaje programación utilizado fue java cuya versión es 1.6, el IDE de desarrollo Netbeans 6.8, librería jhall la cual facilito la realización de la ayuda de la aplicación, librería Loader3DS1_2 y cv97r140 las cuales permiten cargar objetos 3D, librería liquidlnf-0.2.6 que permite cambiar la apariencia de la interfaz gráfica

Recursos Humanos

Recursos Humanos	Cantidad	Horas c/u	Costo por Hora	Costo Total
Director de Tesis	----	-----	-----	----
Desarrolladores	2	500	$3.00	$3000.00
TOTAL				$3000.00

Tabla 64 Recursos Humanos

Recursos Materiales

Recursos Materiales	Cantidad	Costo Unitario	Costo Total
Resma de Papel	4	$4	$16.00
Cartuchos de tinta.	5	$10	$50.00
Internet/horas	500	$1	$500.00
TOTAL			$566.00

Tabla 65 Recursos Materiales

Recursos Técnicos

Recursos Técnicos	Cantidad	Horas	Costo por hora	Costo Total
Computadores	2	500	0.30	$300.00
Impresora canon ip1000	1	16	0.50	$8.00
TOTAL				$308.00

Tabla 66 Recursos Técnicos

Recursos Tecnológicos

Recursos Tecnológicos	Cantidad	Costo Unitario	Costo Total
NetBeans IDE 6.8	1	Gratuito	$0.00
Java JDK 1.6	1	Gratuito	$0.00
Java 3D 1.5	1	Gratuito	$0.00
Enterprise Architec 7.0	2	Trial	$0.00
TOTAL			$0.00

Tabla 67 Recursos Tecnológicos

Costo total del Proyecto

Costo total del Proyecto	Costo Total
Recursos Humanos	$3000.00
Recursos Materiales	$566.00
Recursos Técnicos	$308.00
Recursos Tecnológicos	$00.00
TOTAL	$3874.00

Tabla 68 Costo Total del Proyecto

10. CONCLUSIONES

Una vez terminado nuestro proyecto de tesis se ha concluido lo siguiente:

- El sistema de presentaciones tridimensionales P3D es de gran utilidad cuando se necesita manipular objetos 3d ya que es un programa ligero y permite cargar modelos tridimensionales en varios formatos como 3DS, COB, DEM, DXF, IOB, LWS, NFF, OBJ, PDB, PLAY, SLD, VRT, VTK, WRL.
- La utilización de la tecnología java 3D nos permite desarrollar aplicaciones en corto tiempo ya que es un lenguaje de programación de alto nivel y posee una gran cantidad de documentación y ejemplos que ayudan al desarrollador.
- Se creó el algoritmo para agrupar y distribuir múltiples objetos en forma circular aplicando la ecuación de la circunferencia.
- Para desarrollar el componente de efectos se utilizó movimiento rectilíneo uniforme ya que nos permite recorrer espacios iguales en tiempos iguales, además se definió una unidad de tiempo de 20ms para optimizar el rendimiento del procesador ya que por defecto java controla el paso del tiempo en milisegundos.
- Para la reproducción del sonido se integró el componente mplayer ya que es multiplataforma, soporta una gran cantidad de formatos de audio y video y su integración con aplicaciones java es bastante sencilla comparado con otros componentes o frameworks.
- Para la fácil navegación de las presentaciones se desarrollaron nuevos tipos de nodos con un campo de texto adicional que permite reconocer los diferentes componentes (diapositivas, audio, formas, objetos3D y texto3D) que están incluidos en la presentación.
- Como conclusión final podemos decir que para la liberación del software se hospedo el código fuente así como también la ayuda y los manuales en el sitio web www.sourceforge.net, quedando de esta manera accesible para cualquier persona interesada en la aplicación.

11. RECOMENDACIONES

Una vez concluido el proyecto de tesis y en base a la experiencia adquirida durante el proceso de desarrollo se plantearon las siguientes recomendaciones:

- Emplear el software P3D para crear presentaciones que demanden de poco cantidad texto, puesto que en la aplicación no existe el salto de línea lo cual hace que su desarrollo tome un poco más de tiempo.
- No utilizar la tecnología Java 3D en proyectos que requieran una gran velocidad de renderización tales como videojuegos, ya que el motor de renderizado que posee no sería capaz de cumplir de manera óptima este tipo de tareas, para este fin existen otras opciones como es el caso de OpenGL, JOGL y Ogre3D.
- Poseer conocimientos de geometría (básica y analítica), así como también revisar los temas de física relacionados con movimiento para lograr un mejor rendimiento en los algoritmos de animación y efectos 3D.
- Tener la aplicación mplayer preinstalada en los sistemas Linux dependiendo de su distribución, ya sea través del código fuente o en los formatos soportados por el sistema empleado tales como .deb, .rpm o .bin.
- Para continuar con el desarrollo de la aplicación conformar equipos de trabajo de tal manera que cada grupo tenga funciones específicas, lo que agilitara su proceso.
- El incremento de nuevos componentes y funciones en la aplicación tales como: integrar video, manejo de gráficos estadísticos, revisión ortográfica y tablas pueden ser abordados como nuevos trabajos investigativos por parte de los estudiantes de la carrera de Ingeniería en Sistemas de la Universidad Nacional de Loja.
- Para liberar cualquier tipo de software se debe tener claro bajo qué tipo de licencia se lo quiere realizar y conocer cuáles son los pasos a seguir para realizarlo, para lo cual existen sitios web donde se brinda este servicio de manera gratuita, siendo uno de ellos www.sourceforge.net.

12. BIBLIOGRAFIA

Archivos Digitales:

- Aitken. G., "Moving from C++ to Java", Dr. Dobb's Journal, March 1996, pp. 52-56
- Anuff. E... Java Source Book, New York, NY: John Wiley & Sons, Inc.. 1996
- Arnold, K., and J. Gosling. The Java Programing Lenguaje, Reading. MA: Addison Weley Publishing Company, 1996.
- Boone. B... "Multitasking in Java" Java Report. May/June 1996, pp 27-33
- Java in a nutshell: a desktop quick reference D. Flanagan. Ed. O'Reilly
- The Java tutorial: object-oriented programming for the Internet M. Campione. Ed. Addison-Wesly Programación del lenguaje
- Core packages. J. Gosling. Ed. Addison-Wesley Manual de reference
- The Java language specification J. Gosling. Ed. Addison-Wesley Lenguaje y manual de referencia
- Introduction to Graphics Programming with Java 3D Doug Twilleager

Internet:

- Luis Antonio Fernández Aldana. 2007. Introducción a Java3D. [En línea] , Que es Java3D, Caracteristicas,[http://www.monografias.com/trabajos43/java-tres-d/java-tres-d3.shtml], [Consulta:15 de Noviembre del 2009]
- SUN, 2010. [En línea] Api Java3D,[http://java.sun.com/developer/onlineTraining/java3d/],[Consulta:12 Febrero 2010].
- Wikipedia. 2010. Blender [en línea]. Características, [http://es.wikipedia.org/w/index.php?title=Blender],[Consulta:20 de Septiembre 2010]
- Wikipedia.2009. Herramientas de diseño asistido [En línea] Herramientas de diseño asistido[http://es.wikipedia.org/wiki/Herramientas_de_dise%C3%B1o_asistido#Historia],[Consulta:15 de Marzo del 2010]
- María José Ballesteros, Gloria María Gallego. La Diapositiva [En línea], Ventajas y Desventajas, [http://www.uclm.es/profesorado/ricardo/Diapositivas/Diapositivas_Fca_2004.doc], [Consulta: 20 de enero del 2010]
- RENa, 2010.Metodologia [En línea], Ciencia, Método e Investigación, [http://www.rena.edu.ve/cuartaEtapa/metodologia/tema19.html],[Consulta:12 de enero del 2010]
- Robert Aldo Velásquez Huerta, 2010. Control y Evaluación de Proyectos Informáticos, [En línea], Unidad 1: Metodología de los Proyectos, [http://www.scribd.com/doc/2521982/Metodologia-de-Proyectos#], [Consulta: 12 de enero del 2010]

- Wikipedia, 2009. Programación Orientada a Objetos, [en línea], Programación Orientada a Objetos, [http://es.wikipedia.org/wiki/Programaci%C3%B3n_orientada_a_objetos], [Consulta: 20 de Noviembre del 2009]

- Wikipedia, 2010. Movimiento Rectilíneo,[En línea], Ecuaciones del Movimiento, [http://es.wikipedia.org/wiki/Movimiento_rectil%C3%ADneo], [Consulta: 15 de mayo del 2010]
- WebTaller, 2010. Firma digital de un applet en java, [En línea], Como Configurar la Seguridad de Applet, [http://www.webtaller.com/construccion/lenguajes/java/lecciones/firma-digital-appletjava.
- Php], [Consulta: 20 de Junio del 2010]
- Leonardo Cruz Verduit, 2002. Aspectos metodológicos básicos para la preparación y el empleo de las diapositivas. [En línea], Clasificación de las diapositivas, Editorial CENIC, La Habana, Cuba, [http://bvs.sld.cu/revistas/aci/vol10_4_02/aci030402.htm], [Consulta: 20 de Noviembre del 2009]
- Juan Antonio Medina Romani, 2004. Uso de equipos y sistemas multimedia en el proceso de aprendizaje enseñanza. [En línea], Uso de equipos y sistemas multimedia en el proceso de aprendizaje enseñanza, [http://www.monografias.com/trabajos20/multimedia-en-aprendizaje/multimedia-en-aprendizaje.shtml], [Consulta**:** 15 de Septiembre del 2010]
- Leonardo Muro García , 2007. Licencias de Software. [en línea], Licencias del Software, [http://www.monografias.com/trabajos55/licencias-de-software/licencias-de-software.shtml], [Consulta: 19 de noviembre de 2010]
- Wikipedia, 2010. Arquitectura de Software, [en línea], Arquitectura, Modelos o Vistas, [http://es.wikipedia.org/wiki/Arquitectura_de_software], [consulta: 19 de noviembre de 2010]
- Wikipedia, 2010. Software libre y de código abierto. [en line], Software libre y de código abierto, [http://es.wikipedia.org/wiki/Software_libre_y_de_c%C3%B3digo_abierto], [consulta:19 de noviembre de 2010]
- Laura Marthé Hinojosa Castillo, 2006. Ingeniería de Software. [en línea], Lenguaje Unificado de Modelado. [http://www.monografias.com/trabajos34/ingenieria-software/ingenieria-software.shtml#quees], [consulta: 19 de noviembre de 2010].

13. ANEXOS

ANEXO 1: FICHA DE VALIDACIÓN

UNIVERSIDAD NACIONAL DE LOJA

PROYECTO "CONSTRUCCIÓN DE UNA HERRAMIENTA MULTIMEDIA PARA EL DESARROLLO DE DIAPOSITIVAS TRIDIMENSIONALES BAJO LICENCIA GPL, HACIENDO USO DE LA TECNOLOGÍA JAVA 3D, ORIENTADA AL SECTOR ACADÉMICO Y PROFESIONAL DEL ÁREA DE LA ENERGÍA LAS INDUSTRIAS Y LOS RECURSOS NATURALES NO RENOVABLES".

1ro	FV01
Usuario:	Docente
Rol:	
Nombre:	

Instrucciones:
Señale con una x la respuesta a la pregunta formulada y justifique su respuesta.

1. ACCESIBILIDAD

1.1 ¿Se presentó algún problema al ingresar a la aplicación?
SI ( ) NO ( )

¿Por qué?.....

1.2 ¿El tiempo de respuesta del sistema al ejecutar una acción oscila entre?

a) 0-2 segundos ( )
b) 3-5 segundos ( )
c) 6-10 segundos ( )
d) 11 o más segundos ( )

2. FUNCIONALIDAD

2.1 ¿Tuvo una falla al utilizar las diferentes funciones de la aplicación?

Administrar presentación	SI ( ) NO ( )
Administrar Diapositiva	SI ( ) NO ( )
Administrar componentes gráficos	SI ( ) NO ( )
Administrar galería de objetos	SI ( ) NO ( )
Administrar sonido	SI ( ) NO ( )
Administrar material	SI ( ) NO ( )
Administrar efectos visuales	SI ( ) NO ( )
Administrar texturas	SI ( ) NO ( )

2.2 ¿La información ingresada en la presentación se guardó de forma correcta?

Administrar presentación	SI ( ) NO ( )
Administrar Diapositiva	SI ( ) NO ( )
Administrar componentes gráficos	SI ( ) NO ( )
Administrar galería de objetos	SI ( ) NO ( )
Administrar sonido	SI ( ) NO ( )
Administrar material	SI ( ) NO ( )
Administrar efectos visuales	SI ( ) NO ( )
Administrar texturas	SI ( ) NO ( )

2.3 Según su criterio: ¿El sistema cubre todos los controles, al momento de agregar imágenes, objetos y audio?
SI ( ) NO ( )

¿Dónde faltaría ?.....

3 DISEÑO Y PRESENTACION

3.1 ¿Considera amigable la interfaz gráfica de usuario?
SI ( ) NO ( )

¿Por qué ?.....

3.2 ¿Considera usted que el diseño de la aplicación es apropiada para facilitar la ejecución de sus actividades?
SI ( ) NO ( )

¿Por qué ?.....

3.3 ¿Cómo considera usted la comprensión y utilización del sistema?

a) Fácil ( )
b) Aceptable ( )
c) Complicado ( )

F:…….

ANEXO 2: ANTEPROYECTO

UNIVERSIDAD NACIONAL DE LOJA

AREA DE LA ENERGIA LAS INDUSTRIAS Y LOS RECURSOS NATURALES NO RENOVABLES

ANTEPROYECTO DE TESIS

Tema:

Construcción de una herramienta multimedia para el desarrollo de diapositivas tridimensionales bajo licencia GPL, haciendo uso de la tecnología JAVA 3D, orientada al sector académico y profesional del Área de la Energía las Industrias y los Recursos Naturales No Renovables

AUTORES:

- Alfredo Rolando Macas Chocho
- Klever Daniel Macas Flores

LOJA - ECUADOR

2009

1. TITULO

"Construcción de una herramienta multimedia para el desarrollo de diapositivas tridimensionales bajo licencia GPL, haciendo uso de la tecnología JAVA 3D, la cual está orientado para el sector académico y profesional del Área de la Energía las Industrias y los Recursos Naturales No Renovables "

2. PROBLEMÁTICA

2.1.SITUACIÓN PROBLEMÁTICA

Las personas a nivel mundial necesitan cada vez mejorar su forma de presentar proyectos, trabajos, etc., antes de la aparición de los ordenadores esto se realizaba mediante papelografos o carteles, luego con el surgimiento del computador y sistemas operativos de entorno gráfico con programas de aplicación como power point, impress, etc, se crean diapositivas bidimensionales. Hoy en día muchas personas se encuentran limitadas por el software de presentaciones 2D ya que en varias presentaciones se necesita visualizar objetos del mundo real como, piezas mecánicas, partes de computadores, aparatos, animales, etc. Es decir que todo objeto del mundo real se encuentra en una tercera dimensión y tratar de presentarlo en un ambiente bidimensional no sería lo más adecuado, además todo lo que se puede realizar en un ambiente 3D es imposible realizarlo en un ambiente 2D, como son movimientos, giros de 360º, iluminación, sombras y efectos visuales. En el mercado existe software que permite realizar presentaciones 3D como Xara3D, Macromedia Flash, 3D Estudio Max entre otros. La mayor parte del software que se utiliza es privativo lo que implica costos adicionales, limitaciones de uso y no permiten manipulación del código fuente para adaptarlo a sus necesidades.

En el Ecuador esta realidad no es diferente aunque aquí no se controla mucho el uso de licencias de software, pero existe poco software de este tipo la mayor parte de ellos son muy complicados de utilizar para el usuario, es por ello que surge la necesidad tecnologías que permitan realizar los procesos de forma fácil y rápida.

En la ciudad y provincia de Loja, existe una gran cantidad de estudiantes y profesionales que necesitan este tipo de software para mejorar la calidad de presentación de sus trabajos y proyectos.

La falta de un software de presentaciones en 3D y la utilización de programas alternos para mejor las diapositivas realizadas en 2D ocasiona múltiples inconvenientes como son:

- Incremento de gastos por el pago de licencias de software adicional.
- La representación de objetos del mundo real es limitada.

- Muchos de los programas que existen en mercado actual para presentaciones tridimensionales, carece de una galería de objetos prediseñados.
- Las animaciones de texto y objetos en un entorno bidimensional, es limitado con respecto a un entorno tridimensional.
- Las presentaciones no son muy llamativas ya que muchas de las veces no captan el suficiente interés del público.

La carrera de Ingeniería en Sistemas de la Universidad Nacional de Loja a través del sistema académico modular por objetos de transformación busca la vinculación con la colectividad, formar profesionales capaces de brindar soluciones a los problemas vigentes en el medio, haciendo uso de las diferentes tecnologías y herramientas presentes en el mundo actual.

Quizás este sea uno de los motivos por los cuales la inquietud y el deseo por desarrollar un software de presentaciones en 3D bajo la licencia GPL ya que está orientada principalmente a proteger la libre distribución, modificación y uso de software. Su propósito es declarar que el software cubierto por esta licencia es software libre y protegerlo de intentos de apropiación que restrinjan esas libertades a los usuarios, con el trabajo a realizar se deja una pequeña luz al vasto conocimiento que se puede utilizar en alas de conseguir un mejorar nivel de vida en nuestro país.

Tomando en cuenta todo lo antes mencionado hemos creído conveniente realizar la "*Construcción de una herramienta multimedia para el desarrollo de diapositivas tridimensionales, haciendo uso de la tecnología JAVA 3D, la cual está orientado para el sector académico y profesional del Área de la Energía las Industrias y los Recursos Naturales No Renovables*".

2.2.Problema General de investigación

"Las limitaciones que tiene el software 2D de desarrollo de diapositivas causan que los estudiantes y profesionales limiten sus presentaciones de trabajos o hagan uso de programas adicionales privativos para mejorar la calidad de sus presentaciones, lo que a

su vez representa un costo adicional al costo del paquete de Microsoft office, por lo cual se construirá un software para el desarrollo de diapositivas 3D en la ciudad de Loja en el periodo de septiembre 2009 – septiembre 2010".

2.2.1. Problemas específicos de investigación

- Limitación en la presentación de objetos del mundo real.
- Falta de un asistente configurable para el desarrollo de diapositivas.
- Limitación de efectos visuales 2D con respecto a los 3D.
- Limitación en la reproducción de sonido en diferentes formatos.
- Limitación en la visualización de múltiples objetos en una misma diapositiva.
- Interfaces de usuario complejas.
- Falta de dinámica en las diapositivas 2D.

3. JUSTIFICACION

3.1.Justificación

- **Social:**

  El presente proyecto representara un gran aporte a la colectividad ya que en este se pretende revolucionar la forma de realizar las diapositivas haciéndolas más llamativas y expresivas, además el proyecto estará bajo la licencia GPL que permite al usuario hacer uso del código fuente y colaborar con nuevas mejoras que serán en beneficio del software.

- **Económica:**

  El proyecto se desarrollara bajo el lenguaje java que es uno de los mejores y que además es libre al igual que la mayoría de los entornos de desarrollo para este lenguaje, es decir no se tendrá que pagar licencias.

- **Técnica:**

  El proyecto implica hacer uso de hardware avanzado como tarjetas gráficas, gran cantidad de memoria y poderosos procesadores, hace algunos años esto habría resultado casi imposible o muy costoso, pero con el avance de la tecnología en la actualidad el hardware no representa mucho problema ya que se cuenta con procesadores de hasta 4 núcleos, capacidad de memoria de hasta 4 GB y aceleradores gráficos de hasta 1GB, es decir las características de hardware son más que suficientes.

- **Académica:**

  Con el presente proyecto se pretende aportar nuevos conocimientos que les serán muy útiles a otros estudiantes o personas involucradas en el desarrollo de este tipo de proyectos.

  Así mismo servirá para la obtención del título de Ingeniero en Sistemas.

3.2.Viabilidad

Gracias al conocimiento adquirido en los diferentes módulos de la carrera y a la experiencia acumulada en el desarrollo de proyectos anteriores estamos en capacidad de desarrollar el presente proyecto.

No se requiere la inversión de muchos recursos económicos lo cual será asequible a nuestro presupuesto, además el fácil acceso a las herramientas de programación nos es muy conveniente.

En lo que respecta a la parte técnica el software y el hardware disponible en la actualidad resulta más que suficiente para el desarrollo del presente proyecto.

4. OBJETIVOS

4.1.General

Construir una herramienta multimedia para el desarrollo de diapositivas tridimensionales, haciendo uso de la tecnología JAVA 3D, la cual está orientada al sector académico y profesional del Área de la Energía las Industrias y los Recursos Naturales No Renovables.

4.2.Específicos

- Desarrollar una galería básica de objetos tridimensionales del mundo real.
- Desarrollar un asistente de software configurable para la creación de diapositivas tridimensionales.
- Desarrollar un algoritmo que permita visualizar múltiples objetos en una misma diapositiva.
- Desarrollar un componente de software configurable para los efectos tridimensionales.
- Integrar un componente de software para el sonido.
- Desarrollar un árbol de navegación para los objetos tridimensionales agregados a la diapositiva.
- Implantación y liberación del software.

5. MARCO TEÓRICO

CAPITULO 1

LA DIAPOSITIVA

1.1. Introducción

Las diapositivas son cada uno de los elementos que constituyen la presentación y cada una de ellas podría identificarse con una lámina o página. Se pueden crear y modificar de manera individual.

El número de diapositivas varía en función del contenido de la presentación, pero en general, podemos decir que es aconsejable que cada diapositiva contenga una única idea o elemento de información.

Tradicionalmente, las diapositivas eran "las del proyector" que podían obtenerse a través de fotografías o incluso hacerse manualmente. Hoy en día existen también las diapositivas informatizadas, elaboradas en varios programas con estos fines, que incluso ofrecen la posibilidad de movimiento y sonido.

La diapositiva es un instrumento idóneo para la explicación de determinadas materias donde el elemento visual tiene especial protagonismo: arte, presentación de datos estadísticos, imágenes del mundo natural, esquemas educativos y, en definitiva, la diapositiva debe esta donde interese mantener una imagen fija para flexionar sobre ella**.**

La diapositiva le ofrece la posibilidad de adaptarse a su estilo personal de enseñar y al nivel de la clase. Por su bajo coste y por su simplicidad de uso, la diapositiva es una herramienta insustituible en la enseñanza.

1.2. Características de las diapositivas

- Son mucho más luminosas, brillantes, comunicativas y sugestivas que las fotografías.
- Se proyectan en grande y así puede participar toda la clase.
- Las imágenes son de buena calidad y no cansan la vista.

- Se pueden mostrar al ritmo que el profesor necesite.
- Se puede cambiar su colocación, añadir o eliminar, según convenga.
- Se adapta a todas las áreas de enseñanza.
- Es un recurso que favorece la motivación.

1.3. Diapositivas informatizadas

Las diapositivas informatizadas son documentos informáticos que pueden incluir textos, esquemas, sonidos, animaciones, fragmentos de video…etc., y pueden visionarse por la pantalla del ordenador como si se tratara de una proyección de diapositivas.

Si se dispone de un cañón proyector de video o de una pantalla de cristal líquido y un retroproyector, las diapositivas informáticas pueden proyectarse sobre una pantalla externa.

1.4. Clasificación de las diapositivas

De acuerdo con el tipo de apoyo que brinde al expositor, la diapositiva se clasifica como: de reafirmación, ampliación y complementación.

Cuando un texto describe lo más objetivamente posible lo que representa una imagen, se está en presencia de una diapositiva de reafirmación. El texto reafirma lo que muestra la imagen. Si el texto amplía el mensaje que la imagen muestra, se tiene una diapositiva de ampliación. El texto va más allá de la mera descripción de la imagen. Por último, cuando el texto agrega a la imagen información que esta no muestra ni permite deducir, se trata de una diapositiva de complementación. La imagen actúa como un complemento del texto.

Dentro de los tres tipos de diapositivas señaladas, pueden existir variantes, así sucede con la forma negativa de la diapositiva de reafirmación. El texto de la diapositiva afirma que la limpieza dentro de un laboratorio tiene que ser escrupulosa; sin embargo, la imagen representa a un investigador que trabaja en un laboratorio donde predomina el caos y la desorganización. La imagen, aparentemente, contradice lo que afirma el texto. La imagen en todos los casos apoya al texto, complementa o refuerza el mensaje.

1.5. Diapositiva de Texto

La forma más común de escribir un texto para una diapositiva, antes de la introducción de las computadoras, era mediante las máquinas de escribir. Para lograr una diapositiva de texto aceptable a partir de estos medios, era necesario que el texto fuera escrito en letras mayúsculas, que los tipos de la máquina estuvieran bien limpios, que se golpearan las teclas con la misma fuerza, y se empleara una cinta nueva y de buena calidad.

Durante la última década del siglo XX y hasta el presente, la computadora ha sustituido a la máquina de escribir en la elaboración de diapositivas de texto, como resultado de la aparición en el mercado de múltiples paquetes de programas avanzados que facilitan esa operación, como Microsoft Power Point, Corel Presentations, Lotus Smart Suit, Margi Systems Presenter-to-go, Astound Presentation, entre otros. Dichos programas permiten obtener infinitas variantes, debido a sus muchas prestaciones y herramientas entre las que se encuentran la ilimitada cantidad de fuentes tipográficas, de paletas de colores, el amplio rango de presentaciones y plantillas, así como las extensas bibliotecas con otros muchos recursos.

No obstante, es recomendable de igual forma, escribir el texto en altas y que se presente de forma centrada. Los textos justificados a la derecha o a la izquierda pudieran quedar muy próximos a los bordes de la diapositiva. En el caso que el texto se escriba en altas y bajas, debe utilizarse un tamaño de letra igual o superior a los 12 puntos.

1.6. Diapositivas de Gráficos

Las diapositivas de texto son las de uso más frecuente, pero las de gráficos han adquirido notable importancia en los últimos tiempos, debido a la posibilidad que ofrecen de transmitir eficazmente los resultados de complejas investigaciones, sin la necesidad de engorrosas explicaciones. No en vano se ha afirmado que un gráfico transmite más de mil palabras. Un diseño apropiado de las diapositivas se basa en el empleo mesurado de los textos y la utilización profusa de las ilustraciones. Las diapositivas se distinguen por los tipos de gráficos que presentan, entre las principales tenemos:

a) De línea

b) De bola o pastel

c) De barras o columnas.

Los gráficos de línea se utilizan preferentemente para dar una idea inmediata de los objetivos o de las variables cambiantes. Son los más empleados y regularmente establecen una relación entre dos o más variables. Frecuentemente se utilizan para indicar porcentajes ascendentes o descendentes. Se aconseja utilizar más de dos líneas para realizar comparaciones entre variables. La combinación de colores en estos casos resulta de mucha utilidad. Siempre que sea posible, deben usarse patrones (notación o simbología) que ayuden a identificar las líneas. Se debe evitar el empleo de abreviaturas, acrónimos, siglas, etcétera, salvo las de reconocimiento internacional; las tipografías (letras y números) deben tener un tamaño tal que asegure su adecuada legibilidad y debe evitarse la aglomeración de cifras y datos comparativos.

En el caso de los gráficos de bola o pastel se utilizan principalmente para mostrar el peso relativo de los elementos que forman un sistema. Se emplean unidades relativas, especialmente, el tanto por ciento. El pastel debe tener una proporción acorde a la diapositiva (se debe dar un tamaño adecuado a los márgenes), las secciones del pastel no deben ser más de siete, y ninguna debe representar menos del 5 % del total, se debe estar alerta y comprobar que las partes sumen el 100 %, el tamaño de las rebanadas debe responder al porcentaje que representan (las cifras respectivas deben colocarse preferentemente en su interior), los textos necesarios pueden situarse dentro del pastel o fuera de este (preferiblemente fuera cuando las secciones representan valores pequeños) y el color debe usarse para dar énfasis y estética a la diapositiva, de acuerdo con su influencia psicofisiológica sobre el espectador. Los gráficos de columnas o barras son ideales para representar datos comparativos; ellos ofrecen gran flexibilidad en la ubicación de textos y cifras, así como en el empleo del color. También deben cumplir con requisitos fundamentales para su utilización: se deben comparar las barras horizontales y verticales en la representación de los datos (seleccionar las que se usarán en la diapositiva), nunca se debe hacer un gráfico con más de siete barras, ellas deben ser bien proporcionadas (ni muy estrechas, ni muy anchas). En cuanto a los textos se deben situar fuera de las columnas y las cifras se pueden colocar fuera o dentro, según sea más conveniente desde el punto de vista estético. Sobre los colores, deben usarse

tantos como categorías de barras, y si el gráfico es en blanco y negro, las barras deben estar identificadas con toda claridad. El empleo de relleno no es útil en estos casos.

Otro de los aspectos esenciales en la utilización de diapositivas como medio audiovisual, es la retención del espectador durante la presentación. Se sabe que la capacidad máxima de retención de la atención de un espectador adulto, con coeficiente de inteligencia normal y escolaridad promedio de noveno grado, ante una diapositiva de calidad, no es mayor de 30 segundo. La curva de atención comienza a decrecer pasado ese tiempo, que es cuando el espectador empieza a buscar deficiencias en la presentación o en el contenido de la imagen. Si una buena diapositiva puede retener la atención del espectador durante ese lapso de tiempo, una mala diapositiva es posible que no lo logre por más de tres o cuatro segundos; o lo que es peor, que distraiga en lugar de fijar la atención, ello tiene una relación directa con la calidad de la diapositiva.

1.7. Ventajas y Desventajas

- Las diapositivas permiten ser proyectadas en grandes pantallas, todos pueden verlas claramente.
- Los esquemas, demás elementos audiovisuales, como en el caso de diapositivas informatizadas, llaman la atención de los alumnos, son motivantes.
- Son ideales para dar clase a grandes grupos.
- Para la proyección de las diapositivas la sala debe estar sin luz, siendo esto un inconveniente pero, en cambio para las diapositivas informatizadas no hace falta, y se pueden tomar apuntes…
- Se puede utilizar con cualquier tema o nivel educativo.
- Se pueden facilitar copias en papel, para las diapositivas informatizadas.
- El profesor puede mantenerse cara a los alumnos a la vez que explica, esto mejora la comunicación
- Ayuda al profesor o al exponente recordando mediante los esquemas o fotografías.
- La elaboración resulta fácil y el control es sencillo, quizá se puede presentar alguna dificultad en las diapositivas informatizadas.

1.8. Aprendizaje mediante el uso de diapositivas.

En el proceso de enseñanza-aprendizaje es de vital importancia la utilización de herramienta, técnicas y métodos que permitan que se realice de la mejor manera para de esta forma lograr una comunicación bidireccional mucho más efectiva entre los protagonistas. El uso de diapositivas para la enseñanza se encuentra entre las herramientas multimedia ya que utiliza diferentes tipos de datos dentro de una misma presentación como puede ser texto, audio, animaciones, videos, imágenes tanto fijas como en movimiento.

El empleo de diapositivas en la enseñanza favorece en gran escala al docente ya que mezcla diferentes tecnologías de difusión de la información, impactando varios sentidos a la vez para lograr un efecto mayor en la comprensión del mensaje. Hoy en día la utilización de este tipo de recurso es muy común y a esto le favorece la aparición del computador y software diseñado con este fin lo cual facilita aún más su uso.

El computador es considerado como una de las mejores herramientas multimedia ya que este posee los componentes esenciales para la elaboración de presentaciones, el óptimo uso de la misma depende de la creatividad y destreza que posea el docente.

Entre los usos que se da a este tipo de herramienta multimedia esta la creación de presentaciones de diapositivas mediante software dedicado a este propósito, para el desarrollo de estas presentaciones es necesario conocer el uso de estos programa que tiene muchas virtudes, entre las cuales se encuentran: edición de textos, colocación de imágenes fijas y en movimiento, edición de dibujos libres, colocación de música y todo tipo de sonido así como también colocación de videos tipo películas de cine, colocación de fondos de pantalla especiales, colocación de efectos especiales de sonido y movimiento, etc., las que pueden ser colocadas en cada diapositiva, a conveniencia de la persona que las edita.

Para presentar los archivos diseñados, para pequeños o grandes auditorios, se necesita contar con los siguientes elementos:

- Proyector multimedia, que permite ver las imágenes en pantalla gigante.
- Parlantes, que permite escuchar en alto volumen la música y otros sonidos.

- Equipo de sonido y micrófonos que permite escuchar la voz del expositor en altoparlantes para que pueda ser escuchado por todo el auditorio, cuando se trata del dictado de conferencias.

CAPITULO 2

JAVA 3D

2.1 Que es Java3D

Java3D es una interfaz de programación utilizada para realizar aplicaciones y applets con gráficos en tres dimensiones. Proporciona a los desarrolladores un alto nivel para crear y manipular objetos geométrios 3D y para construir las estructuras utilizadas en el **renderizado** de dichos objetos. Se pueden describir grandes **mundos virtuales** utilizando estos constructores, que proporcionan a Java3D la suficiente información para hacer un renderizado de forma eficiente.

Los objetos geométricos creados por los constructores residen en un **universo virtual**, que luego es renderizado. El API está diseñado con flexibilidad para crear universos virtuales precisos de una amplia variedad de tamaños, desde astronómicos a subatómicos.

A pesar de toda esta funcionalidad, la API es sencilla de usar. Los detalles de renderizado se manejan automáticamente. Aprovechándose de los Threads, Java3D es capaz de renderizar en paralelo.

2.2 Características

El modelado de Java3D se basa en múltiples objetivos, siendo el principal el rendimiento. Se tomaron diferentes decisiones relativas al modelado de tal forma que las implementaciones de Java3D proporcionaran el mejor rendimiento posible a las aplicaciones de usuario. En particular, cuando se realizan distribuciones, se elige la alternativa que permite obtener mejores prestaciones en tiempo de ejecución. Otros objetivos importantes de Java3D son:

- Proporcionar un amplio conjunto de utilidades que permitan crear mundos en 3D interesantes. Se evitó incluir características no esenciales o que se podrían colocar directamente sobre Java3D.
- Proporcionar un paradigma de programación orientado a objeto de alto nivel para permitir a los desarrolladores generar sofisticadas aplicaciones y applets de forma rápida.

- Proporcionar soporte a cargadores en tiempo de ejecución. Esto permite que Java3D se adapte a un gran número de formatos de ficheros, como pueden ser formatos específicos de distintos fabricantes de CAD, formatos de intercambio o VRML 1.0 (*Virtual Reality Modelling Language*) y VRML 2.0.

Las aplicaciones en Java3D construyen los distintos elementos gráficos como objetos separados y los conectan unos con otros mediante una estructura en forma de árbol denominada **grafo de escena**. La aplicación manipula los diferentes objetos utilizando los métodos de acceso, de modificación y de unión definidos en su interfaz.

2.3 Geometrías en java 3D

Hay tres formas principales de crear contenidos geométricos. Una forma es usar las clases de utilidades geométricas para box, cone, cylinder, y sphere. Otra forma es especificar coordenadas de vértices para puntos, segmentos de líneas y/o superficies poligonales. Una tercera forma es usar un cargador geométrico.

2.4 Sistema de Coordenadas del Mundo Virtual

Un ejemplar de la clase VirtualUniverse sirve como raíz para el escenario gráfico de todos los programas Java 3D. El término 'Universo Virtual' comúnmente se refiere al espacio virtual de tres dimensiones que rellenan los objetos Java 3D. Cada objeto del universo virtual establece un sistema de coordenadas Cartesianas. Un objeto Locale sirve como punto de referencia para los objetos visuales en un universo virtual. Con un Locale en un SimpleUniverse, hay un sistema de coordenadas en el universo virtual.

El sistema de coordenadas del universo virtual Java 3D es de mano derecha. El eje X es positivo hacia la derecha, el eje Y es positivo hacia arriba y el eje Z es positivo hacia el espectador, con todas las unidades en metros.

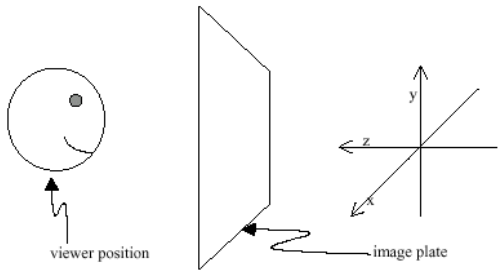


Figura 2. 1 Orientación de Ejes en un Mundo Virtual

2.5 Definición Básica de Objeto Visual

Un ejemplar de Shape3D Define un Objeto Visual, un nodo Shape3D de escenario gráfico define un objeto visual. Shape3D es una de las subclases de la clase Leaf: por lo tanto, los objetos Shape3D sólo pueden ser hojas en un escenario gráfico. El objeto Shape3D no contiene información sobre la forma o el color de un objeto visual. Esta información está almacenada en los objetos NodeComponent referidos por el objeto Shape3D. Un objeto Shape3D puede referirse a un componente nodo Geometry y a un componente nodo Appearance.

En los escenarios gráficos de la página anterior, el símbolo de objeto genérico (rectángulo) fue utilizado para representar el objeto ColorCube. El sencillo escenario gráfico de la Figura 2-2 muestra un objeto visual representado como una hoja Shape3D (triángulo) y dos NodeComponents (óvalos) en lugar del rectángulo genérico.

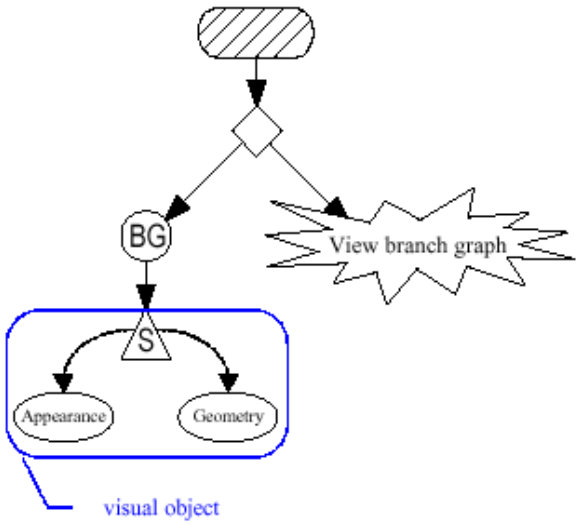


Figura 2. 2 Un Objeto Shape3D Define un Objeto Visual en un Escenario Gráfico.

Un objeto visual se puede definir usando sólo un objeto Shape3D y un nodo componente Geometry. Opcionalmente, el objeto Shape3D también se refiere a un nodo componente Appearance. Los constructores de Shape3D muestran que se pueden crear sin referencias a componentes nodos, con sólo una referencia a un nodo Geometry, o con referencias a ambos tipos de componentes.

Mientras que el objeto Shape3D no esté vivo o compilado, las referencias a los componentes pueden modificarse con varios. Estos métodos pueden usarse sobre objetos Shape3D vivos o compilados si se configuran las capacidades del objeto primero.

2.6 NodeComponent

Los objetos NodeComponent contienen las especificaciones exactas de los atributos de un objeto visual. Cada una de las muchas subclases de NodeComponent define ciertos atributos visuales. La Figura 2-3 muestra una parte del árbol del API Java 3D que contiene las clases NodeComponent y sus descendientes.

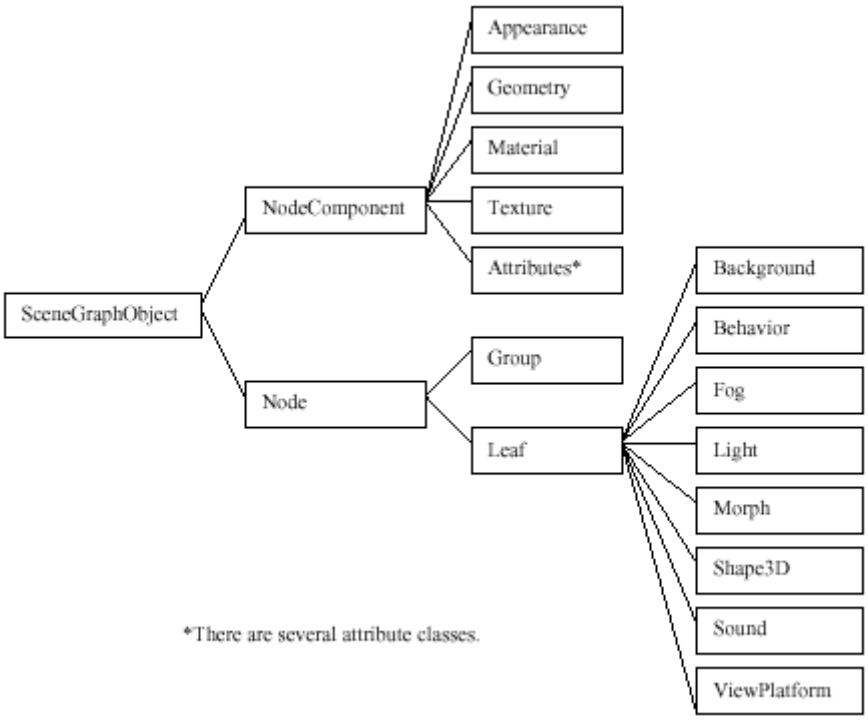


Figura 2. 3 Visión Parcial del Árbol de Clases de Java 3D Mostrando las Subclases de NodeComponent.

2.7 Clases de Utilidades Geométricas

Esta sección cubre las clases de utilidad para crear gráficos primitivos geométricos como cajas, conos, cilindros y esferas. Los primitivos geométricos son la segunda forma más fácil para crear contenidos en un universo virtual. La más fácil es usar la clase ColorCube.

Las clases primitivas proporcionan al programador más flexibilidad que la clase ColorCube. Un objeto ColorCube define la geometría y el color en un componente Geometry. Consecuentemente, todo en el ColorCube es fijo, excepto su tamaño. El tamaño de un ColorCube sólo se especifica cuando se crea.

Un objeto primitivo proporciona más flexibilidad especificando la forma sin especificar el color. En una clase de utilidad geométrica primitiva, el programador no puede cambiar la geometría, pero puede cambiar la apariencia. Las clases primitivas le dan al programador la flexibilidad de tener varios ejemplares de la misma forma geométrica primitiva donde cada una tiene una apariencia diferente haciendo una referencia a un NodeComponent de apariencia diferente.

Las clases de utilidad Box, Cone, Cylinder y Sphere están definidas en el paquete com.sun.j3d.utils.geometry. En la Figura 2.4 podemos ver la porción del paquete com.sun.j3d.utils.geometry que contiene las clases primitivas.

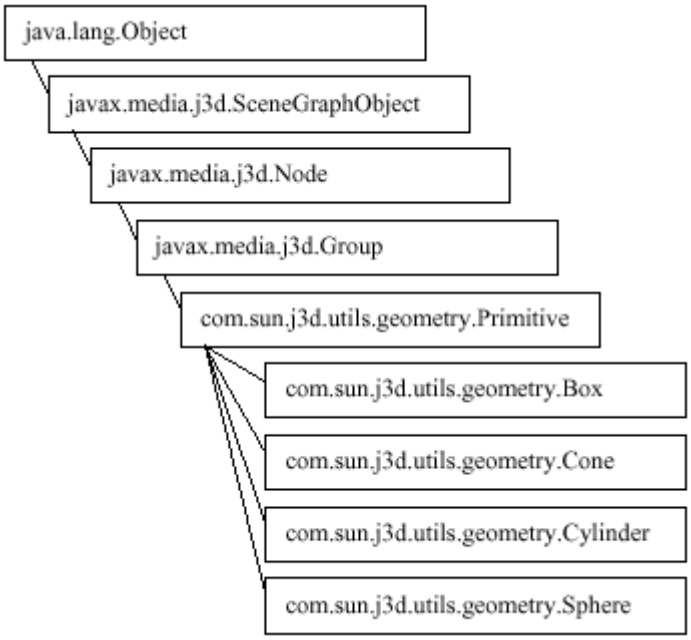


Figura 2. 4 Árbol de Clases de las Clases Geométricas Primitivas

2.7.1 Box

La clase geométrica Box crea cubos de 3 dimensiones. Los valores por defecto para la longitud, anchura y altura son 2 metros, con el centro en el origen, resultando en un cubo con esquinas en (-1, -1, -1) y (1, 1, 1). La longitud, la anchura y la altura pueden especificarse en el momento de la creación del objeto.

2.7.2 Cone

La clase Cone define conos centrados en el origen y con el eje central alineado con el eje Y. Los valores por defecto para el radio y la altura son 1,0 y 2,0 metros respectivamente.

2.7.3 Cylinder

La clase Cylinder crea objetos cilíndricos con sus eje central alineado con el eje Y. Los valores por defecto para el radio y la altura son 1,0 y 2,0 metros respectivamente.

2.7.4 Sphere

La clase Sphere crea objetos visuales esféricos con el centro en el origen. El radio por defecto es de 1,0 metros.

2.8 Clases Matemáticas

Para crear objetos visuales, se necesitan la clase Geometry y sus subclases. Muchas de éstas subclases describen primitivos basados en vértices, como puntos, líneas y polígonos rellenos. A continuación veremos varias clases matemáticas (Point*, Color*, Vector*, TexCoord*) usadas para especificar datos relacionados con los vértices

Todas estas clases matemáticas están en el paquete javax.vecmath.*. Este paquete define varias clases Tuple* como superclases genéricas abstractas. Otras clases más útiles descienden de las clases Tuple. En la Figura 2-5, podemos ver algunas de las clases del árbol.

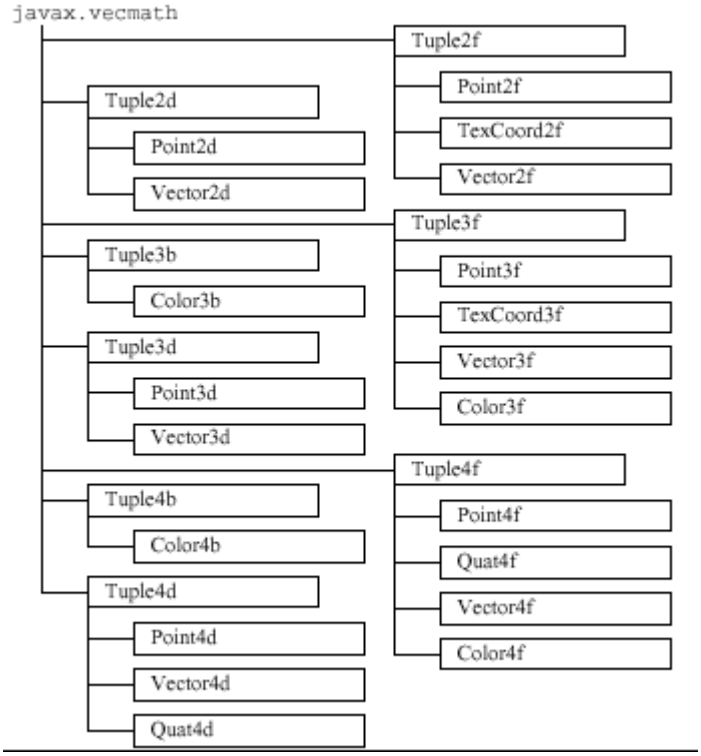


Figura 2. 5 Árbol de clases del paquete de clases matemáticas.

Cada vértice de un objeto visual podría especificar hasta cuatro objetos javax.vecmath, representado coordenadas, colores, superficies normales, y coordenadas de textura. Normalmente se usan las siguientes clases:

- Point* (para coordenadas)
- Color* (para colores)
- Vector* (para superficies normales)
- TexCoord* (para coordenadas de textura)

Observa que las coordenadas (objetos Point*) son necesarios para posicionar cada vértice. Los otros datos son opcionales, dependiendo de cómo se renderice el primitivo. Por ejemplo, se podría definir un color (un objeto Color*) para cada vértice y los colores del primitive se interpolan entre los colores de los vértices. Si se permite la iluminación, serán necesarias las superficies normales (y por lo tanto los objetos Vector*). Si se permite el mapeo de texturas, podrían necesitarse las coordenadas de texturas.

2.8.1 Clases Point

Los objetos Point* normalmente representan coordenadas de un vértice, aunque también pueden representar la posición de una imagen, fuente de un punto de luz, localización espacial de un sonido, u otro dato posicional. Los constructores de las clases Point* son muy similares a los de Tuple*, excepto en que devuelven objetos Point*.

2.8.2 Clases Color

Los objetos Color* representan un color, que puede ser para un vértice, propiedad de un material, niebla, u otro objeto visual. Los colores se especifican con Color3* o Color4*, y sólo para datos de byte o coma flotante de simple precisión. Los objetos Color3* especifican un color como una combinación de valores rojo, verde y azul (RGB). Los objetos Color4* especifican un valor de transparencia, además del RGB. Para los tipos de datos de tamaño byte, los valores de colores van desde 0 hasta 255. Para tipos de datos de coma flotante de simple precisión, los valores van entre 0,0 y 1,0.

2.8.3 Clases Vector

Los objetos Vector* frecuentemente representan superficies normales en vértices aunque también pueden representar la dirección de una fuente de luz o de sonido. De nuevo, los constructores de las clases Vector* son similares a los de Tuple*. Sin embargo, los objetos Vector* añaden muchos métodos que no se encuentran en las clases Tuple*.

2.8.4 Clases TexCoord

Hay sólo dos clases TexCoord* que pueden usarse para representar las coordenadas de textura de un vértice: TexCoord2f y TexCoord3f. TexCoord2f mantiene las coordenadas de textura como una pareja de coordenadas (s, t); TexCoord3f como un trio (s, t, r).

Los constructores para las clases TexCoord* también son similares a los de Tuple* como las clases Color*, las clases TexCoord* tampoco tienen métodos adicionales, por eso sólo tratan con los métodos que heredan de sus superclases.

2.9 Clases Geometry

En los gráficos 3D por ordenador, todo, desde el más sencillo triángulo al más complicado de los modelos Jumbo, está modelado y renderizado con datos basados en vértices. Con Java 3D, cada objeto Shape3D debería llamar a su método setGeometry () para referenciar uno y sólo uno objeto Geometry. Para ser más preciso, Geometry es una superclase abstracta, por eso los objetos referenciados son ejemplares de una de sus subclases.

Estas subclases se dividen en tres categorías principales:

- Geometría basada en vértices no indexados (cada vez que se renderiza un objeto visual, sus vértices sólo podrían usarse una vez)
- Geometría basada en vértices indexados (cada vez que se renderiza un objeto visual, sus vértices se podrían reutilizar)
- Otros objetos visuales (las clases Raster, Text3D, y CompressedGeometry)

Esta sección cubre las dos primeras categorías. El árbol de clases de Geometry y sus subclases se muestran en la Figura 2.6.

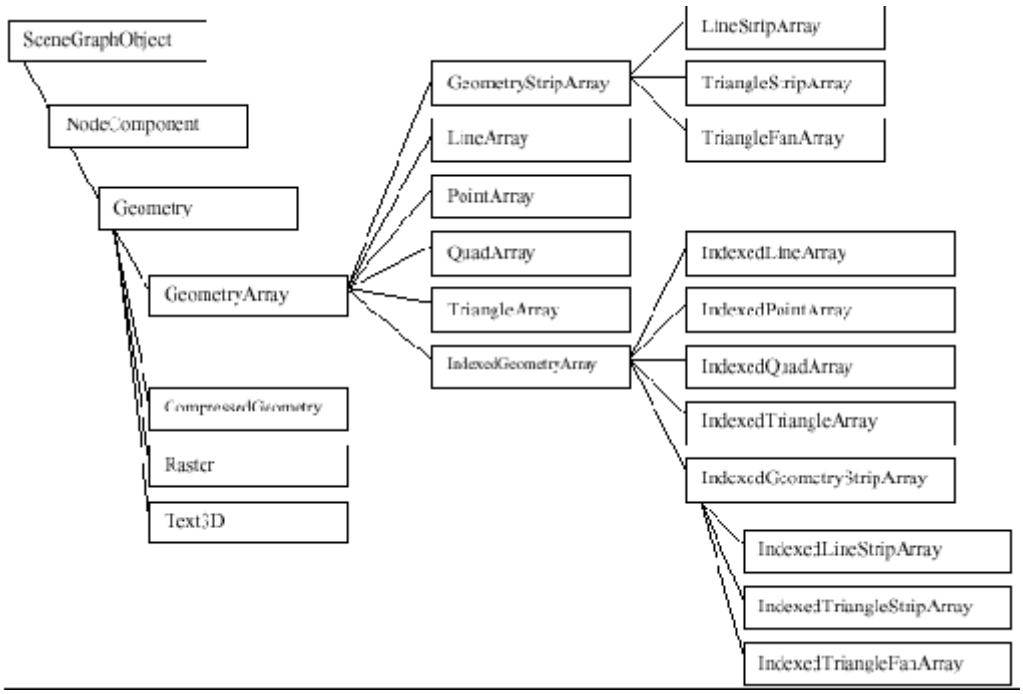


Figura 2. 6 Árbol de clases de Geometry.

2.10 Clase GeometryArray

Como se podría deducir de los nombres de las clases, las subclases de Geometry podrían usarse para especificar puntos, líneas y polígonos rellenos (triángulos y cuadriláteros). Estos primitivos basados en vértices son subclases de la clase abstracta GeometryArray, que indica que cada una tiene un array que mantiene los datos de los vértices.

Por ejemplo, si se usa un objeto GeometryArray para especificar un triángulo, se define un array de tres elementos: un elemento para cada vértice. Además de la localización de las coordenadas, se pueden definir otros tres arrays opcionales para almacenar el color, la superficie normal, y las coordenadas de textura. Estos arrays que contienen las coordenadas, los colores, las superficies normales y las coordenadas de texturas, son los "arrays de datos".

Hay tres pasos en la vida de un objeto **GeometryArray**:

Paso 1: Construcción de un objeto GeometryArray vacío.

Cuando se construye por primera vez un objeto GeometryArray, se deben definir dos cosas:

- El número de vértices (arrays de elementos) necesarios.
- El tipo de datos (coordenadas de localización, color, superficie normal, y/o coordenadas de textura) a almacenar en cada vértice. Esto se llama formato del vértice.

Paso 2: Rellenar con Datos el Objeto GeometryArray.

Después de construir el objeto GeometryArray, asignamos valores a los arrays, correspondiendo a los formatos de los vértices asignados. Esto se podría hacer vértice por vértice, o usando un array para asignar datos a muchos vértices con una sola llamada del método.

Paso 3: Hacer que los Objetos Shape3D Referencien a los Objetos GeometryArray.

Los objetos Shape3D se añaden a un BranchGroup, que en algún lugar se añade al escenario gráfico general. (Al contrario que los objetos GeometryArray, que son NodeComponents, Shape3D es una subclase de Node, por eso pueden ser añadidos como hijos al escenario gráfico.)

2.10.1 Subclases de GeometryArray

La clase GeometryArray es una superclase abstracta para subclases más útiles, como LineArray. La Figura 2.7 muestra el árbol de subclases de GeometryArray. La principal distinción entre estas subclases es cómo el renderizador Java 3D decide renderizar sus vértices.

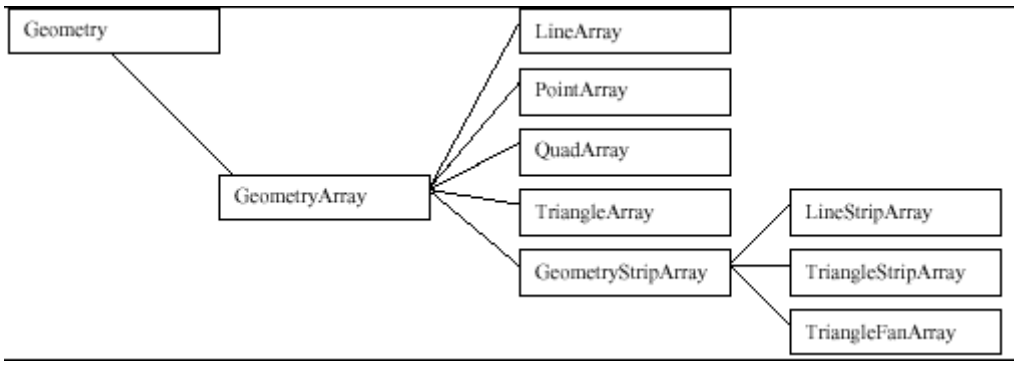


Figura 2. 7 Subclases no indexadas de GeometryArray.

La Figura 2.8 muestra ejemplos de las cuatro subclases de GeometryArray: PointArray, LineArray, TriangleArray, y QuadArray. En esta figura, los tres conjuntos más a la izquierda muestran los mismos seis vértices renderizando seis puntos, tres líneas, o dos triángulos. La cuarta imagen muestra cuatro vértices definiendo un cuadrilátero.

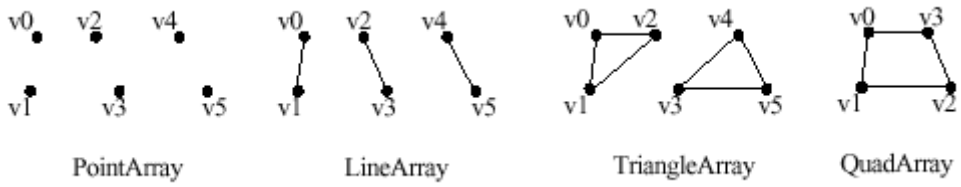


Figura 2. 8 Subclases de GeometryArray.

Por defecto, se rellena el interior de los triángulos y cuadriláteros. En la última sección, aprenderemos los atributos que pueden influenciar en el modo de renderizado de los primitivos rellenos.

2.11 Contenidos sencillos en java 3D

2.11.1 Cargadores

Una clase Loader lee ficheros de escenas 3D (no ficheros Java 3D) y crea representaciones Java 3D de sus contenidos que pueden ser añadidos selectivamente a un mundo Java 3D y argumentados por otro código Java 3D. El paquete com.sun.j3d.loaders proporciona el contenido principal para convertir los ficheros creados en otras aplicaciones en aplicaciones Java 3D. Las clases cargadoras implementan el interface Loader definido en el paquete com.sun.j3d.loaders.

Como hay una gran variedad de formatos de ficheros para propósitos de representación de escenas 3D (por ejemplo, .obj, .vrml, etc.) y siempre habrá más formatos de ficheros, el código real para cargar un fichero no forma parte de Java 3D o del paquete loaders; sólo se incluye el interface para el mecanismo de carga. Con la definición del interface, el usuario de Java 3D puede desarrollar clases cargadoras de ficheros con el mismo interface que las otras clases cargadoras.

2.11.1.1 Cargadores Disponibles Públicamente

En Java 3D existen varias clases cargadoras. A continuación se listan algunos ejemplos:

Formato de Fichero	**Descripción**
3DS	3D-Studio
DXF	AutoCAD Drawing Interchange File
OBJ	Wavefront
SLD	Solid Works (prt and asm files)
WRL	Virtual Reality Modeling Language

Tabla 2. 1 Cargadores Java 3D disponibles Públicamente

Esta gran variedad de cargadores existe para hacer más sencilla la escritura de cargadores para los diseñadores Java 3D. Las clases Loader son implementaciones del interface Loader que baja el nivel de dificultad para escribir un cargador. Como en el ejemplo, un programa que carga un fichero 3D realmente usa un cargador y un objeto escena. El cargador lee, analiza y crea la representación Java 3D de los contenidos del fichero. El objeto escena almacena el escenario gráfico creado por el cargador. Es posible cargar escenas desde más de un fichero (del mismo formato) usando el mismo objeto cargador y crear múltiples objetos escena. Los ficheros de diferentes formatos pueden combinarse en un programa Java 3D usando las clases cargadoras apropiadas.

La clase LoaderBase proporciona una implementación para cada uno de los tres métodos load () del interface Loader. LoaderBase también implementa dos constructores.

2.11.2 Texto 3D

La forma de añadir texto a un mundo virtual Java 3D es crear un objeto Text3D para texto. El Text3D crea texto usando geometría. La geometría textual de un objeto Text3D es una extrusión de la fuente.

Crear un objeto Text3D es un poco más complicado que crear un objeto Text2D. El primer paso es crear un objeto Font3D con el tipo de fuente, el tamaño y el estilo seleccionado. Luego se crea un objeto Text3D para una cadena particular usando el objeto Font3D. Como la clase Text3D es una subclase de Geometry, el objeto Text3D es un NodeComponent que es referenciado por uno o más objetos Shape3D:

Figura 2. 9 Punto de Referencia por Defecto y Extrusión de un Objeto 3DText

El texto de un objeto Text3D puede orientarse de una gran cantidad de formas. La orientación se especifica como el camino de dirección. Las direcciones son right, left, up, y down.

Cada objeto Text3D tiene un punto de referencia. El punto de referencia para un objeto Text3D es el origen del objeto. El punto de referencia de cada objeto se define por la combinación del camino y la alineación del texto. La Tabla 2.2 muestra los efectos de las especificaciones del camino y la alineación sobre la orientación del texto y la situación del punto de referencia.

	ALIGN_FIRST (Default)	ALIGN_CENTER	ALIGN_LAST
PATH_RIGHT (Default)	Text3D	Text3D	Text3D
PATH_LEFT	D3txeT	D3txeT	D3txeT
PATH_DOWN	T e x t 3 D	T e x t 3 D	T e x t 3 D

PATH_UP	D 3 t x e T	D 3 t x e T	D 3 t x e T

Tabla 2. 2 Alineación y Orientación del texto 3D

Los objetos Text3D tienen superficies. La adicción de un paquete de apariencia que incluye un objeto Material a un objeto Shape3D referenciando la geometría Text3D permitirá la iluminación del objeto Text3D.

2.11.2.1 Clases Usadas en la Creación de Objetos Text3D

Esta sección presenta el material de referencia para tres clases usadas en la creación de objetos Text3D: Text3D, Font3D, y FontExtrusion, en este orden. La Figura 2.10 muestra el árbol de clases de Text3D.

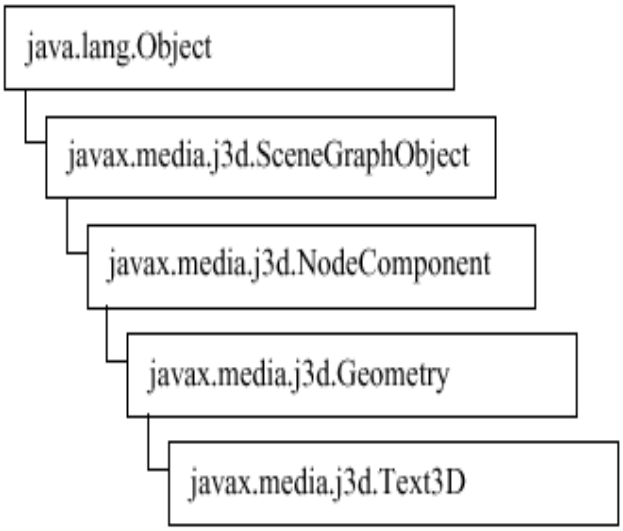


Figura 2. 10 Árbol de Clases de Text3D

Cada objeto Text3D se crea desde un objeto Font3D. Un sólo objeto Font3D puede usarse para crear un número ilimitado de objetos Text3D. Un objeto Font3D contiene la extrusión geométrica de cada carácter en el tipo de letra. Un objeto Text3D copia las geometrías para formar la cadena especificada. Los objetos Font3D pueden ser

recolectados por el recolector de basura sin afectar a los objetos Text3D creados a partir de él.

2.11.3 Fondo

Por defecto, el fondo de un universo virtual Java 3D es negro sólido. Sin embargo, podemos especificar otros fondos para nuestros mundos virtuales. El API Java 3D proporciona una forma fácil de especificar un color sólido, una imagen, una geometría o una combinación de éstos como fondo.

Cuando especificamos una imagen para el fondo, se sobrescribe la especificación del color de fondo, si existe. Cuando se especifica una geometría, se dibuja sobre el color de fondo o la imagen.

La única parte espinosa es la especificación de un fondo geométrico. Toda la geometría de fondo se especifica como puntos en una esfera. Si nuestra geometría es un PointArray, que podría representar estrellas a años luz, o un TriangleArray, que podría representar montañas en la distancia. La geometría de fondo se proyecta sobre el infinito cuando se renderiza.

Los objetos Background tienen límites de aplicación, lo que nos permite que se puedan especificar diferentes fondos para diferentes regiones del mundo virtual. Un nodo Background está activo cuando su región de aplicación se interseca con el volumen de activación del ViewPlatform.

Si están activos varios nodos Background, el nodo que está más "cercano" al ojo será el utilizado. Si no hay ningún nodo Background activo, la ventana se mostrará en negro. Sin embargo, la definición de "más cercano" no está especificada. Por cercano, se elige el fondo con los límites de aplicación más internos que encierra la ViewPlatform.

Es improbable que nuestra aplicación necesite iluminar la geometría del fondo en realidad el sistema visual humano no puede percibir los detalles visuales a grandes distancias. Sin embargo, una geometría de fondo si puede ser sombreada. La geometría del fondo podría no contener luces, pero las luces definidas en el escenario gráfico pueden influenciar en la geometría del fondo.

2.11.3.1 La clase Background

La Figura 2.11 muestra el árbol de clase de la clase Background. Como una extensión de la clase Leaf, un ejemplar de la clase Background puede ser un hijo de un objeto Group.

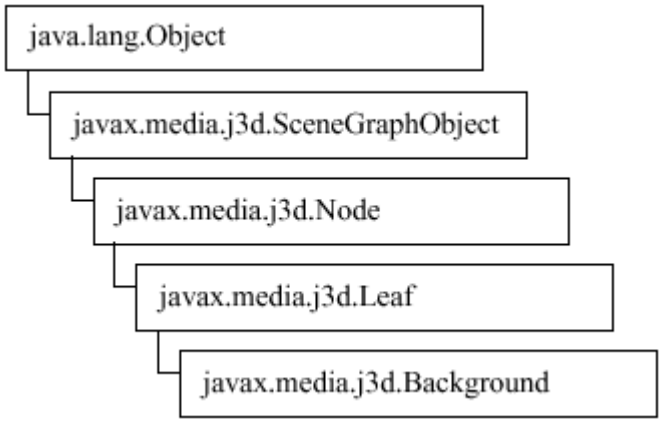


Figura 2. 11 Árbol de Clases de Background.

2.11.4 BoundingLeaf

Los Bounds (límites) se usan con luces, comportamientos, fondos y una gran variedad de otras aplicaciones en Java 3D. Los Bounds permiten al programador variar la acción, la apariencia, y/o el sonido sobre el campo virtual. La especificación de Bounds también permite al sistema de renderizado de Java 3D mejorar la ejecución del recortado y por lo tanto mejorar el rendimiento.

La especificación típica de límites utiliza un objeto Bounds para limitar una región. En el escenario gráfico resultante, los objetos Bounds se mueven con los objetos que lo referencian. Esto está bien para muchas aplicaciones; sin embargo, podría haber situaciones en las que fuera deseable tener la región límite que se moviera independientemente de los objetos que usan los límites.

Por ejemplo, si un mundo incluye una fuente de luz estacionaria que ilumina unos objetos en movimiento, los límites de la luz deberían incluir el objeto en movimiento. Una forma de manejar esto podría ser crear los límites lo suficientemente grandes como

para incluir todos los lugares donde se mueve el objeto. Esta no es la mejor respuesta en muchos casos. Una mejor solución es usar un BoundingLeaf. Situado en el escenario gráfico con el objeto visual, el BoundingLeaf se mueve con el objeto visual independientemente de la fuente de luz. La Figura 2-12 muestra un escenario gráfico con un una luz que usa un nodo BoundingLeaf.

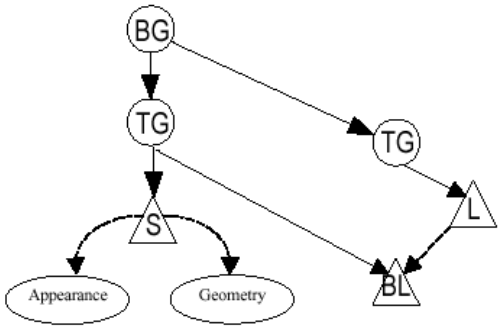


Figura 2. 12 Movimiento de un objeto BoundlingLeaf

2.12 Interacción en java 3D

La interacción y la animación se especifican con objetos Behavior. La clase Behavior es una subclase abstracta que proporciona el mecanismo para incluir código que modifique el escenario gráfico. La clase Behavior, y sus descendientes, son enlaces a código del usuario que proporciona las modificaciones para los gráficos y los sonidos del universo virtual.

El propósito del objeto Behavior en un escenario gráfico es modificar el propio escenario gráfico, o los objetos que hay dentro de él, en respuesta a algunos estímulos. Un estímulo puede ser una pulsación de tecla, un movimiento del ratón, la colisión de objetos, el paso del tiempo, algún otro evento, o una combinación de estos. Los cambios producidos incluyen la adicción de objetos al escenario gráfico, la eliminación de objetos, cambio de atributos de los objetos del escenario gráfico, reordenación de los objetos del escenario gráfico, o una combinación de estos. Las posibilidades sólo están limitadas por las capacidades de los objetos del escenario gráfico.

2.12.1 Animación contra Interacción

Si un usuario navega en un programa donde se proporciona un comportamiento, la vista se moverá en respuesta a eventos del teclado y/o ratón. El movimiento de la plataforma de la vista es una interacción porque es el resultado directo de una acción del usuario.

Sin embargo, otras cosas podrían cambiar como resultado del movimiento de la plataforma de la vista, (por ejemplo, comportamientos de cartelera o LOD). Los cambios causados como resultado del movimiento de la plataforma de vista son indirectamente causados por el usuario y por lo tanto son animaciones.

2.12.2 Behavior Básico

Como se explicó en la sección anterior, las clases Behavior se usan en muchas aplicaciones Java 3D y de muchas formas. Es importante entender las consideraciones de funcionamiento y programación de estas clases. Esta sección explica la clase Behavior, ofrece una receta para programar clases de comportamientos personalizadas, y muestra una aplicación de ejemplo que usa una clase Behavior.

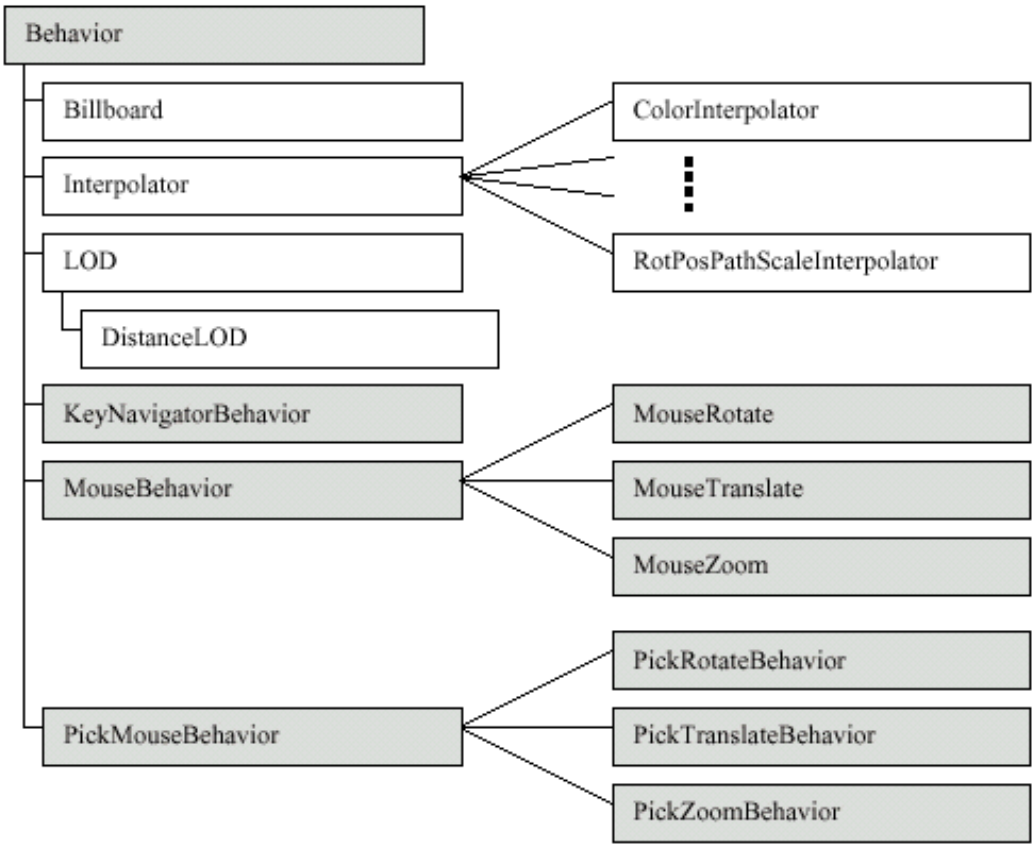


Figura 2. 13 Árbol de Subclases de Behavior

2.12.3 Mecanismo de Behaviors

Una clase de comportamiento personalizado implementa los métodos de inicialización y processStimulus de la clase abstracta Behavior. Por supuesto, la clase de

comportamiento personalizado, también tiene al menos un constructor y también podría tener otros métodos.

La mayoría de los comportamientos actuarán sobre un objeto del escenario gráfico para afectar al comportamiento. El objeto sobre el que actúa un comportamiento es referido como el objeto del cambio. Es a través de este objeto, u objetos, que el comportamiento afecta al mundo virtual. Aunque es posible tener un comportamiento que no tenga un objeto del cambio, la mayoría lo tienen.

El comportamiento necesita una referencia a su objeto(s) de cambio para poder realizar los cambios de comportamiento. Se puede usar el constructor para seleccionar la referencia del objeto de cambio. Si no se hace, otro método de la clase de comportamiento personalizado debe almacenar esta información. En cualquier caso, la referencia se hace en el momento en que se construye el escenario gráfico, que es el primer cálculo de comportamiento.

El método de inicialización se invoca cuando el escenario gráfico que contiene la clase de comportamiento se vuelve vivo. Este método de iniciación es responsable de seleccionar el evento de disparo inicial para el comportamiento y seleccionar la condición inicial de las variables de estado del comportamiento. El disparo se especifica como un objeto WakeupCondition, o una combinación de objetos WakeupCondition.

El método processStimulus se invoca cuando ocurre el evento de disparo especificado para el comportamiento. Este método es responsable de responder al evento. Como se pueden codificar muchos eventos en un sólo objeto WakeupCondition (por ejemplo, varias acciones de teclado podrían estár codificados en un WakeupOnAWTEvent), esto incluye la descodificación del evento. El método processStimulus responde al estímulo, normalmente modificando el objeto de cambio, y, cuando es apropiado, reseteando el disparo.

2.12.4 Disparo de los Comportamientos

Los comportamientos activados se disparan por la ocurrencia de uno o más estímulos especificados. El estímulo de disparo para un comportamiento se especifica usando descendientes de la clase WakeupCondition.

La clase abstracta, WakeupCondition, es la base para todas las clases de disparo del API Java 3D. Cinco clases extienden WakeupCondition, una es la clase abstracta WakeupCriterion, las otras cuatro permiten la composición de múltiples condiciones de disparo en una única condición de disparo. La Figura 2.14 muestra el árbol de clases.

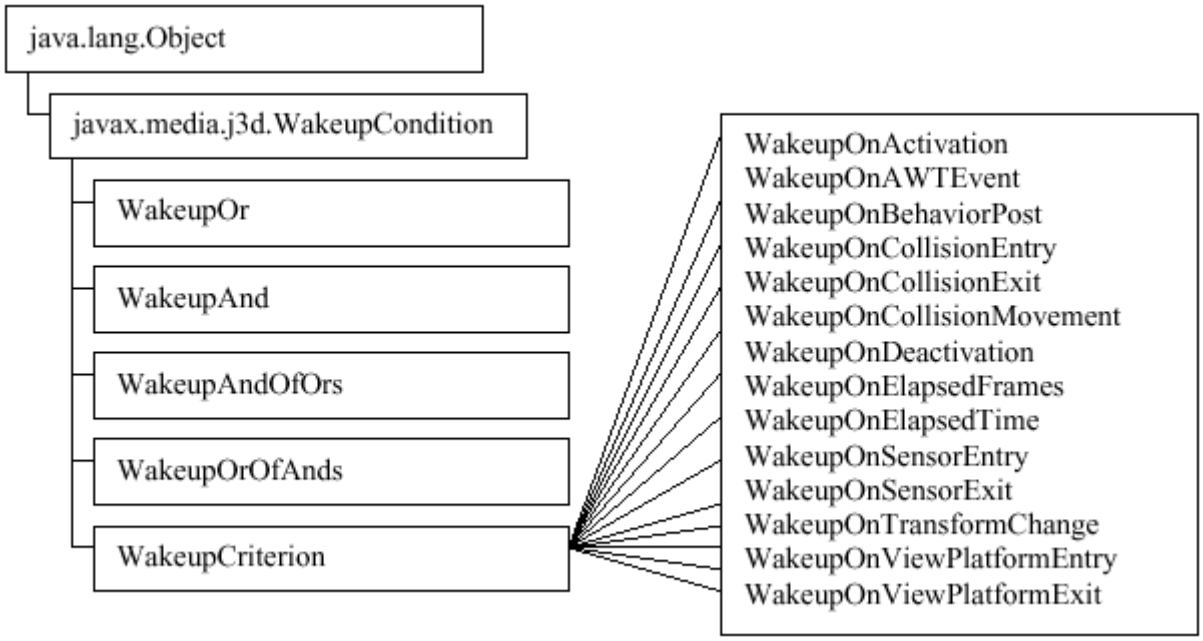


Figura 2. 14 Árbol de Clases para WakeupCondition y clases relacionadas.

2.12.5 Clases WakeupCriterion Específicas

La Tabla 2.4 presenta las 14 clases WakeupCriterion específicas. Estas clases se usan para especificar las condiciones de disparo de los objetos behavior. Los ejemplares de estas clases se usan individualmente o en combinaciones.

Clase Criterio	Disparo
WakeupOnActivation	En la primera detección de una intersección del volumen de activación de un ViewPlatform con la región límite del objeto.
WakeupOnAWTEvent	Cuando ocurre un evento AWT específico.
WakeupOnBehaviorPost	Cuando un objeto behavior envía un evento específico.

WakeupOnCollisionEntry	En la primera detección de colisión del objeto especifico con otro objeto del escenario gráfico.
WakeupOnCollisionExit	Cuando el objeto especifico no colisiona con ningún otro objeto del escenario gráfico.
WakeupOnCollisionMovement	Cuando el objeto especificado se mueve mientras colisiona con otro objeto del escenario gráfico.
WakeupOnDeactivation	Cuando el volumen de activación de un ViewPlatform deja de interactuar con los límites del objeto.
WakeupOnElapsedFrames	Cuando ha pasado un número determinado de frames.
WakeupOnElapsedTime	Cuando ha pasado un número de segundos determinado.
WakeupOnSensorEntry	En la primera detección de cualquier sensor que intersecciones con los límites especificados.
WakeupOnSensorExit	Cuando un sensor que intersecciona va con los límites del objeto deja de interseccionar con los límites especificados.
WakeupOnTransformChange	Cuando cambia la transformación dentro de un TransformGroup especificado.
WakeupOnViewPlatformEntry	En la primera detección de intersección del volumen de activación de un ViewPlatform con los límites especificados.
WakeupOnViewPlatformExit	Cuando el volumen de activación de una vista deja de intereseccionar con los límites especificados.

Tabla 2. 3 Clases WakeupCriterion específicas.

2.12.6 Clases de Comportamientos Útiles para la Navegación por Teclado

Hasta este momento, el espectador ha estado en una localización fija con una orientación fija. El poder mover el espectador es una capacidad importante en muchas aplicaciones de los gráficos 3D. Java 3D es capaz de mover el espectador.

La Figura 2.15 muestra la rama gráfica básica para un universo virtual de Java 3D. En esta figura, se considera la plataforma de la visión transform. Si se cambia la transformación, el efecto es mover, o reorientar, o ambas, al espectador. De esto, podemos ver que el diseño básico de la navegación del teclado es simple: hacemos que un objeto behavior cambie la transformación de la vista de la plataforma en respuesta a los movimientos dominantes.

Este diseño simple es exactamente el modo de trabajo de las clases utilitarias del teclado de Java 3D.

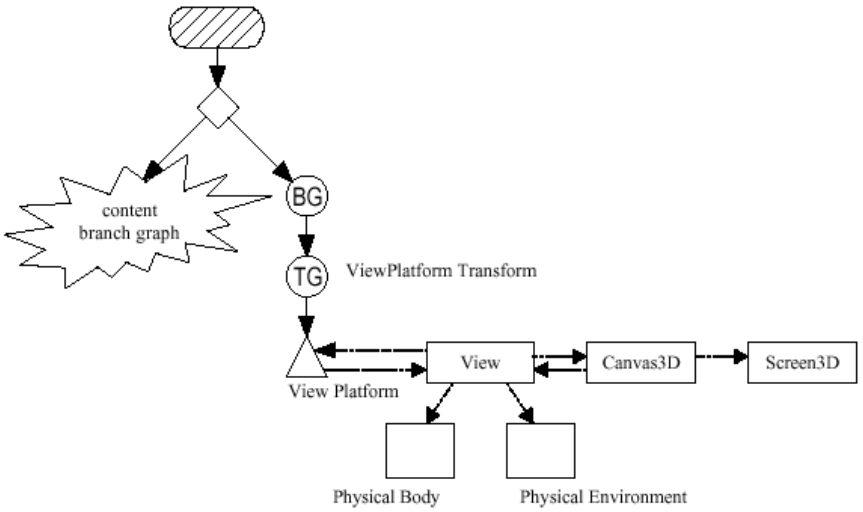


Figura 2. 15 Rama gráfica básica para un universo virtual de Java 3D

2.12.6.1 Clases KeyNavigatorBehavior y KeyNavigator

La utilidad de navegación del teclado se implementa como dos clases. En el tiempo de ejecución hay dos objetos. El primer objeto es el objeto KeyNavigatorBehavior, el segundo es un objeto KeyNavigator. El objeto KeyNavigatorBehavior realiza todas las funciones típicas de una clase de comportamiento, excepto que llama al objeto KeyNavigator para realizar la función de processStimulus. La clase KeyNavigator toma el AWTEvent y lo procesa bajo al nivel de pulsaciones de teclas individuales. La Tabla

siguiente muestra el efecto de las pulsaciones de teclas individuales. KeyNavigator implementa el movimiento con aceleración.

Tecla	Movimiento	Movimiento Alt-tecla
<-	rotar a la izquierda	traslación lateral izquierda
->	rotar a la derecha	traslación lateral derecha
^	mover hacia adelante	
V	mover hacia atrás	
PgUp	rotar arriba	translación hacia arriba
PgDn	rotar abajo	translación hacia abajo
+	aumenta la distancia de salto (y vuelve al origen)	
-	reduce la distancia de salto	
=	vuelve al centro del universo	

Tabla 2. 4 Movimientos de KeyNavigatorBehavior

2.12.7 Clases de Utilidad para Interactuar con el Ratón

El paquete de comportamientos de ratón (com.sun.j3d.utils.behaviors.mouse) contiene las clases del comportamiento en las cuales el ratón se utiliza como entrada de información para la interacción con los objetos visuales. Incluyendo las clases para traslaciones (moviéndose en un plano paralelo a la placa de la imagen), enfocando (que mueve hacia atrás y adelante), y los objetos visuales que rotan en respuesta a los movimientos del ratón.

La siguiente tabla resume las tres clases específicas del comportamiento del ratón incluidas en el paquete. Además de estas tres clases, está la clase abstracta MouseBehavior, y el interface MouseCallback. Esta clase abstracta y el interface se utilizan en la creación de las clases específicas del comportamiento del ratón y son útiles para crear comportamientos personalizados del ratón.

Clase MouseBehavior	Acción en Respuesta a la Acción del Ratón	Acción del ratón
MouseRotate	Rota el objeto visual sin moverlo.	Botón izquierdo pulsado con

		movimiento del ratón.
MouseTranslate	Traslada el objeto visual en un plano paralelo al plato de imagen.	Botón derecho pulsado con movimiento del ratón.
MouseZoom	Traslada el objeto visual en un plano orthogonal al plato de imagen.	Botón central pulsado con movimiento del ratón.

Tabla 2. 5 Sumario de las clases específicas de MouseBehavior

2.12.7.1 MouseNavigation

Las tres clases específicas de comportamiento del ratón se pueden utilizar para crear un universo virtual en el cual el ratón se utilice para la navegación. Cada una de las clases específicas del comportamiento del ratón tiene un constructor que toma un solo parámetro entero para las banderas. Cuando se utiliza MouseBehavior.INVERT_INPUTS como argumento a este constructor, el comportamiento del ratón responde en la dirección opuesta. Este comportamiento inverso es apropiado para cambiar el transform ViewPlatform. Es decir las clases del comportamiento del ratón se pueden utilizar para el control navegacional.

2.12.8 Picking y sus clases de Utilidad

Picking (Elección) le da al usuario una forma de interactuar con objetos visuales individuales en la escena.

Picking está implementado por un comportamiento típicamente disparado por eventos de botones del ratón. En la selección de un objeto visual, el usuario sitúa el puntero del ratón sobre el objeto elegido y pulsa el botón del ratón. El objeto behavior se dispara por la pulsación de este botón y empieza la operación de selección. Se proyecta un rayo dentro del mundo virtual desde la posición del puntero del ratón paralela con la proyección. Se calcula la intersección de este rayo con los objetos del mundo virtual. El objeto visual que intersecciona más cerca al plato de la imagen se selecciona para interacción. La Figura 2.16 muestra un rayo de selección proyectado en un mundo virtual.

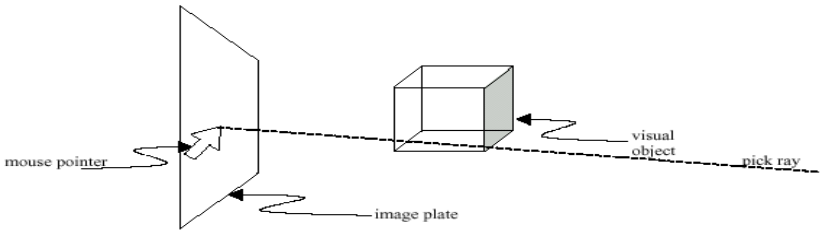


Figura 2. 16 Proyección de un PickRay en el Mundo Virtual

En algunos casos la interacción no se hace directamente con el objeto seleccionado, sino con un objeto en el camino del escenario gráfico hasta el objeto. La determinación del objeto para un posterior procesamiento no siempre es sencilla. Si un objeto visual cúbico que va a ser rotado está compuesto por seis objetos Shape3D individuales junto con seis objetos TransformGroup, como en el escenario gráfico de la Figura 2.17, no es el objeto TransformGroup sobre el objeto Shape3D interseccionado el que necesita ser modificado. El 'cubo' se rota por la manipulación del objeto TransformGroup que es hijo del objeto BranchGroup en el escenario gráfico. Por esta razón, el resultado de algunas operaciones de selección es devolver el path del escenario gráfico para su posterior procesamiento.

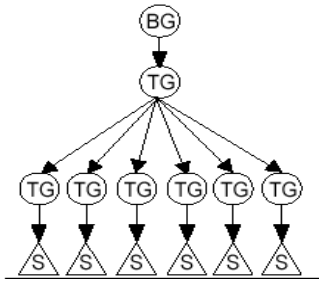


Figura 2. 17 Diagrama del Escenario Gráfico del Cubo de Shape3D planos.

La comprobación de intersecciones es necesita mucho cálculo. Por lo tanto, la selección es cara y se vuelve más cara con la complejidad de la escena.

Hay dos aproximaciones básicas para usar las características de selección de Java 3D, usar objetos de clases picking, o crear clases picking personalizadas y usar ejemplares de estas clases. El paquete picking incluye clases para pick/rotate, pick/translate, y pick/zoom.

Cómo un objeto comportamiento picking operará sobre cualquier objeto del escenario gráfico (con las capacidades apropiadas), sólo se necesita proporcionar un objeto picking.

PickPoint

Los objetos PickPoint representan un punto para selección. Como una subclase de PickShape, los objetos PickBounds se usan con BranchGroup y Locale así como con clases del paquete picking.

PickRay

Los objetos PickRay representan un rayo (un punto y una dirección) para selección. Como una subclase de PickShape, los objetos PickBounds se usan con BranchGroup y Locale así como con clases del paquete picking.

PickSegment

Los objetos PickSegment representan un segmento de línea (definida por dos puntos) para selección. Como una subclase de PickShape, los objetos PickBounds se usan con BranchGroup y Locale así como con clases del paquete picking.

SceneGraphPath

La clase SceneGraphPath se usa en la mayoría de las aplicaciones de selección. Esto es porque normalmente la selección implica encontrar un camino de escenario gráfico en el que se encuentra un objeto para permite la manipulación del objeto o de un objeto TransformGroup en el path.

Un objeto SceneGraphPath representa el camino del escenario gráfico hacia el objeto elegido permitiendo la manipulación del objeto o de un objeto TransformGroup en el camino del objeto.

2.12.9 Clases Generales del Paquete Picking

Incluidas en el paquete com.sun.j3d.utils.behaviors.picking hay varias clases de comportamientos generales y específicos. Las clases generales son útiles para crear

nuevos comportamientos de selección, entre las que se incluyen PickMouseBehavior, PickObject, y PickCallback.

Clase PickMouseBehavior

Esta es la clase base para los comportamientos de selección específicos proporcionados en el paquete. También es útil para extender clases de comportamientos de selección personalizados.

Clase PickObject

La clase PickObject proporciona métodos para determinar qué objeto fue seleccionado por una operación de selección del usuario. Una amplia variedad de métodos resulta de las distintas formas posibles de aplicaciones de selección. Es útil crear clases de selección personalizadas.

Interface PickingCallback

El interface PickingCallback proporciona un marco de trabajo para extender una clase de selección existente. En particular cada una de las clases específicas implementa este interface permitiendo al programador proporcionar un método que sea llamado cuando la operación de selección tenga lugar.

Clase Intersect

La clase Intersect proporciona varios métodos para comprobar la intersección de un objeto PickShape (clase corazón) y geometrías primitivas. Esta clases es útil para la creación de clases picking personalizadas.

2.12.10 Clases de Comportamientos Picking Específicas

Incluidas en el paquete com.sun.j3d.utils.behaviors.picking hay clases de comportamientos específicas: PickRotateBehavior, PickTranslateBehavior, y PickZoomBehavior. Estas clases permiten al usuario interactuar con un objeto seleccionado con el ratón. Los comportamientos individuales responden a los diferentes botones del ratón (izquierdo=rotar, derecho=trasladar, central=zoom). Estas clases son subclases de PickMouseBehavior.

PickRotateBehavior

Esta clase permite al usuario seleccionar y rotar interactivamente un objeto visual. El usuario usa el botón izquierdo del ratón para seleccionar y rotar. Se puede usar un ejemplar de PickRotateBehavior en conjunción con otras clases de selección específicas.

PickTranslateBehavior

Esta clase permite al usuario seleccionar y trasladar interactivamente un objeto visual. El usuario usa el botón derecho del ratón para seleccionar y trasladar. Se puede usar un ejemplar de PickTranslateBehavior en conjunción con otras clases de selección específicas.

PickZoomBehavior

Esta clase permite al usuario seleccionar y hacer zoom interactivamente un objeto visual. El usuario usa el botón central del ratón para seleccionar y hacer zoom. Se puede usar un ejemplar de PickZoomBehavior en conjunción con otras clases de selección específicas.

2.13 Animación en java 3D

Al igual que las interacciones, las animaciones Java 3D se implementan usando objetos Behavior. Como podrás imaginar, se puede crear cualquier animación personalizada usando objetos Behavior. Sin embargo, el API Java 3D proporciona varias clases útiles para crear animaciones sin tener que crear una nueva clase.

Los Interpoladores y los Objetos Alpha Proporcionan Animaciones Basadas en el Tiempo. Las acciones de Interpolator incluyen el cambio de localización, orientación, tamaño, color o transparencia de un objeto visual. Todos los comportamientos interpoladores podrían implementarse creando una clase behavior personalizada; sin embargo, usar un interpolador hace más sencilla la creación de estas animaciones. Las clases Interpolator existen para otras acciones, incluyendo algunas combinaciones de estas acciones.

2.14 Iluminación en java 3D

Java 3D sombrea los objetos visuales basándose en la combinación de sus características materiales y en las luces del universo virtual. El sombreado resulta de aplicar un modelo de iluminación a un objeto visual en presencia de fuentes de luz. La siguiente sección da una descripción del modelo de iluminación usado en el renderizador de Java 3D y cómo la luz interactúa con las características materiales para proporcionar sombreado.

2.14.1 Sombreado en Java 3D

Sombrear objetos visuales en Java 3D depende de muchos factores. Esta sección proporciona una descripción abreviada del modelo de iluminación de Java 3D, del modelo de color, y del modelo de sombreado.

2.14.2 Modelo de Iluminación

En el mundo real, los colores que percibimos son una combinación de las características físicas del objeto, de las características de las fuentes de luz, de las posiciones relativas de los objetos a las fuentes de luz, y del ángulo desde el cual se ve el objeto. Java 3D utiliza un modelo de iluminación para aproximar la física del mundo real.

La ecuación del modelo de iluminación depende de tres vectores: la superficie normal (n), la dirección de la luz (l), y la dirección al ojo del espectador (e) además de las características materiales del objeto y de las características de la luz. La Figura 2.18 muestra los tres vectores para dos vértices de una superficie esférica. Los vectores para cada vértice pueden tener distintas direcciones dependiendo de especificidades de la escena. Cuando los vectores de la luz y del ojo varían, se calculan en tiempo de ejecución. Por lo tanto, cada vértice de la esfera potencialmente se renderiza como una sombra diferente.

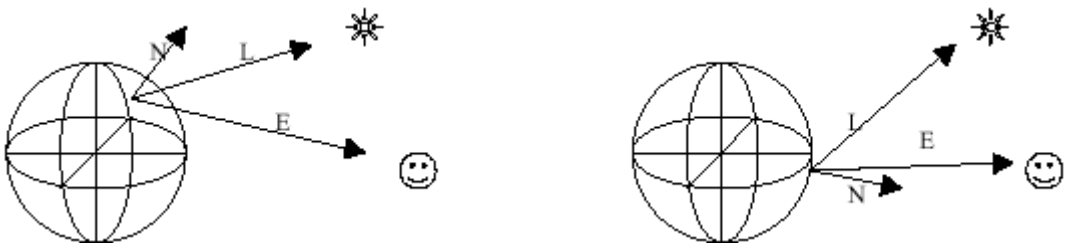


Figura 2. 18 Vectores de Luz, Superficie Normal y Ojo del espectador.

El modelo de iluminación incorpora tres tipos de reflexiones de luz del mundo real: ambiente, difuso, y especular. La reflexión ambiente resulta de la luz ambiente, luz constante de bajo nivel, en una escena. La reflexión difusa es la reflexión normal de una fuente de luz desde un objeto visual. Las reflexiones especulares son las reflexiones sobre iluminadas de una fuente de luz sobre un objeto, que ocurren en ciertas situaciones.

La Figura 2.19 muestra una esfera y un plano renderizado por Java 3D... La parte más oscura de la esfera exhibe sólo la reflexión ambiente. El centro de la esfera está iluminado por la luz difusa y ambiente. Con una esfera azul y una luz blanca, la reflexión difusa es azul. La parte más brillante de la esfera es el resultado de la reflexión especular con reflexiones ambiente y difusa.

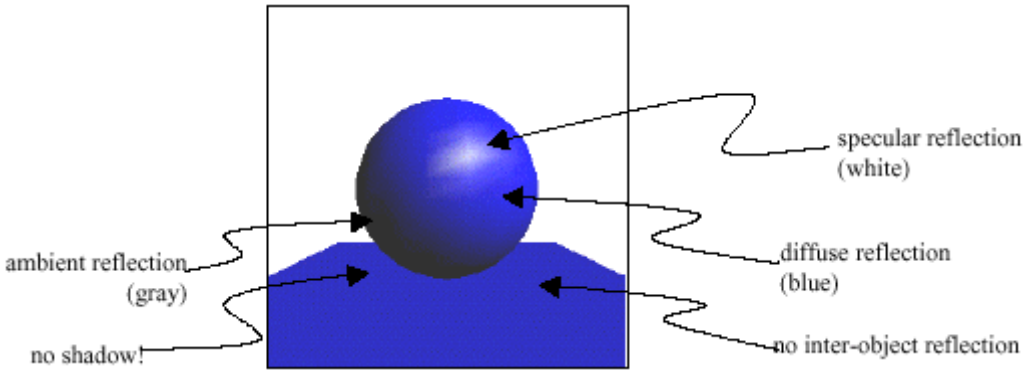


Figura 2. 19 Esfera sombreada y un Plano.

2.14.3 Clase Light

El API Java 3D proporciona cuatro clases para luces. Todas se derivan de la clase Light. La Figura 2.20 muestra la jerarquía de clases de Java 3D relacionada con las luces. Light, una clase abstracta, proporciona los métodos y las constantes de capacidades asociadas para manipular el estado, color, y los límites de un objeto Light. El estado de la luz es un booleano que activa y desactiva la luz.

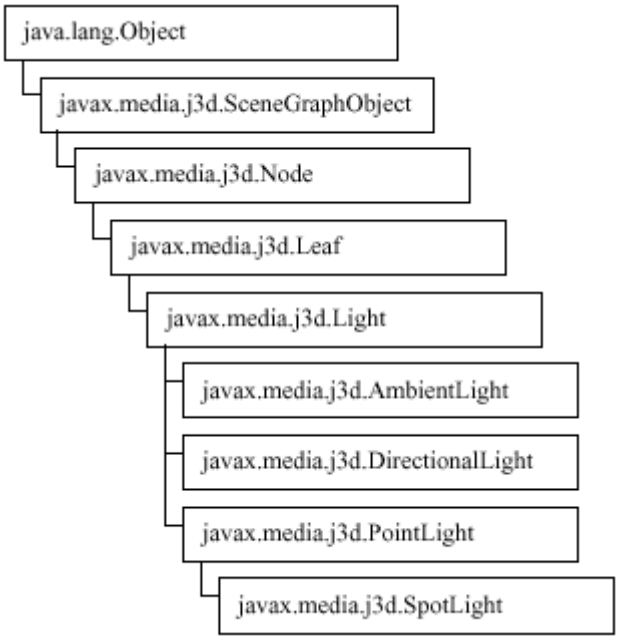


Figura 2. 20 El Árbol de Clases del API Java 3D con las clases Light

El siguiente bloque de referencia lista los métodos y las constantes de la clase Light. Debemos recordar que los límites seleccionados con setInfluencingBounds () activan una luz cuando el objeto bounds referenciado intersecciona con la vista.

2.14.3.1 Luz Ambiente

Los objetos de luz ambiente proporcionan luz de la misma intensidad en todas las localizaciones y en todas las direcciones. Los objetos de luz ambiente modelan la luz reflejada desde otros objetos visuales. Si miramos la superficie inferior de nuestro escritorio, veremos la parte inferior del escritorio aunque ninguna fuente de luz esté dando directamente en esa superficie (a menos que tengamos una lámpara bajo el escritorio). La luz que brillaba hacia arriba en el fondo del escritorio se reflejó en el suelo y en otros objetos. En ambientes naturales con muchos objetos, la luz se refleja desde muchos objetos para proporcionar la luz ambiente. La clase AmbientLight de Java 3D simula este efecto.

2.14.3.2 Luz Direccional

Una fuente DirectionalLight aproxima fuentes de luz muy distantes tales como el sol. Al contrario que las fuentes AmbientLight, las fuentes DirectionalLight proporcionan una sola dirección al brillo de luz. Para los objetos iluminados con una fuente DirectionalLight, el vector de luz es constante.

La Figura 2.21 muestra dos vértices de la misma esfera que están siendo iluminados por una fuente DirectionalLight. El vector de luz es igual para estos dos y para todos los vértices. Puesto que todos los vectores de luz de una fuente DirectionalLight son paralelos, la luz no se atenúa. En otras palabras, la intensidad de una fuente DirectionalLight no varía con la distancia al objeto visual y la fuente DirectionalLight.

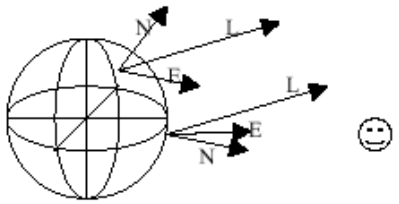


Figura 2. 21 El Vector de Luz es Constante para Fuentes DirectionalLight.

2.14.3.3 Punto de Luz

Un PointLight es el contrario de un DirectionalLight. Es una fuente de luz omnidireccional cuya intensidad se atenúa con la distancia y tiene una localización. Los objetos PointLight se aproximan a bombillas, velas, u otras fuentes de luz sin reflectores o lentes.

Un modelo de ecuación cuadrática modela la atenuación de las fuentes PointLight. La Figura 2.22 ilustra la relación de un objeto PointLight con una esfera. Observa que los vectores de luz no son paralelos.

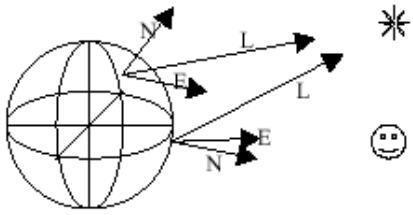


Figura 2. 22 El vector de Luz varía para una fuente PointLight.

2.15 Texturas en java 3D

El texturado, también llamado mapeo de textura, es una manera de añadir riqueza visual a una superficie sin la adición de los detalles geométricos finos. La riqueza visual la proporciona una imagen, también llamada textura, que da el aspecto del detalle

superficial para el objeto visual. La imagen se mapea dentro de la geometría del objeto visual en el momento de la renderización. De ahí el término mapeo de textura.

El aspecto de muchos objetos del mundo real depende de su textura. La textura de un objeto es realmente la geometría relativamente fina de la superficie de un objeto. Para apreciar la superficie del papel las texturas juegan con el aspecto de los objetos del mundo real, consideremos una alfombra. Incluso cuando todas las fibras de una alfombra son del mismo color la alfombra no aparece con un color constante debido a la interacción de la luz con la geometría de las fibras. Aunque Java 3D es capaz de modelar la geometría de las fibras individuales de la alfombra, los requisitos de memoria y el funcionamiento de la renderización para una alfombra del tamaño de una habitación modelada a tal detalle harían dicho modelo inútil. Por otra parte, tener un polígono plano de un solo color no hace un reemplazo convincente para la alfombra en la escena renderizada.

La Figura 2.23 muestra algunas de las texturas como podemos ver, una textura puede proporcionar riqueza visual a objetos de distintos tamaños.

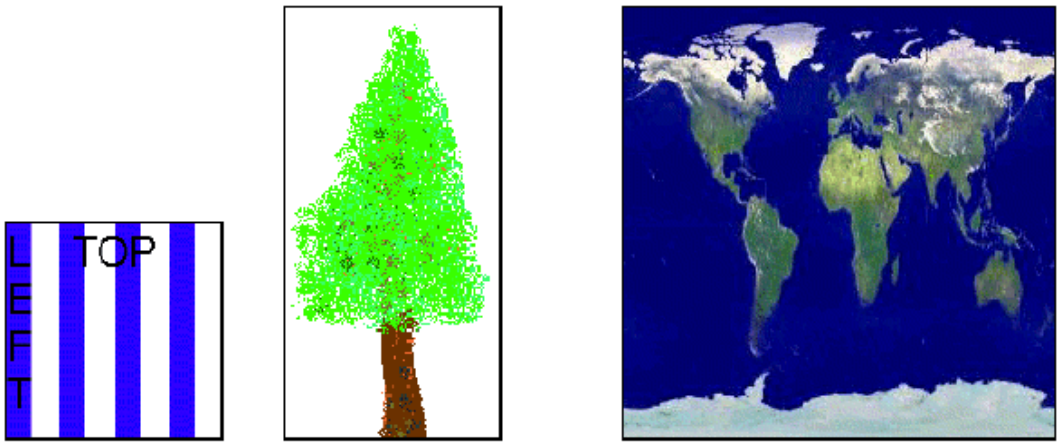


Figura 2. 23 Imágenes usadas como Texturas en los Programas de Ejemplo.

2.15.1 Texturado Básico

El texturado de polígonos en un programa de Java 3D se consigue a través de la creación del manojo de apariencia apropiado y cargando la imagen de la textura dentro de él, especificando la localización de la imagen de la textura en la geometría, y fijando los atributos de texturado. Como veremos, especificar texturas puede ser muy complejo.

Afortunadamente, hay clases de utilidad para ayudarnos en el proceso y las configuraciones de los valores por defecto para las opciones texturado son las apropiadas para las aplicaciones de texturado básicas.

2.15.2 La Clase NewTextureLoader

Los programas Java 3D que usan texturas pueden tener una gran cantidad de líneas sólo para cargar las texturas y crear los manojos de apariencia. Se puede ahorrar algo de programación y, más importante, memoria en tiempo de ejecución compartiendo manojos de apariencia cuando sea apropiado. Sin embargo, esto no reduce mucho la cantidad de programación. Se pueden conseguir otras reducciones de programación creando una clase para crear los manojos de apariencia de textura. El desafío de crear esta clase consiste en el requisito del observador de imagen para el objeto TextureLoader.

El objeto Canvas3d o un Applet pueden servir como el observador de imagen, pero tener una referencia a un cierto componente por todas partes en el programa puede ser fastidioso. Para tratar esta inconveniencia, se ha extendido la clase TextureLoader que elimina la necesidad de un componente observador de imagen. En su lugar se utiliza un solo método para especificar un observador de imagen para todas las aplicaciones futuras del cargador de textura.

Los constructores de NewTextureLoader son iguales a los de TextureLoader excepto en que ninguno requiere un componente observador de imagen.

2.15.3 Coordenadas de Textura

La imagen de la textura se crea para caber en la geometría basándose en la especificación de las coordenadas de textura. El proceso real es asociar los texels de la textura a los pixeles de la geometría cuando es renderizada. Cada pixel de una textura se llama un texel, o un 'elemento de textura'. Éste es el proceso de mapeado de la textura.

El mapeo de textura comienza con la especificación de las coordenadas de textura para los vértices de la geometría. Mientras se renderiza cada pixel del triángulo texturado, se calculan las coordenadas de la textura en el pixel desde los vértices del triángulo. La interpolación Trilinear de las coordenadas de textura de los vértices determina las

coordenadas de textura para el pixel y por lo tanto, el texel de la imagen de la textura usada en el color final del pixel.

La Figura 2.24 ilustra el proceso de interpolación trilinear para un pixel del ejemplo. La representación se hace en orden de scan. El pixel, P, para el mapeo de textura está justo en el centro del scan actual en el triángulo de la izquierda de la ilustración. Se han asignado las coordenadas de textura para cada uno de los vértices del triángulo. Se han etiquetado TC1, TC2, y TC3. Estas coordenadas de textura son el punto de partida para la interpolación trilinear. Las primeras dos interpolaciones lineares determinan las coordenadas de la textura a lo largo de las caras del triángulo en el scan (puntos etiquetados A y B en la figura). La tercera interpolación se hace entre estos dos puntos.

Las coordenadas de textura que resultan para P son (0,75, 0,6). En la derecha de la figura está la textura. Usando las coordenadas de textura calculadas para P se selecciona el texel.

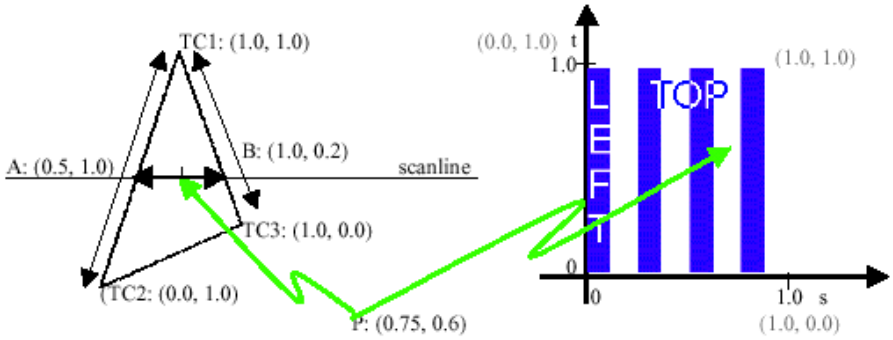


Figura 2. 24 Imagen del Mapeo de Textura

La selección del Texel no se explica completamente en el ejemplo anterior. La especificación de la sección de filtración da más detalles sobre la selección del texel. Otro detalle todavía no explicado es la interacción entre el color del texel, otras fuentes del color, y el color final del pixel. El modo de valor por defecto se 'substituye' con el cuál se utiliza el color del texel como el color del pixel, pero hay otros modos.

En este punto del capítulo se han utilizado todas las texturas en su orientación ordinaria. La Figura 2.25 muestra planos con algunas de las posibles orientaciones de textura sólo seleccionando las coordenadas de textura en los vértices.

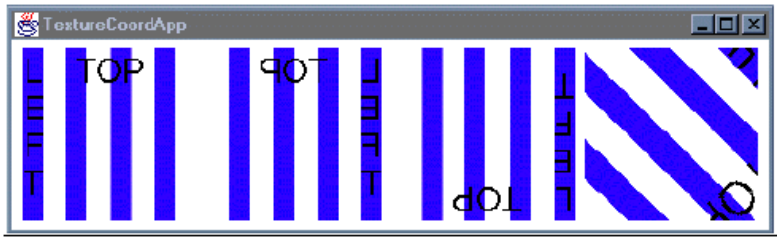


Figura 2. 25 Algunas de las posibles orientaciones de una Textura en un Plano.

2.15.4 Opciones de Textura

Texture2D, es una extensión de Texture. Algunas de las opciones básicas para el texturado se implementan en esta clase. Puesto que Texture es una clase abstracta, sus configuraciones se harán a través de un objeto Texture2D o Texture3d. Las configuraciones son el modo de límites, filtros, y el formato de la textura.

2.15.5 Modo de Límites: Envolver o Abrazar

La configuración del modo de límites determina lo que ocurre cuando el mapeo tiene lugar si las coordenadas de textura van más allá del rango 0 a 1 del espacio de la imagen. Las opciones son envolver la imagen, o abrazar la imagen. Envolver, significa repetir la imagen según sea necesario, es el valor por defecto. Abrazar utiliza el color del borde de la imagen en cualquier lugar fuera del rango 0 a 1. Estas configuraciones se hacen independientemente en las dimensiones s y t.

2.15.6 Formato de Textura

La última configuración de la clase Texture es la del formato de textura. El formato de textura es una declaración de cuántos valores hay por texel y cómo esos valores afectan a los pixels. Por ejemplo, una configuración del formato de textura de INTENSITY indica que solo el valor del texel será utilizado para rojo, verde, azul, y los valores de alpha del pixel. Una configuración del formato de textura de RGB indica que los tres valores del texel serán utilizados para los valores rojo, verde, y azul del pixel mientras que el valor de alpha del pixel sigue siendo igual.

Formato de Textura	Valores por Texel	Modifica Color del Pixel	Modifica alpha del Pixel
INTENSITY	1	si, R=G=B	si, R=G=B=A
LUMINANCE	1 (sólo color)	si, R=G=B	No
ALPHA	1 (sólo alpha)	no	Si
LUMINANCE_ALPHA	2	si, R=G=B	Si
RGB	3	si	No
RGBA	4	si	Si

Tabla 2. 6 Formato de textura

2.15.7 Texture3d

Como el nombre implica, un objeto Texture3d contiene una imagen tridimensional de la textura. Puede ser que pensemos en él como un volumen de color. La clase Texture3d es una extensión de Texture, así que todas las características de la clase Texture se aplican a Texture3d. La única característica que Texture3d tiene

6. METODOLOGIA PARA LA EJECUCION DE LA INVESTIGACIÓN

6.1 MATRIZ DE CONSISTENCIA GENERAL

<table>
<tr><td colspan="4">PROBLEMA GENERAL DE INVESTIGACIÓN (ENUNCIADO):

“Limitaciones del software 2D de desarrollo de diapositivas causan que los estudiantes y profesionales limiten sus presentaciones de trabajos o hagan uso de programas adicionales privativos para mejorar la calidad de sus diapositivas, lo que a su vez representa un costo adicional al costo del paquete de Microsoft office, por lo cual se construirá un software para el desarrollo de diapositivas 3D en la ciudad de Loja en el periodo de septiembre 2009 – septiembre 2010”</td></tr>
<tr><td>TEMA</td><td>OBJETO DE INVESTIGACIÓN</td><td>OBJETIVO DE LA INVESTIGACIÓN</td><td>HIPÓTESIS DE INVESTIGACIÓN</td></tr>
<tr><td>Construcción de una herramienta multimedia para el desarrollo de diapositivas tridimensionales bajo licencia GPL, haciendo uso de la tecnología JAVA 3D, la cual está orientado para el sector académico y profesional del Área de la Energía las Industrias y los Recursos Naturales No Renovables</td><td>Construcción del software de desarrollo de diapositivas tridimensionales utilizando la tecnología java3D.</td><td>Desarrollar e implantar un software de desarrollo de diapositivas tridimensionales utilizando la tecnología java3D para el sector estudiantil de la ciudad y provincia de Loja.</td><td>Con el desarrollo de este SW los estudiantes lograran realizar presentaciones de gran calidad y de una manera sencilla y rápida.</td></tr>
</table>

6.2 MATERIALES, MÉTODOS Y TÉCNICAS DE TRABAJO

La Metodología es el estudio científico que nos enseña a descubrir nuevos conocimientos determinados para ordenar la actividad que desea cumplir.

Técnicas.

Para poder recolectar la información útil nos hemos apoyado en las siguientes técnicas:

La Entrevista: Es la técnica más significativa y productiva de que dispone el analista para recabar datos, la utilizamos para obtener la información en forma verbal, a través de preguntas al personal comunes y a las personas que requieren de éstos servicios.

La Observación: La técnica de la observación es muy útil para reconocer la forma en que se labora en la Institución. El propósito de la observación es múltiple, nos permite determinar; ¿Qué se está haciendo?, ¿Cómo se está haciendo?, ¿Quién lo hace?, ¿Cuándo se lleva a cabo?, ¿Cuánto tiempo toma?, ¿Dónde de hace?, ¿Por qué se hace?, las cuales son pautas que nos servirán para el desarrollo de nuestro proyecto.

Métodos.

Los Métodos que utilizaremos en el presente trabajo de investigación serán los siguientes:

Método Científico: Lo utilizaremos como guía principal de toda la investigación, ya que a través de este se planteó el problema, los objetivos: general y específico, además nos permitirá la organización, procesamiento, análisis, e interpretación de la información obtenida acerca de la Institución.

Método Inductivo-Deductivo.- los cuales siguen el proceso analítico sintético que satisfacen los requerimientos propios de las ciencias informáticas (recolección de datos, análisis de la información e interpretación de los hechos y descubrimiento de nuevos procedimientos).

Método Ciclo de Vida de un Sistema: Comprende las diferentes etapas por la que tiene que pasar un sistema, éste método guía a los desarrolladores a establecer los principales elementos que intervendrán en el desarrollo, las mejores guías para implementar y las tácticas que se tomarán en las diferentes etapas:

Análisis: Se trata de utilizar las diferentes técnicas para recoger la información, seleccionar y categorizar para poder utilizar en la siguiente etapa de la planificación sin tener dificultad en futuro.

Diseño: Con la información seleccionada se elabora un prototipo que permitirá definir la apariencia principal que tomará el Software y hacemos una breve idea de las prestaciones que dará.

Desarrollo: Para poder empezar con la codificación es necesario seguir las especificaciones del prototipo final que se realizó en la etapa de diseño, de esta manera disminuimos conflictos no esperados.

Pruebas: Una vez terminada la codificación de la aplicación, realizamos las respectivas pruebas para comprobar que el Software esté realizando lo deseado y que los resultados sean los correctos.

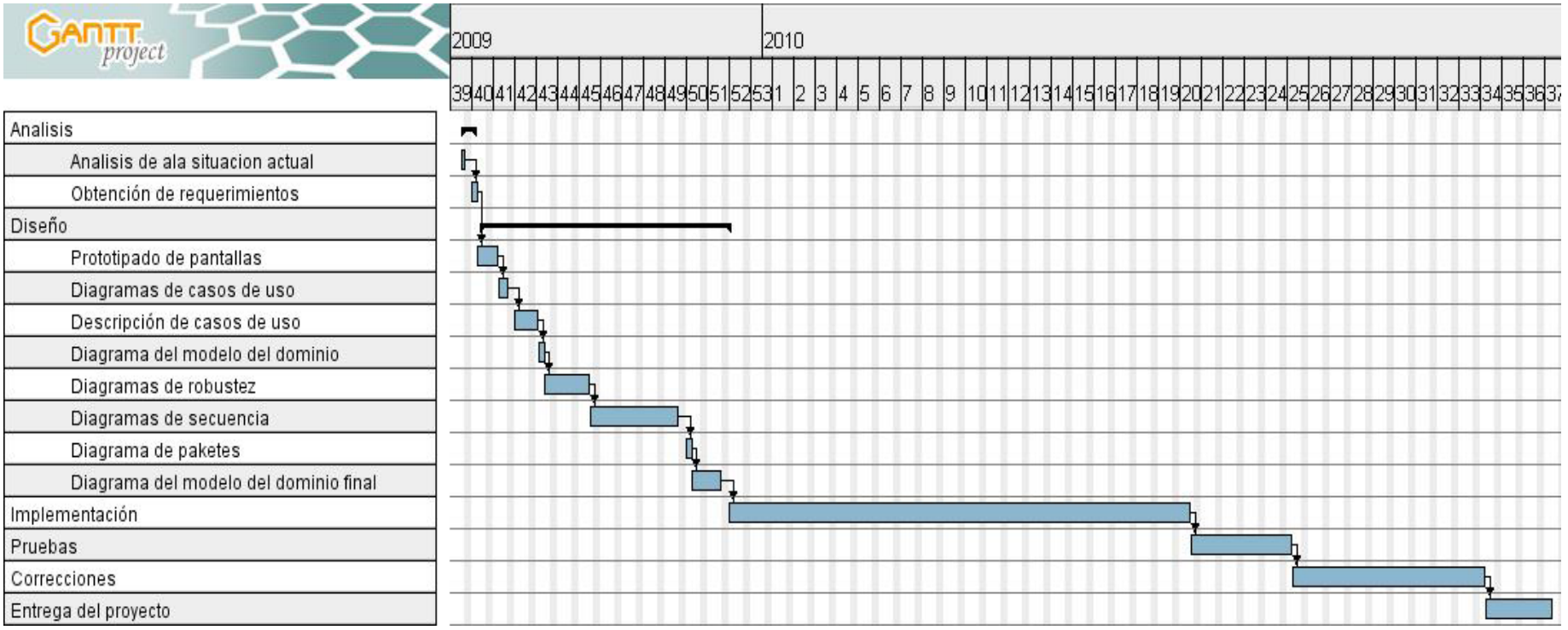
Implementación: Comprobado ya el Software implantamos la Aplicación en el lugar que se lo necesita o para lo que fue creado, teniendo en cuenta ciertos requerimientos como Recursos de Hardware y Software.

Mantenimiento: Luego de un tiempo prudencial (2-3 meses) se comprueba si el Software sigue realizando las tareas correctas, caso contrario se harán las modificaciones correspondientes, cabe destacar que además del software puede tener actualizaciones periódicas denominadas como mantenimiento.

Metodología para el proceso de desarrollo de software.

Finalmente la metodología de desarrollo del software que utilizaremos para la creación del proyecto es ICONIX, la cual se utiliza para analizar, modelar y diseñar, diferentes aplicaciones, casos de uso que son manejados dentro del contexto del problema. ICONIX. Es utilizado para capturar los requerimientos del cliente, en un sistema nuevo. Este proceso hace fuerte el uso de la herramienta UML, (Lenguaje Unificado de Modelado) que sirve para modelar la interacción del sistema.

7. CRONOGRAMA DE ACTIVIDADES



Lista de tareas

Nombre	Fecha de inicio	Fecha de fin	Recursos
Analisis	25/09/09	30/09/09	Alfredo Macas Kleber Macas
Analisis de ala situacion actual	25/09/09	28/09/09	
Obtención de requerimientos	28/09/09	30/09/09	
Diseño	1/10/09	22/12/09	Alfredo Macas Kleber Macas
Prototipado de pantallas	30/09/09	7/10/09	
Diagramas de casos de uso	7/10/09	12/10/09	
Descripción de casos de uso	12/10/09	20/10/09	
Diagrama del modelo del dominio	20/10/09	22/10/09	
Diagramas de robustez	22/10/09	6/11/09	
Diagramas de secuencia	6/11/09	7/12/09	
Diagrama de paketes	7/12/09	9/12/09	
Diagrama del modelo del dominio final	9/12/09	21/12/09	
Implementación	21/12/09	21/05/10	Alfredo Macas Kleber Macas
Pruebas	21/05/10	23/06/10	Alfredo Macas Kleber Macas
Correcciones	23/06/10	25/08/10	Alfredo Macas Kleber Macas
Entrega del proyecto	25/08/10	16/09/10	Alfredo Macas Kleber Macas

8. PRESUPUESTO Y FINANCIAMIENTO.

Recursos Humanos

Recursos Humanos	Cantidad	Horas c/u	Costo por Hora	Costo Total
Director de Tesis	----	-----	-----	----
Desarrolladores	2	500	$3.00	$3000.00
TOTAL				$3000.00

Recursos Materiales.

Recursos Materiales	Cantidad	Costo Unitario	Costo Total
Resma de Papel	2	$4	$8.00
Cartuchos de tinta.	5	$10	$50.00
Internet/horas	100	$1	$100.00
Flash Memory (1GB) Kingston.	1	-----	------
TOTAL			$158.00

Recursos Técnicos.

Recursos Técnicos	Cantidad	Horas	Costo por hora	Costo Total
Computadores	2	500	0.30	$300.00
Impresora	1	10	0.10	$1.00
TOTAL				$301.00

Recursos Tecnológicos.

Recursos Tecnológicos	Cantidad	Costo Unitario	Costo Total
NetBeans IDE 6.0	1	Gratuito	$0.00
Eclipse ganymede	1	Gratuito	$0.00
Java JDK 1.6	1	Gratuito	$0.00
Java 3D 1.5	1	Gratuito	$0.00
Omondo.EclipseUML.2008	3	Gratuito	$0.00
TOTAL			$0.00

Resumen del Presupuesto

Resumen del Presupuesto	Costo Total
Recursos Humanos	$3000.00
Recursos Materiales	$158.00
Recursos Técnicos	$301.00
Recursos Tecnológicos	$00.00
SUBTOTAL	$3459.00
Imprevistos 10 %	$345.9
TOTAL	$3804.9

9. BIBLIOGRAFÍA

Archivos Digitales:

- Aitken. G., "Moving from C++ to Java", Dr. Dobb's Journal, March 1996, pp. 52-56
- Anuff. E... Java Source Book, New York, NY: John Wiley & Sons, Inc.. 1996
- Arnold, K., and J. Gosling. The Java Programing Lenguaje, Reading. MA: Addison Weley Publishing Company, 1996.
- Boone. B... "Multitasking in Java" Java Report. May/June 1996, pp 27-33
- Core packages. J. Gosling. Ed. Addison-Wesley Manual de reference
- Introduction to Graphics Programming with Java 3D Doug Twilleager
- Java in a nutshell: a desktop quick reference D. Flanagan. Ed. O'Reilly
- The Java tutorial: object-oriented programming for the Internet M. Campione. Ed. Addison-Wesly Programación del lenguaje
- The Java language specification J. Gosling. Ed. Addison-Wesley Lenguaje y manual de referencia

Internet:

- SUN, 2010. [En línea] Api Java3D,[http://java.sun.com/developer/onlineTraining/java3d/],[Consulta:12 Febrero 2010].
- Juan Antonio Medina Romani, 2004. Uso de equipos y sistemas multimedia en el proceso de aprendizaje enseñanza. [En línea], Uso de equipos y sistemas multimedia en el proceso de aprendizaje enseñanza, [http://www.monografias.com/trabajos20/multimedia-en-aprendizaje/multimedia-en-aprendizaje.shtml], [Consulta: 15 de Septiembre del 2010]
- Luis Antonio Fernández Aldana. 2007. Introducción a Java3D. [En línea] , Que es Java3D, Caracteristicas,[http://www.monografias.com/trabajos43/java-tres-d/java-tres-d3.shtml], [Consulta:15 de Noviembre del 2009]
- Leonardo Muro García , 2007. Licencias de Software. [en línea], Licencias del Software, [http://www.monografias.com/trabajos55/licencias-de-software/licencias-de-software.shtml], [Consulta: 19 de noviembre de 2010]
- Laura Marthé Hinojosa Castillo, 2006. Ingeniería de Software. [en línea], Lenguaje Unificado de Modelado. [http://www.monografias.com/trabajos34/ingenieria-software/ingenieria-software.shtml#quees], [consulta: 19 de noviembre de 2010].
- RENa, 2010.Metodologia [En línea], Ciencia, Método e Investigación, [http://www.rena.edu.ve/cuartaEtapa/metodologia/tema19.html],[Consulta:12 de enero del 2010]

10. ANEXOS

10.1 MATRIZ DE CONSISTENCIA ESPECÍFICA

➢ **PROBLEMA ESPECÍFICO:** Limitación en la presentación de objetos del mundo real.			
OBJETIVO ESPECÍFICO	**HIPÓTESIS ESPECÍFICA**	**UNIDAD DE OBSERVACIÓN**	**SISTEMA CATEGORIAL**
Desarrollar una galería básica de objetos tridimensionales del mundo real.	Con la galería de objetos tridimensionales prediseñados se le dará al usuario la facilidad de crear de manera fácil y rápida sus diapositivas, además se podrá agregar nuevos objetos.	• Herramienta de modelado y diseño tridimensional Blender 3D.	• Introducción a Blender • Volumen I de la Documentación de Blender - Guía de Usuario.

➢ **PROBLEMA ESPECÍFICO:** Falta de un asistente configurable para el desarrollo de diapositivas.			
OBJETIVO ESPECÍFICO	**HIPÓTESIS ESPECÍFICA**	**UNIDAD DE OBSERVACIÓN**	**SISTEMA CATEGORIAL**
Desarrollar un asistente de software configurable para la creación de diapositivas tridimensionales	Con el asistente de software se lograra crear de manera fácil y rápida las diapositivas.	• Software de presentaciones Impres.	• Análisis y diseño de un asistente de software.

➢ **PROBLEMA ESPECÍFICO:** Limitación en la visualización de múltiples objetos en una misma diapositiva.			
OBJETIVO ESPECÍFICO	**HIPÓTESIS ESPECÍFICA**	**UNIDAD DE OBSERVACIÓN**	**SISTEMA CATEGORIAL**
Desarrollar un algoritmo que permita agrupar múltiples objetos en una misma diapositiva.	Con el algoritmo se podrá agrupar y visualizar varios objetos en una misma diapositiva.	• Tecnología java 3D.	• Análisis y diseño de algoritmos.

- **PROBLEMA ESPECÍFICO:** Limitación de efectos visuales 2D con respecto a los 3D.

OBJETIVO ESPECÍFICO	HIPÓTESIS ESPECÍFICA	UNIDAD DE OBSERVACIÓN	SISTEMA CATEGORIAL
Desarrollar un componente de software configurable para los efectos tridimensionales	Con el componente de efectos visuales tridimensionales se lograran realizar diapositivas más elegante y llamativas para el público.	• Tecnología java 3D.	• Vince, John. 3D Computer Animation. Addison−Wesley. • Java 3D Programain Daniel Selman • http://java.sun.com/products/java-media/3d.

➢ **PROBLEMA ESPECÍFICO:** Limitación en la reproducción de sonido en diferentes formatos.			
OBJETIVO ESPECÍFICO	**HIPÓTESIS ESPECÍFICA**	**UNIDAD DE OBSERVACIÓN**	**SISTEMA CATEGORIAL**
Desarrollar un componente de software para el sonido.	Con el desarrollo de un componente de sonido el usuario podrá agregar sonido de formatos diferentes, en las diapositivas.	• Motor de reproducción de sonido mplayer.	• API mplayer.

- **PROBLEMA ESPECÍFICO:** Interfaces de usuario complejas.

OBJETIVO ESPECÍFICO	HIPÓTESIS ESPECÍFICA	UNIDAD DE OBSERVACIÓN	SISTEMA CATEGORIAL
Desarrollar un árbol de navegación para los objetos tridimensionales agregados a la diapositiva	Con el desarrollo del árbol de navegación el usuario podrá tener un mejor control de los objetos que utiliza para desarrollar la diapositiva.	• Tecnología Java	• API de Java.

10.2. MATRIZ DE OPERATIVIDAD DE LOS OBJETIVOS ESPECÍFICOS

➢ **OBJETIVO ESPECÍFICO:** Desarrollar una galería básica de objetos tridimensionales del mundo real.						
ACTIVIDAD O TAREA	**METODOLOGÍA**	**FECHA**		**RESPONSABLES**	**PRESUPUESTO**	**RESULTADOS ESPERADOS**
		INICIO	**FINAL**			
Análisis Diseño Codificación	• ICONIX	01/10/2009	28/11/2009	Alfredo Macas Klever Macas	$ 400	Galería básica de objetos tridimensionales.

➢ **OBJETIVO ESPECÍFICO:** Desarrollar un asistente de software configurable para la creación de diapositivas tridimensionales.						
ACTIVIDAD O TAREA	**METODOLOGÍA**	**FECHA**		**RESPONSABLES**	**PRESUPUESTO**	**RESULTADOS ESPERADOS**
		INICIO	**FINAL**			
Análisis Diseño Codificación	• ICONIX	01/12/2009	25/01/2010	Alfredo Macas Klever Macas	$ 800	Asistente de software.

➢ **OBJETIVO ESPECÍFICO:** Desarrollar un algoritmo que permita agrupar múltiples objetos en una misma diapositiva.						
ACTIVIDAD O TAREA	**METODOLOGÍA**	**FECHA**		**RESPONSABLES**	**PRESUPUESTO**	**RESULTADOS ESPERADOS**
		INICIO	**FINAL**			
Análisis Diseño Codificación	• ICONIX	01/02/2010	01/03/2010	Alfredo Macas Klever Macas	$ 500	Algoritmo para manipulación de varios objetos.

➢ **OBJETIVO ESPECÍFICO** Desarrollar un componente de software configurable para los efectos tridimensionales.						
ACTIVIDAD O TAREA	**METODOLOGÍA**	**FECHA**		**RESPONSABLES**	**PRESUPUESTO**	**RESULTADOS ESPERADOS**
		INICIO	**FINAL**			
Análisis Diseño Codificación	• ICONIX	05/03/2010	01/06/2010	Alfredo Macas Klever Macas	$ 400	Componente de efectos tridimensionales.

➢ **OBJETIVO ESPECÍFICO:** Integrar un componente de software para el sonido.						
ACTIVIDAD O TAREA	**METODOLOGÍA**	**FECHA**		**RESPONSABLES**	**PRESUPUESTO**	**RESULTADOS ESPERADOS**
		INICIO	**FINAL**			
Análisis Integración	• ICONIX	05/06/2010	25/06/2010	Alfredo Macas Klever Macas	$ 400	Componente de sonido integrado.

➢ **OBJETIVO ESPECÍFICO:** Desarrollar un árbol de navegación para los objetos tridimensionales agregados a la diapositiva.						
ACTIVIDAD O TAREA	**METODOLOGÍA**	**FECHA**		**RESPONSABLES**	**PRESUPUESTO**	**RESULTADOS ESPERADOS**
		INICIO	**FINAL**			
Análisis Diseño Codificación.	ICONIX	01/07/2010	01/09/2010	Alfredo Macas Klever Macas	$ 400	Árbol de navegación.

➢ **OBJETIVO ESPECÍFICO:** Implantación y liberación del software.						
ACTIVIDAD O TAREA	**METODOLOGÍA**	**FECHA**		**RESPONSABLES**	**PRESUPUESTO**	**RESULTADOS ESPERADOS**
		INICIO	**FINAL**			
-Implantar y liberar el código fuente del Software -Capacitación de Usuarios	Enseñanza/ Aprendizaje	15/09/2010	05/10/2010	Alfredo Macas Klever Macas	$ 400	Aceptación por parte de los nuevos usuarios del software.

10.3. MATRIZ DE CONTROL DE RESULTADOS

N	Resultados	Fecha	Docente
1	Galería básica de objetos tridimensionales.	28/11/2009	
2	Asistente de software.	25/01/2010	
3	Algoritmo para manipulación de varios objetos.	01/03/2010	
4	Componente de efectos tridimensionales.	01/06/2010	
5	Componente de sonido integrado.	25/06/2010	
6	Árbol de navegación.	01/09/2010	
7	Aceptación por parte de los nuevos usuarios del software.	05/10/2010	