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

# Título: Análisis y Diseño de una herramienta para la creación, edición y visualización de Centros Expositivos Virtuales.

**Autor:** Lisandra Zaylín Rado Naranjo.

**Tutores:** MSc. Lidiexy Alonso Hernández.
Ing. Minardo Gollún González López.

**Cotutor:** MSc. Manuel Villanueva Betancourt.

**Ciudad de la Habana**

**Junio 2011**

DECLARACIÓN DE AUTORÍA

Declaro que soy la única autora de este trabajo y autorizo al tutor Minardo Gollún González López y al Centro de Informática Industrial (CEDIN), de la Universidad de las Ciencias Informáticas, para que hagan el uso que estimen pertinente con este trabajo.

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

**Lisandra Zaylín Rado Naranjo.** _____
Firma del Autor

**MSc. Lidiexy Alonso Hernández.** _____
Firma del Tutor

**Ing. Minardo Gollún González López.** _____
Firma del Tutor

**MSc. Manuel Villanueva Betancourt.** _____
Firma del Tutor

# Datos de Contacto

Nombre y Apellidos: Minardo Gollún González López

Edad: 28 años

Ciudadanía: cubano

Institución: Universidad de las Ciencias Informáticas (UCI)

Título: Ingeniero en Ciencias Informáticas

Categoría Docente: Instructor

E-mail: mgonzalezl@uci.cu

Graduado de la UCI, con cinco años de experiencia en el tema de Realidad virtual, y jefe del proyecto Paseos Virtuales en el Centro de Desarrollo de Informática Industrial.

# *Agradecimientos:*

*A toda mi familia y amistades, por haber puesto su granito de arena que ha permitido ser la persona que soy hoy, por transmitirme buenos consejos, apoyo incondicional y dedicación.*

*A mi mamá por su dedicación, comprensión y apoyo, por sus consejos, por confiar en mí, darme ánimo y ayudarme a seguir mis sueños.*

*A mi abuela Isabel por acogerme como una hija, por sus consejos, apoyo y dedicación, por todo su empeño y por ayudar a construir mis sueños.*

*A mi papá por su apoyo y dedicación.*

*A mi tía Rita por estar siempre presente en cada momento que la he necesitado, por hacerme sentir como una hija, por todo su cariño y apoyo.*

*A mis hermanos por su cariño.*

*A mis tutores por toda la ayuda brindada.*

*A mi novio quien ha sido mi tutor durante mis 5 años de estudio universitario y quien se ha comportado como mi compañero, mi hermano, mi padre, madre y abuela, por su compañía y apoyo durante este tiempo, por comprenderme en momentos difíciles y exigirme éxito en mis estudios.*

# *Dedicatoria:*

*A mi mamá por desear tanto como yo este momento.*

*A mi abuela Isabel por haber puesto en mi todo su empeño.*

*A mis hermanos que sirva de ejemplo y vean que todo lo que uno se propone lo logra.*

# *Resumen:*

Desde el año 2002 en la Universidad de las Ciencias Informáticas el Grupo de Desarrollo de Paseos Virtuales (GDPV) ha desarrollado paseos virtuales en tres dimensiones y ha identificado la carencia de una herramienta especializada en la creación, edición y visualización de paseos virtuales como un problema que afecta directamente el tiempo de entrega de sus productos.

En el presente trabajo se desarrolla una propuesta de diseño de una herramienta para la creación, edición y visualización de Centros Expositivos Virtuales. En el cuerpo del documento podrá encontrar un estudio exhaustivo del estado del arte donde se introducen conceptos básicos asociados a los paseos virtuales. Se presenta una descripción general del proceso actual de realización de paseos virtuales, se definen y especifican las funcionalidades que debe cumplir la herramienta y se modela el sistema. Se muestran los resultados de aplicar varias disciplinas de la ingeniería del software como la ingeniería de requisitos, el análisis y el diseño, entre otras, que en su totalidad arrojan todos los artefactos necesarios para conformar una propuesta completa de solución que permiten la implementación de la herramienta.

**Palabras clave:** Exposiciones virtuales, ingeniería del software, paseos virtuales, realidad virtual.

## Índice:

## Índice de Tablas:



## Índice de Figuras:

# Introducción

El surgimiento y desarrollo de las nuevas Tecnologías de la Información y la Comunicación (TIC) ha invadido todo el universo informativo en la actualidad. Dentro de ellas, la realidad virtual es una de las ramas de la informática, que ha despertado un alto interés por lograr digitalizar en detalle casi todo mundo real.

En la actualidad una gran variedad de industrias utilizan la realidad virtual para muchos fines, entre los cuales está ayudar a comercializar sus servicios y productos. Una de las aplicaciones de la realidad virtual con abundante mercado en el mundo son los Paseos Virtuales[1]. En Cuba se han creado paseos virtuales, pero se realizan mediante el uso de videos o vistas 360. En la Universidad de las Ciencias Informáticas (UCI) desde el año 2002 se creó el GDPV, donde se ha ganado experiencia en la realización de aplicaciones de este tipo. Desde sus inicios el Grupo de Desarrollo ha realizado pequeños paseos virtuales usando muchas de las herramientas más populares en estas labores y se ha podido identificar sus ventajas y desventajas.

Durante el proceso de desarrollo para obtener un paseo virtual se realizan un conjunto de actividades como: el Análisis y Diseño de Software, Implementación, entre otras, que dan lugar a un sistema que recrea un escenario virtual representado en video, vistas 360 o de forma interactiva[2]. Cuando se concluye la etapa de implementación de un paseo virtual si el equipo de desarrollo necesita modificar los recorridos, la posición de los modelos o realizar algún otro cambio en el escenario se debe realizar nuevamente todo el proceso de implementación lo cual afecta el tiempo total necesario para la entrega del producto.

Las visitas o paseos virtuales tienen gran aceptación por el público, pero debido al alto costo de su desarrollo se limita el uso de estas técnicas a entidades que posean las facilidades económicas para financiar el gasto que representa cada exposición. Las entidades responsables de las áreas expositivas cubanas no se encuentran exentas de esta situación, adicionado al déficit económico, el bloqueo económico norteamericano al país limita aún más la capacidad de contratación de algún servicio o adquisición de productos de software para el desarrollo de áreas expositivas virtuales.

[1] Paseos Virtuales: simulación de un entorno empleando tecnología de realidad virtual (ver: Glosario de Términos).

[2] Interactiva: Se refiere a que el usuario tenga la posibilidad de desplazarse por el escenario y obtener información de los objetos que conforman la escena.

Partiendo de la situación antes expuesta se plantea el siguiente **problema de investigación** a resolver: ¿Cómo obtener la especificación y documentación que permita implementar una herramienta informática para la creación, edición y visualización de Centros Expositivos Virtuales interactivos?

Luego, a partir del problema formulado, se define como **objeto de estudio**: El análisis y diseño de herramientas informáticas para realidad virtual.

Para contribuir a la solución del problema planteado se define como: **objetivo general** el siguiente: Diseñar una herramienta para la creación, edición y visualización de Centros Expositivos Virtuales interactivos.

Por tanto, el **Campo de acción** será: El análisis y diseño de herramientas informáticas para la manipulación de Escenas Virtuales y la creación de Paseos Virtuales interactivos.

Para darle cumplimiento a los objetivos propuestos se precisan las siguientes **tareas de investigación** a cumplir:

- Elaboración del marco teórico a partir del estado del arte relacionado con el diseño de herramientas de realidad virtual.
- Caracterización de las herramientas de manejo de escenas 3D y diseño de Paseos Virtuales.
- Identificación de las herramientas y lenguajes para el desarrollo de la aplicación.
- Identificación de los requisitos del software.
- Diseño de Prototipo de interfaz.
- Elaboración de los modelos del Análisis y Diseño del software.
- Evaluación de la propuesta de solución.

**Idea a defender:**

El análisis y diseño dotará al GDPV con la documentación y especificaciones necesarias que le permita la implementación de una herramienta para la creación, edición y visualización de Centros Expositivos Virtuales interactivo.

En el progreso de la investigación científica se emplearon los siguientes métodos:

**Métodos del nivel teórico:**

- Método histórico lógico se empleó para realizar un estudio valorativo de la evolución de los diferentes lenguajes, metodologías y software posibles a utilizar en el desarrollo de la herramienta y de sus funcionalidades.

- La inducción-deducción es utilizado en la investigación para sintetizar información de interés relacionada con el trabajo que se presenta.
- El analítico–sintético se utilizó al analizar toda la información relacionada con el tema de tesis, para extraer los elementos más importantes de cada documento consultado.

**Métodos del nivel empírico:**

- Análisis de todas las fuentes de información relacionadas con el tema.
- La entrevista para obtener información sobre las herramientas de diseño más utilizadas, procesos de desarrollos, entre otros.

**Posibles resultados:**

A partir de los resultados de la investigación realizada se espera obtener el análisis y diseño de una herramienta para la creación y edición de Centros Expositivos Virtuales interactivos.

**Estructura del trabajo**

El contenido del trabajo está estructurado de la siguiente forma.

**Capítulo 1. Fundamentación teórica.**

En el presente capítulo, mediante un estudio del arte, se introducen conceptos básicos asociados a los paseos virtuales y aspectos relacionados con la problemática planteada en la introducción del trabajo. Se caracterizan las principales metodologías de desarrollo de software existentes pretendiendo dejar sentadas las bases teóricas para un correcto análisis y diseño del software. Se presentan lenguajes y herramientas que pudieran ser útiles para el desarrollo de la investigación.

**Capítulo 2. Características del sistema.**

En el presente capítulo se muestra el proceso actual para el desarrollo de un paseo virtual y se describe todo el funcionamiento del sistema que se propone. Se presenta el modelado de negocio y se especifica tanto los requisitos funcionales como los no funcionales. Son seleccionadas las herramientas, el lenguaje y la metodología a utilizar en el desarrollo del sistema.

**Capítulo 3. Análisis y diseño del sistema.**

Se define el modelo de análisis y diseño de la herramienta propuesta, se modela el diagrama de clases de análisis y del diseño y los diagramas de interacción. Se Diseña el prototipo de interfaz de la herramienta.

# Capítulo 1. Fundamentación teórica.

## Introducción

En el presente capítulo, mediante un estudio del arte, se introducen conceptos básicos asociados a los paseos virtuales y aspectos relacionados con la problemática planteada en la introducción del trabajo. Se caracterizan las principales metodologías, lenguajes y herramientas de desarrollo de software existentes, pretendiendo dejar sentadas las bases teóricas para un correcto análisis y diseño del software.

## 1.1 Realidad virtual.

Numerosas son las definiciones atribuidas al término Realidad Virtual, quizás tantas como el número de autores que se han acercado al tema. Al realizar una búsqueda del significado de dicha expresión se puede apreciar en los resultados obtenidos que existe similitud en lo que hablan los autores al respecto. A continuación son citadas algunas definiciones.

*"Una simulación por ordenador de un sistema real o imaginario que permite a un usuario realizar operaciones en el sistema simulado y muestra los efectos en tiempo real".*

*"Una forma de interacción persona-ordenador en el que un entorno real o imaginario es simulado y los usuarios pueden interactuar y manipular ese mundo. Los usuarios viajan en el mundo simulado moviéndose hacia donde quiera estar, e interactuar y manipular objetos simulados. En los entornos virtuales de mayor éxito, los usuarios sienten que están realmente presentes en el mundo simulado y que su experiencia en el mundo virtual coincide con lo que iba a experimentar en el medio ambiente que se simula…"* (Answer, 2009).

El autor considera que: Un sistema para poder ser considerado de realidad virtual debe ser capaz, en determinado grado, de estimular y engañar los sentidos a los que se dirige.

La realidad virtual es posible clasificarla en dos tipos dependiendo de la participación del usuario:

- Realidad Virtual Inmersivas, permite que las personas se perciban dentro del entorno virtual tridimensional generado artificialmente, estimulando varios de los sentidos mediante el uso de varios dispositivos.
- Realidad Virtual no Inmersivas, también conocida como Realidad Virtual de Escritorio, se dedica fundamentalmente a estimular o engañar el sentido de la vista mediante una computadora, un mouse (ratón) y un teclado, es una de las formas de realidad virtual más económica.

La presente investigación se centrará en el campo de la realidad virtual que se ocupa de estimular y engañar el sentido visual mediante la representación digital de un entorno tridimensional donde el usuario pueda interactuar intuitivamente y en tiempo real[3] con los objetos que encuentre dentro de él.

### 1.1.1 Aplicaciones de la realidad virtual.

Las aplicaciones de la realidad virtual que existen en la actualidad sobre actividades de la vida cotidiana son diversas: permite conocer elementos que de otro modo no estarían al alcance de las personas: recorrer las venas y arterias del cuerpo humano como en microsubmarino, realizar cirugías, recreación de edificios y sitios de interés histórico o artístico, diseño de productos y maquinaria, simuladores aéreos, terrestres y marinos. Aquí es posible mezclar los conocimientos adquiridos con la fantasía y poner a prueba hipótesis sin el riesgo de destruir objetos o entornos delicados.

## 1.2 Paseos Virtuales.

Los Paseos Virtuales son unas de las aplicaciones de la realidad virtual con abundante mercado en todo el mundo, básicamente son la simulación de un entorno empleando tecnología de realidad virtual, pueden ser realizados a áreas existentes con determinado grado de realismo visual, permiten el enriquecimiento de la escena e incluso la representación de un área totalmente inexistente.

### 1.2.1 Aplicaciones de los paseos virtuales.

La creación de un paseo virtual puede realizarse con muchos objetivos que pueden ser atractivos para una extensa masa de personas. La posibilidad de conocer un lugar sin la necesidad de desplazarse más allá de su escritorio es muy tentativa y útil. Reproducir en la memoria de una computadora el recorrido de una pirámide egipcia o cavernas prehistóricas cuya visita constituye un riesgo para su conservación, o un paseo por sitios de interés histórico o artístico son algunas de las aplicaciones de los paseos virtuales que se pueden mencionar. En la actualidad una gran variedad de industrias utilizan esta tecnología para ayudar a comercializar sus servicios y productos. En los últimos años, la calidad, usabilidad y accesibilidad de los recorridos virtuales ha mejorado considerablemente, con algunos sitios web que permiten al usuario navegar por los viajes haciendo clic en los mapas o los planos integrados de suelo.

[3] Tiempo Real se refiere a la representación en pantalla de las imágenes generadas con una frecuencia no menor a 30 imágenes por segundo.

## 1.3 Centros expositivos virtuales.

Los centros expositivos virtuales son un tipo de paseo virtual que recrean lugares que en su interior exponen objetos de gran valor abiertos al servicio y al desarrollo del conocimiento artístico, histórico, estético, natural y social de la población. Los Museos, Galerías de Artes y otros son espacios al servicio de la sociedad que adquieren, conservan, investigan, comunican y exponen o exhiben, con propósitos de estudio, educación y deleite, colecciones de arte, científicas, objetos valiosos, siempre con un valor cultural o patrimonial.

## 1.4 Lenguajes de programación y modelado.

### 1.4.1 Lenguajes de programación. C++ y Python.

Un lenguaje de programación es un idioma artificial diseñado para expresar automatizaciones que pueden ser llevadas a cabo por máquinas como las computadoras. Es un modo práctico para que los seres humanos puedan dar instrucciones a un equipo. Con la utilización de los lenguajes de programación es posible crear programas que controlen el comportamiento físico y lógico de una máquina y expresar algoritmos con precisión. Está formado por un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones.

- **C++.**

Es un lenguaje de programación estandarizado por la norma ISO/IEC 14882:1998. Se trata de un lenguaje compilado. Entre sus principales características está el soporte para la programación orientada a objetos y el soporte de plantillas o programación genérica, además de ser un lenguaje de alto nivel que está considerado como un lenguaje potente al poder trabajar tanto en bajo, como en alto nivel. Posee dos propiedades fundamentales que son difíciles de encontrar en otros lenguajes que son la posibilidad de redefinir los operadores, comúnmente conocido como la sobrecarga de operadores, y la identificación de tipos en tiempo de ejecución. Además presenta una biblioteca estándar muy poderosa. Esta creación de Bjarne Stroustrup[4] es muy usada en el ámbito educacional y profesional, siendo uno de los lenguajes más

[4] Bjarne Stroustrup es un científico en computación y Catedrático de Ciencias de la Computación en la Universidad A&M de Texas. Es reconocido mundialmente por ser el creador del lenguaje de programación C++. Miércoles 25 de

populares en la implementación de gráficos por computadoras. (Seta, 2010). C++ es, con mucha diferencia, el lenguaje más popular, seguido a larga distancia de Java y Python. (González Duque, 2011)

- **Python.**

Python es un lenguaje de programación creado por Guido Van Rossum a principios de los años 90 cuyo nombre está inspirado en el grupo de cómicos ingleses "Monty Python". Es un lenguaje similar a Perl, pero con una sintaxis muy limpia y que favorece un código legible. Se trata de un lenguaje interpretado o de script, con tipado dinámico, fuertemente tipado, multiparadigma y orientado a objetos. Su sintaxis simple, clara y sencilla; el tipado dinámico, el gestor de memoria, la gran cantidad de librerías disponibles y la potencia del lenguaje, entre otros, hacen que desarrollar una aplicación en Python sea sencillo y muy rápido. Python no es adecuado sin embargo para la programación de bajo nivel o para aplicaciones en las que el rendimiento sea crítico. (González Duque, 2008).

### 1.4.2 Lenguaje de modelado. UML.

Desde el momento en que se concibe un software es necesario establecer comunicación entre los involucrados con el producto de forma tal que el equipo obtenga la información que les será útil durante el proceso de desarrollo del software. Con la utilización del lenguaje de modelado es posible representar un sistema que será objeto de automatización o ya automatizado y describir las actividades que se deben realizar para construir un determinado producto.

- **UML.**

El Lenguaje Unificado de Modelado por sus siglas en inglés (UML) es la especificación más utilizada por OMG [5](OMG, 2011). Es un lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de software. Se usa para entender, diseñar, hojear, configurar, mantener, y controlar la información sobre tales sistemas. Está pensado para usarse con todos los métodos de desarrollo, etapas del ciclo de vida, dominios de aplicación y medios. Entre sus principales ventajas está la de unificar notaciones previas. UML ayuda al usuario a entender la realidad de la

---

Junio de 2008. Visitar [http://www.dosideas.com/noticias/actualidad/109-entrevista-a-bjarne-stroustrup-creador-de-c.html].

[5] OMG ® (Object Management Group) es una organización internacional, abierta, sin fines de lucro de la industria informática. OMG desarrolla estándares de modelado e integra una amplia gama de tecnologías. [Disponible en: www.omg.org].

tecnología y la posibilidad de que reflexione antes de invertir y gastar grandes cantidades en proyectos que no estén seguros en su desarrollo, reduciendo el coste y el tiempo empleado en la construcción de las piezas que constituirán el modelo.

**Algunas de las propiedades de UML como lenguaje de modelado estándar son:**

- Concurrencia, es un lenguaje distribuido y adecuado a las necesidades de conectividad actuales y futuras.
- Ampliamente utilizado por la industria desde su adopción por OMG.
- Reemplaza a decenas de notaciones empleadas con otros lenguajes.
- Modela estructuras complejas.
- Las estructuras más importantes que soportan tienen su fundamento en las tecnologías orientadas a objetos, tales como objetos, clase, componentes y nodos.
- Emplea operaciones abstractas como guía para variaciones futuras, añadiendo variables si es necesario.
- Comportamiento del sistema: casos de uso, diagramas de secuencia y de colaboraciones, que sirven para evaluar el estado de las máquinas (Rumbaugh, y otros, 2000).

## 1.5 Metodologías de desarrollo de software.

Las metodologías de desarrollo de software de forma general son usadas para planificar, estructurar y controlar el proceso de desarrollo de software. Se han definido un gran número de estas en función de las dimensiones de los proyectos en lo que se aplicarán y otras muchas variables que han conllevado a un desarrollo constante y fructífero, actualmente existen muchas como Scrum o Modelo de Diagnóstico Común (Common Diagnostic Model, CDM), entre las más utilizadas se encuentran las que se describen en los siguientes epígrafes.

### 1.5.1 Programación Extrema (XP).

Se inició el 6 de marzo 1996. Programación Extrema es uno de los más populares procesos ágiles de desarrollo de software. Su mayor propósito es la satisfacción del cliente. Programación Extrema faculta a los desarrolladores para responder con seguridad a las necesidades cambiantes de los clientes, incluso a finales del ciclo de vida. (Wells, 2009). Es una disciplina de desarrollo de software basado en los valores de la simplicidad, la comunicación y la retroalimentación. Su acción consiste en llevar todo el equipo junto

con la presencia de prácticas sencillas, con suficiente información para que el equipo pueda ver dónde están y para ajustar las prácticas a su situación particular. (Ron, 2001). Una de las características que destaca a esta metodología es que el volumen de información que describe es mucho menor que la generada por las metodologías pesadas.

### 1.5.2 RUP.

El Proceso Unificado de Desarrollo (RUP) es un proceso de desarrollo de software y a la vez, es un conjunto de actividades dirigidas a transformar los requerimientos del cliente en un sistema de software. Aumenta la productividad del equipo de desarrollo, permitiendo a cada miembro compartir un lenguaje común (UML[6]), un proceso y una vista de cómo revelar el software. RUP es un proceso configurable y es soportado por herramientas que automatizan partes grandes del proceso. Genera una amplia documentación de los elementos que la componen. En su modelación define como sus principales elementos a los trabajadores, actividades, artefactos y el flujo de actividades.

**Principios de RUP.**

- Adaptar el proceso.
- Equilibrar las prioridades de los interesados que están enfrentadas.
- Colaborar con otros equipos.
- Demostrar el valor de forma iterativa.
- Elevar el nivel de abstracción.
- Centrarse continuamente en la calidad.

**El ciclo de vida de RUP.**

La vida de un sistema transcurre a través de ciclos de desarrollo, desde que comienza hasta que termina, en cada ciclo se repite el proceso unificado de desarrollo. Cada uno consta de cuatro fases (Inicio, elaboración, construcción, transición) y concluye con una versión del producto. Cada fase se subdivide en iteraciones. Una iteración es una secuencia de actividades con un plan establecido y criterios de evaluación, cuyo resultado es una versión del software.

**El ciclo de vida de RUP se caracteriza por ser:**

- Dirigido por casos de uso.
- Centrado en la arquitectura.

[6] Lenguaje Unificado de Modelado.

- Iterativo e Incremental. (IBM, 2009).

## 1.6 Herramientas de diseño de software.

Uno de los elementos a tener en cuenta en la elaboración de un software es la organización y el cumplimiento de las tareas de forma correcta y eficiente. Las Herramientas CASE (Computer Aided Software Engineering, en español Ingeniería de Software Asistida por Computadora) fueron desarrolladas para automatizar dichos procesos y facilitar las tareas de coordinación de los eventos que necesitan ser mejorados en el ciclo de desarrollo de software. Las Herramientas CASE brindan asistencia al equipo de desarrollo de software durante el ciclo de vida de desarrollo de un software, permiten, entre otras cosas, reducir el tiempo de construcción de aplicaciones y mejorar su calidad, con un menor coste en términos de tiempo y dinero.

### 1.6.1 Rational Rose Enterprise.

Es un producto de IBM [7] que proporciona un lenguaje de modelado común que permite crear en menos tiempo software de calidad. Entre sus principales características se pueden destacar las siguientes:

- Es compatible con el lenguaje UML (Unified Modeling Language) y es uno de los productos más completos de la familia Rational Rose.
- Soporta la ingeniería directa e inversa para algunas de las construcciones más comunes de Java 1.5.
- Es capaz de analizar la calidad del código y de generar código gracias a las capacidades de sincronización configurable entre el modelo y el código.
- Se integra con otras herramientas de desarrollo del ciclo vital de IBM Rational.
- Sistemas operativos admitidos: Windows (IBM, 2011).

### 1.6.2 Visual Paradigm para UML.

Visual Paradigm para UML (VP-UML) es una herramienta de diseño UML y herramienta CASE diseñada para la ayuda al desarrollo de software. VP-UML soporta estándares de la industria clave, tales como

[7] **International Business Machines** (**IBM**), conocida coloquialmente como **el Gigante Azul,** es una empresa multinacional estadounidense que fabrica y comercializa herramientas, programas y servicios relacionados con la informática. IBM tiene su sede en Armonk (Nueva York, Estados Unidos) y está constituida desde el 15 de junio de 1911, pero lleva operando desde 1888.

Lenguaje de Modelado Unificado (UML), SysML, BPMN, XMI, ofrece un completo conjunto de herramientas de los equipos de desarrollo de software necesario para la captura de requisitos, la planificación de programas, la planificación de controles, la clase de modelado, modelado de datos entre otros. (Visual Paradigm, 2011)

**Lista de características:**

- Soporte de UML versión 2.1.
- Diagramas de Procesos de Negocio - Proceso, Decisión, Actor de negocio, Documento.
- Ingeniería de ida y vuelta. Ingeniería inversa - Código a modelo, código a diagrama.
- Generación de código - Modelo a código, diagrama a código.
- Editor de Detalles de Casos de Uso - Entorno todo-en-uno para la especificación de los detalles de los casos de uso, incluyendo la especificación del modelo general y de las descripciones de los casos de uso.
- Diagramas de flujo de datos.
- Generador de informes para generación de documentación.
- Distribución automática de diagramas - Reorganización de las figuras y conectores de los diagramas UML.
- Importación y exportación de ficheros XMI.
- Integración con Visio - Dibujo de diagramas UML con plantillas (stencils) de MS Visio.
- Editor de figuras.

## 1.7 Entorno integrado de desarrollo (IDE). QT Creator.

Un IDE es un entorno de programación que ha sido empaquetado como un programa de aplicación. Consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica (GUI). En el epígrafe se presentan las principales características del QT Creator, por ser elegido por el GDPV para quienes se realiza la presente investigación.

Qt Creator es un IDE multi-plataforma, puede ejecutarse sobre Windows, Linux/X11 y Mac OS. Entre otras características cuenta con:

- Editor de Código sofisticado que ofrece soporte para la edición de C + + y QML (JavaScript),

- Puede integrarse con la mayoría de los sistemas de control de versiones, incluyendo Git, Subversion, Perforce, CVS y Mercurial.
- Ofrece dos editores visuales integrados: Qt Designer para la construcción de interfaces de usuario usando los Qt Widgets[8], y Qt Quick Designer para el desarrollo de interfaces de usuario con el lenguaje QML. (Nokia, 2008)

## 1.8 Motor Gráfico. Object Oriented Graphics Engine (Ogre).

En el epígrafe se presentan las principales características del motor gráfico Ogre, por ser elegido por el GDPV para quienes se realiza la presente investigación.El motor gráfico es el que proporciona funciones de renderizado 2D, 3D y de sprites[9]. Se suele apoyar en bibliotecas/APIs [10]gráficas como OpenGL o DirectX.

OGRE 3D es un motor de gráficos 3D de código abierto. Es flexible y orientado a objetos, además de ser portable a múltiples entornos como Windows, MacOS X o Linux. Ha sido diseñado con la idea de hacer más sencillo el desarrollo de juegos 3D. Explota al máximo las posibilidades de las tarjetas gráficas. El motor está muy extendido y es muy valorado por la comunidad de código abierto. Con él se han desarrollado muchos juegos, algunos de ellos con carácter comercial. (Tirado Granados, 2008)

## 1.9 Herramientas de diseño gráfico.

Con el desarrollo de este epígrafe no se pretende realizar una comparación entre las herramientas de diseño gráfico, sino comprender las principales funcionalidades y características comunes que pueden ser útiles durante la investigación. Las herramientas de diseño gráfico permiten realizar la modelación y manipulación de objetos y entornos virtuales, brindan un conjunto de funciones que facilitan el trabajo con escenarios virtuales y algunas posibilitan la creación de paseos virtuales. El conocimiento de estas herramientas para la investigación que se realiza es muy importante ya que permitirá identificar elementos que pudieran ser útiles para realizar la propuesta de solución.

[8] Qt Widgets: Se refiere a los componentes visuales que brinda el IDE.

[9] Sprites: Tipo de mapa de bits dibujados en la pantalla de ordenador por hardware gráfico especializado.

[10] APIs: Interfaz de programación de aplicaciones.

### 1.9.1 SketchUp Pro.

Es un software de modelado 3D potente e intuitivo, que permite a los diseñadores explorar, comunicar y presentar complejos conceptos de diseño. El sencillo y potente conjunto de herramientas y el sistema de dibujo inteligente facilitan la creación y manipulación de modelos, mientras que las funciones de importación/exportación te ofrecen la flexibilidad y funcionalidad necesarias para el proceso de trabajo profesional (3D Profesional, 2011).

### 1.9.2 Autodesk 3ds Max y Autodesk 3ds Max Design.

Herramienta de diseño gráfico que proporcionan potentes funciones integradas de modelado, animación, renderización y composición en 3D que multiplican rápidamente la productividad de los artistas y diseñadores. Aunque ambas versiones comparten la tecnología y la funcionalidad básicas, una ofrece herramientas y experiencias específicas a los desarrolladores de juegos, realizadores de efectos visuales y diseñadores gráficos, mientras que la otra tiene características especializadas para los arquitectos, diseñadores, ingenieros y especialistas en visualización (3D Profesional, 2011)**.**

### 1.9.3 Autodesk Maya

Es una herramienta de diseño gráfico que ofrece a los artistas un flujo de trabajo creativo completo, con todas las herramientas para realizar animación, modelado, simulación, efectos visuales, renderización, rastreo de movimiento y composición en 3D dentro de una plataforma de producción sumamente ampliable. Toda esta funcionalidad se reúne en una única aplicación que aporta un valor excepcional a los artistas gráficos. Ahora disponible para el sistema operativo Mac OS X de 64 bits, Maya 2011 proporciona una interfaz de usuario revitalizada (3D Profesional, 2011)**.**

### 1.9.4 Blender

Es la principal suite en código abierto para la creación de contenido 3D, disponible para los principales sistemas operativos bajo la General Public License (GNU) Blender Fundation, 2011. Blender es una suite completamente integrada multi-plataformas para la creación de gráficos 3D, permite el modelado, animación de los modelos, además de la creación y reproducción de video juegos; todo esto en un paquete fácil de descargar y gratis, además de estar disponible para las plataformas: Windows, Linux, Irix, Sun Solaris, FreeBSD o Mac OS X.

**Entre las ventajas del Blender como programa para el modelado 3D están:**

- Posee muchas herramientas que pueden ser usadas mediante un atajo de teclado, disminuyendo el tiempo de realización.
- Posee la mayoría de las herramientas existentes para volver el Modelado 3D. Ejemplo de esto son las herramientas: Subdividir, Extruir, Rotar, Escalar, Mover, Suavizar, Girar, entre muchas otras.
- Cuenta con una importante serie de modificadores para lograr efectos deseados en los modelos. (Bevel, Array, Boolean, Curve, Displacement
- A diferencia de muchos programas privativos como el 3D Max y el Maya, posee una poderosa herramienta de Esculpir.
- Permite trabajar cómodamente con modelos a bajo polígonos, optimizándolos así, para su posterior utilización en la creación de aplicaciones gráficas en tiempo real, como paseos virtuales y video juegos. Blender trabaja con polígonos de 3 o 4 vértices, es decir con triángulos y cuadriláteros. (González López, 2011)

## 1.10 Herramientas para realizar trazabilidad de requisitos.

La administración de requisitos es una parte esencial para controlar la complejidad, riesgo, alcance del proyecto, y definir los roles y criterios para un software o un proyecto de negocio exitoso. Las herramientas para realizar trazabilidad de requisitos permiten modelar requisitos y proveer una trazabilidad completa de cada requisito hasta los entregables finales y el comportamiento del sistema.

### 1.10.1 Rational RequisitePro

IBM Rational RequisitePro es una herramienta de gestión de requisitos que permite gestionar los requisitos de los equipos de proyecto, escribir buenos casos prácticos, mejorar la trazabilidad, reforzar la colaboración, reducir las tareas de remodelación de los proyectos y aumentar la calidad.

**Características de la herramienta.**

- Evita repeticiones y duplicaciones gracias a la integración avanzada en tiempo real con Microsoft Word.
- Gestiona la complejidad con vistas de rastreabilidad detalladas que muestran relaciones padre-hijo.
- Reduce los riesgos de los proyectos con la visualización de requisitos que pueden verse afectados por cambios en los requisitos en sentido ascendente o descendente.

- Logra la colaboración de equipos distribuidos geográficamente a través de una interfaz web escalable totalmente funcional e hilos de debate.
- Captura y analiza información de requisitos con personalización y filtrado detallado de atributos.
- Mejora la productividad haciendo un seguimiento de los cambios mediante comparaciones de las versiones del proyecto con líneas base de proyecto basadas en XML.
- Ajusta los objetivos empresariales con los productos finales del proyecto mediante la integración con varias herramientas en la plataforma de desarrollo y distribución de software de IBM Rational.
- Sistemas operativos admitidos: Windows. (IBM, 2011)

### 1.10.2 Open Source Requirements Management Tool (OSRMT).

Es una herramienta de Software Libre pensada para asistir en todo el Ciclo de Vida del Desarrollo del Software. Permite la descripción avanzada de diversos tipos de requisitos y garantiza la trazabilidad entre todos los documentos relacionados con la ingeniería de requisitos (funcionalidades, requisitos, casos de uso, casos de prueba). La herramienta integra módulos de Administración y Configuración, Gestión de Documentos de la Ingeniería de Requisitos, Trazabilidad entre documentos de trabajo e Informes y estadísticas.

Además de las funcionalidades ya mencionadas, este sistema provee:

- Gestión de la configuración: versionado y registro de los cambios realizados en los diferentes elementos.
- Gestión de usuarios y permisos.
- Herramientas de migración para los diversos cambios de versiones.
- Múltiples idiomas (importación y exportación para dar soporte a diversos idiomas).
- Importar y exportar información en XML y mediante línea de comandos.
- Exportar información en HTML mediante línea de comandos.
- Informes:
    - Básicos.
    - Específicos creados por el usuario.
    - A partir de los resultados de búsquedas avanzadas.
    - Exportados a HTML PDF.

## 1.11 El análisis y diseño de un software.

Como parte del proceso de desarrollo de un software inicialmente se define los objetivos y el alcance del proyecto, obteniéndose como uno de los principales resultados un listado de casos de uso y una lista con los factores de riesgo del proyecto. Se presenta una visión general de los procesos que existen en el negocio a sistematizar para comprender a que se dedica el mismo, así como establecer una comunicación entre el cliente y el equipo de desarrollo. Una vez que es alcanzado este punto, se define qué es lo que debe hacer el sistema, mediante la captura de los requisitos que este debe cumplir, se obtiene una vista externa del sistema, que en el lenguaje del cliente, describe lo que se espera de él a través del diagrama de casos de uso. A partir de aquí se debe profundizar en los casos de usos detallándolos de manera que permitan reflejar una vista interna del sistema descrita con el lenguaje de los desarrolladores. Es en este momento donde el análisis y diseño del software comienza a jugar un papel fundamental en el proceso de desarrollo.

### 1.11.1 Análisis de un software.

El análisis del software es una tarea dentro de la ingeniería que permite especificar el funcionamiento, los datos y el rendimiento de un determinado programa, indica la interfaz del software con otros elementos del sistema y establece las restricciones que debe cumplir el software. Durante esta etapa se realiza un análisis profundo de los requisitos con el objetivo de obtener una comprensión más precisa y una descripción que sea fácil de mantener y que ayude a estructurar el sistema entero incluyendo su arquitectura. En el análisis se enfatiza en los aspectos internos del sistema y se utiliza el lenguaje de los desarrolladores para describir los resultados. En este período son detallados los casos de usos y se especifican los requisitos con el uso de un lenguaje más formal para lograr refinar los requisitos del sistema. Además son estructurados los requisitos de forma tal que facilite su comprensión, preparación, modificación y en general su mantenimiento. (Pressman, 2001).

**Artefactos del Análisis.**

- *Modelo de análisis.*
- *Clases del análisis.*
- *Realización de casos de uso del análisis.*
- *Paquete del análisis.*
- *Descripción de la arquitectura.*

### 1.11.2 Diseño de software.

El diseño del software es una de las actividades técnicas que se requiere para construir y verificar un software. Cada actividad que se desarrolle durante esta fase transforma la información de manera que dé lugar por último a un software de computadora validado. En el diseño es donde se modela el sistema y se determina su arquitectura de manera que soporte todos los requisitos tanto funcionales como no funcionales. Durante el diseño se desarrollan, revisan y documentan los refinamientos progresivos de la estructura de datos, arquitectura, interfaces y datos procedimentales de los componentes del software. Como resultado de su elaboración se obtiene representaciones del software que permiten evaluar su calidad.

**Artefactos del diseño.**

- *Modelo de diseño.*
- *Clases del diseño.*
- *Realización de los casos de uso del diseño.*
- *Subsistema de diseño.*
- *Interfaz.*
- *Modelo de despliegue.*
- Descripción de la arquitectura. (Jacobson [et al.]., 2000).

## Consideraciones del capítulo.

Con la conclusión del capítulo se logró conformar un compendio de conceptos y caracterizaciones que permiten una mejor comprensión del estado del arte y de los contenidos que se mostrarán en los próximos capítulos del presente trabajo. Se caracterizó la metodología de desarrollo XP y la metodología RUP con el objetivo de seleccionar en el próximo capítulo la apropiada para guiar el proceso de desarrollo de la propuesta de solución. Se profundizó en las principales características del Lenguaje Unificado de Modelado, de los lenguajes de programación C++ y Python y de algunas herramientas CASE, para entender sus funciones y determinar cuál será utilizada en el desarrollo del producto. Se identificaron las principales características del IDE QT Creator y del motor gráfico Ogre con el objetivo de adquirir conocimientos necesarios para enfrentar la el diseño de la propuesta de solución. Se realizó un estudio de las herramientas que existen actualmente y que pueden ser utilizadas para crear paseos virtuales y se identificó que no existe ninguna herramienta con este fin desarrollada en Cuba y a nivel internacional existen algunas que pueden ser utilizadas pero solo permiten la creación de paseos virtuales no interactivos en forma de vistas 360 o video. El análisis de las herramientas de diseño gráfico resumido en epígrafes anteriores, permitirá seleccionar algunas de las funcionalidades que deberá cumplir la herramienta para la creación, edición y visualización de Centros Expositivos Virtuales interactivos.

# Capítulo 2. Características del sistema.

## Introducción

En el presente capítulo se describe el proceso que se realiza actualmente para la obtención de un paseo virtual y se propone una solución. Se presenta el modelado de negocio y se especifica tanto los requisitos funcionales como los no funcionales que debe cumplir el sistema y son definidos los casos de uso. Se modela el diagrama de casos de uso para representar la interacción del usuario con los módulos de la herramienta y se realiza una descripción textual de cada caso de uso. Son seleccionadas las herramientas, el lenguaje y la metodología a utilizar en el desarrollo del sistema.

## 2.1 Proceso actual de realización de un paseo virtual.

Uno de los primeros pasos que se realiza para enfrentar el proceso de desarrollo de un software es entender el funcionamiento de los procesos que existen en el negocio. A partir de este punto se define qué es lo que debe hacer el sistema, mediante la captura de los requisitos que este debe cumplir y se profundiza en los casos de uso detallándolos de manera que permitan reflejar una vista interna del sistema.

Actualmente el GDPV para obtener un paseo virtual realiza las actividades comunes del ciclo de desarrollo de un producto informático, entre las cuales está el Modelado del Negocio, como resultado se obtiene la información necesaria de los procesos a automatizar, Captura de Requisitos se obtiene la especificación de los requisitos que debe cumplir el sistema, Análisis y Diseño donde se modela el sistema que se va a construir, Implementación y Pruebas. Una de las actividades que se consideran con mayor importancia por el peso que representa para la obtención de un paseo es la implementación.

En esta etapa haciendo uso de herramientas de diseño gráfico los artistas digitales modelan y ensamblaban los objetos que conforman la escena del paseo virtual, que posteriormente se entregan a los programadores quienes implementan la lógica de la aplicación. Para completar la lógica de la aplicación se implementa la interfaz gráfica con los componentes que permiten al usuario interactuar con el sistema, se implementa el código necesario para cargar la escena y todos sus modelos, los recorridos disponibles para el usuario final en el paseo virtual, además de los detalles que dan realismo a los objetos como: luz, sombra o efectos visuales como niebla. Se implementan las relaciones de la aplicación con un motor gráfico para renderizar la escena, la biblioteca de tratamiento de colisiones, sonido y física, esta última es

utilizada para asignar propiedades físicas a los elementos de la escena. Cuando se concluye la etapa de implementación de un paseo virtual si el equipo de desarrollo necesita modificar los recorridos, la posición de los modelos o realizar algún otro cambio en el escenario se debe realizar nuevamente todo el proceso desde el trabajo de los artistas digitales.

## 2.2 Modelo de Dominio.

El modelo de dominio permite comprender el funcionamiento del negocio a través de la modelación de los procesos identificados, conceptos y sus relaciones. Mediante el modelo conceptual que se muestra a continuación se explica los conceptos significativos del negocio con el objetivo de lograr un conocimiento básico del vocabulario y de los conceptos que se incluyen en los requerimientos. Los principales conceptos involucrados en el desarrollo de un Paseo Virtual se describen a continuación.

**Diseñador gráfico:** Persona que crea los modelos gráficos.

**Modelos gráficos:** Representación gráfica de objetos y planos mediante el uso de aplicaciones informáticas.

**Probador:** Persona que realiza las pruebas al sistema para verificar que cumple con los requisitos de sofware.

**Desarrollador:** Persona que realiza la implementación del sistema.

**Herramientas de diseño gráfico:** Aplicaciones informáticas utilizadas para diseñar gráficos.

**Herramientas de pruebas:** Aplicaciones utilizadas para apoyar la realización de las pruebas al sistema.

**IDEs:** Entorno de desarrollo integrado (en inglés *integrated development environment*) es un programa informático compuesto entre otras herramientas por: un editor de código, un compilador, un depurador y un constructor de interfaz gráfica que permiten implementar otro programa.

**Motor gráfico:** Proveedor de motor para los gráficos 2D y 3D, motor físico o detector de colisiones, sonidos, animación, inteligencia artificial, redes, administración de memoria y un escenario gráfico.

**Componentes:** Programas o ficheros de códigos.

**Paseo virtual:** Simulación de un entorno empleando tecnología de realidad virtual.

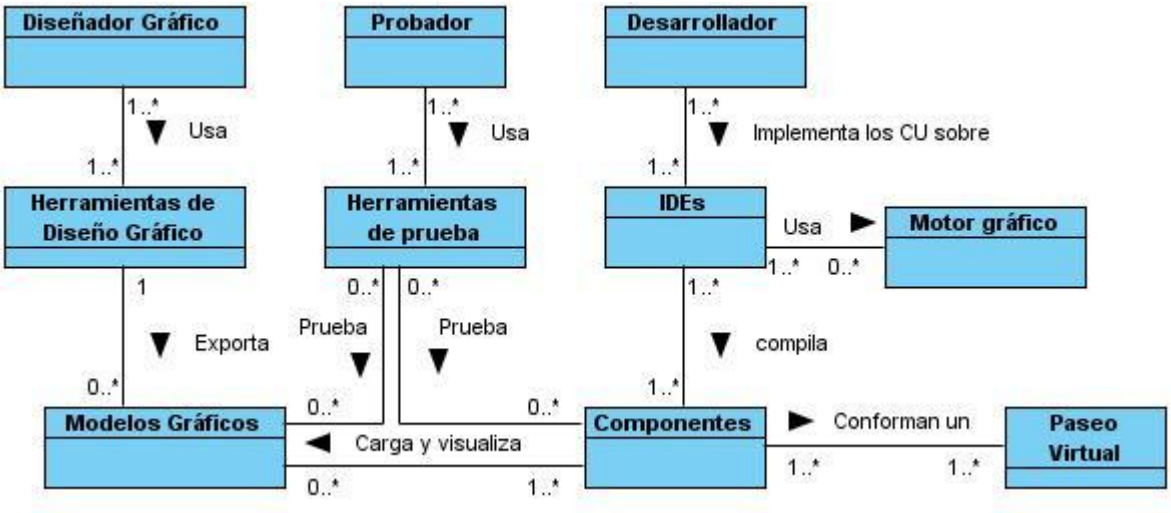



**Figura 1. Modelo de Dominio.**

## 2.3 Objeto de automatización.

Se pretende diseñar una herramienta que permita automatizar las actividades enmarcadas en la etapa de implementación de un paseo virtual interactivo de tal forma que no sea necesario efectuar la implementación de los componentes y parte del proceso de edición de la escena.

## 2.4 Propuesta de solución.

Se propone realizar el Análisis y Diseño de una herramienta para Crear, Editar y Visualizar Centros Expositivos Virtuales interactivos, por lo que se plantea que la herramienta a diseñar conste de dos módulos los cuales se describen a continuación.

- El módulo Editor permitirá al usuario cargar el entorno virtual al cual se le realizará el paseo, cargar objetos decorativos o expositivos y manipularlos, asociarle información y moverlo hacia una posición válida del escenario. Además debe permitir definir una trayectoria de recorrido del paseo y realizar detección de colisiones, así como adicionar un sonido para ambientar el entorno. En caso de no desear alguno de los aspectos mencionados la herramienta debe posibilitar la eliminación de estos. Luego de haber realizado los cambios en el entorno virtual la herramienta debe permitir realizar una salva con todos los elementos del paseo virtual creado.

- El módulo Visualizador mostrará al usuario el paseo previamente creado. Debe permitir definir la velocidad y altura de la cámara para realizar el recorrido. Se debe mostrar la información asociada a los objetos y resaltar los objetos expositivos al pasar el mouse por encima de ellos. Debe dar la posibilidad de iniciar y pausar el recorrido, ofrecer una ayuda del paseo y cerrar el visualizador del paseo. Los modelos deben ser creados en una herramienta de diseño gráfico en las coordenadas 0.0.

### 2.4.1 Metodologías, lenguajes y herramientas a utilizar.

Se seleccionó la metodología de desarrollo RUP porque el GDPV requiere una documentación exhaustiva del análisis y diseño de la herramienta y solicitaron que se desarrollara la investigación utilizando dicha metodología ya que poseen experiencia trabajando con la misma. Se seleccionó el lenguaje de programación C++ por ser un lenguaje compilado además de ser el soportado por el IDE QT Creator seleccionado por ser libre y multiplataforma y ser el utilizado por el GDPV. Se utilizará y el motor gráfico Ogre por solicitud del cliente. La herramienta CASE que se utilizará será el Visual Paradigm por ser una herramienta multiplataforma que soporta el ciclo de vida completo de desarrollo de software, la UCI paga su licencia y además soporta el lenguaje de modelado UML el cual ha sido seleccionado para especificar y visualizar artefactos del sistema. Para la administración de los requisitos se seleccionó el RequisitePro pues permite reducir los riesgos de los proyectos con la visualización de requisitos que pueden verse afectados por cambios en los requisitos en sentido ascendente o descendente y es actualmente la utilizada por el Grupo de Desarrollo de Paseos Virtuales para quienes se desarrolla el presente trabajo. En el diseño de la herramienta se utilizará el framework Property Browser para modificar las propiedades de los objetos, seleccionado por el GDPV quienes poseen experiencia desarrollando aplicaciones con el mismo.

## 2.5 Especificación de los requisitos de software.

Los requisitos son una especificación de lo que el sistema debe cumplir para satisfacer al cliente. Estos pueden dividirse en requisitos funcionales y requisitos no funcionales. En los próximos epígrafes se especifican los requisitos tanto funcionales como no funcionales para cada uno de los módulos que conforman la herramienta que se propone.

### 2.5.1 Requisitos funcionales.

Los requisitos funcionales son capacidades o condiciones que el sistema debe cumplir. En las siguientes tablas se especifican los requisitos funcionales de la herramienta para Crear, Editar y Visualizar Centros Expositivos Virtuales.

**Requisitos funcionales del módulo Editor.**

| **Nº** | **Nombre** | **Descripción** |
|---|---|---|
| RF 1 | Administrar proyecto. | Maneja las funcionalidades Crear, Salvar Eliminar y Cargar un paseo virtual. |
| RF 1.1 | Crear proyecto. | Crea una carpeta con el nombre del paseo virtual a diseñar, dentro de esta se crea una carpeta que contendrá los recursos (modelos gráficos, cámaras, recorridos, luces) del paseo virtual y el fichero principal del paseo. El fichero y la carpeta de los recursos tendrán el mismo nombre. |
| RF 1.2 | Salvar proyecto. | Guarda los cambios realizados al paseo cargado en la carpeta creada según la distribución descrita en el requisito RF 1.1. |
| RF 1.3 | Eliminar proyecto. | Elimina el proyecto, la carpeta del paseo con el fichero principal y todos sus modelos gráficos. |
| RF 1.4 | Cargar proyecto. | Carga el proyecto seleccionado en el editor para luego realizar cambios. |
| RF 2 | Administrar modelos del sistema. | Adiciona o elimina modelos al sistema. |
| RF 2.1 | Adicionar modelo al sistema. | Adiciona un modelo al sistema. |
| RF 2.2 | Eliminar modelo del sistema. | Elimina un modelo del sistema. Antes de eliminarlo se debe preguntar al usuario si realmente está seguro de eliminarlo definitivamente del sistema. |
| RF 3 | Administrar objetos del entorno. | Maneja las funcionalidades relacionadas con los objetos decorativos o expositivos que se mostrarán dentro del entorno. |
| RF 3.1 | Cargar objeto en el | Carga un objeto y lo muestra en el área de visualización. |

| | | |
|---|---|---|
| | entorno. | |
| RF 3.2 | Mover objeto del entorno. | Mover un objeto hacia cualquier área válida en el entorno. |
| RF 3.3 | Alinear objetos. | Al mover un objeto se mostrará una línea que indica la posición del objeto seleccionado con respecto a otro objeto cercano, permitiendo alinearlo con otro objeto del escenario. |
| RF 3.4 | Activar sombra. | Muestra u oculta la sombra de un objeto en el entorno. |
| RF 3.5 | Cambiar vista del objeto. | Cambia el ángulo de vista de todos los objetos. Las vistas pueden ser Frente, Arriba, Derecha, Izquierda y Libre en Perspectiva. |
| RF 3.6 | Cambiar modo de visualización. | Cambia la forma de visualizar los objetos gráficos de la escena. La forma de visualización puede ser: Sólido, Puntos o Malla. |
| RF 3.7 | Eliminar objeto del entorno. | Elimina el objeto seleccionado del entorno y del área de visualización. |
| RF 4 | Modificar objeto decorativo. | Modifica el tamaño de un objeto decorativo y la cantidad que puede existir en el paseo. |
| RF 4.1 | Escalar objetos decorativos. | Aumenta o disminuye el tamaño de un objeto decorativo. |
| RF 4.2 | Duplicar objetos decorativos. | Realiza una copia de un objeto decorativo en el entorno. |
| RF 5 | Modificar información de los objetos. | Crea y modifica la información referente a un objeto y permite clasificarlo en decorativo o expositivo. |
| RF 5.1 | Actualizar información de objeto. | Modifica la información asociada a un objeto. |
| RF5.2 | Definir tipo de objeto. | Clasifica el objeto en decorativo o expositivo. |
| RF 6 | Administrar sonido | Adiciona un sonido al entorno, permite la reproducción recursiva, asociarlo a un objeto o escenario y da la posibilidad de activarlo o desactivarlo. |
| RF 6.1 | Adicionar sonido. | Adiciona un sonido al sistema. |
| RF 6.2 | Eliminar sonido. | Elimina un sonido del sistema. |

| | | |
|---|---|---|
| RF 6.3 | Repetir sonido. | Realiza la reproducción recursiva de un sonido. |
| RF 6.4 | Asociar sonido a la escena. | Asigna un sonido a un entorno. |
| RF 6.5 | Quitar sonido de la escena. | Quita un sonido asociado a un entorno. |
| RF 7 | Administrar cámara | Adiciona una cámara al paseo, permite modificar su altura y velocidad así como definir una trayectoria y eliminarla del paseo. |
| RF 7.1 | Adicionar cámara. | Adiciona una cámara al paseo. |
| RF 7.2 | Modificar altura. | Aumenta o disminuye la altura de una cámara. |
| RF 7.3 | Definir trayectoria. | Define el recorrido que realizará la cámara dentro del paseo. |
| RF 7.4 | Eliminar cámara. | Elimina la cámara del paseo. |
| RF 7.5 | Eliminar Trayectoria. | Elimina la trayectoria definida por el actor. |
| RF 8 | Detectar colisiones. | Detecta la cercanía entre objetos. |
| RF 9 | Exportar paseo. | Guarda todos los cambios realizados al paseo y se guarda en la carpeta del visualizador. |
| RF 10 | Listar recursos. | Lista los nombres de los proyectos y los nombres de los modelos que se encuentren en el sistema. |

**Tabla 1. Requisitos funcionales del módulo Editor.**

**Requisitos funcionales del módulo Visualizador.**

| **N°** | **Nombre** | **Descripción** |
|---|---|---|
| RF 1 | Inicializar Datos. | Lista en el visualizador los nombres de los paseos previamente creados en el módulo Editor que se encuentren en la carpeta de proyectos de la aplicación. |
| RF 2 | Definir modo de paseo. | Define la forma en que se realizará el recorrido por el centro expositivo virtual, el recorrido puede realizarse de forma libre (utilizando los controles del teclado) o dirigido (se muestra el centro virtual a través del recorrido de la cámara). |
| RF 3 | Editar cámara. | Modifica la velocidad y la altura de la cámara mediante los dispositivos de entrada, de esta forma es posible realizar el |

| | | |
|---|---|---|
| | | recorrido por el centro virtual. |
| RF 3.1 | Modificar velocidad. | Disminuye o aumenta la velocidad con que se moverá la cámara en el paseo. |
| RF 3.2 | Modificar altura. | Disminuye o aumenta la altura a la que estará la cámara durante el paseo. |
| RF 3.3 | Pausar paseo. | Disminuye la velocidad de la cámara a valor cero. |
| RF 4 | Mostrar información asociada a un objeto. | Muestra la información asociada a un objeto en una pequeña ventana al hacer clic con el mouse encima del mismo. |
| RF 5 | Resaltar objetos expositivos. | Al pasar el mouse por encima de un objeto el puntero cambia de forma indicando que puede ser seleccionado. |
| RF 6 | Comenzar paseo. | Carga todos los elementos: modelos, luces, recorridos, cámara del paseo seleccionado y lo muestra en pantalla completa. |
| RF 7 | Cerrar paseo. | Cierra la vista previa del paseo virtual y se muestran el resto de las ventanas que conforman el Editor. |
| RF 8 | Mostrar ayuda del paseo. | Muestra información sobre cómo realizar el recorrido por el entorno virtual. |

**Tabla 2. Requisitos funcionales del módulo Visualizador.**

### 2.5.2 Requisitos no funcionales.

Los requisitos no funcionales son propiedades o cualidades que el producto debe cumplir. Debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable, rápido o confiable. Para la herramienta a desarrollar se especifican los siguientes requisitos no funcionales:

**Requisitos no funcionales del módulo Editor.**

1. Usabilidad.

RnF 1.1 Utilización del sistema. Los usuarios del sistema serán diseñadores de Paseos Virtuales deberán poseer conocimientos mínimos para: rotar, mover y escalar modelos gráficos en alguna de las herramientas de diseño de modelos 3D.

2. Portabilidad.

RnF 3.1 Ejecución de la herramienta. La herramienta permitirá ser compilada y ejecutada sobre la plataforma Windows y Linux.

3. Restricciones en el diseño y la implementación.

RnF 3.1 Lenguajes de programación. Se utilizará el lenguaje de programación C++.

RnF 3.2 Herramientas de desarrollo. Para la implantación del sistema se utilizará la herramienta de desarrollo QT Creator.

RnF 3.3 Herramientas de diseño gráfico. Se utilizará la herramienta Blender para la modelación de gráficos.

RnF 3.4 Implementación del sistema. Se regirá por la filosofía de Programación Orientada a Objetos y se utilizará el motor gráfico Ogre.

4. Hardware

RnF 4.1 Instalación de la herramienta. Para la instalación del software la computadora deberá cumplir con los siguientes requisitos:

- Memoria RAM de 1 Gb o Superior.
- Velocidad de Microprocesador a 3.0 GHz o superior.
- Recursos de Video de 256 Mb o superior.

5. Interfaz o apariencia externa.

RnF 5.1 Interfaz de Usuario. La interfaz de la herramienta será sencilla para facilitar el uso de la misma, lo cual se evidencia en lo siguiente: las opciones de servicios además de tener su icono identificador tendrá el texto que muestre la opción en cuestión con su función descrita en una frase breve para un reconocimiento rápido por el usuario. Constará de un menú para listar los nombres de los proyectos que se encuentren en el sistema en la parte inferior izquierda. En la parte superior izquierda se mostrará un menú para listar los objetos que se encuentren en el entorno y otro menú en la parte inferior derecha que muestre un listado de los nombres de los modelos que se encuentren en el sistema. Contendrá además una barra de herramienta que muestre iconos que darán respuesta a las funcionalidades.

6. Ayuda y documentación en línea.

RnF 6.1 Se contará con un manual de ayuda que describa cada uno de los servicios que brinda la aplicación.

**Requisitos no funcionales del módulo Visualizador.**

1. Usabilidad.

RnF 1.2 Utilización del sistema. Los usuarios del sistema serán visitantes interesados en observar un paseo virtual determinado con habilidades en la manipulación del teclado y mouse (ratón) de una computadora.

2. Rendimiento.

RnF 2.1 Velocidad de representación de imágenes. Representar las imágenes o cuadros (frame) a una velocidad mínima de 30 frames por segundo.

3. Portabilidad.

RnF 3.1 Ejecución de la herramienta. La herramienta permitirá ser compilada y ejecutada sobre la plataforma Windows y Linux.

4. Restricciones de diseño.

RnF 4.1 Lenguajes de programación. Se utilizará el lenguaje de programación C++.

RnF 4.2 Herramientas de desarrollo. Para la implementación del sistema se utilizará el IDE QT Creator.

RnF 4.3 Herramientas de diseño gráfico. Se utilizará la herramienta Blender para la modelación de gráficos.

RnF 4.4 Implementación del sistema. Se regirá por la filosofía de Programación Orientada a Objetos y se utilizará el motor gráfico Ogre.

5. Hardware

RnF 5.1 Instalación de la herramienta. Para la instalación del software la computadora destino deberá cumplir con los siguientes requisitos.

- Memoria RAM de 1 Gb o Superior.
- Velocidad de Microprocesador a 3.0 GHz o superior.
- Recursos de Video de 128 Mb o superior.

6. Interfaz.

RnF 6.1 Interfaz de Usuario. Para facilitar el uso la interfaz de la herramienta las opciones de servicios se representarán con un icono identificador y al colocar el ratón encima de alguno de estos iconos debe mostrarse un texto que muestre la opción en cuestión con su función descrita en una frase breve para un reconocimiento rápido por el usuario. Constará de un menú para listar los nombres de los proyectos que se encuentren en el sistema en la parte superior izquierda.

Contendrá además una barra de herramienta que muestre iconos que darán respuesta a las funcionalidades.

7. Ayuda y documentación en línea.

RnF 7.1 Manual de ayuda. Contará con un manual de ayuda con información referente a cómo utilizar el teclado y el mouse (ratón) para desplazarse por el paseo virtual y acceder a la información de los objetos.

## 2.6 Definición de los casos de uso.

Una vez recopilados los requisitos se pueden definir los casos de uso los cuales facilitarán una descripción de cómo el sistema se usará. El caso de uso describe la manera en que los actores interactúan con el sistema. (Jacobson [et al.], 2000).

**Definición de los actores.**

Un actor representa papeles que las personas (o dispositivos) juegan como impulsores del sistema. Los actores que interactuarán con la herramienta para Crear, Editar y Visualizar Centros Expositivos Virtuales se describen a continuación.

| Actores | Justificación |
|---|---|
| Diseñador del Paseo. | Es la persona que interactúa con el módulo Editor de la herramienta para crear o editar un paseo virtual a un centro expositivo. |
| Usuario. | Es la persona que interactúa con el módulo Visualizador de la herramienta para realizar el recorrido por uno de los paseos virtuales. |

**Tabla 3. Descripción de los actores que interactúan con la herramienta.**

**Listados de casos de uso (CU):**

**Módulo Editor.**

CU-1. Administrar proyecto.

CU-2. Administrar modelos del sistema.

CU-3. Administrar objetos del entorno.

CU-4. Modificar objeto decorativo.

CU-5 Modificar información de los objetos.

**Módulo Visualizador.**

CU-1. Inicializar Datos.

CU-2. Definir modo de paseo.

CU-3. Editar cámara.

CU-4. Mostrar información asociada a un objeto.

CU-5. Resaltar objetos expositivos.

CU-6. Administrar sonido.
CU-7. Administrar cámara.
CU-8. Detectar colisiones.
CU-9. Exportar paseo.
CU-10. Listar recursos.

CU-6. Comenzar paseo.
CU-7. Cerrar paseo.
CU-8. Mostrar ayuda del paseo.

### 2.6.1 Diagrama de casos de uso.

La representación de los diagramas de casos de uso permite especificar la funcionalidad y el comportamiento de un sistema mediante su interacción con los actores del sistema.

En las siguientes figuras se muestra la vista de casos de uso para los módulos Editor y Visualizador que representan los casos de usos arquitectónicamente significativos. En el Anexo 1 del documento se podrán observar los diagramas de casos de uso representando todos los casos de uso definidos para ambos módulos.


**Figura 2. Diagrama de casos de uso arquitectónicamente significativos del módulo Editor.**

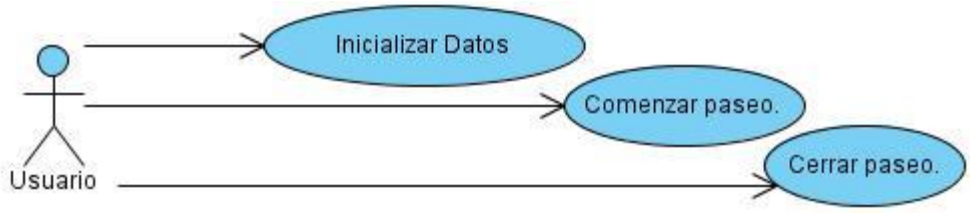



**Figura 3. Diagrama de casos de uso arquitectónicamente significativos del módulo Visualizador.**

### 2.6.2 Descripción textual de los casos de usos.

Mediante la descripción textual de un caso de uso se describe paso a paso la secuencia de eventos que los actores realizan para completar un proceso a través del sistema. Se especifican las acciones que

realizan los actores sobre el sistema y la respuesta que surge como consecuencia de cada evento. Seguidamente se muestran las descripciones textuales de los casos de uso arquitectónicamente significativos. Pueden verse en el Anexo 2 las descripciones textuales de todos los casos de uso definidos para la herramienta.

**Casos de uso expandidos para el módulo Editor.**

| **Caso de uso** | |
|---|---|
| **CU-1** | Administrar proyecto. |
| **Propósito** | El objetivo del caso de uso es que el diseñador del paseo pueda crear un proyecto, salvar un proyecto, cargar un proyecto en escena para luego editarlo o eliminar un proyecto de la carpeta del sistema. |
| **Prioridad** | Crítico. |
| **Actores:** Diseñador del Paseo. | |
| **Resumen:** El diseñador del paseo inicializa el caso de uso cuando selecciona una de las acciones siguientes: Crear Proyecto, Salvar Proyecto, Cargar Proyecto o Eliminar Proyecto. Al seleccionar la opción Crear Proyecto se crea una carpeta con el nombre del proyecto en la cual se salvará el fichero principal del paseo y una carpeta que contendrá los recursos (cámara, luces, recorridos, modelos) del paseo virtual. El fichero y la carpeta de recursos tendrán el mismo nombre. Los cambios efectuados al proyecto serán guardados en la capeta especificada. El caso de uso permite además eliminar el paseo virtual diseñado. | |
| **Referencias** | RF 1, RF 1.1, RF 1.2, RF 1.3, RF 1.4. |
| **Precondiciones** | No puede existir un proyecto con el mismo nombre. |
| **Poscondiciones** | Se crea una carpeta con el nombre del proyecto en la cual se salva el fichero principal del paseo y una carpeta que contendrá los modelos del paseo virtual. Se muestra en pantalla el área de diseño y los objetos almacenados en el sistema. En caso de haber seleccionado la opción eliminar proyecto, se elimina el proyecto seleccionado de la carpeta del sistema. |
| **Curso Normal de Eventos** | |
| **Acción del actor** | **Respuesta del sistema** |
| 1. El diseñador selecciona unas de las opciones siguientes: Crear Proyecto, Salvar Proyecto o | 1.1 Si el diseñador selecciona la opción Crear Proyecto ir al Escenario 1 Crear Proyecto. |

| | |
|---|---|
| Eliminar Proyecto. | 1.2 Si el diseñador selecciona la opción Salvar Proyecto ir al Escenario 2 Salvar Proyecto.<br>1.3 Si el diseñador selecciona la opción Eliminar Proyecto ir al Escenario 3 Eliminar Proyecto.<br>1.4 Si el diseñador selecciona la opción Cargar Proyecto ir al Escenario 3 Cargar Proyecto. |
| **Escenario 1 Crear Proyecto** | |
| **Curso Normal de Eventos** | |
| **Acción del actor** | **Respuesta del sistema** |
| 1. El diseñador elige la opción Crear Proyecto.<br>2. El diseñador teclea el nombre del proyecto.<br>3. El diseñador selecciona la opción aceptar. | 1.1 El sistema muestra una venta al diseñador para que teclee el nombre del proyecto a crear.<br>3.1 El sistema verifica la existencia de un proyecto con igual nombre al que introduce el diseñador.<br>3.2 Crea una carpeta con el nombre del proyecto, dentro de esta carpeta crea el fichero principal del paseo y una carpeta que contendrá los modelos del paseo virtual. El fichero y las carpetas son creados con el mismo nombre del proyecto. Finaliza el caso de uso. |
| **Curso Alterno de Eventos** | |
| **Acción del actor** | **Respuesta del sistema** |
| 3.3 El diseñador inserta un nombre de proyecto que ya existe en el sistema.<br>3.4 El diseñador selecciona la opción cancelar. | 3.3 El sistema muestra el mensaje “Ya existe un Proyecto con este nombre...” indicando al diseñador que ya el proyecto existe en el sistema.<br>3.5 El sistema cancela la operación, culmina así el caso de uso. |
| **Escenario 2 Salvar Proyecto** | |
| **Curso Normal de Eventos** | |
| **Acción del actor** | **Respuesta del sistema** |

| | |
|---|---|
| 1. El diseñador elige la opción Salvar Proyecto. | 1.1 Guarda los cambios realizados al paseo en la carpeta creada según la distribución descrita en el Escenario 1. Finaliza así el caso de uso. |
| **Escenario 3 Eliminar Proyecto** | |
| **Curso Normal de Eventos** | |
| **Acción del actor** | **Respuesta del sistema** |
| 1. El diseñador elige la opción Eliminar Proyecto.<br>2. El diseñador selecciona la opción aceptar. | 1.1 El sistema muestra el siguiente mensaje de alerta: ¿Desea eliminar el proyecto del sistema?<br>2.1 El sistema elimina el proyecto, la carpeta del paseo con el fichero principal y todos sus modelos, cámara, recorridos luces. Finaliza el caso de uso. |
| **Curso Alterno de Eventos** | |
| **Acción del actor** | **Respuesta del sistema** |
| 2. El diseñador elige la opción cancelar. | 2.1 El sistema cancela la operación y culmina así el caso de uso. |
| **Escenario 4 Cargar Proyecto** | |
| **Curso Normal de Eventos** | |
| **Acción del actor** | **Respuesta del sistema** |
| 1. El diseñador selecciona un paseo virtual de la lista de paseos y elige la opción Cargar Paseo. | 1.1 El sistema muestra en el editor el paseo seleccionado con todos sus recursos (cámara, modelos, recorridos, luces). |

**Tabla 4. Descripción textual del CU-1. Editor.**

| **Caso de uso** | |
|---|---|
| **CU-2** | Administrar modelos del sistema. |
| **Propósito** | El objetivo del caso de uso es que el diseñador del paseo pueda adicionar o eliminar modelos del sistema. |
| **Prioridad** | Crítico. |
| **Actores:** Diseñador del Paseo. | |

<table>
<tr><td colspan="2">Resumen: El diseñador del paseo inicializa el caso de uso cuando selecciona la opción adicionar modelo al sistema. Antes de eliminar un modelo del sistema se pregunta al usuario si realmente está seguro de eliminarlo definitivamente del sistema.</td></tr>
<tr><td>Referencias</td><td>RF 2, RF 2.1, RF 2.2.</td></tr>
<tr><td>Poscondiciones</td><td>Se adiciona o elimina modelos del sistema.</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 diseñador selecciona unas de las opciones siguientes: Adicionar Modelo al Sistema o Eliminar Modelo del Sistema.</td><td>1.1 Si el diseñador selecciona la opción Adicionar Modelo al Sistema ir al Escenario 1 Adicionar Modelo al Sistema.<br>1.2 Si el diseñador selecciona la opción Eliminar Modelo del Sistema ir al Escenario 2 Eliminar Modelo del Sistema.</td></tr>
<tr><td colspan="2">Escenario 1 Adicionar Modelo al Sistema</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 diseñador elige la opción Adicionar Modelo al Sistema o selecciona uno de los modelos que se encuentren en la escena.<br>2. El diseñador selecciona el modelo desde la ubicación donde se encuentre.<br>3. El diseñador da clic en la opción aceptar.</td><td>1.1 En el caso de haber elegido la opción Adicionar Modelo al Sistema el sistema abre una venta como el explorador de Windows para que el diseñador busque el modelo. Si el diseñador inicializó el caso de uso al dar clic derecho sobre el modelo y elegir la opción adicionar al sistema, El sistema agrega el modelo como un recurso gráfico más del sistema disponible para cargarlo en los proyectos.<br>3.1 El sistema agrega el modelo como un recurso gráfico más del sistema disponible para cargarlo en los proyectos y finaliza el caso de uso.</td></tr>
<tr><td colspan="2">Curso Alterno de Eventos</td></tr>
<tr><td>Acción del actor</td><td>Respuesta del sistema</td></tr>
</table>

| | |
|---|---|
| 3.2 El diseñador selecciona un modelo que ya existe en el sistema.<br>3.4 El diseñador elige la opción aceptar. | 3.3 El sistema muestra el mensaje: "El modelo "nombre" ya existe. Desea cambiar el nombre del modelo por "nombre_1".<br>3.5 El sistema adiciona el modelo con el nombre "nombre_1".Finaliza el caso de uso. |
| **Escenario 2 Eliminar Modelo del Sistema** | |
| **Curso Normal de Eventos** | |
| **Acción del actor** | **Respuesta del sistema** |
| 1. El diseñador elige la opción Eliminar Modelo del Sistema.<br>2. El diseñador elige la opción aceptar. | 1.1 Muestra el siguiente mensaje de alerta: ¿Desea eliminar el modelo: "nombre"?<br>2.1 Elimina el modelo de la carpeta del sistema y finaliza el caso de uso. |
| **Curso Alterno de Eventos** | |
| **Acción del actor** | **Respuesta del sistema** |
| 2. El diseñador elige la opción cancelar. | 2.1 El sistema cancela la operación y culmina así el caso de uso. |

**Tabla 5. Descripción textual del CU-2. Editor.**

| **Caso de uso** | |
|---|---|
| **CU-3** | Administrar objetos del entorno. |
| **Propósito** | El objetivo del caso de uso es que el diseñador del paseo pueda cargar modelos en la escena, moverlos, alinearlo respecto a otro objeto, activar su sombra y modificar la forma de visualizarlos en la escena. |
| **Prioridad** | Crítico. |
| **Actores:** Diseñador del Paseo. | |
| **Resumen:** El diseñador del paseo inicializa el caso de uso cuando selecciona una de las opciones para administrar los objetos decorativos o expositivos que se mostrarán dentro del entorno. El caso de uso permite cargar modelos en escena, moverlos, alinearlo respecto a otro objeto, activar sombra y modificar la forma de verlos. | |

| | |
|---|---|
| **Referencias** | RF 3, RF 3.1, RF 3.2, RF 3.4, RF 3.5, RF 3.6, RF 3.7. |
| **Precondiciones** | Los objetos deben estar añadidos al sistema. |
| **Poscondiciones** | Se insertan los objetos en el entorno.<br>Se modifica la posición de un objeto en el entorno.<br>Se genera sombra de un objeto.<br>Se modifica la vista del objeto,<br>Se modifica el modo de visualización de un objeto<br>Se eliminar un objeto del entorno. |

| **Curso Normal de Eventos** | |
|---|---|
| **Acción del actor** | **Respuesta del sistema** |
| 1. El diseñador selecciona unas de las opciones siguientes: Cargar Modelos en Escena, Mover Modelo, Alinear Objeto, Activar Sombra, Cambiar Vista del Objeto, Cambiar modo de visualización o Eliminar objeto del Entorno. | 1.1 Si el diseñador selecciona la opción Cargar Modelos en Escena ir al Escenario 1.<br>1.2 Si el diseñador selecciona la opción Mover Modelo ir al Escenario 2.<br>1.3 Si el diseñador selecciona la opción Alinear Objeto ir al Escenario 3.<br>1.4 Si el diseñador selecciona la opción Activar Sombra ir al Escenario 4.<br>1.5 Si el diseñador selecciona la opción Cambiar Vista del Objeto ir al Escenario 5.<br>1.6 Si el diseñador selecciona la opción Cambiar modo de Visualización ir al Escenario 6.<br>1.7 Si el diseñador selecciona la opción Eliminar objeto del Entorno ir al Escenario 7. |
| **Escenario 1 Cargar Modelos en Escena** | |
| **Curso Normal de Eventos** | |
| **Acción del actor** | **Respuesta del sistema** |

| | |
|---|---|
| 1. El diseñador elige un objeto de la lista de objetos que se encuentran en el sistema y lo arrastra hacia el área de visualización. | 1.1 El sistema pinta el objeto en el área de visualización, si el objeto ya se encuentra en la escena el sistema le asocia un nuevo nombre. El formato del nuevo nombre será: “nombre_número”. |
| **Escenario 2 Mover Modelo** | |
| **Curso Normal de Eventos** | |
| **Acción del actor** | **Respuesta del sistema** |
| 1. El diseñador selecciona un objeto en el área de visualización y lo desplaza por el espacio. | 1.1 El sistema posiciona el objeto en la posición seleccionada por el diseñador. |
| **Curso Alterno de Eventos** | |
| **Acción del actor** | **Respuesta del sistema** |
| 1. El diseñador selecciona un objeto en el área de visualización y lo ubica en una posición ocupada por otro objeto. | 1.1 El sistema coloca el objeto en la posición no ocupado por otro objeto más cercana. |
| **Escenario 3 Alinear Objeto** | |
| **Curso Normal de Eventos** | |
| **Acción del actor** | **Respuesta del sistema** |
| 1. El diseñador selecciona un objeto en el área de visualización y lo desplaza hacia una posición cercana a otro objeto. | 1.1 El sistema muestra unas líneas que indican la posible alineación con el objeto más cercano a la posición donde se moverá el objeto desplazado. |
| **Escenario 4 Activar Sombra** | |
| **Curso Normal de Eventos** | |
| **Acción del actor** | **Respuesta del sistema** |

| | |
|---|---|
| 1. El diseñador selecciona un objeto en el área de visualización y elige la opción Activar Sombra del objeto.<br>2. El diseñador selecciona un objeto en el área de visualización y elige la opción Desactivar Sombra del objeto. | 1.1 El sistema muestra la sombra del objeto seleccionado en el área de visualización.<br>2.1 El sistema oculta la sombra del objeto seleccionado. |
| **Escenario 5 Cambiar Vista del Objeto** | |
| **Curso Normal de Eventos** | |
| **Acción del actor** | **Respuesta del sistema** |
| 1. El diseñador elige una de las opciones para cambiar la vista de la escena o de los objetos. Las opciones de vistas pueden ser Frente, Arriba, Derecha, Izquierda y Libre en Perspectiva. | 1.1 El sistema posiciona la escena y todos sus elementos de la forma especificada por el diseñador. |
| **Escenario 6 Cambiar modo de visualización** | |
| **Curso Normal de Eventos** | |
| **Acción del actor** | **Respuesta del sistema** |
| 1. El diseñador elige una de las opciones para cambiar la forma de visualizar la escena o los objetos. La forma de visualización puede ser Sólido, Puntos o Malla. | 1.1 El sistema visualiza la escena y todos sus elementos de la forma especificada por el diseñador. |
| **Escenario 7 Eliminar Objeto del Entorno** | |
| **Curso Normal de Eventos** | |
| **Acción del actor** | **Respuesta del sistema** |
| 1. El diseñador elige un objeto del área de visualización y selecciona la opción eliminar objeto del entorno. | 1.1 El sistema elimina el objeto seleccionado del entorno. |

**Tabla 6. Descripción textual del CU-3. Editor.**

| **Caso de uso** | |
|---|---|
| **CU-7** | Administrar cámara. |
| **Propósito** | El objetivo del caso de uso es que el diseñador del paseo pueda adicionar y modificar la cámara que guiará el recorrido del paseo virtual. |
| **Prioridad** | Crítico. |
| **Actores:** Diseñador del Paseo. | |
| **Resumen:** El diseñador del paseo inicializa el caso de uso cuando adiciona o elimina un recorrido del paseo, adiciona una cámara y modifica su altura, velocidad o la elimina del paseo. | |
| **Referencias** | RF 7, RF 7.1, RF 7.2, RF 7.3, RF 7.4, RF 7.5. |
| **Precondiciones** | Debe estar definido el recorrido a realizar por el paseo. |
| **Poscondiciones** | Se crea la cámara que guiará el recorrido por el paseo.<br>Se define una trayectoria.<br>Se modifica la altura de una cámara.<br>Se elimina una cámara del paseo. |
| **Curso Normal de Eventos** | |
| **Acción del actor** | **Respuesta del sistema** |
| 1. El diseñador elige una de las opciones siguientes: Definir trayectoria, Adicionar cámara, Eliminar cámara, Modificar altura de cámara. | 1.1 Si el diseñador selecciona la opción Definir trayectoria ir al Escenario 1.<br>1.2 Si el diseñador selecciona la opción Adicionar cámara ir al Escenario 2.<br>1.3 Si el diseñador selecciona la opción Eliminar cámara ir al Escenario 3.<br>1.4 Si el diseñador selecciona la opción Modificar altura ir al Escenario 4. |
| **Escenario 1 Definir Trayectoria** | |
| **Curso Normal de Eventos** | |
| **Acción del actor** | **Respuesta del sistema** |

| | |
|---|---|
| 1. El diseñador elige la opción adicionar un recorrido y define mediante el clic izquierdo del mouse puntos que van definiendo el recorrido (spline) hasta dar clic derecho que indica que finalizó de definir la trayectoria del recorrido. | 1.1 El sistema visualiza la línea que se forma entre un punto y el anterior y finalizada la entrada de puntos adiciona el recorrido a la escena. |
| **Escenario 2 Adicionar Cámara** | |
| **Curso Normal de Eventos** | |
| 1. El diseñador elige la opción adicionar cámara. | 1.1 El sistema adiciona la cámara a la escena en el primer punto definido por el diseñador y la visualiza en el área de visualización. Finaliza el caso de uso. |
| **Curso Alterno de Eventos** | |
| **Acción del actor** | **Respuesta del sistema** |
| 1. El diseñador coloca la cámara en un área en que no se haya definido un recorrido. | 1.2 El sistema le avisa al diseñador que el área no está permitida cancelando la acción. Finaliza el caso de uso. |
| **Escenario 3 Eliminar Cámara** | |
| **Curso Normal de Eventos** | |
| **Acción del actor** | **Respuesta del sistema** |
| 1. El diseñador elige la opción eliminar cámara. | 1.1 El sistema elimina la cámara de la escena. Finaliza el caso de uso. |
| **Escenario 4 Modificar Altura** | |
| **Curso Normal de Eventos** | |
| **Acción del actor** | **Respuesta del sistema** |
| 1. El diseñador elige la cámara y modifica la altura desplazándola con el mouse o modificando el atributo altura del objeto. | 1.1 El sistema modifica el atributo altura de la cámara y la visualiza en la nueva posición. Finaliza así el caso de uso. |
| **Escenario 5 Eliminar Trayectoria** | |
| **Curso Normal de Eventos** | |
| **Acción del actor** | **Respuesta del sistema** |

| | |
|---|---|
| 1. El diseñador elige la opción eliminar trayectoria. | 1.1 El sistema elimina la trayectoria definida por el diseñador. Finaliza así el caso de uso. |

**Tabla 7. Descripción textual del CU-7. Editor.**

| **Caso de uso** | |
|---|---|
| **CU-9** | Exportar paseo. |
| **Propósito** | El objetivo del caso de uso es salvar todos los cambios efectuados al paseo y exportarlo a la carpeta del visualizador. |
| **Prioridad** | Crítico. |
| **Actores:** Diseñador del Paseo. | |
| **Resumen:** El diseñador del paseo inicializa el caso de uso cuando da clic en la opción Exportar paseo. Se salvan todos los cambios realizados al paseo en la carpeta de proyecto del visualizador. | |
| **Referencias** | RF 9. |
| **Precondiciones** | Debe existir al menos un paseo almacenado en el sistema. |
| **Poscondiciones** | Se guardan todos los cambios efectuados al paseo. |
| **Curso Normal de Eventos** | |
| **Acción del actor** | **Respuesta del sistema** |
| 1. El diseñador elige la opción Exportar paseo.<br>2. El diseñador selecciona los proyectos que desea salvar para el módulo visualizador.<br>3. El diseñador introduce el nombre de la carpeta donde se guardarán los proyectos seleccionados y acepta la operación. | 1.1 El sistema salva todos los cambios realizados al paseo en la carpeta del proyecto y muestra todos los proyectos que existen en el sistema.<br>2.1 El sistema muestra una ventana al diseñador para que introduzca la dirección de la carpeta donde se guardarán los proyectos seleccionados.<br>3.1 El sistema salva en la carpeta todos los proyectos seleccionados, además de un registro para facilitar la lectura de los recursos (cámara, recorridos, luces, modelos), finalizando así el caso de uso. |

**Tabla 8. Descripción textual del CU-9. Editor.**

**Casos de uso expandidos para el módulo Visualizador.**

| **Caso de uso** | |
|---|---|
| **CU-1** | Inicializar Datos. |
| **Propósito** | El objetivo del caso de uso es que el usuario pueda disponer de todos los paseos virtuales que posea el visualizador en su carpeta de proyectos. |
| **Prioridad** | Crítico. |
| **Actores:** Usuario. | |
| **Resumen:** El usuario inicializa el caso de uso cuando ejecuta el Visualizador. El Visualizador lista los paseos previamente creados en el módulo Editor que se encuentren en la carpeta de proyectos de la aplicación para posibilitar la selección del paseo que desee visualizar el usuario. | |
| **Referencias** | RF 1. |
| **Precondiciones** | El paseo debe estar creado con el módulo Editor. |
| **Poscondiciones** | Se muestra en el área de visualización los paseos que se encuentren en el sistema. |
| **Curso Normal de Eventos** | |
| **Acción del actor** | **Respuesta del sistema** |
| 1. El usuario ejecuta el Visualizador. | 1.1 Se abre el sistema, y se listan los paseos que se encuentren en la carpeta de proyectos de la aplicación. |

**Tabla 9. Descripción textual del CU-1. Visualizador.**

| **Caso de uso** | |
|---|---|
| **CU-6** | Comenzar paseo. |
| **Propósito** | El objetivo del caso de uso que el usuario pueda visualizar el paseo en pantalla completa. |
| **Prioridad** | Crítico. |
| **Actores:** Usuario. | |
| **Resumen:** El caso de uso se inicializa cuando el usuario selecciona de la lista de paseos disponibles el que desea visualizar. | |
| **Referencias** | RF 6. |
| **Precondiciones** | El paseo debe estar disponible en la lista de paseos del visualizador. |

| **Poscondiciones** | Se muestra el paseo en pantalla completa. |
|---|---|
| **Curso Normal de Eventos** | |
| **Acción del actor** | **Respuesta del sistema** |
| 1. El usuario selecciona un paseo virtual de los disponibles en la lista de paseos del visualizador. | 1.1 El sistema carga los recursos (luces, cámara, recorridos, modelos) del paseo seleccionado y lo muestra en pantalla completa. Finaliza así el caso de uso. |

**Tabla 10. Descripción textual del CU-6. Visualizador.**

| **Caso de uso** | |
|---|---|
| **CU-7** | Cerrar paseo. |
| **Propósito** | El objetivo del caso de uso que el usuario pueda visualizar la pantalla inicial del Visualizador y el resto de las opciones. |
| **Prioridad** | Crítico. |
| **Actores:** Usuario. | |
| **Resumen:** El caso de uso se inicializa cuando el usuario selecciona la opción cerrar paseo. | |
| **Referencias** | RF 7. |
| **Precondiciones** | El paseo debe estar cargado en el área de visualización. |
| **Poscondiciones** | Se muestra la pantalla inicial del Visualizador. |
| **Curso Normal de Eventos** | |
| **Acción del actor** | **Respuesta del sistema** |
| 1. El usuario selecciona la opción cerrar paseo. | 1.1 El sistema cierra el paseo virtual que se encuentra en ejecución, visualizando la pantalla inicial y mostrando el resto de las opciones del Visualizador. Finaliza el caso de uso. |

**Tabla 11. Descripción textual del CU-7. Visualizador.**

## 2.7 Matriz de trazabilidad entre los requisitos y los casos de usos del sistema.

En el presente epígrafe se muestra la matriz de trazabilidad bidireccional que se generó entre los requisitos y los casos de uso de la herramienta que se propone. Llevar la trazabilidad bidireccional entre los requisitos y los casos de usos permite que cuando el cliente solicite un cambio el equipo pueda encontrar exactamente qué se necesita modificar, cuántos elementos, dónde se ubican y por tanto proporcionar datos para el cálculo del esfuerzo y tiempo de desarrollo, elementos que aportan la información necesaria a la hora de tomar una decisión sobre ese cambio.

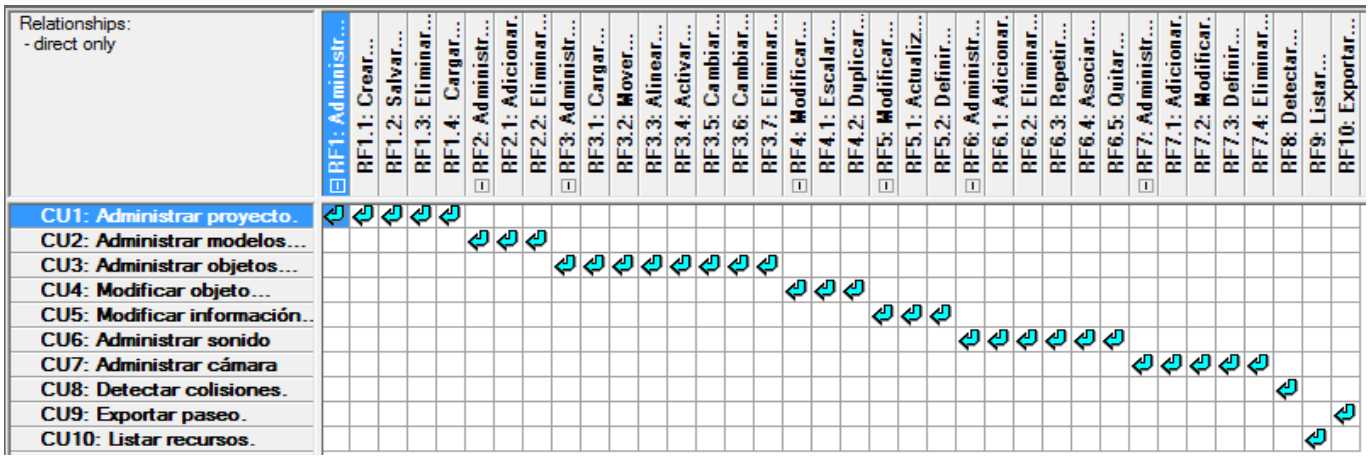



**Figura 4. Matriz de Trazabilidad entre Requisitos y Casos de Uso del Módulo Editor.**

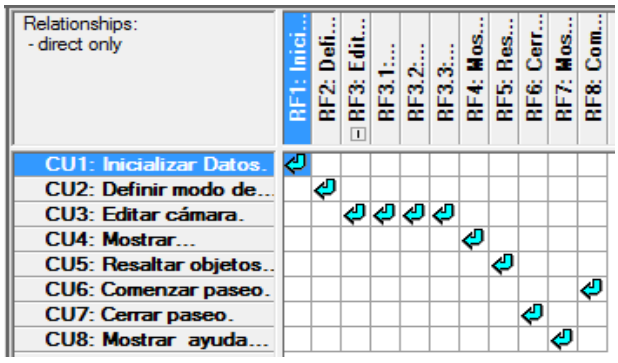



**Figura 5. Matriz de Trazabilidad entre Requisitos y Casos de Uso del Módulo Visualizador.**

## Consideraciones del capítulo.

Se describió el proceso que actualmente se realiza para la creación de un paseo virtual, para identificar los problemas existentes y precisar los subprocesos que serán objeto de automatización. Se realizó el modelo de dominio para obtener una mejor comprensión de los conceptos y procesos que se manejan dentro del negocio a automatizar. Se presentó la propuesta del sistema a desarrollar como parte de la solución al problema planteado. Se especificaron y describieron los requisitos que la herramienta debe cumplir, en total se capturaron 39 requisitos funcionales para el módulo “Editor” y 11 para el módulo “Visualizador”, además de 9 no funcionales para el “Editor” y 10 para el “Visualizador”. Se realizó la definición, representación y descripción textual de los casos de uso del sistema con el objetivo de especificar las funcionalidades del sistema y los actores que interactuarán con el mismo. Se describieron paso a paso la secuencia de eventos que los actores realizarán para inicializar un caso de uso y la respuesta del sistema ante cada acción del actor, se definieron 10 casos de usos para el “Editor” y 8 para el “Visualizador”, en total se describieron 38 escenarios, lo que representa un promedio de 2,2 escenarios por caso de uso. Se realizó la matriz de trazabilidad entre los casos de uso y los requisitos para documentar la trazabilidad bidireccional entre los requisitos y casos de uso. De forma general con la conclusión del presente capítulo se logró describir el sistema que se desea construir lo cual dará paso a una nueva etapa del desarrollo del trabajo para darle cumplimiento a los objetivos trazados.

# Capítulo 3. Análisis y diseño del sistema.

## Introducción

En el presente capítulo se realiza un análisis de los requisitos que se describieron en la captura de los requisitos y se diseña el sistema mediante el lenguaje de los desarrolladores para su posterior desarrollo. Se modela el sistema a través de los modelos del análisis y del diseño. Se describe textualmente cómo se lleva a cabo y se ejecuta cada caso de uso definido para el sistema a través de los diagramas de interacción. Se representa y describe la vista lógica del sistema donde se especifican las clases arquitectónicamente significativas y se diseña un prototipo de interfaz para la herramienta.

## 3.1 Modelos conceptuales de la herramienta.

Los modelos conceptuales que se muestran a continuación son una forma de explicar a los desarrolladores los conceptos significativos asociados a la creación edición y visualización de paseos virtuales que pueden servir como base para un posterior análisis y diseño de la herramienta. En la propuesta del sistema se definió que la herramienta a desarrollar estará compuesta por dos módulos por lo que se realizó un modelo conceptual para cada uno de ellos.

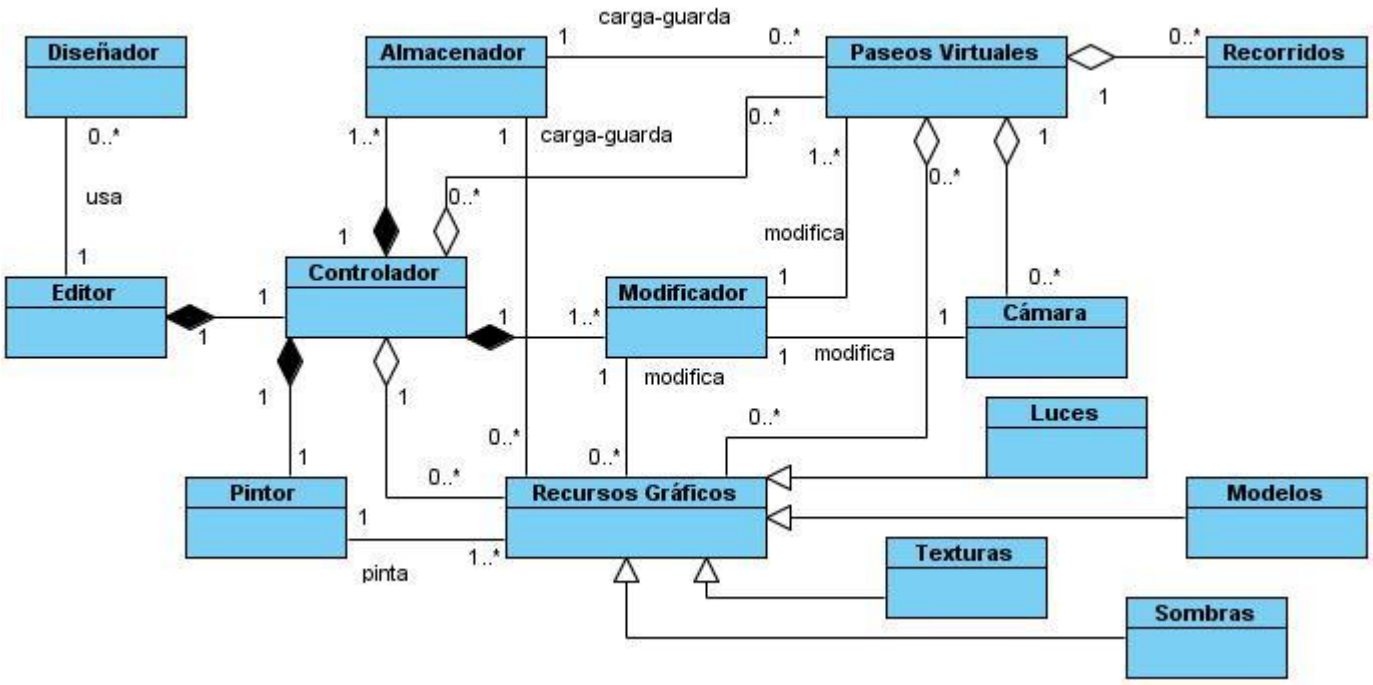



**Figura 6. Modelo conceptual del módulo Editor.**

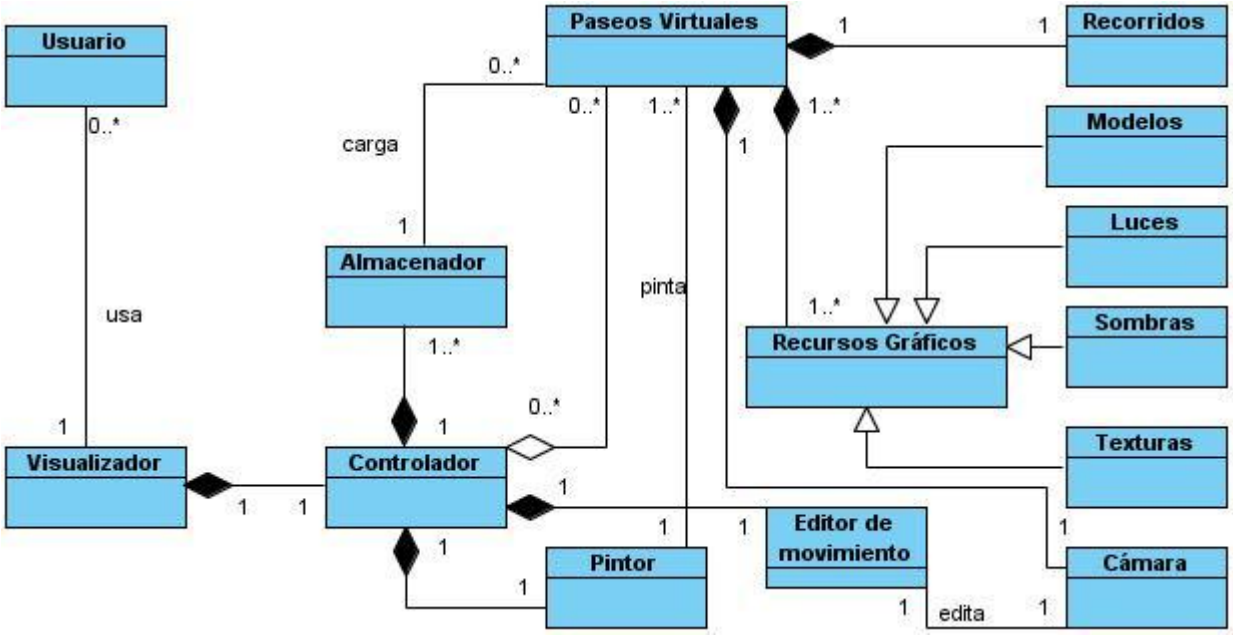



**Figura 7. Modelo conceptual del módulo Visualizador.**

## 3.2 Análisis y diseño de la herramienta.

El análisis proporciona una visión general del sistema que puede ser más difícil de obtener mediante el estudio de los resultados del diseño y la implementación debido a que contienen demasiados detalles. Un Modelo de análisis ofrece una especificación más precisa de los requisitos que la que se obtuvieron como resultado de la captura de requisitos, se describe utilizando el lenguaje de los desarrolladores, y puede por tanto introducir un mayor formalismo y ser utilizado para razonar sobre los funcionamientos internos del sistema, puede considerarse como una primera aproximación al modelo de diseño y es por tanto una entrada fundamental cuando se da forma al sistema en el diseño y en la implementación. El modelo de diseño se crea tomando el modelo de análisis como punto de partida, es el centro de atención al final de la fase de elaboración y el comienzo de las iteraciones de la construcción.

### 3.2.1 Vista lógica.

La arquitectura es la base estructural de un sistema informático, lo describe desde varios puntos de vista; contiene lineamientos muy importantes que establecen esquemas a los que se deben ajustar los componentes del sistema. La misma se representa en 4+1 vistas arquitectónicas que son: la vista lógica, vista de procesos, vista de despliegue, vista de implementación y la vista de casos de uso que es la rectora. En el capítulo anterior se realizó la vista de casos de uso arquitectónicamente significativos, partiendo de ella se realiza el diseño de la vista lógica de la herramienta que se propone.

La herramienta que se propone realizar es una aplicación de escritorio la cual solo necesitará de una computadora para su ejecución. En la siguiente figura se representa la organización del sistema en capas utilizando el patrón 3 capas que permite simplificar y comprender el desarrollo del sistema reduciendo la dependencia de forma que las capas más bajas no sean conscientes de ningún detalle de las superiores. En cada una de las capas se representan las clases arquitectónicamente significativas organizadas en paquetes.

1. Capa de presentación:

Paquete Interfaz de usuario: Contiene las clases arquitectónicamente significativas que representan las interfaces de usuario encargadas de enviar las peticiones del cliente a las clases que se encuentran en la capa de aplicación.

2. Capa de negocio:

Paquete Lógica del negocio: Contiene las clases y funcionalidades de control y ejecución de las peticiones

que llegan desde la capa de interfaz de usuario. Sirve como intermediario o mediado entre la lógica de presentación y la lógica de negocio.

Paquete de Dominio: Contiene las clases del negocio y sus funcionalidades.

3. Capa de datos:

Acceso a Datos: Contiene la clase y funcionalidades de acceso a los datos es decir la información almacena en ficheros. Sirve como intermediario o mediado entre la lógica de datos y la lógica de negocio.

CEV Ficheros: Recoge las especificaciones del almacenamiento de los datos en ficheros.

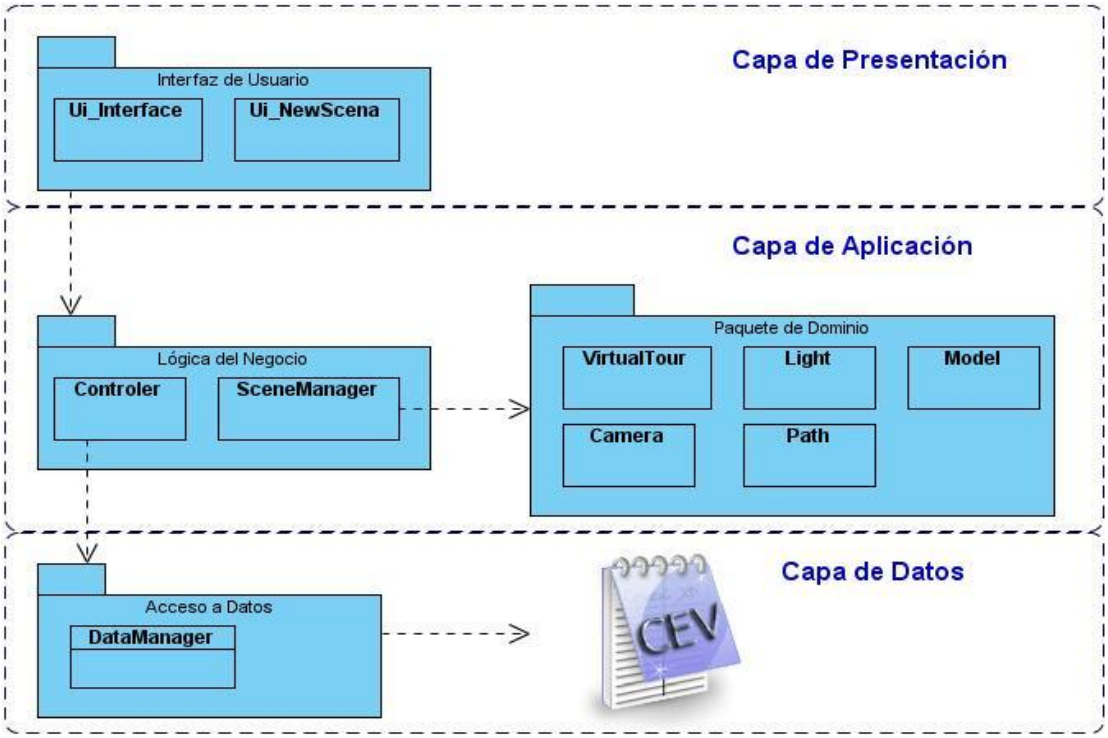



**Figura 8. Vista Lógica de la herramienta.**

### 3.2.2 Diagramas de clases.

Los diagramas de clases se utilizan para mostrar clases y sus relaciones, para saber las clases que participan en la realización de un caso de uso y las responsabilidades que deberán cumplir. En el modelo de análisis se utilizan tres estereotipos diferentes sobre clases (clase interfaz, clase de control y clase entidad). Las clases diseñadas en el análisis especifican y sugieren clases de diseño más refinadas que se adaptan al entorno de implementación. El diagrama de clases del diseño incluye más detalles que el

diagrama de clases del modelo de análisis lo cual es necesario para la adaptación del modelo de diseño al entorno de la implementación.

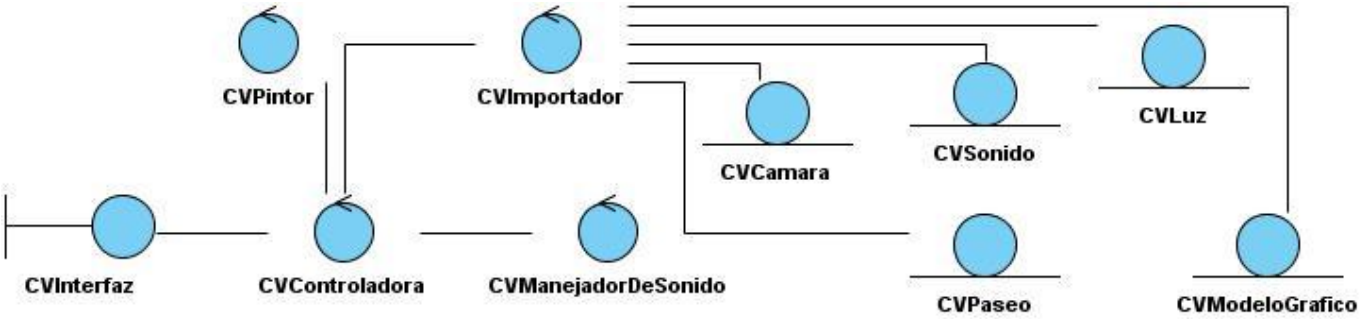



**Figura 9. Diagrama de clases del análisis. Módulo Visualizador.**

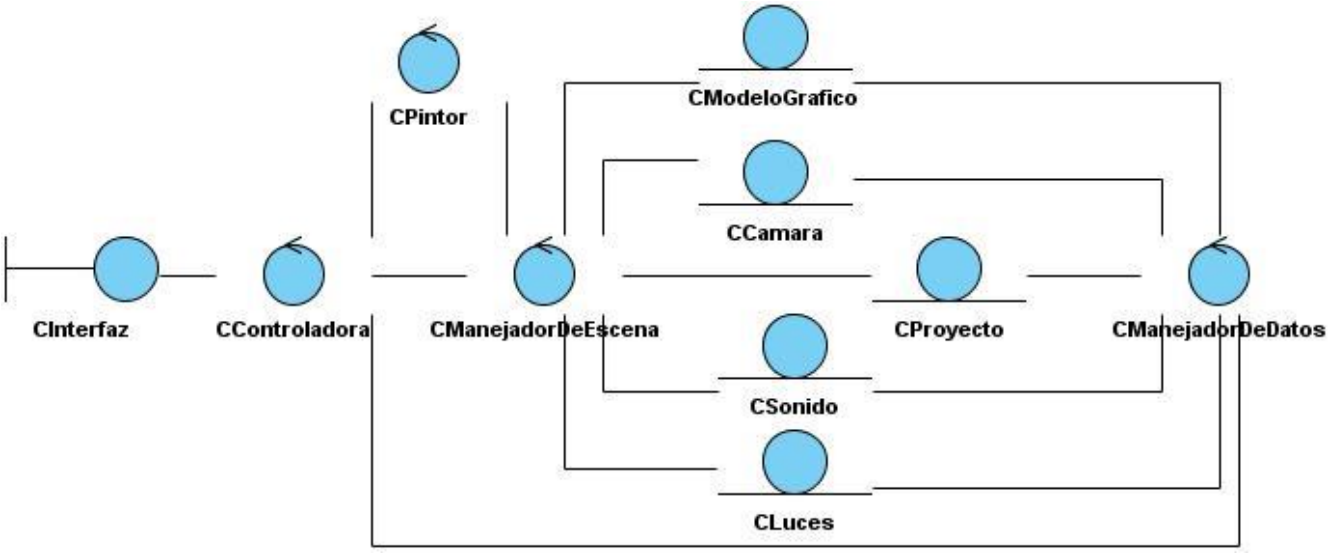



**Figura 10. Diagrama de clases del análisis. Módulo Editor.**

Las Figuras 8 y Figura 9 representan las clases identificadas durante el análisis de la herramienta para llevar a cabo la realización de los casos de uso de forma general.

Leyenda:

- Clases que se encuentran en la capa Presentación.
- Clases que se encuentran en la capa Aplicación.
- Clases que se encuentran en la capa Acceso a Datos.

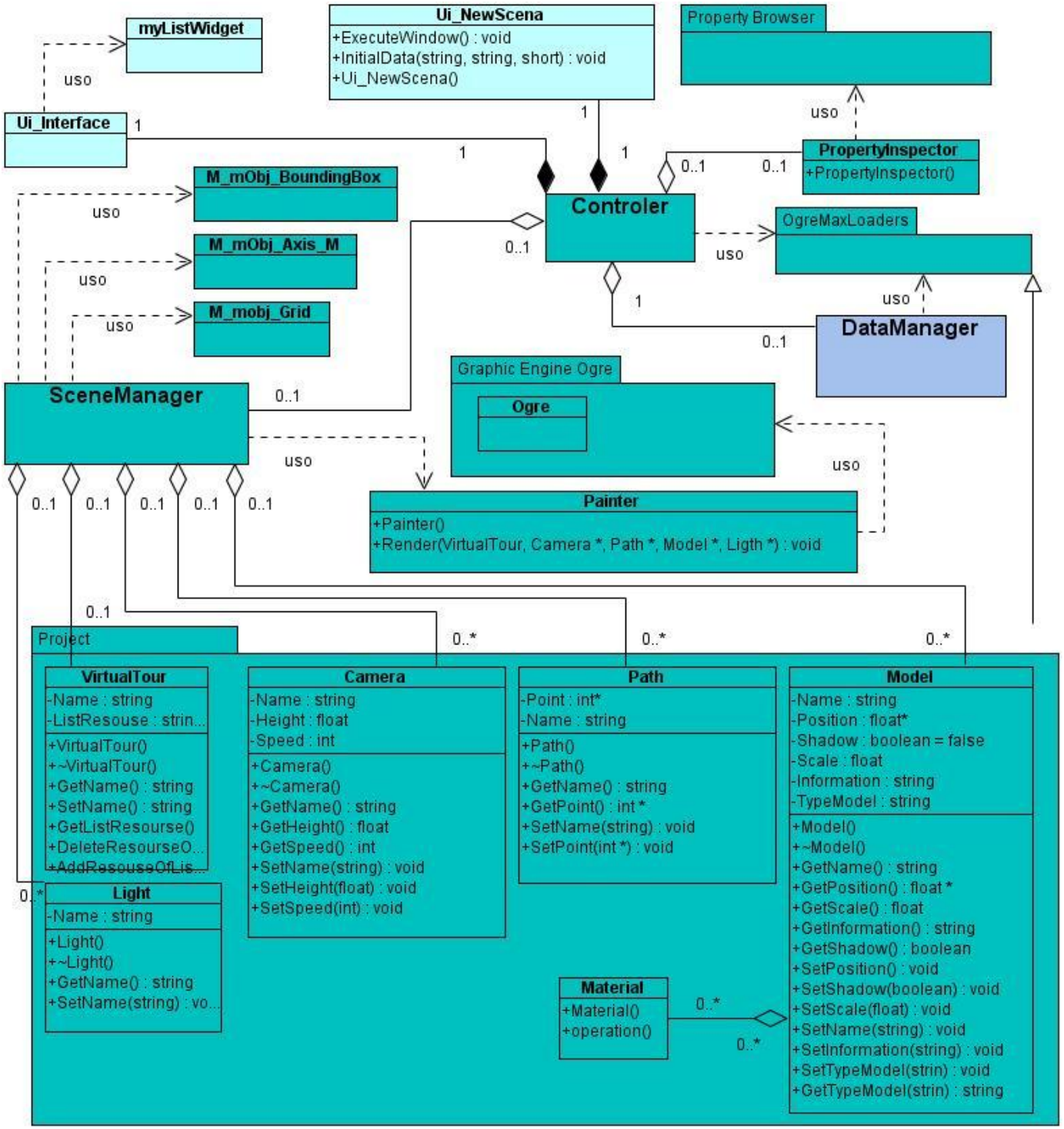



**Figura 11. Diagrama de clases del diseño del módulo Editor.**

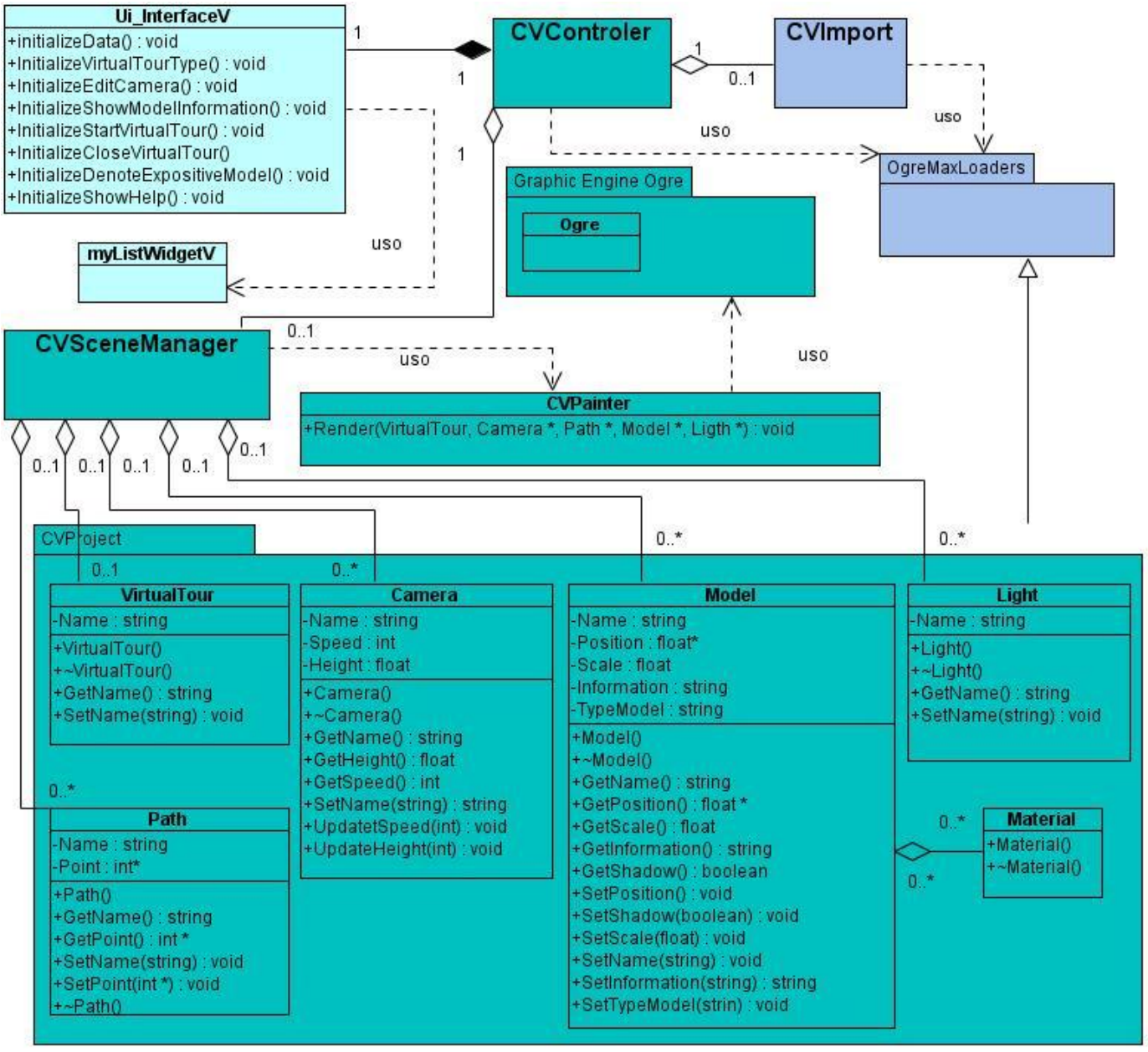



**Figura 12. Diagrama de clases del diseño del módulo Visualizador.**

Los diagramas de clases del diseño que se muestran en la Figura 10 y Figura 11 representan las clases, sus relaciones y sus responsabilidades.

### 3.2.3 Descripción de las clases.

A continuación se realiza una breve descripción de las clases arquitectónicamente significativas para el diseño de la herramienta. Las descripciones más detalladas de todas las clases identificadas pueden encontrarse en el Anexo 3 del documento.

**Clases arquitectónicamente significativas del módulo Editor.**

"SceneManager".Clase controladora que tiene la responsabilidad de administrar los datos de las entidades que maneja el sistema.

"Controler". Clase controladora que tiene la responsabilidad de enviar las respuestas a las peticiones que realiza el usuario hacia la clase Interfaz. Contiene todas las funcionalidades que dan respuesta a los casos de uso del sistema.

"DataManager". Clase controladora que tiene la responsabilidad de administrar los ficheros que almacenan los datos persistentes del sistema.

La estructura detallada de las clases anteriormente descritas se puede observar de forma consecutiva en el Anexo 6 mediante su representación en UML.

**Clases arquitectónicamente significativas del módulo Visualizador.**

"CVSceneManager".Clase controladora que tiene la responsabilidad de administrar los datos de las entidades que maneja el sistema.

"CVControler". Clase controladora que tiene la responsabilidad de enviar las respuestas a las peticiones que realiza el usuario hacia la clase Interfaz. Contiene todas las funcionalidades que dan respuesta a los casos de uso del sistema.

"CVImport". Clase controladora que tiene la responsabilidad de administrar los ficheros que almacenan los datos persistentes del sistema.

La estructura detallada de las clases anteriormente descritas se puede observar de forma consecutiva en el Anexo 6 mediante su representación en UML.

### 3.2.4 Realización de casos de uso.

La realización de los casos de uso se describe durante el análisis y el diseño a través de los diagramas de colaboración y los diagramas de secuencia. Los diagramas de colaboración se utilizan en el modelo de análisis mostrando cómo el control pasa de un objeto a otro a medida que se lleva a cabo el caso de uso, y los mensajes que se envían entre los objetos. El nombre del mensaje indica el motivo del objeto que realiza la llamada en su interacción con el objeto invocado. La realización de un caso de uso en el modelo

de diseño describe como se realiza el caso de uso en términos de las clases de diseño correspondientes. Para modelar las interacciones entre los objetos del diseño se utiliza el diagrama de secuencia. En las siguientes figuras se representan los diagramas de secuencia de los casos de uso arquitectónicamente significativos que fueron realizados durante el diseño, el resto de los diagramas de cada realización de caso de uso se podrán encontrar en el Anexo 4. Para cada escenario de un caso de uso se realizó un diagrama de colaboración y uno de secuencia.

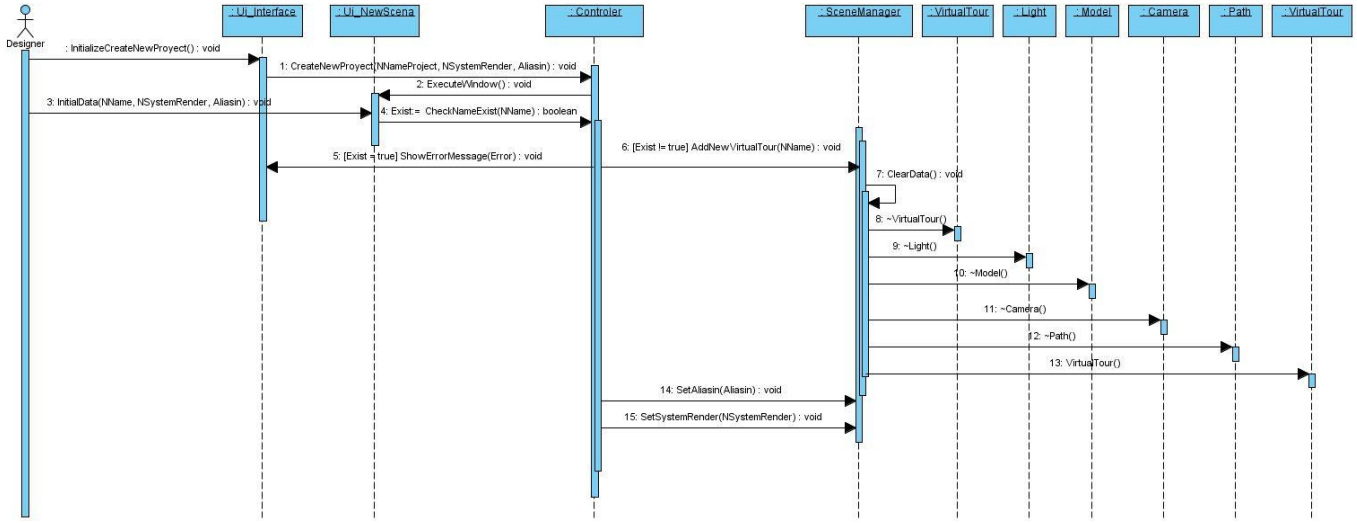



**Figura 13. Diagrama de secuencia CU-1 Administrar Proyecto Escenario Crear Proyecto.**

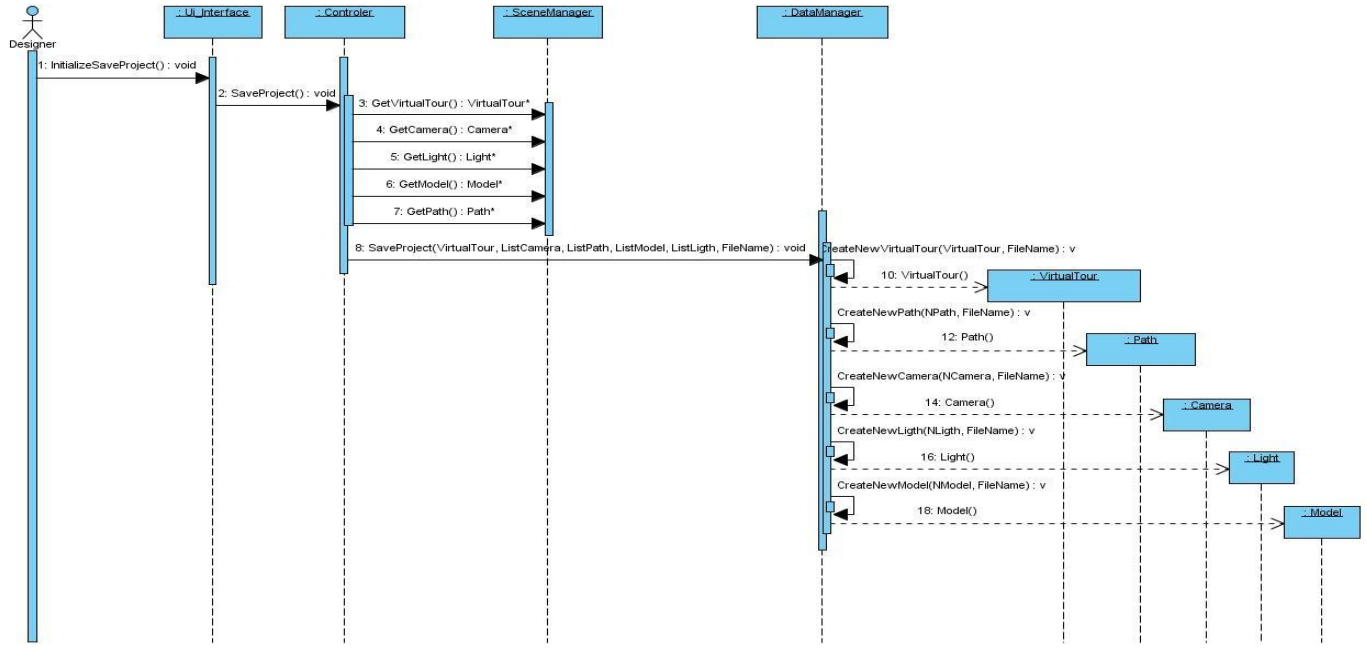



**Figura 14. Diagrama de secuencia CU-1 Administrar Proyecto Escenario Salvar Proyecto.**

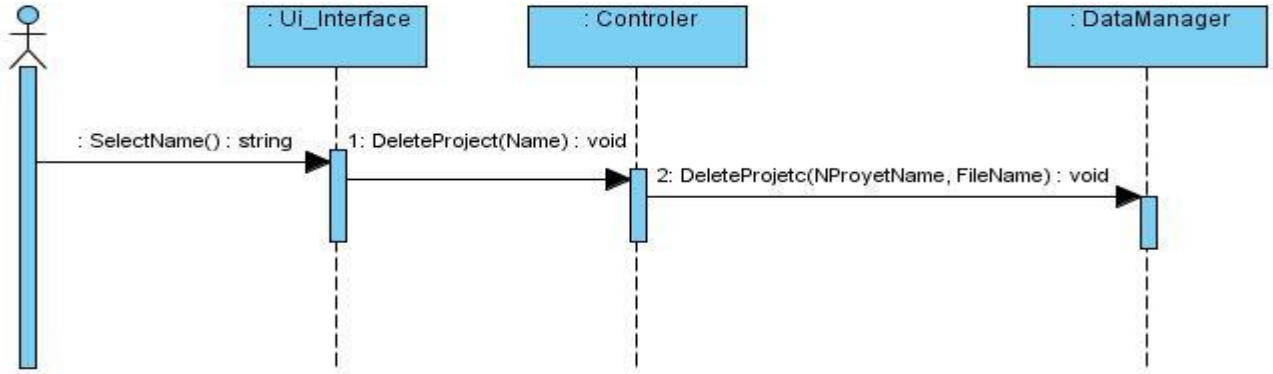



**Figura 15. Diagrama de secuencia CU-1 Administrar Proyecto Escenario Eliminar Proyecto.**

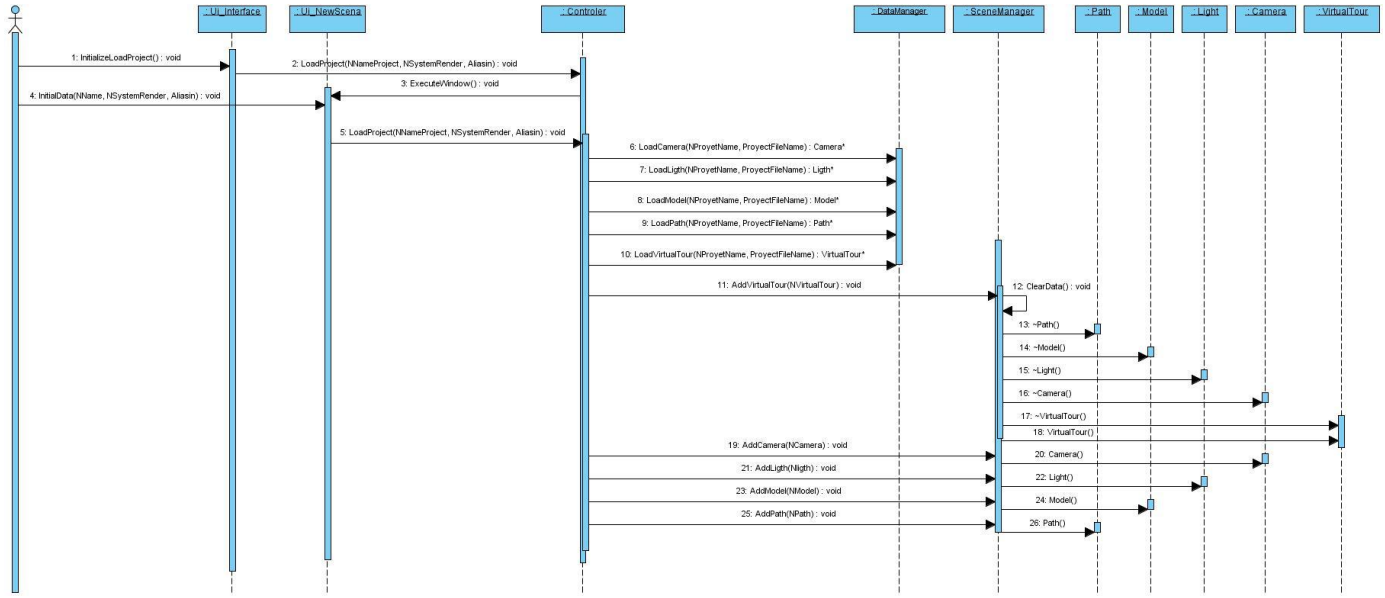



**Figura 16. Diagrama de secuencia CU-1 Administrar Proyecto Escenario Cargar Proyecto.**

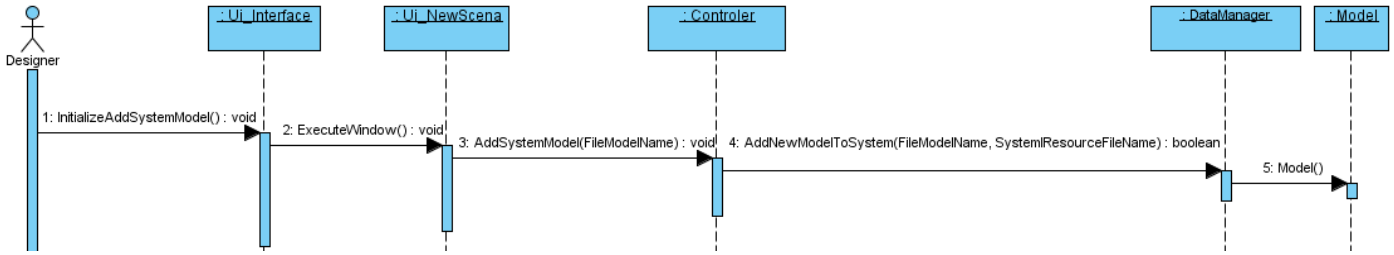



**Figura 17. Diagrama de secuencia CU-2 Administrar Modelos del Sistema Escenario Adicionar Modelo.**

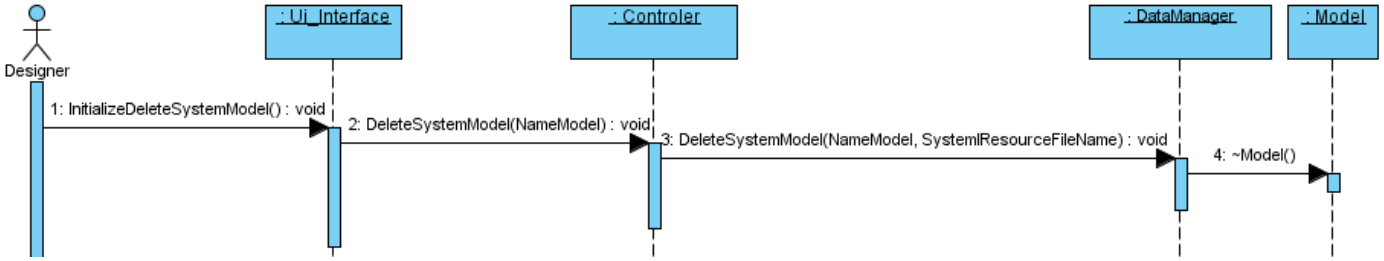



**Figura 18. Diagrama de secuencia CU-2 Administrar Modelos del Sistema Escenario Eliminar Modelo.**

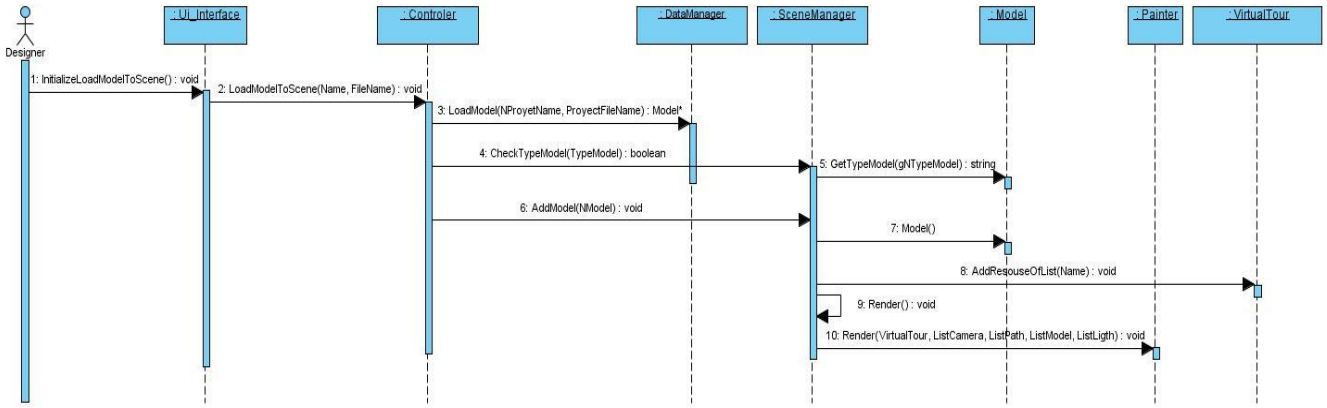



**Figura 19. Diagrama de secuencia CU-3 Administrar Objetos del Entorno Escenario Cargar Objeto en Entorno.**

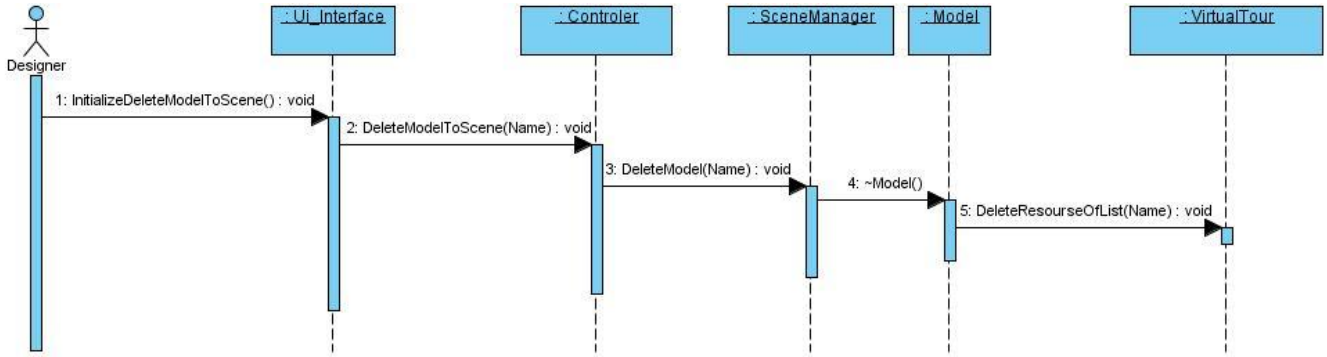



**Figura 20. Diagrama de secuencia CU-3 Administrar Objetos del Entorno Escenario Eliminar Objeto de Entorno.**

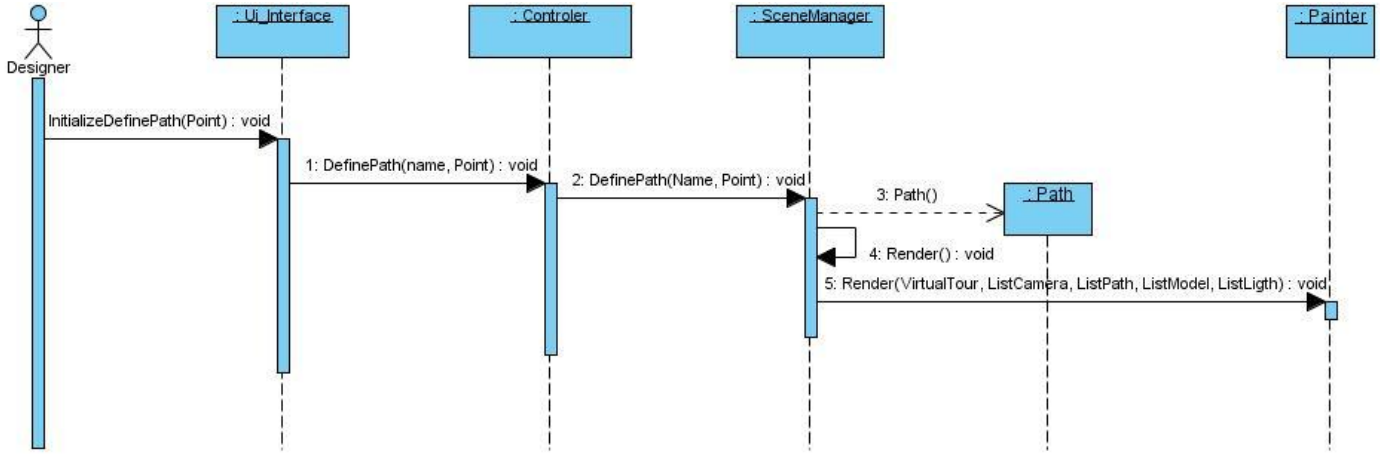



**Figura 21. Diagrama de secuencia CU-7 Administrar Cámara Escenario Definir Trayectoria.**

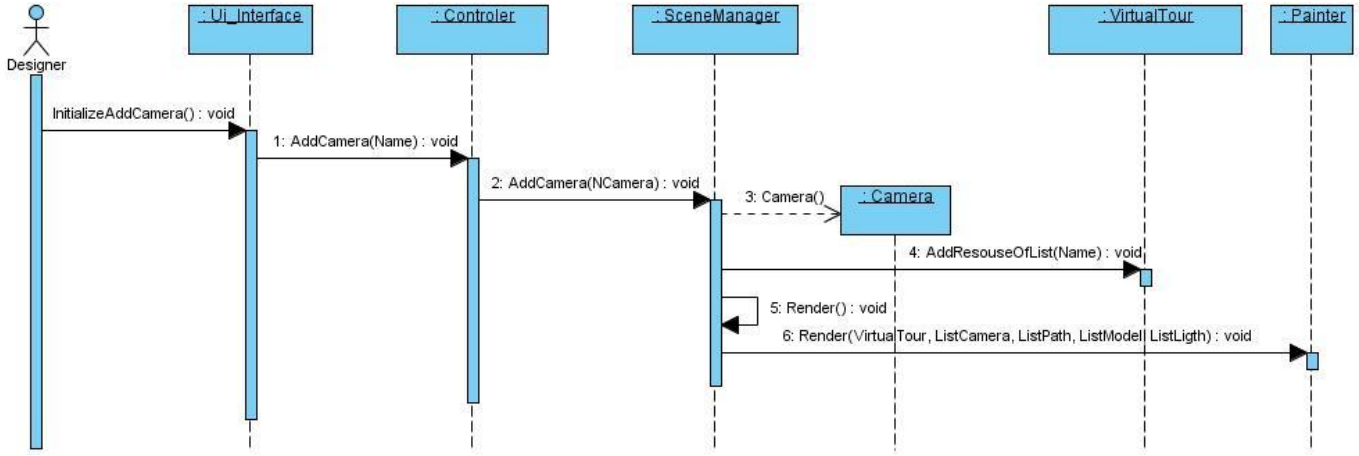



**Figura 22. Diagrama de secuencia CU-7 Administrar Cámara Escenario Adicionar Cámara.**

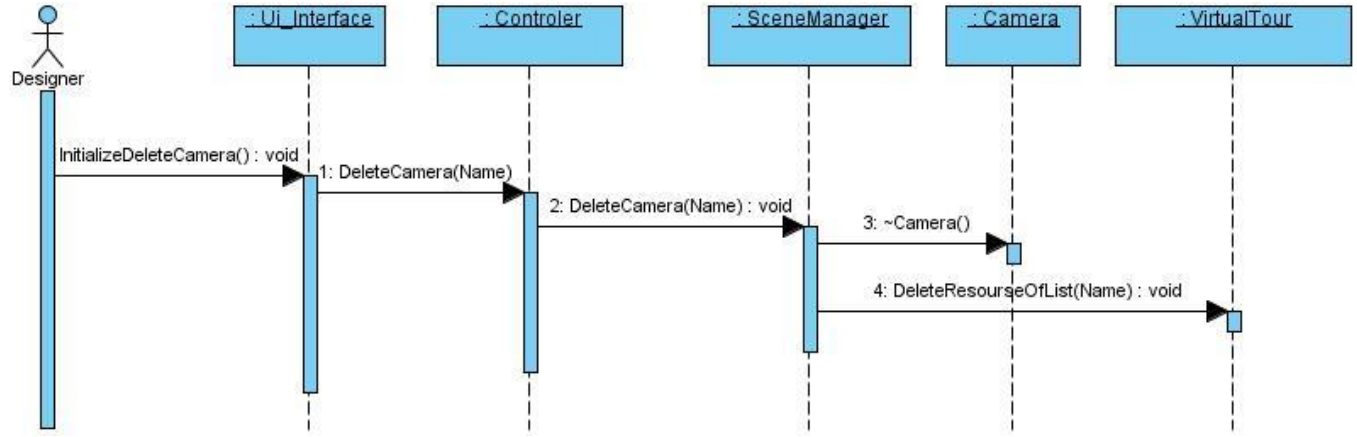



**Figura 23. Diagrama de secuencia CU-7 Administrar Cámara Escenario Eliminar Cámara.**

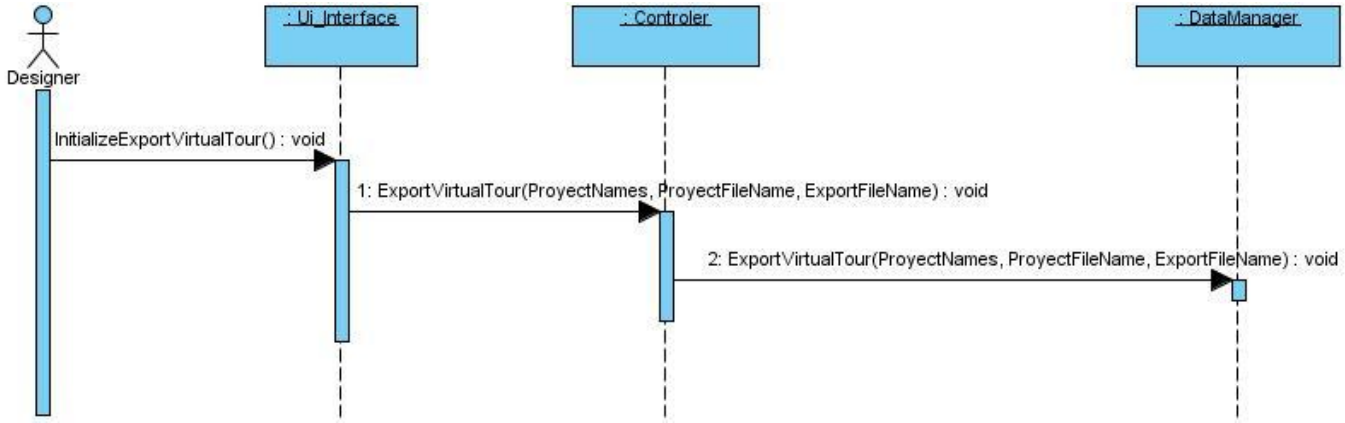



**Figura 24. Diagrama de secuencia CU-9 Exportar Paseo.**

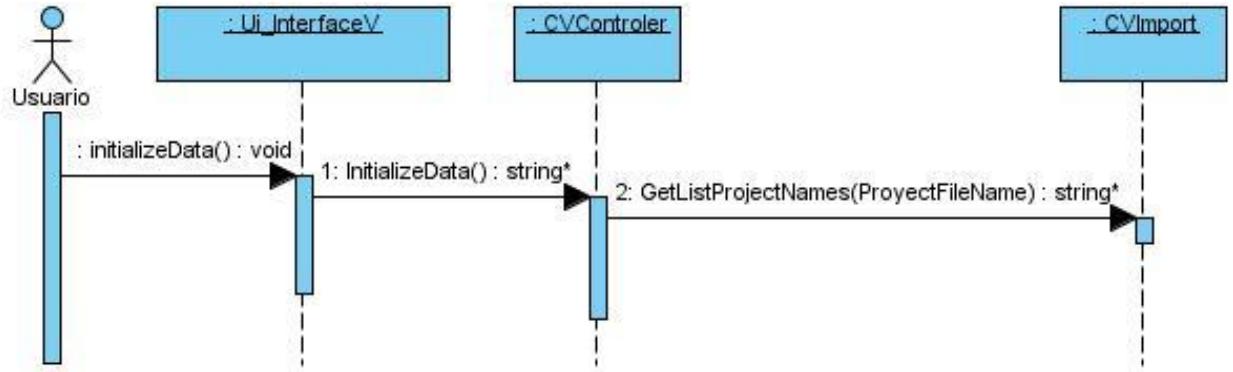



**Figura 25. Diagrama de secuencia CU-1 Inicializar Datos.**

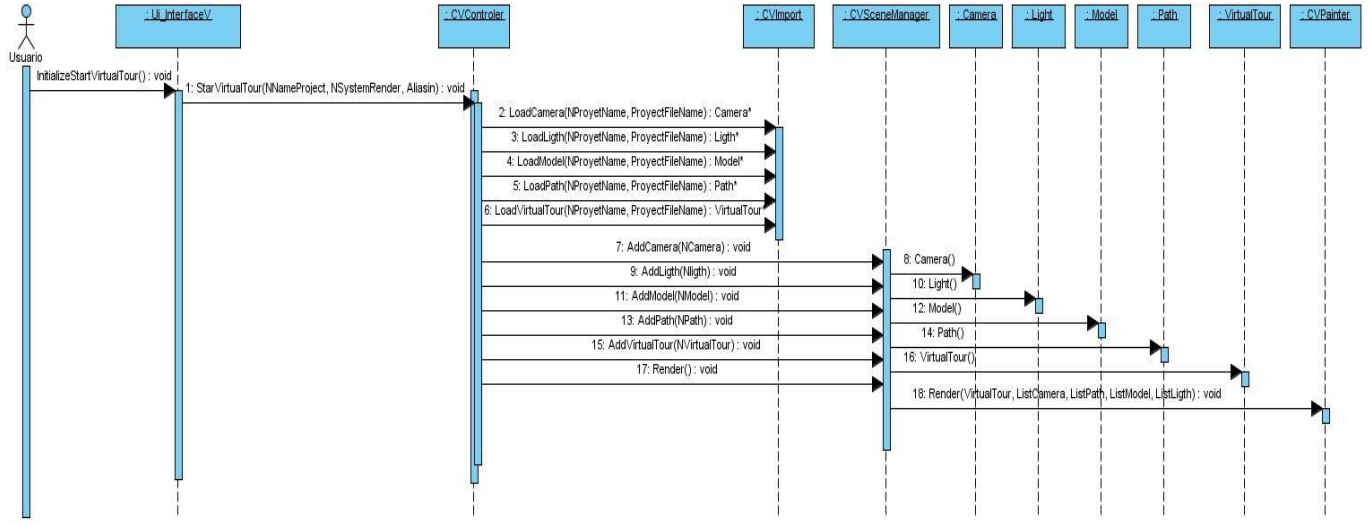



**Figura 26. Diagrama de secuencia CU-6 Comenzar Paseo.**

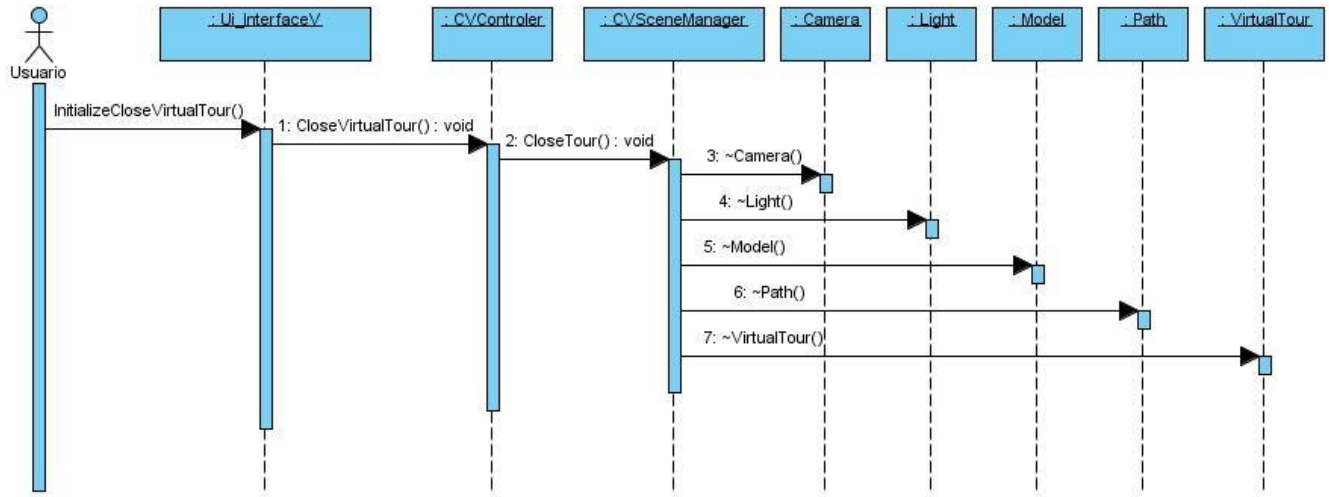



**Figura 27. Diagrama de secuencia CU-7 Cerrar Paseo.**

### 3.2.5 Prototipo de interfaz.

El diseño de interfaces de usuario es una tarea que ha adquirido relevancia en el desarrollo de un sistema. Un prototipo de interfaz representa un conjunto de elementos que le permiten al usuario interactuar con el sistema. Durante el presente epígrafe se muestra el prototipo de interfaz diseñado para ambos módulos de la herramienta teniendo en cuenta los requisitos solicitados por el cliente y las funcionalidades

definidas para el sistema y se describen los elementos de la interfaz que le dan respuesta a los casos de uso arquitectónicamente significativos. En las figuras se resaltan mediante un recuadro los elementos que dan respuesta a los casos de uso y pueden identificarse con una letra.

En la Figura 28 se representa la interfaz de usuario del módulo editor la cual de forma general está conformada por 7 áreas que se describen a continuación.

Barra de Títulos: Se muestra el nombre de la herramienta, el proyecto que esté cargado en escena y los botones para maximizar, minimizar o cerrar la herramienta.

Área “A”: Se resalta en un recuadro la barra de tarea donde se encuentra elementos de acceso rápido que le dan respuesta algunas de las funcionalidades del sistema.

Área “B”: Se resalta en un recuadro el componente Explorer, permite listar los nombres de los objetos que se encuentren en la escena (área D).

Área “C”: Proyectos, componente que permite listar los nombres de los proyectos que se encuentran en la carpeta del sistema.

Área “D”: Render, área de edición y visualización de la escena.

Área “E”: Propiedades, componente que muestra los atributos de los recursos (modelos, cámara, recorridos, luces) de la escena y permite editarlos.

Área “F”: Mallas, componente que permite listar los modelos gráficos que se encuentren en el sistema.

Área “G”: Cargar Proyecto, ventana que permite introducir los datos necesarios para crear o cargar un proyecto.

A continuación se mencionan los componentes de la propuesta de interfaz de usuario que dan respuesta a los casos de uso arquitectónicamente significativos del módulo Editor:

CU-1: Administrar proyecto.

La aplicación da respuesta mediante los componentes de la interfaz situados en las áreas “A”, “C” y “G” de la Figura 28.

CU-2: Administrar modelos del sistema.

La aplicación da respuesta mediante uno de los componentes que se encuentra en la barra de herramienta y el área “A" y “F”. Ver Figura 28.

CU-3: Administrar objetos del entorno.

La aplicación da respuesta mediante uno de los componentes que se encuentra en el área “A”, “B” y “F”. Ver Figura 28.

CU-7: Administrar cámara.

La aplicación da respuesta mediante uno de los componentes que se encuentra en la barra de herramienta y el componente Propiedades del área “E”. Ver Figura 28.

CU-9: Exportar paseo.

La aplicación da respuesta mediante uno de los componentes que se encuentra en el área “A”. Ver Figura 28.

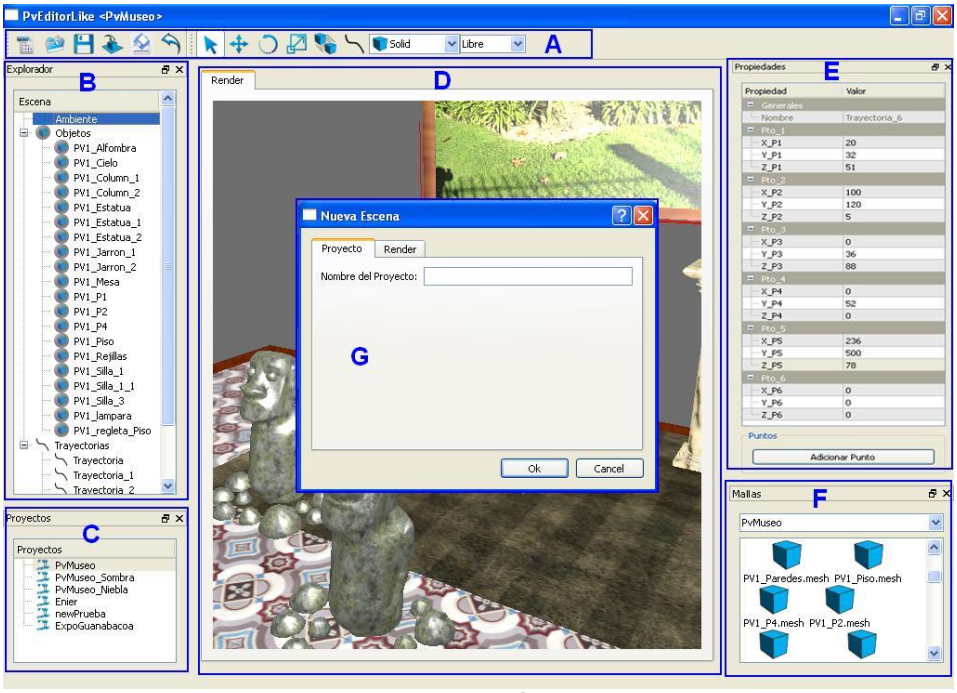



**Figura 28. Interfaz Gráfica del Editor.**

En las Figuras 29 y 30 se muestra la interfaz de usuario del módulo Visualizador, se describen a continuación los componentes de la interfaz.

Figura 29-A: Área donde se resalta en un recuadro la barra de tarea donde se encuentra elementos de acceso rápido que le dan respuesta algunas de las funcionalidades del sistema.

Figura 29-B: Área de Proyectos, componente que permite listar los nombres de los proyectos que se encuentran en la carpeta del sistema.

Figura 29-C: Área que muestra la ayuda que especifica información referente a cómo utilizar el teclado y el mouse (ratón) para desplazarse por el paseo virtual.

Figura 30-B: Área de Render, área de visualización que muestra un paseo virtual.

Figura 30-C: Área para confirmar cerrar un paseo.

A continuación se mencionan los componentes de la propuesta de interfaz de usuario que dan respuesta a los casos de uso arquitectónicamente significativos del módulo Visualizador:

CU-1. Inicializar Datos.

La aplicación da respuesta mediante los componentes que se encuentran en el área B (Paseos Virtuales), Ver Figura 29-A.

CU-6. Comenzar paseo.

La aplicación da respuesta mediante uno de los componentes que se encuentra en el área B. Ver Figura 29-A.

CU-7. Cerrar paseo.

La aplicación da respuesta mediante los componentes que se encuentran en el área A y C. Ver Figura 30-A.

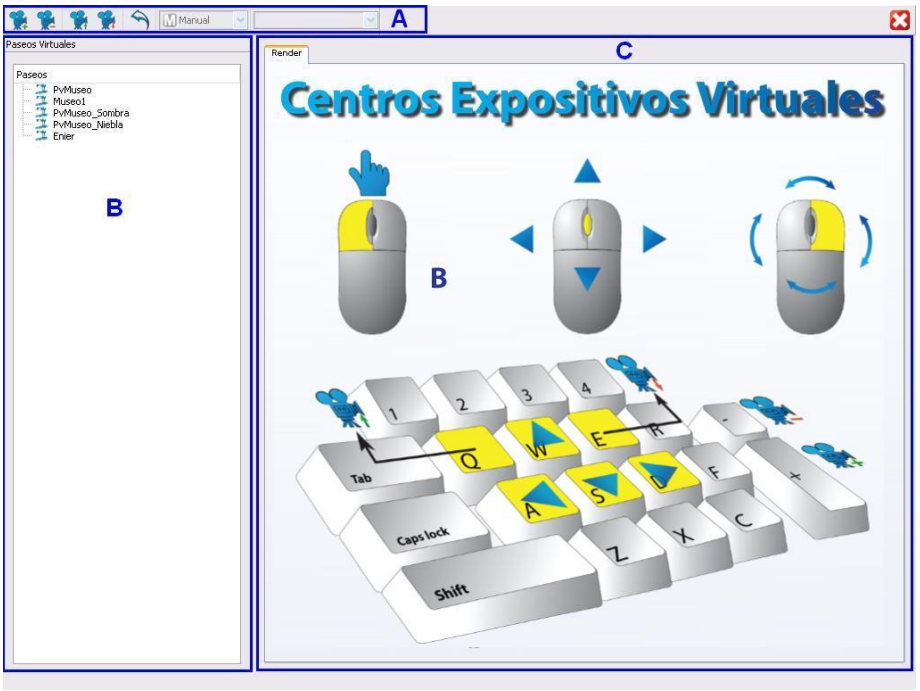



**Figura 29. Interfaz Gráfica del Visualizador.**

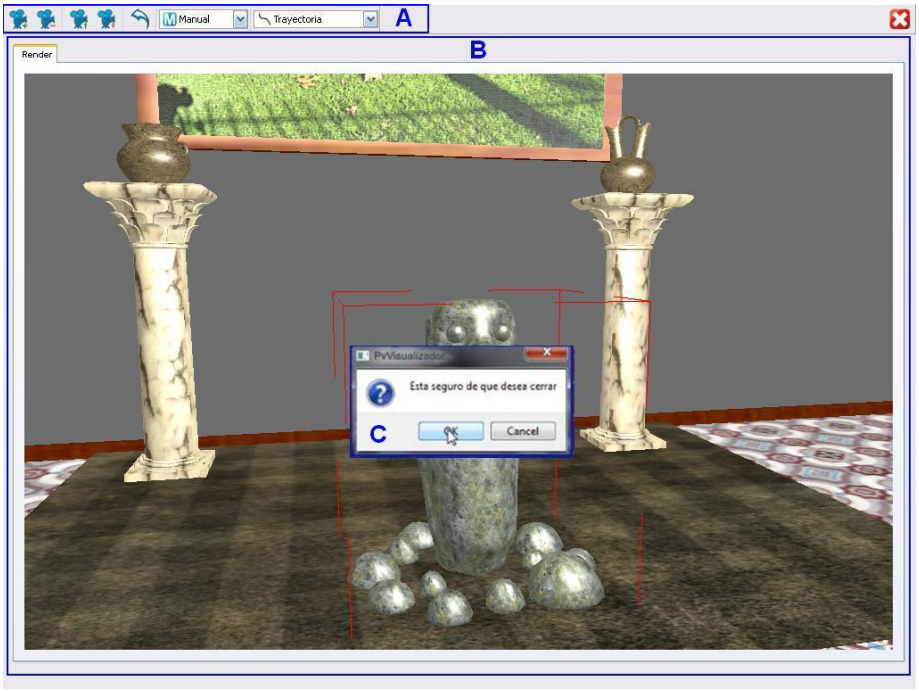



**Figura 30. Interfaz Gráfica del Visualizador.**

## Consideraciones del capítulo.

En el capítulo se esbozaron los modelos conceptuales de los módulos edición y visualización de la herramienta lo que da un primer acercamiento en términos conceptuales a la propuesta de solución. Se crearon y representaron los diagramas de clases del análisis y los diagramas de colaboración para cada escenario de los casos de uso para obtener una especificación más precisa de los requisitos y describir la realización de los casos de uso. Se realizaron los diagramas de clases del diseño de ambos módulos y se describieron las principales clases, de esta forma se obtuvieron las clases que participan en la realización de un caso de uso y sus responsabilidades. Se realizaron los diagramas de secuencia del diseño para cada escenario de cada caso de uso para obtener de forma más detallada las interacciones entre los objetos del diseño y finalmente se diseñó una propuesta de interfaz sencilla para dar respuesta a las funcionalidades de la herramienta.

# Conclusiones.

Con la conclusión del trabajo realizado se demuestra que el objetivo de la investigación se alcanzó satisfactoriamente. Se logró obtener la especificación y documentación que permite implementar una herramienta para la creación, edición y visualización de Centros Expositivos Virtuales interactivos mediante la descripción de los requisitos y casos de uso identificados para el sistema, la representación técnica y el diseño del prototipo de interfaz que da respuesta a las funcionalidades identificadas para la herramienta. El diseño obtenido como resultado de la investigación realizada se está aplicando actualmente en la implementación de una herramienta.

# Recomendaciones.

Es importante para futuros desarrollos:

- Realizar un estudio sobre las herramientas libres para la Administración de requisitos.
- Implementar y generalizar el uso de la propuesta actual.
- Identificar nuevos requisitos que van surgiendo con el desarrollo de nuevas aplicaciones y que permitirán ir actualizando la propuesta actual.

# Referencias Bibliográficas.

**Alexander , Ian F. y Stevens, Richard . 2002.** *Writing Better Requirements.* Reino Unido : Addison Wesley, 2002. 0-321-13163-0.

**3D Profesional. 2011.** www.3dprofesional.com. *software.* [En línea] 3D Profesional, 2011. [Citado el: 14 de Marzo de 2011.] http://www.3dprofesional.com/software/index.html.

**Answer. 2009.** Answer.com. *Answer.com.* [En línea] Answers Corporation, 1 de Noviembre de 2009. [Citado el: 2 de Noviembre de 2010.] http://www.answers.com/topic/virtual-reality.

**Blender Fundation. 2011.** blender. *Blender.* [En línea] Blender Fundation, 23 de Marzo de 2011. [Citado el: 23 de Marzo de 2011.] http://www.blender.org/.

**Corporation, Microsoft. 2009.** *Microsoft Application Arquitecture Guide.* 2da Edición. Estados Unidos : Microsoft Corporation, 2009. 9780735627109.

**González Duque, Raúl . 2011.** Mundo Geek . *El lenguaje de los grandes programadores .* [En línea] 2011. [Citado el: 29 de Mayo de 2011.] http://mundogeek.net/archivos/2011/05/08/el-lenguaje-de-los-grandes-programadores/.

**—. 2008.** Universidad Nacional de San Luis, Argentina. *Python para todos.* [En línea] 6 de Mayo de 2008. [Citado el: 24 de Enero de 2011.] www.dirinfo.unsl.edu.ar/jornadas/img/ebooks/pythonparatodos.pdf.

**González López, Minardo Gollún. 2011.** Informática XIV Convención y Feria Internacional 2011. *XIV Convención y Feria Internacional Informática 2011.* [En línea] 07 de Febrero de 2011. [Citado el: 09 de Mayo de 2011.] www.informaticahabana.cu. 978-959-7213-01-07.

**Grupo Soluciones Innova S.A. 2011.** rational.com. *Grupo Soluciones. GSInnova.* [En línea] Grupo Soluciones Innova S.A, 4 de Marzo de 2011. [Citado el: 4 de Marzo de 2011.] http://www.rational.com.ar/apertura/acerca.html.

**IBM. 2011.** Rational Rose Enterprise. *www.ibm.com.* [En línea] IBM, 2011. [Citado el: 16 de 02 de 2011.] http://www-142.ibm.com/software/products/es/es/enterprise/.

**—. 2011.** developerWorks. *www.ibm.com.* [En línea] 2011. [Citado el: 3 de Febrero de 2011.] http://www.ibm.com/developerworks/rational/library/content/03July/1000/1251/1251_bestpractices_TP026B.pdf.

**—. 2011.** IBM. *IBM. Rational Rose Enterprise.* [En línea] IBM, 2011. [Citado el: 23 de Enero de 2011.] http://www-142.ibm.com/software/products/es/es/enterprise/.

**—. 2011.** Rational RequisitePro. *www-142.ibm.com.* [En línea] IBM, 3 de Marzo de 2011. [Citado el: 3 de Marzo de 2011.] http://www-142.ibm.com/software/products/es/es/reqpro/.

**Jacobson, Ivar, Booch, Grady y Rumbaugh, James. 2000.** *El proceso unificado de desarrollo de software.* España : Addison Wesley, 2000. 84-7829-036-2.

**Larman, Craig. 1999.** *UML Y PATRONES. Introducción al análisis y diseño orientado a objetos.* México : PRENTICE HALL, 1999. 970-17-0261-1.

**MIT-Microsoft Alliance . 2003.** iCampus the MIT-Microsoft Alliance. *Proyect: Games to Teach.* [En línea] MIT, 6 de 2003. [Citado el: 23 de 3 de 2011.] http://icampus.mit.edu/projects/GamesToTeach.shtml.

**Nokia, Corporation,. 2008.** QT Nokia. *Qt Creator IDE and tools.* [En línea] Nokia Corporation, 2008. [Citado el: 1 de Junio de 2011.] http://qt.nokia.com/products/developer-tools/.

**OMG, Object Management Group. 2011.** OMG weset the standard. *Unified Modeling Language.* [En línea] 14 de Febrero de 2011. [Citado el: 15 de Febrero de 2011.] http://www.uml.org/.

**Pressman, Roger S. 2001.** *Ingeniería del Software. Un enfoque práctico. Quinta edisión.* España : McGraw-Hill, 2001. 8448132149.

**Ron , Jeffries. 2001.** XProgramming.com. *What is Extreme Programming?* [En línea] 8 de Noviembre de 2001. [Citado el: 16 de Enero de 2011.]

**Rumbaugh, James, Jacobson, Ivar y Booch, Grady. 2000.** *El lenguaje unificado de modelado. Manual de referencia.* España : Addison Wesley, 2000. 8478290370 .

**Seta, Leonardo De. 2010.** DosIdeas. *www.dosideas.com.* [En línea] DosIdeas, 3 de 11 de 2010. [Citado el: 3 de 11 de 2010.] http://www.dosideas.com/noticias/actualidad/109-entrevista-a-bjarne-stroustrup-creador-de-c.html.

**Tirado Granados, Gonzalo. 2008.** *Introducción a Ogre 3D.* [PDF] 2008.

**Visual Paradigm. 2011.** Visual Paradigm for UML. *www.visual-paradigm.com.* [En línea] Visual Paradigm, 3 de Marzo de 2011. [Citado el: 3 de Marzo de 2011.] http://www.visual-paradigm.com/product/vpuml/.

**Wells, Don. 2009.** extremeprogramming. *Extreme Programming: A gentle introduction.* [En línea] 8 de Noviembre de 2009. [Citado el: 25 de Enero de 2011.] http://www.extremeprogramming.org/.

# Bibliografía Consultada

**Alexander , Ian F. y Stevens, Richard . 2002.** *Writing Better Requirements.* Reino Unido : Addison Wesley, 2002. 0-321-13163-0.

**Agüera, Ivan Obeso. 2000.** Estudio de herramientas de análisis y gestión de requisitos. 2000.

**Cecilia. 2008.** blog ipcorp. [En línea] 11 de Diciembre de 2008. http://www.ipcorp.com.ar/blog/2008/12/11/osrmt-open-source-requirements-management-tool/.

**Código, Planeta. 2008.** Planeta código. [En línea] 18 de Junio de 2008. http://www.planetacodigo.com/wiki/glosario:matriz_de_trazabilidad.

**Corporation, Microsoft. 2009.** *Microsoft Application Arquitecture Guide.* 2da Edición. Estados Unidos : Microsoft Corporation, 2009. 9780735627109.

**Eva. 2005.** Entorno virtual de aprendizaje. *Eva.* [En línea] Ingeniería de software, 5 de Junio de 2005. http://eva.uci.cu/mod/resource/view.php?inpopup=true&id=22434.

**Hernández, Enrique Orallo. 2002.** El Lenguaje Unificado de Modelado (UML). 2002.

**Hill, Mc Graw. 1999.***INGENIERÍA DEL SOFTWARE Un enfoque práctico. .* EEUU : Sexta edición, Capítulo 1, pág. 5., 1999.

**MIT-Microsoft Alliance . 2003.** iCampus the MIT-Microsoft Alliance. *Proyect: Games to Teach.* [En línea] MIT, 6 de 2003. [Citado el: 23 de 3 de 2011.] http://icampus.mit.edu/projects/GamesToTeach.shtml.

**Obeso, Ivan Agüera. 1999.***Estudio de herramientas de análisis y gestión de requisitos.* 1999.

**Ponjuán Dante, G. 2004.***Aspectos introductorios acerca de los sistemas en: Sistemas de información, Principios y aplicaciones.* La Habana : Félix Varela, 2004.

**Prof. Juan Carlos Gutiérrez Lázaro. 2008-2009.** UML Diagramas de clases y casos de uso. [En línea] 2008-2009. http://www.fdi.ucm.es/profesor/jcgutierrez/Tema%202/02UML-1.pdf.

**RAE. 2001.** Real Academia Española. [En línea] 2001. http://www.rae.es/rae.html..

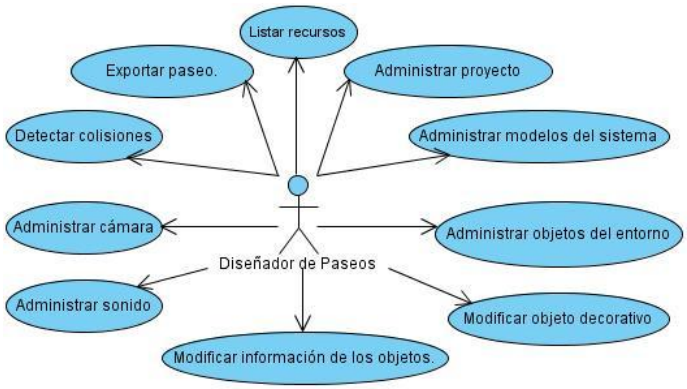
**Sommerville, Ian. 2005.** INGENIERÍA DEL SOFTWARE. Séptima edición . Madrid : PEARSON EDUCACIÓN.S.A, 2005, pág. 5.

# Glosario de términos.

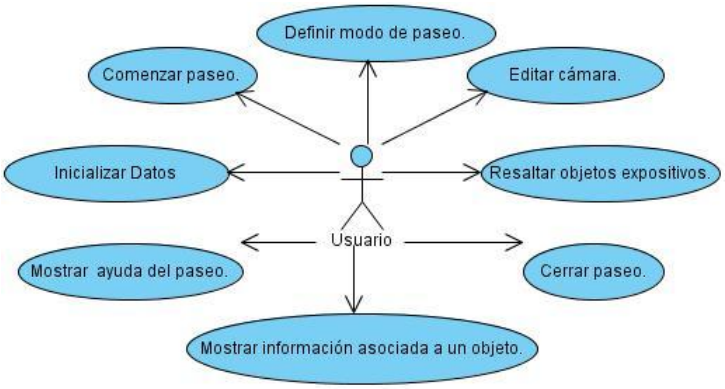
- **Aliasing:** Las líneas, especialmente las que están casi horizontalmente o verticales, aparecen dentadas o irregulares debido a su representación por píxeles. A este escalonamiento se le conoce como aliasing.
- **AntiAliasing:** Técnicas que se usan para reducir el Aliasing.
- **Centros expositivos virtuales:** Paseos Virtuales destinados a la exposición de objetos.
- **Escena**: Área virtual que contiene los modelos, luces, recorridos, cámara y todos los elementos necesarios para visualizar un paseo.
- **GDPV:** Grupo de Desarrollo de Paseos Virtuales.
- **Material:** Combinación de luces, colores e imágenes usados para definir una apariencia.
- **Modelo:** Representación abstracta, comúnmente estructurado por triángulos, colores y materiales que representan un objeto.
- **Paseos Virtuales:** Aplicaciones de la realidad virtual con abundante mercado en todo el mundo, básicamente son la simulación de un entorno empleando tecnología de realidad virtual.
- **Path:** Lista de vectores que definen una trayectoria a través de la escena.
- **Píxel:** Abreviatura de “picture element” Es la menor unidad homogénea en color que forma parte de una imagen digital.
- **Proyección:** Conversión de coordenadas 3D del mundo a las coordenadas 2D de un plano.
- **Perspectiva:** Referente a “proyección perspectiva”, determina los tamaños de los objetos basándose en la distancia de los objetos al plano de proyección.
- **Render:** (Como se usa en el documento) corresponde a la creación y representación como imagen 2D, de los modelos gráficos en una escena.
- **Textura:** Imagen que sirve de “piel” a los modelos en un mundo virtual.

# Anexos.

## Anexo 1. Diagrama de casos de uso.




**Anexo 1. Diagrama de casos de uso del módulo Editor.**




**Anexo 2. Diagrama de casos de uso del módulo Visualizador.**

## Anexo 2. Descripción textual de los casos de uso.

**Casos de uso expandidos para el módulo Editor.**

| **Caso de uso** | |
|---|---|
| **CU-4** | Modificar objeto decorativo. |
| **Propósito** | El objetivo del caso de uso es que el diseñador del paseo pueda modificar el tamaño de un objeto decorativo y la cantidad que puede existir en el paseo. |
| **Prioridad** | Secundario. |
| **Actores:** Diseñador del Paseo. | |

| | |
|---|---|
| **Resumen:** El diseñador del paseo inicializa el caso de uso cuando selecciona una de las opciones para modificar el tamaño y la cantidad de objetos decorativos de un mismo tipo que puede existir en el paseo. | |
| **Referencias** | RF 4, RF 4.1, RF 4.2. |
| **Precondiciones** | El objeto debe estar añadido al sistema. |
| **Poscondiciones** | Se modifica el tamaño o la cantidad de objetos del entorno. |

| **Curso Normal de Eventos** | |
|---|---|
| **Acción del actor** | **Respuesta del sistema** |
| 1. El diseñador selecciona un objeto decorativo y le aplica una de las opciones siguientes: Escalar objeto decorativo, Duplicar objeto decorativo. | 1.1 Si el diseñador selecciona la opción Escalar objetos decorativo ir al Escenario 1.<br>1.2 Si el diseñador selecciona la opción Duplicar objeto decorativo ir al Escenario 2. |

| **Escenario 1 Escalar objeto decorativo** | |
|---|---|
| **Curso Normal de Eventos** | |
| **Acción del actor** | **Respuesta del sistema** |
| 1. El diseñador modifica las dimensiones del objeto seleccionando la opción escalar y arrastrando el mouse sobre el (los) eje(s) de coordenada que se desea escalar el objeto, o modificando los atributos correspondientes a las dimensiones del objeto ubicado en el inspector de propiedades. | 1.1 El sistema modifica las dimensiones del objeto y lo muestra con las nuevas dimensiones en el espacio de visualización. |

| **Escenario 2 Duplicar objeto decorativo** | |
|---|---|
| **Curso Normal de Eventos** | |
| **Acción del actor** | **Respuesta del sistema** |

| | |
|---|---|
| 1. El diseñador elige la opción duplicar objeto. | 1.1 El sistema adiciona un nuevo modelo en el área de visualización con las mismas características del modelo seleccionado. El nombre del nuevo modelo tendrá el siguiente formato: “nombre del modelo duplicado_número”. |

| **Caso de uso** | |
|---|---|
| **CU-5** | Modificar información de los objetos. |
| **Propósito** | El objetivo del caso de uso es que el diseñador del paseo pueda modificar la información asociada a un objeto que se encuentre en el entorno o cambiarle el atributo “Tipo de objeto”. |
| **Prioridad** | Secundario. |
| **Actores:** Diseñador del Paseo. | |
| **Resumen:** El diseñador del paseo inicializa el caso de uso cuando Crea y modifica la información referente a un objeto y lo clasifica en decorativo o expositivo. | |
| **Referencias** | RF 5, RF 5.1, RF 5.2. |
| **Precondiciones** | El objeto debe estar añadido al sistema. |
| **Poscondiciones** | Se asocia una información a un objeto y se define el tipo de objeto que se encuentre en el escenario. |
| **Curso Normal de Eventos** | |
| **Acción del actor** | **Respuesta del sistema** |
| 1. El caso de uso se inicializa cuando el diseñador selecciona una de las opciones siguientes: Actualizar información de objeto, Definir tipo de objeto. | 1.1 Si el diseñador selecciona la opción Actualizar información de objeto ir al Escenario 1.<br>1.2 Si el diseñador selecciona la opción Definir tipo de objeto ir al Escenario 2. |
| **Escenario 1 Actualizar información de objeto** | |
| **Curso Normal de Eventos** | |
| **Acción del actor** | **Respuesta del sistema** |

| | |
|---|---|
| 1. El diseñador selecciona un objeto y modifica el texto en la propiedad del objeto Texto. | 1.1 El sistema guarda la actualización realizada en la propiedad del objeto. |
| **Escenario 4 Definir tipo de Objeto** | |
| **Curso Normal de Eventos** | |
| **Acción del actor** | **Respuesta del sistema** |
| 1. El diseñador selecciona un objeto y elige una de las opciones de la propiedad Tipo del objeto. Las opciones de la propiedad son Decorativo, Expositivo o Escenario. | 1.1 El sistema asigna la opción de la propiedad seleccionada al objeto. |

| **Caso de uso** | |
|---|---|
| **CU-6** | Administrar sonido. |
| **Propósito** | El objetivo del caso de uso es que el diseñador del paseo pueda adicionar y modificar un sonido al entorno. |
| **Prioridad** | Opcional. |
| **Actores:** Diseñador del Paseo. | |
| **Resumen:** El diseñador del paseo inicializa el caso de uso cuando adiciona un sonido al entorno y realiza modificaciones al mismo. | |
| **Referencias** | RF 6, RF 6.1, RF 6.2, RF 6.3, RF 6.4, RF 6.5. |
| **Poscondiciones** | Se asocia un sonido al entorno y se modifica el atributo que permite repetir sonido. Se adicionar sonido o eliminar un sonido del sistema. |
| **Curso Normal de Eventos** | |
| **Acción del actor** | **Respuesta del sistema** |
| 1. El diseñador elige una de las opciones siguientes: Adicionar sonido, Eliminar sonido, Repetir sonido, Asociar sonido. | 1.3 Si el diseñador selecciona la opción Adicionar sonido ir al Escenario 1.<br>1.4 Si el diseñador selecciona la opción Eliminar sonido ir al Escenario 2.<br>1.5 Si el diseñador selecciona la opción Repetir sonido ir al Escenario 3. |

| | |
|---|---|
| | 1.6 Si el diseñador selecciona la opción Asociar sonido ir al Escenario 4. |
| **Escenario 1 Adicionar Sonido** | |
| **Curso Normal de Eventos** | |
| **Acción del actor** | **Respuesta del sistema** |
| 1. El diseñador elige la opción Adicionar Sonido.<br>2. El diseñador selecciona el sonido desde la ubicación donde se encuentre.<br>3. El diseñador da clic en la opción aceptar. | 1.1 El sistema abre una venta como el explorador de Windows para que el usuario busque el sonido.<br>3.1 El sistema agrega el sonido como un recurso del sistema disponible para cargarlo en los proyectos. Finaliza el caso de uso |
| **Curso Alterno de Eventos** | |
| **Acción del actor** | **Respuesta del sistema** |
| 2.1 El diseñador selecciona un sonido que ya existe en el sistema.<br>3.4 El diseñador da clic en la opción cancelar. | 2.2 El sistema muestra el mensaje “El sonido “nombre” ya existe”.<br>3.5 El sistema cancela la operación culmina así el caso de uso. |
| **Escenario 2 Eliminar Sonido** | |
| **Curso Normal de Eventos** | |
| 1. El diseñador elige la opción Eliminar sonido.<br>2. El diseñador da clic en la opción aceptar. | 1.1 El sistema muestra el siguiente mensaje: ¿Desea eliminar el sonido: “nombre” del sistema?<br>2.1 El sistema elimina el sonido del sistema y finaliza el caso de uso. |
| **Escenario 3 Repetir Sonido** | |
| **Curso Normal de Eventos** | |
| **Acción del actor** | **Respuesta del sistema** |
| 1. El diseñador selecciona la opción repetir sonido. | 1.1 El sistema indica que el sonido se reproducirá cíclicamente. Finaliza el caso de uso. |
| **Escenario 4 Asociar Sonido** | |
| **Curso Normal de Eventos** | |

| Acción del actor | Respuesta del sistema |
|---|---|
| 1. El diseñador selecciona una escena y le asocia un sonido. | 1.1 El sistema asocia el sonido seleccionado al entorno. Finaliza el caso de uso. |

| **Caso de uso** | |
|---|---|
| **CU-8** | Detectar colisiones. |
| **Propósito** | El objetivo del caso de uso es que el diseñador del paseo no pueda ubicar más de un objeto en la misma posición del espacio. |
| **Prioridad** | Opcional. |
| **Actores:** Diseñador del Paseo. | |
| **Resumen:** El diseñador del paseo inicializa el caso de uso cuando intenta ubicar un objeto en una posición ocupada por otro objeto. | |
| **Referencias** | RF 8. |
| **Precondiciones** | Debe haber más de un objeto en la escena. |
| **Poscondiciones** | El paseo se crea con detección de colisiones. |

| **Curso Normal de Eventos** | |
|---|---|
| **Acción del actor** | **Respuesta del sistema** |
| 1. El diseñador selecciona un objeto y activa la propiedad detectar colisiones. | 1.1 El sistema adiciona al objeto la características “colisiona”, y verifica la posición de él relativa al resto de los objetos del entorno con esta característica, impidiendo que ocupe el mismo espacio físico de otros objetos con esta propiedad. |

| **Caso de uso** | |
|---|---|
| **CU-10** | Listar recursos. |
| **Propósito** | El objetivo del caso de uso es listar los nombres de los proyectos y los nombres de los modelos que se encuentren en el sistema. |
| **Prioridad** | Secundario |
| **Actores:** Diseñador del Paseo. | |

| | |
|---|---|
| **Resumen:** El diseñador del paseo inicializa el caso de uso cuando ejecuta el módulo Editor. El sistema muestra el listado de los nombres de los proyectos y el listado de nombres de los modelos que se encuentren en el sistema. | |
| **Referencias** | RF 10. |
| **Poscondiciones** | Se muestra el listado de los nombres de los proyectos y el listado de nombres de los modelos que se encuentren en el sistema. |
| **Curso Normal de Eventos** | |
| **Acción del actor** | **Respuesta del sistema** |
| 1. El diseñador ejecuta el módulo Editor. | 1.1 El sistema muestra el listado de los nombres de los proyectos y los nombres de los modelos que se encuentren en el sistema. |

**Casos de uso expandidos para el módulo Visualizador.**

| **Caso de uso** | |
|---|---|
| **CU-2** | Definir modo de paseo. |
| **Propósito** | El objetivo del caso de uso es que el usuario realice el paseo por el centro expositivo virtual del modo seleccionado. |
| **Prioridad** | Secundario. |
| **Actores:** Usuario. | |
| **Resumen:** El caso de uso se inicializa cuando el usuario define la forma en que se realizará el recorrido por el centro expositivo virtual, el recorrido puede realizarse de forma libre (utilizando los controles del teclado) o dirigido (se muestra el centro virtual a través del recorrido de la cámara). | |
| **Referencias** | RF 2. |
| **Precondiciones** | El paseo debe estar cargado en el área de visualización. |
| **Poscondiciones** | Se define el modo en que se realizará el recorrido por el centro expositivo virtual. |
| **Curso Normal de Eventos** | |
| **Acción del actor** | **Respuesta del sistema** |
| 1. El usuario selecciona el tipo de recorrido a realizar. | 1.1 El sistema muestra el centro expositivo virtual con el tipo de recorrido seleccionado. |

| **Caso de uso** | |
|---|---|
| **CU-3** | Editar cámara. |
| **Propósito** | El objetivo del caso de uso es que el usuario pueda modificar la velocidad y la altura de la cámara. |
| **Prioridad** | Secundario. |
| **Actores:** Usuario. | |
| **Resumen:** El caso de uso se inicializa cuando el usuario modifica la velocidad o la altura de la cámara. | |
| **Referencias** | RF 3, RF 3.1, RF 3.2, RF 3.3. |
| **Precondiciones** | El paseo debe estar cargado en el área de visualización. |
| **Poscondiciones** | Aumenta o disminuye la velocidad de la cámara o la altura. |
| **Curso Normal de Eventos** | |
| **Acción del actor** | **Respuesta del sistema** |
| 1. El usuario aumenta o disminuye la velocidad o la altura de la cámara. | 1.1 El sistema aumenta o disminuye la velocidad o la altura de la cámara. |

| **Caso de uso** | |
|---|---|
| **CU-4** | Mostrar información asociada a un objeto. |
| **Propósito** | El objetivo del caso de uso es que el usuario pueda observar la información asociada a un objeto. |
| **Prioridad** | Secundario. |
| **Actores:** Usuario. | |
| **Resumen:** El caso de uso se inicializa cuando el usuario selecciona un objeto al hacer clic encima del mismo y se muestra la información asociada al objeto en una pequeña ventana. | |
| **Referencias** | RF 4. |
| **Precondiciones** | El paseo debe estar cargado en el área de visualización. |
| **Poscondiciones** | Se muestra la información asociada a un objeto seleccionado. |
| **Curso Normal de Eventos** | |
| **Acción del actor** | **Respuesta del sistema** |
| 1. El usuario selecciona un objeto. | 1.1 El sistema muestra la información asociada al |

| | |
|---|---|
| | objeto en una pequeña ventana. |

| **Caso de uso** | |
|---|---|
| **CU-5** | Resaltar objetos expositivos. |
| **Propósito** | El objetivo del caso de uso es resaltar los objetos expositivos que se encuentran en el centro expositivo virtual. |
| **Prioridad** | Secundario. |
| **Actores:** Usuario. | |
| **Resumen:** El caso de uso se inicializa cuando el usuario pasa el mouse por encima de un objeto, el puntero cambia de forma indicando que el objeto puede ser seleccionado para ver su información asociada. | |
| **Referencias** | RF 5. |
| **Precondiciones** | El paseo debe estar cargado en el área de visualización. |
| **Poscondiciones** | Resalta el objeto al pasar el mouse por encima del mismo. |
| **Curso Normal de Eventos** | |
| **Acción del actor** | **Respuesta del sistema** |
| 1. El usuario mueve el mouse por encima de uno de los objetos expositivos. | 1.1 El sistema cambia el cursor señalando que el objeto puede ser seleccionado. Finaliza el caso de uso. |

| **Caso de uso** | |
|---|---|
| **CU-8** | Mostrar ayuda del paseo. |
| **Propósito** | El objetivo del caso de uso es mostrar una ayuda al usuario con información sobre cómo realizar el recorrido por el entorno virtual. |
| **Prioridad** | Secundario. |
| **Actores:** Usuario. | |
| **Resumen:** El caso de uso se inicializa cuando el usuario selecciona la opción mostrar ayuda. Luego de haber seleccionado la opción se muestra la ayuda del paseo con información sobre cómo realizar el recorrido por el entorno virtual. | |

| | |
|---|---|
| **Referencias** | RF 8. |
| **Precondiciones** | El paseo debe estar cargado en el área de visualización. |
| **Poscondiciones** | Se muestra la ayuda del paseo con información sobre cómo realizar el recorrido por el entorno virtual. |

| **Curso Normal de Eventos** | |
|---|---|
| **Acción del actor** | **Respuesta del sistema** |
| 1. El usuario elige la opción mostrar ayuda. | 1.1 El sistema muestra la ayuda del paseo con información sobre cómo realizar el recorrido por el entorno virtual. Finaliza el caso de uso. |

## Anexo 3. Descripción de las clases de la herramienta.

| **Nombre:** Controler | |
|---|---|
| **Tipo:** Controladora | |
| **Atributo** | **Tipo** |
| ProyectFileName | string |
| SystemlResourceFileName | string |
| ProyectNames | string* |
| SystemResourceNames | string* |
| **Responsabilidad**: | |
| **Nombre:** | +Controler() |
| **Descripción:** | Constructor de la clase. |
| **Nombre:** | +AddSystemModel(FileModelName : string) : void |
| **Descripción:** | Da respuesta al requisito cargar modelos al sistema. |
| **Nombre:** | +AddSound(FileSoundName : string) : void |
| **Descripción:** | Agrega un sonido al sistema. |
| **Nombre:** | -CheckNameExist(NName : string) : boolean |
| **Descripción:** | Chequea si existe un modelo o proyecto con igual nombre al que le pasan por parámetro. |
| **Nombre:** | +CreateNewProyect(NNameProject : string, NSystemRender : string, Aliasin : short) : |

| | |
|---|---|
| | void |
| **Descripción:** | Solicita los datos necesarios para crear un proyecto para luego crearlo en la carpeta de proyectos del sistema. |
| **Nombre:** | +DeleteSystemModel(NameModel : string) : void |
| **Descripción:** | Elimina un modelo de la carpeta de modelos del sistema. |
| **Nombre:** | +DeleteModelToScene(Name : string) : void |
| **Descripción:** | Elimina un modelo de la escena de un paseo virtual. |
| **Nombre:** | +DeleteProject() : void |
| **Descripción:** | Elimina un proyecto de la carpeta de proyectos del sistema. |
| **Nombre:** | +GetProyectFileName() : string |
| **Descripción:** | Retorna la dirección de la carpeta donde se encuentran los proyectos del sistema. |
| **Nombre:** | +GetSystemlResourceFileName() : string |
| **Descripción:** | Retorna dirección de la carpeta donde se encuentran los modelos del sistema. |
| **Nombre:** | -ListProjectNames() : void |
| **Descripción:** | Actualiza el atributo ProyectNames (lista de nombres de los proyectos del sistema). |
| **Nombre:** | -ListSystemResourceNames() : void |
| **Descripción:** | Actualiza el atributo SystemResourceNames (lista de nombres de los modelos del sistema). |
| **Nombre:** | +LoadProject(NNameProject : string, NSystemRender : string, Aliasin : short) : void |
| **Descripción:** | Carga en el editor un proyecto existente en la carpeta del sistema. |
| **Nombre:** | +LoadModelToScene(Name : string, FileName : string) : void |
| **Descripción:** | Carga un modelo en la escena de un proyecto. |
| **Nombre:** | +SaveProject() : void |
| **Descripción:** | Salva un proyecto y todos los cambios realizados. |
| **Nombre:** | +MoveModel(Name : string, Position : float *) : void |
| **Descripción:** | Actualiza la posición de un objeto. |
| **Nombre:** | +AlignModel(Name : string, Position : float *) : void |
| **Descripción:** | Actualiza la posición de un objeto con respecto al objeto más cercano que se encuentre en el escenario de un paseo virtual. |

| | |
|---|---|
| **Nombre:** | +ActivateShadow(Nombre : string, Shadow : boolean) : void |
| **Descripción:** | Modifica el atributo Shadow (sombra) de un modelo. |
| **Nombre:** | +ChangeViewModel(NViewModel : char) |
| **Descripción:** | Modifica la forma de visualizar un modelo que se encuentre en la escena de un paseo. |
| **Nombre:** | +ChangeRenderType(NRenderType : char) : void |
| **Descripción:** | Modifica la forma de visualizar la escena de un paseo. |
| **Nombre:** | +ScaleModel(Name : string, NScale : float) : void |
| **Descripción:** | Modifica la representación de un modelo que se encuentre en la escena aumentado o disminuyendo sus dimensiones. |
| **Nombre:** | +CopyModel(Name : string) : void |
| **Descripción:** | Realiza una copia de un modelo seleccionado. |
| **Nombre:** | +UpdtateInformationModel(Name : string, Information : string) : void |
| **Descripción:** | Modifica el atributo Information (Información) de un modelo. |
| **Nombre:** | +SetTypeModel(Name : string, TypeModel) : void |
| **Descripción:** | Modifica el atributo TypeModel (Tipo de modelo) de un modelo. |
| **Nombre:** | +DeleteSound(Name : string) : void |
| **Descripción:** | Elimina un sonido del sistema. |
| **Nombre:** | +AddCamera(Name : string) : void |
| **Descripción:** | Adiciona una cámara al escenario de un paseo virtual. |
| **Nombre:** | +DeleteCamera(Name : string) |
| **Descripción:** | Elimina una cámara del escenario de un paseo virtual. |
| **Nombre:** | +SetHeightPath(Name : string, NHeight : float) : void |
| **Descripción:** | Modifica el atributo HeightPath (altura) de una cámara. |
| **Nombre:** | +ExportVirtualTour(ProyectNames : string *, ProyectFileName : string, ExportFileName : string) : void |
| **Descripción:** | Exporta los paseos que se encuentren en la carpeta del sistema. |
| **Nombre:** | +ColisionDetection(NameModel : String) : void |
| **Descripción:** | Permite realizar la detección de colisiones entre objetos. |

| **Nombre:** DataManager | |
|---|---|
| **Tipo:** Controladora | |
| **Responsabilidad**: | |
| **Nombre:** | +DataManager() |
| **Descripción:** | Constructor de la clase. |
| **Nombre:** | +AddSystemModel(FileModelName : string) : void |
| **Descripción:** | Lee los datos de un modelo desde una dirección de fichero y lo crea en la carpeta de recursos del sistema. |
| **Nombre:** | +AddSound(NSound : Sound, FileName : string) : void |
| **Descripción:** | Lee los datos de un sonido desde una dirección de fichero y lo crea en la carpeta de recursos del sistema. |
| **Nombre:** | -CreateNewVirtualTour(VirtualTour : VirtualTour, FileName : string) : void |
| **Descripción:** | Crea un proyecto en la carpeta de proyectos del sistema. |
| **Nombre:** | -CreateNewCamera(NCamera : Camera, FileName : string) : void |
| **Descripción:** | Crea y guarda en un fichero un objeto de tipo cámara en la dirección donde se encuentra el proyecto. |
| **Nombre:** | -CreateNewPath(NPath : Path, FileName : string) : void |
| **Descripción:** | Crea y guarda en un fichero un objeto de tipo recorrido en la dirección donde se encuentra el proyecto. |
| **Nombre:** | -CreateNewLigth(NLigth : Ligth, FileName : string) : void |
| **Descripción:** | Crea y guarda en un fichero un objeto de tipo luz en la dirección donde se encuentra el proyecto. |
| **Nombre:** | -CreateNewModel(NModel : Model, FileName : string) : void |
| **Descripción:** | Crea y guarda en un fichero un objeto de tipo modelo en la dirección donde se encuentra el proyecto. |
| **Nombre:** | +DeleteSystemModel(NameModel : string, SystemlResourceFileName : string) : void |
| **Descripción:** | Elimina el fichero que contiene los datos de un modelo de la carpeta de recursos del sistema. |

| | |
|---|---|
| **Nombre:** | +DeleteProjetc(NProyetName : string, FileName : string) : void |
| **Descripción:** | Elimina los ficheros que contienen los datos de un paseo virtual de la carpeta del sistema. |
| **Nombre:** | +GetListProjectNames(ProyectFileName : string) : string * |
| **Descripción:** | Devuelve la lista de nombres de los proyectos que se encuentran en la carpeta del sistema. |
| **Nombre:** | +GetSystemResourceNames(SystemlResourceFileName : string) : string * |
| **Descripción:** | Devuelve la lista de nombres de los modelos que se encuentran en la carpeta del sistema. |
| **Nombre:** | +LoadVirtualTour(NProyetName : string, ProyectFileName : string) : VirtualTour* |
| **Descripción:** | Lee un fichero que contiene los datos del paseo virtual que le pasan por parámetro y devuelve un objeto de este tipo. |
| **Nombre:** | +LoadModel(NProyetName : string, ProyectFileName : string) : Model * |
| **Descripción:** | Lee un fichero que contiene los datos del modelo que le pasan por parámetro y devuelve un objeto de este tipo. |
| **Nombre:** | +LoadPath(NProyetName : string, ProyectFileName : string) : Path * |
| **Descripción:** | Lee un fichero que contiene los datos de la cámara que le pasan por parámetro y devuelve un objeto de este tipo. |
| **Nombre:** | +LoadPath(NProyetName : string, ProyectFileName : string) : Path * |
| **Descripción:** | Lee un fichero que contiene los datos del recorrido que le pasan por parámetro y devuelve un objeto de este tipo. |
| **Nombre:** | +SaveProject(VirtualTour : VirtualTour, ListCamera : Camera *, ListPath : Path *, ListModel : Model *, ListLigth : Ligth *, FileName : string) : void |
| **Descripción:** | Salva los cambios realizados a proyecto, guarda todos los datos de sus recursos en ficheros que se crean dentro de la carpeta del proyecto. |
| **Nombre:** | +ExportVirtualTour(ProyectNames : string *, ProyectFileName : string, ExportFileName : string) : void |
| **Descripción:** | Salva los ficheros de todos los proyectos que se encuentren en la carpeta del sistema para la carpeta del módulo visualizador. |

| **Nombre:** SceneManager | |
|---|---|
| **Tipo:** Controladora | |
| **Atributo** | **Tipo** |
| -SystemRender | string |
| -Aliasing | short |
| -RenderType | char |
| -ViewModel | char |
| **Responsabilidad**: | |
| **Nombre:** | +SceneManager() |
| **Descripción:** | Constructor de la clase. |
| **Nombre:** | +SetSystemRender(NSystemRender : string) : void |
| **Descripción:** | Permite actualizar el atributo SystemRender para definir la biblioteca con que se va a realizar el render. |
| **Nombre:** | +SetAliasin(Aliasin : short) : void |
| **Descripción:** | Permite actualizar el atributo Aliasing para definir la rigurosidad con que se va aplicar el método aliasing. |
| **Nombre:** | +SetRenderType(NRenderType : char) : void |
| **Descripción:** | Permite actualizar el atributo RenderType para definir la forma de pintar. |
| **Nombre:** | +SetViewModel(NViewModel : char) : void |
| **Descripción:** | Permite actualizar el atributo ViewModel para definir la vista de la escena. |
| **Nombre:** | +SetPosition(Nombre : string, Posicion : float *) : void |
| **Descripción:** | Permite actualizar la posición de un objeto en el escenario. |
| **Nombre:** | +SetShadow(Nombre : string, Shadow : boolean) : void |
| **Descripción:** | Permite activar o desactivar la sombra de un objeto en el escenario actualizando el atributo de este objeto. |
| **Nombre:** | +SetScale(Name : string, NScale : float) : void |
| **Descripción:** | Permite actualizar el atributo NScale (tamaño) de un modelo. |
| **Nombre:** | +SetInformation(Name : string, NInformation : string) |
| **Descripción:** | Permite actualizar el atributo Información (Información) de un modelo. |

| | |
|---|---|
| **Nombre:** | +SetHeight(Name : string, NHeight : float) |
| **Descripción:** | Permite actualizar el atributo Height (Altura) de un modelo. |
| **Nombre:** | +Render() : void |
| **Descripción:** | Método que permite pintar los recursos del proyecto. |
| **Nombre:** | +AddVirtualTour(NVirtualTour : VirtualTour *) : void |
| **Descripción:** | Crea un objeto de tipo paseo virtual. |
| **Nombre:** | +AddCamera(NCamera : Camera *) : void |
| **Descripción:** | Crea un objeto de tipo cámara. |
| **Nombre:** | +AddPath(NPath : Path *) : void |
| **Descripción:** | Crea un objeto de tipo recorrido. |
| **Nombre:** | +AddModel(NModel : Model *) : void |
| **Descripción:** | Crea un objeto de tipo modelo. |
| **Nombre:** | +AddLigth(Nligth : Ligth *) : void |
| **Descripción:** | Crea un objeto de tipo luz. |
| **Nombre:** | +ClearData() : void |
| **Descripción:** | Libera los datos que se encuentren en SceneManager. |
| **Nombre:** | +GetVirtualTour() : VirtualTour * |
| **Descripción:** | Devuelve los objetos existentes de tipo paseo virtual. |
| **Nombre:** | +GetCamera() : Camera * |
| **Descripción:** | Devuelve los objetos existentes de tipo cámara. |
| **Nombre:** | +GetPath() : Path * |
| **Descripción:** | Devuelve los objetos existentes de tipo recorrido. |
| **Nombre:** | +GetModel() : Model * |
| **Descripción:** | Devuelve los objetos existentes de tipo modelo. |
| **Nombre:** | +GetLight() : Project.Light * |
| **Descripción:** | Devuelve los objetos existentes de tipo luz. |
| **Nombre:** | +DeleteCamera(Name : string) : void |
| **Descripción:** | Elimina un objeto de tipo cámara. |
| **Nombre:** | +DeleteLigth(Name : string) : void |

| | |
|---|---|
| **Descripción:** | Elimina un objeto de tipo luz. |
| **Nombre:** | +DeleteModel(Name : string) : void |
| **Descripción:** | Elimina un objeto de tipo modelo. |
| **Nombre:** | +DeletePath(Name : string) : void |
| **Descripción:** | Elimina un objeto de tipo recorrido. |

| **Nombre:** CVControler | |
|---|---|
| **Tipo:** Controladora | |
| **Atributo** | **Tipo** |
| ProyectFileName | string |
| ListProyectNames | string* |
| **Responsabilidad**: | |
| **Nombre:** | +CVControler () |
| **Descripción:** | Constructor de la clase. |
| **Nombre:** | +InitializeData() : void |
| **Descripción:** | Muestra la lista de nombres de los proyectos (ListProyectNames) que existen en la carpeta del sistema. |
| **Nombre:** | -ListProjectNames() : void |
| **Descripción:** | Actualiza la lista de nombres de los proyectos que existen en la carpeta del sistema. |
| **Nombre:** | +DefineVirtualTourType() : void |
| **Descripción:** | Modifica el atributo TourMode (modo de paseo virtual) para mostrar el paseo dirigido o libre. |
| **Nombre:** | +EditCamera() : void |
| **Descripción:** | Modifica la velocidad y altura de la cámara posibilitando el recorrido por el paseo. |
| **Nombre:** | +StarVirtualTour(NNameProject : string, NSystemRender : string, Aliasin : short) : void |
| **Descripción:** | Carga un paseo virtual en el visualizador y lo muestra en pantalla completa. |
| **Nombre:** | +CloseVirtualTour() : void |
| **Descripción:** | Cierra un paseo virtual cargado en la pantalla del visualizador. |
| **Nombre:** | +ShowModelInformation() : void |

| | |
|---|---|
| **Descripción:** | Muestra la información asociada a un objeto. |
| **Nombre:** | +DenoteExpositiveModel() : void |
| **Descripción:** | Resalta un objeto de tipo expositivo. |
| **Nombre:** | +ShowHelp() : void |
| **Descripción:** | Muestra la ayuda del paseo. |

| **Nombre:** CVSceneManager | |
|---|---|
| **Tipo:** Controladora | |
| **Atributo** | **Tipo** |
| -SystemRender | string |
| -Aliasing | short |
| -TourMode | boolean |
| **Responsabilidad**: | |
| **Nombre:** | +CVSceneManager() |
| **Descripción:** | Constructor de la clase. |
| **Nombre:** | +SetSystemRender(NSystemRender : string) : void |
| **Descripción:** | Permite actualizar el atributo SystemRender para definir la biblioteca con que se va a realizar el render. |
| **Nombre:** | +SetAliasin(Aliasin : short) : void |
| **Descripción:** | Permite actualizar el atributo Aliasing para definir la rigurosidad con que se va aplicar el método aliasing. |
| **Nombre:** | +Render() : void |
| **Descripción:** | Método que permite pintar los recursos del proyecto. |
| **Nombre:** | +CloseTour() : void |
| **Descripción:** | Elimina la instancia de objetos perteneciente un proyecto que se encuentre en el SceneManager y visualiza la pantalla inicial. |
| **Nombre:** | +EditCamera(height : int) : void |
| **Descripción:** | Permite actualizar la altura de una cámara. |
| **Nombre:** | +AddVirtualTour(NVirtualTour : VirtualTour *) : void |

| | |
|---|---|
| **Descripción:** | Crea un objeto de tipo paseo virtual. |
| **Nombre:** | +AddCamera(NCamera : Camera *) : void |
| **Descripción:** | Crea un objeto de tipo cámara. |
| **Nombre:** | +AddPath(NPath : Path *) : void |
| **Descripción:** | Crea un objeto de tipo recorrido. |
| **Nombre:** | +AddModel(NModel : Model *) : void |
| **Descripción:** | Crea un objeto de tipo modelo. |
| **Nombre:** | +AddLigth(Nligth : Ligth *) : void |
| **Descripción:** | Crea un objeto de tipo luz. |
| **Nombre:** | +ClearData() : void |
| **Descripción:** | Libera los datos que se encuentren en SceneManager. |
| **Nombre:** | +GetVirtualTour() : VirtualTour * |
| **Descripción:** | Devuelve los objetos existentes de tipo paseo virtual. |
| **Nombre:** | +GetCamera() : Camera * |
| **Descripción:** | Devuelve los objetos existentes de tipo cámara. |
| **Nombre:** | +GetPath() : Path * |
| **Descripción:** | Devuelve los objetos existentes de tipo recorrido. |
| **Nombre:** | +GetModel() : Model * |
| **Descripción:** | Devuelve los objetos existentes de tipo modelo. |
| **Nombre:** | +GetLight() : Project.Light * |
| **Descripción:** | Devuelve los objetos existentes de tipo luz. |
| **Nombre:** | +DeleteCamera(Name : string) : void |
| **Descripción:** | Elimina un objeto de tipo cámara. |
| **Nombre:** | +DeleteLigth(Name : string) : void |
| **Descripción:** | Elimina un objeto de tipo luz. |
| **Nombre:** | +DeleteModel(Name : string) : void |
| **Descripción:** | Elimina un objeto de tipo modelo. |
| **Nombre:** | +DeletePath(Name : string) : void |
| **Descripción:** | Elimina un objeto de tipo recorrido. |

| **Nombre:** CVImport | |
|---|---|
| **Tipo:** Controladora | |
| **Responsabilidad**: | |
| **Nombre:** | +CVImport() |
| **Descripción:** | Constructor de la clase. |
| **Nombre:** | +GetListProjectNames(ProyectFileName : string) : string * |
| **Descripción:** | Devuelve la lista de nombres de los proyectos que se encuentran en la carpeta del sistema. |
| **Nombre:** | +GetSystemResourceNames(SystemlResourceFileName : string) : string * |
| **Descripción:** | Devuelve la lista de nombres de los modelos que se encuentran en la carpeta del sistema. |
| **Nombre:** | +LoadVirtualTour(NProyetName : string, ProyectFileName : string) : VirtualTour* |
| **Descripción:** | Lee un fichero que contiene los datos del paseo virtual que le pasan por parámetro y devuelve un objeto de este tipo. |
| **Nombre:** | +LoadModel(NProyetName : string, ProyectFileName : string) : Model * |
| **Descripción:** | Lee un fichero que contiene los datos del modelo que le pasan por parámetro y devuelve un objeto de este tipo. |
| **Nombre:** | +LoadPath(NProyetName : string, ProyectFileName : string) : Path * |
| **Descripción:** | Lee un fichero que contiene los datos del recorrido que le pasan por parámetro y devuelve un objeto de este tipo. |
| **Nombre:** | +LoadLigth(NProyetName : string, ProyectFileName : string) : Ligth * |
| **Descripción:** | Lee un fichero que contiene los datos de la luz que le pasan por parámetro y devuelve un objeto de este tipo. |
| **Nombre:** | +LoadCamera(NProyetName : string, ProyectFileName : string) : Project.Camera * |
| **Descripción:** | Lee un fichero que contiene los datos de la cámara que le pasan por parámetro y devuelve un objeto de este tipo. |

| **Nombre:** Ui_Interface |
|---|

| |
|---|
| **Tipo:** Interfaz |
| **Descripción** |
| Clase interfaz que tiene la responsabilidad de enviar las peticiones del usuario hacia la clase Controler y mostrar la respuesta. Contiene todas las llamadas a las funcionalidades que dan respuesta a los casos de uso. |
| **Nombre:** Ui_NewScena |
| **Tipo:** Interfaz |
| **Descripción** |
| Clase interfaz que tiene la responsabilidad de enviar las peticiones del usuario hacia la clase Controler. Es la encargada de enviar los datos a la clase para ejecutar la función de crear y cargar proyecto. |
| **Nombre:** Painter |
| **Tipo:** Controladora |
| **Descripción** |
| Clase que tiene la responsabilidad de visualizar los objetos que se adicionen a la escena. |
| **Nombre:** VirtualTour |
| **Tipo:** Entidad |
| **Descripción** |
| Representa un objeto de tipo VirtualTour. Permite almacenar temporalmente los datos correspondientes a esta entidad. |
| **Nombre:** Light |
| **Tipo:** Entidad |
| **Descripción** |
| Representa un objeto de tipo Light. Permite almacenar temporalmente los datos correspondientes a esta entidad. |
| **Nombre:** Path |
| **Tipo:** Entidad |
| **Descripción** |
| Representa un objeto de tipo Path. Permite almacenar temporalmente los datos correspondientes a esta entidad. |

| |
|---|
| **Nombre:** Model |
| **Tipo:** Entidad |
| **Descripción** |
| Representa un objeto de tipo Model. Permite almacenar temporalmente los datos correspondientes a esta entidad. |
| **Nombre:** Camera |
| **Tipo:** Entidad |
| **Descripción** |
| Representa un objeto de tipo Camera. Permite almacenar temporalmente los datos correspondientes a esta entidad. |
| **Nombre:** Material |
| **Tipo:** Entidad |
| **Descripción** |
| Representa un objeto de tipo Material. Permite almacenar temporalmente los datos correspondientes a esta entidad. |
| **Nombre:** PropertyInspector |
| **Tipo:** Controladora |
| **Descripción** |
| Su responsabilidad es modificar la información de los objetos. |
| **Nombre:** M_mObj_BoundingBox |
| **Tipo:** Entidad |
| **Descripción** |
| Clase que se utiliza para representar una caja que mueve los modelos en la escena. |
| **Nombre:** M_mObj_Axis_M |
| **Tipo:** Entidad |
| **Descripción** |
| Clase que se utiliza para representar los ejes de coordenadas de un modelo o de la escena. |
| **Nombre:** M_mobj_Grid |
| **Tipo:** Entidad |

| **Descripción** |
|---|
| Clase que se utiliza para representar una rejilla de referencia en la escena. |
| **Nombre:** qt_property_inspector |
| **Tipo:** Paquete |
| **Descripción** |
| Paquete que permite modificar la información de los objetos. |
| **Nombre:** OgreMaxLoaders |
| **Tipo:** Paquete |
| **Descripción** |
| Paquete que permite manipular los ficheros del sistema. Disponible en: www.sourceforge.net/projects/tinyxml |
| **Nombre:** Graphic Engine Ogre |
| **Tipo:** Paquete |
| **Descripción** |
| Motor gráfico. Disponible en: www.ogre3d.org |

## Anexo 4. Diagramas de Interacción.

### Diagramas de colaboración.

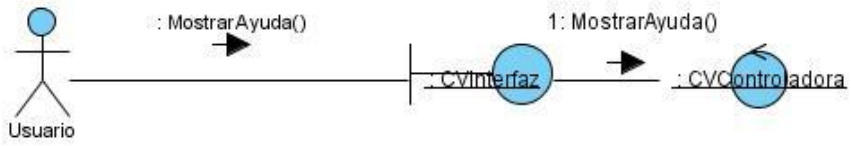



**Diagrama de colaboración CU- Mostrar ayuda.**

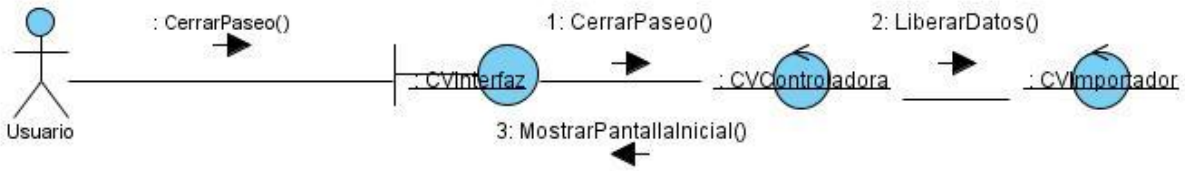



**Diagrama de colaboración CU- Cerrar paseo.**

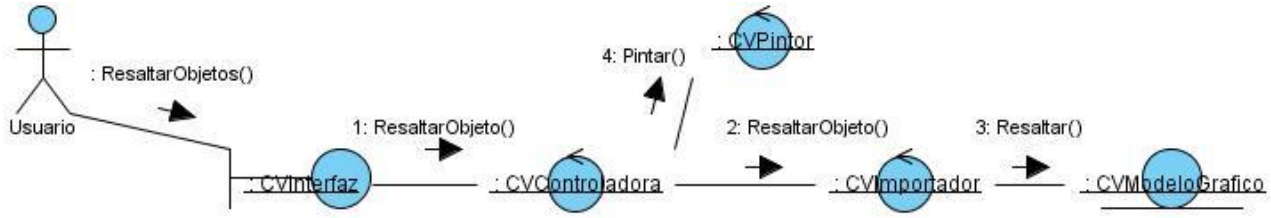



**Diagrama de colaboración CU- Resaltar objetos expositivos.**

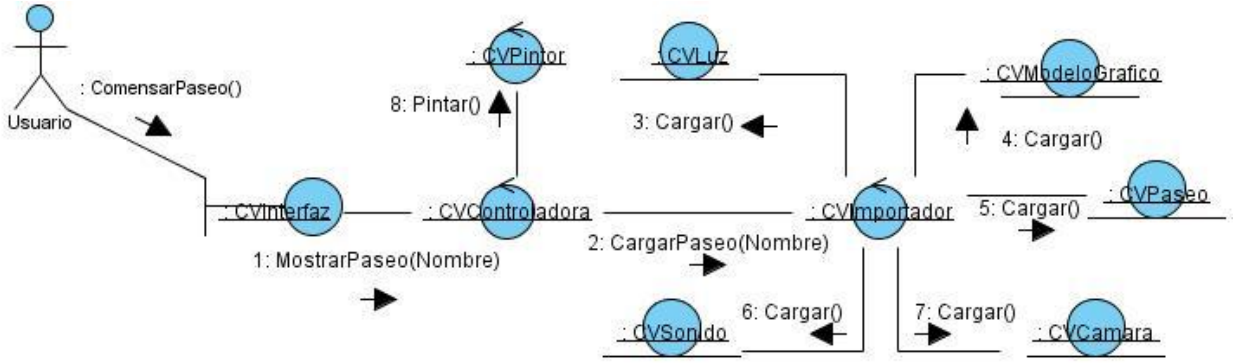



**Diagrama de colaboración CU- Comenzar paseo.**

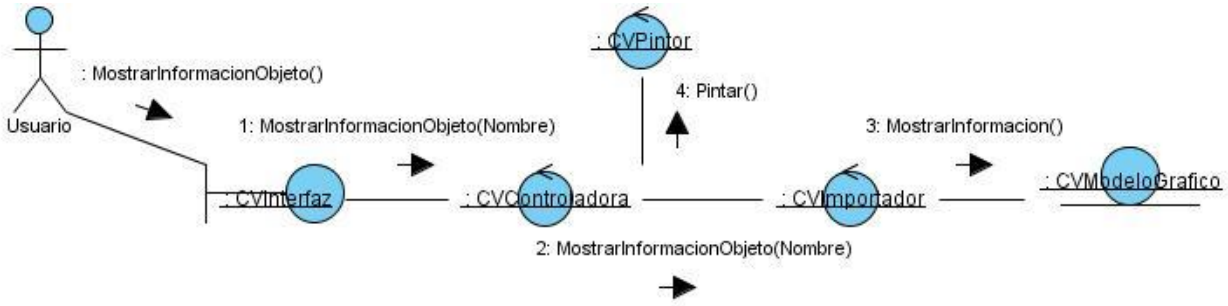



**Diagrama de colaboración CU- Mostar información asociada a un objeto.**

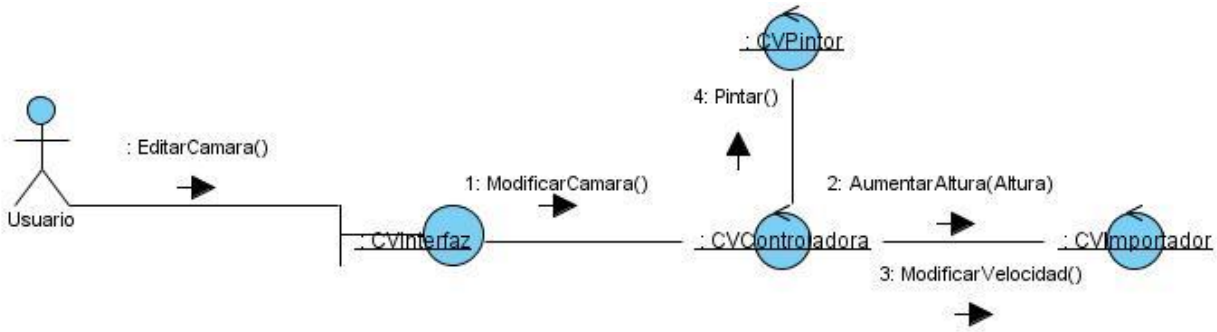



**Diagrama de colaboración CU- Editar cámara.**

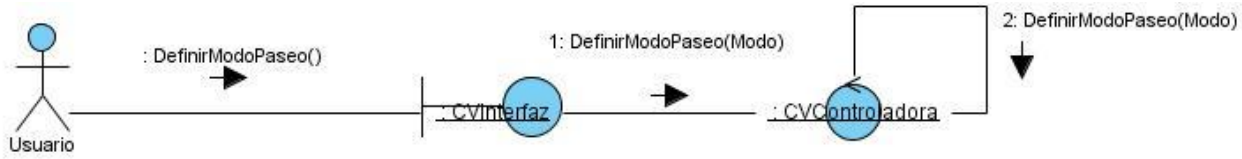



**Diagrama de colaboración CU-Definir modo de paseo.**

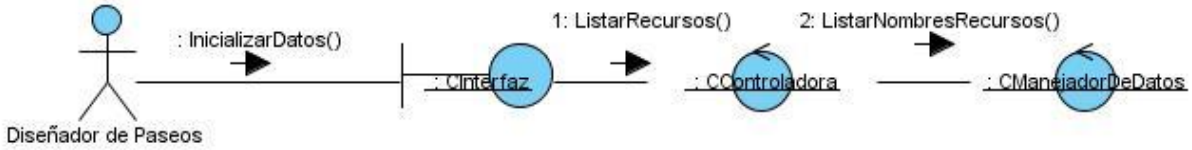



**Diagrama de colaboración CU- Listar recursos.**

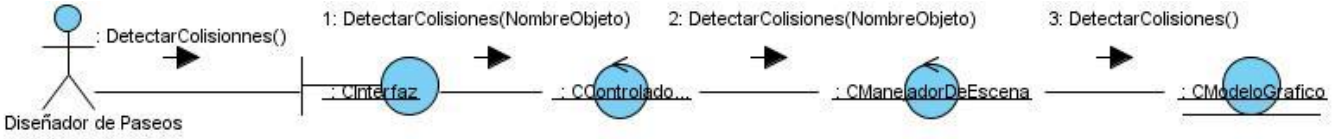



**Diagrama de colaboración CU- Detectar colisiones.**

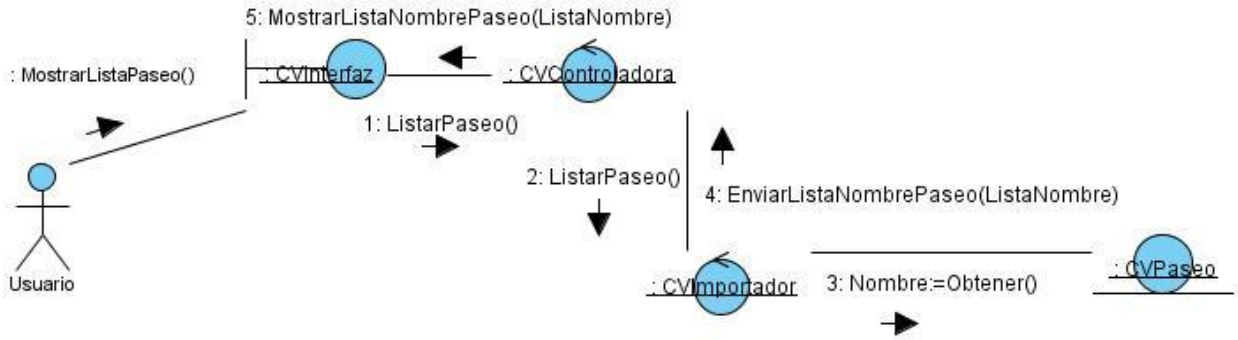



**Diagrama de colaboración CU-Inicializar datos.**

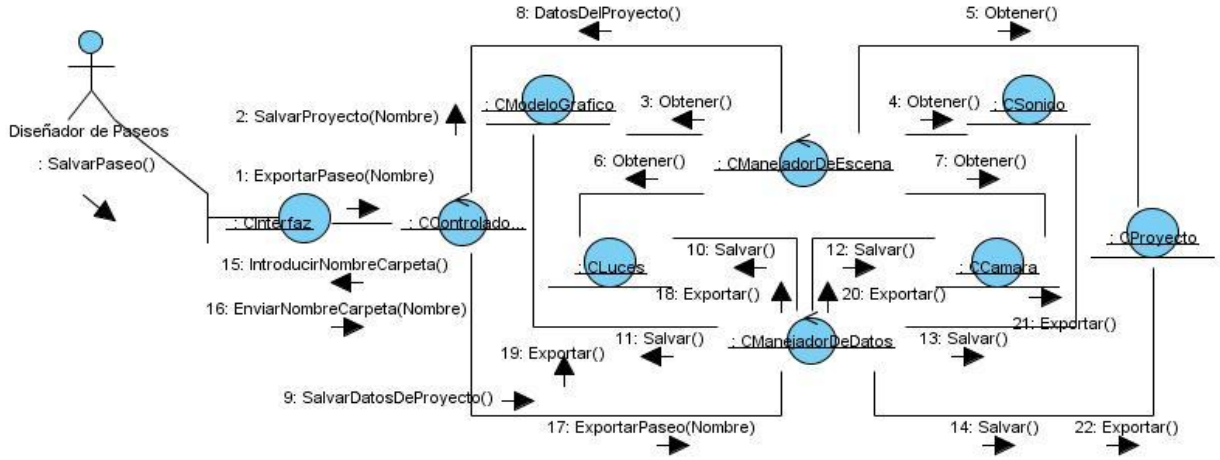



**Diagrama de colaboración CU-Exportar paseo.**

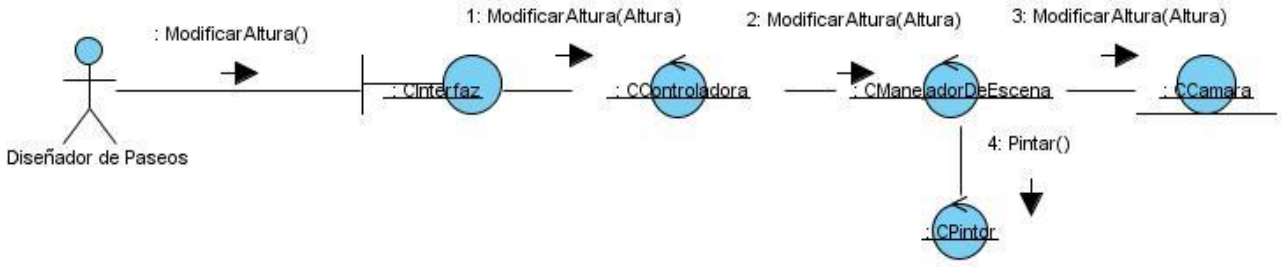



**Diagrama de colaboración CU- Administrar Cámara escenario Modificar altura.**

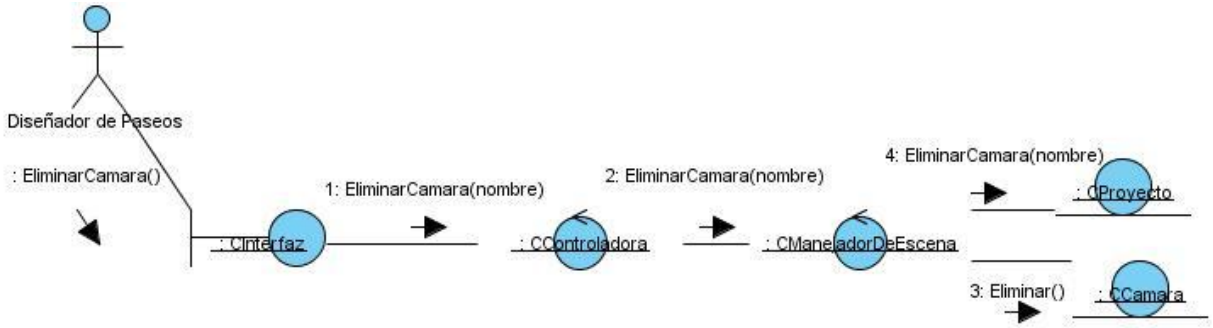



**Diagrama de colaboración CU- Administrar Cámara escenario Eliminar cámara.**

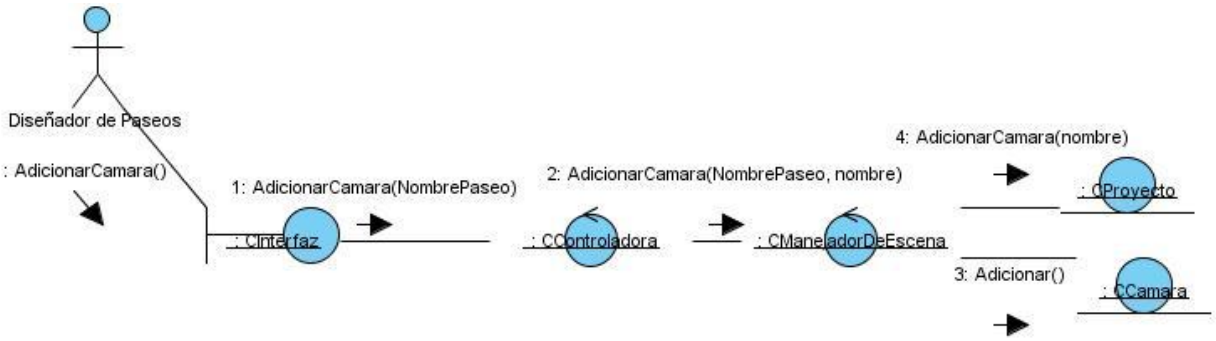



**Diagrama de colaboración CU- Administrar Cámara escenario Adicionar cámara.**

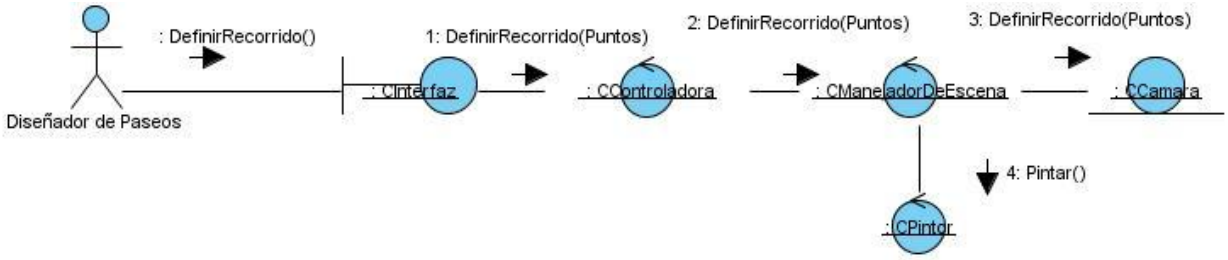



**Diagrama de colaboración CU- Administrar Cámara escenario Definir recorrido.**

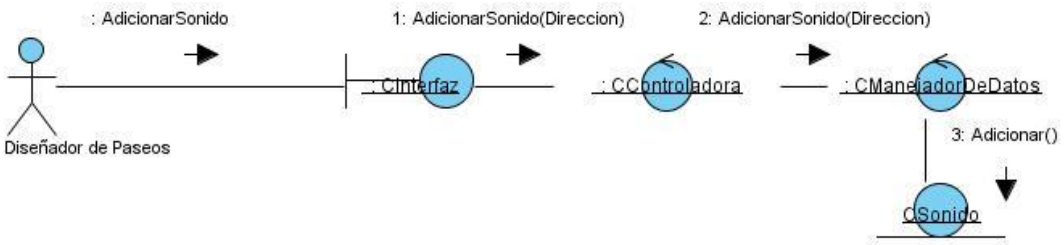



**Diagrama de colaboración CU- Administrar Sonido escenario Adicionar sonido.**

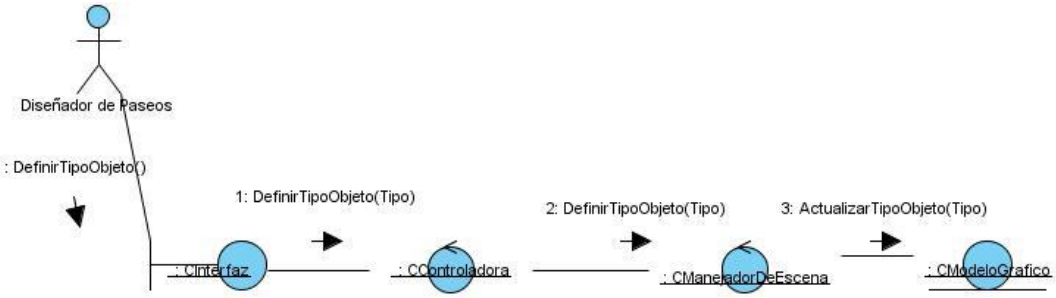



**Diagrama de colaboración CU- Modificar información de objeto escenario Definir tipo de objeto.**

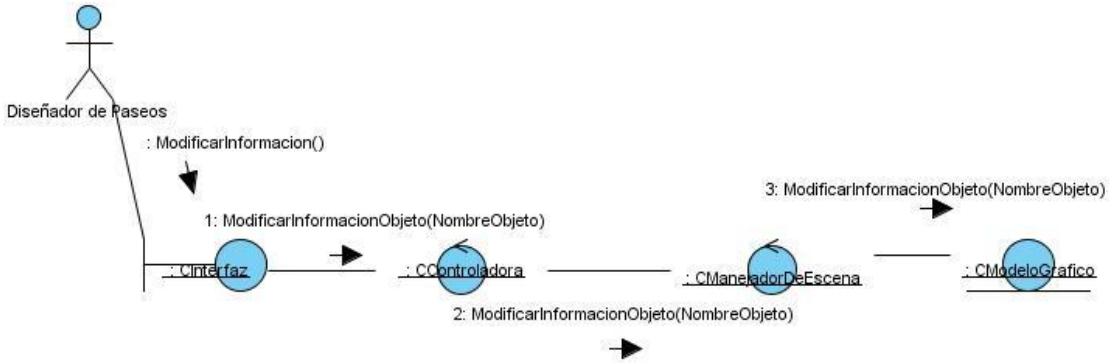



**Diagrama de colaboración CU- Modificar información de objeto escenario Actualizar información de objeto.**

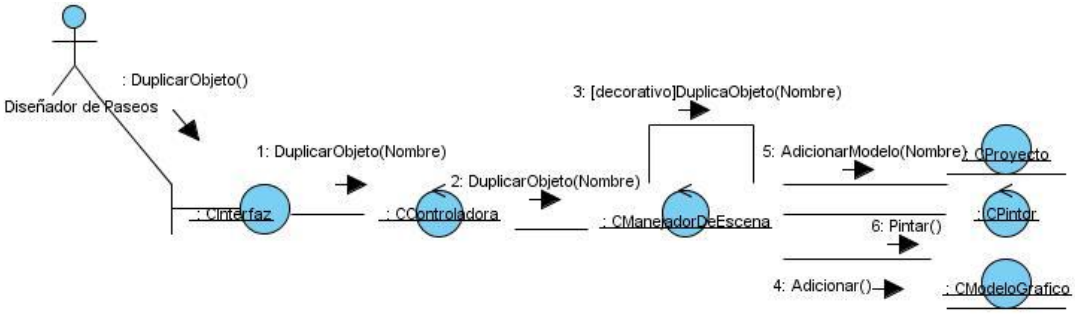



**Diagrama de colaboración CU-Modificar objeto decorativo escenario Duplicar objeto.**

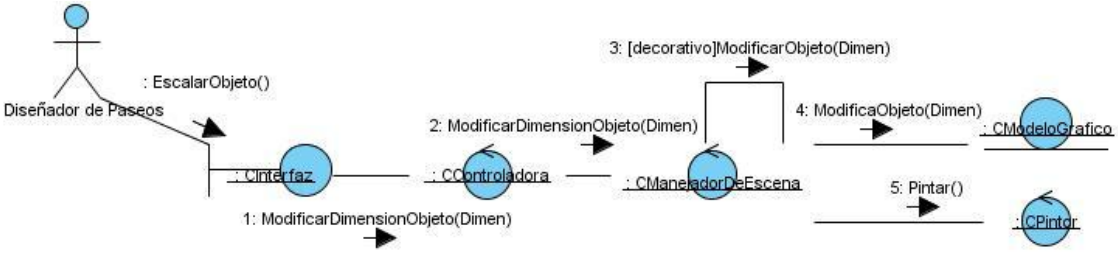



**Diagrama de colaboración CU-Modificar objeto decorativo escenario Escalar objeto.**

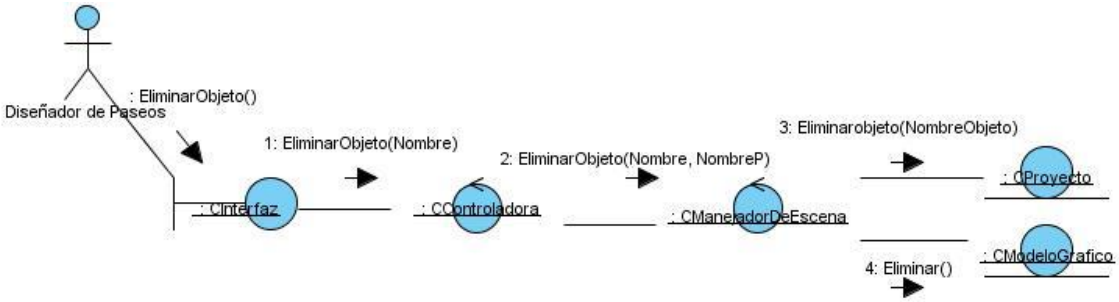



**Diagrama de colaboración CU- Administrar objetos del entorno escenario Eliminar objeto.**

.

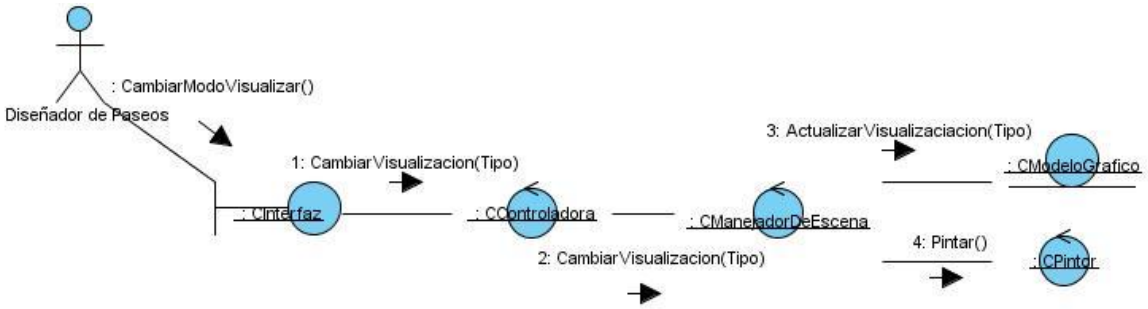



**Diagrama de colaboración CU-Administrar objetos del entorno escenario Cambiar modo de visualización.**

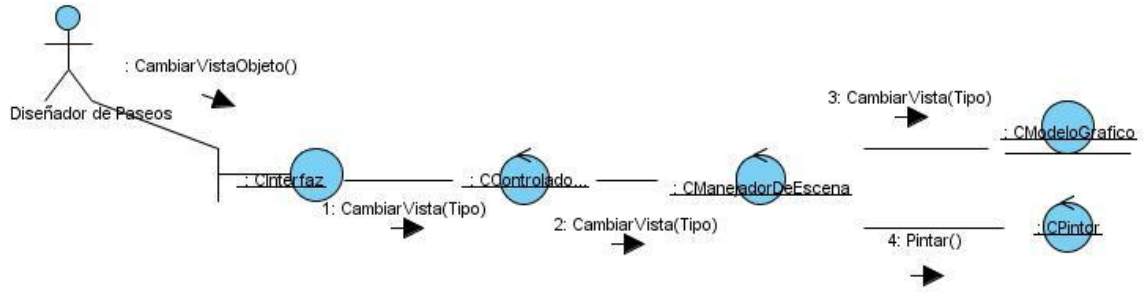



**Diagrama de colaboración CU-Administrar objetos del entorno escenario Cambiar vista de objeto.**

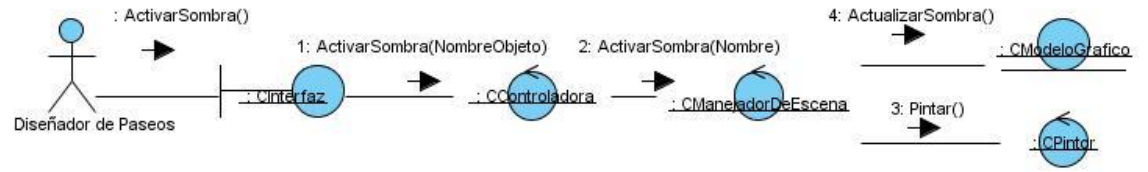



**Diagrama de colaboración CU-Administrar objetos del entorno escenario activar sombra.**

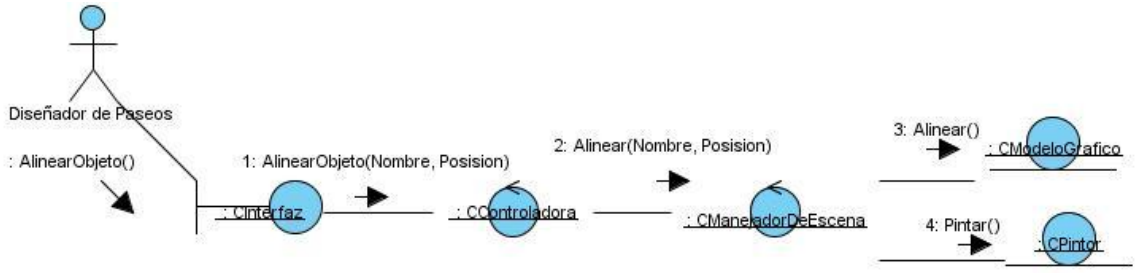



**Diagrama de colaboración CU-Administrar objetos del entorno escenario Alinear objeto.**

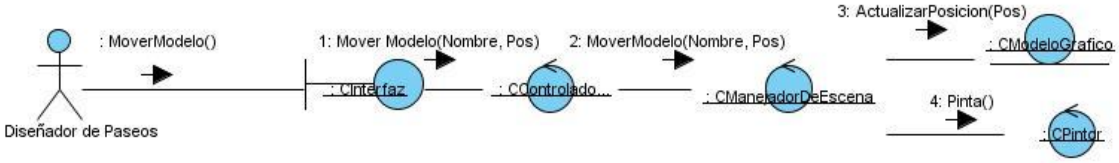



**Diagrama de colaboración CU-Administrar objetos del entorno escenario Mover objeto.**


**Diagrama de colaboración CU-Administrar modelos del sistema escenario Adicionar modelo.**

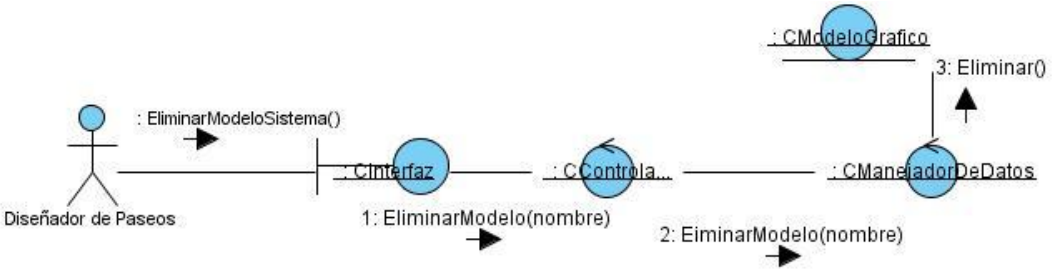



**Diagrama de colaboración CU-Administrar modelos del sistema escenario Eliminar modelo.**

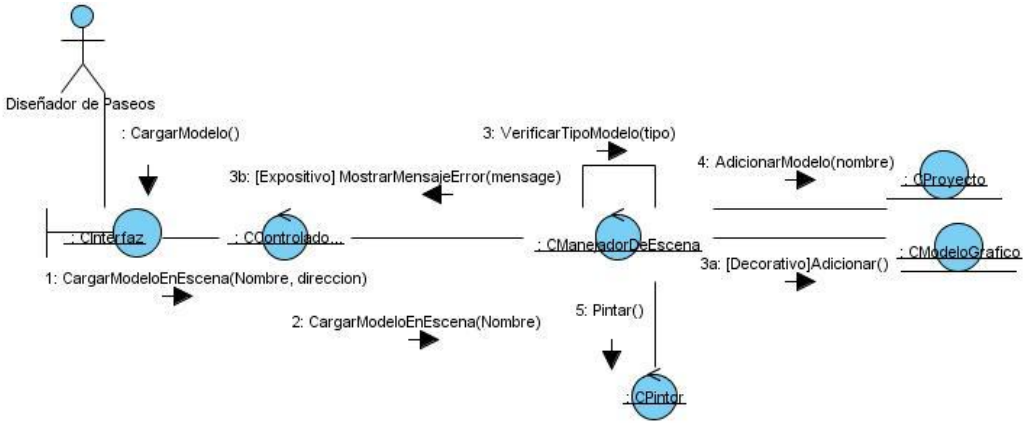



**Diagrama de colaboración CU-Administrar objetos del entorno escenario Cargar objeto en el entorno.**

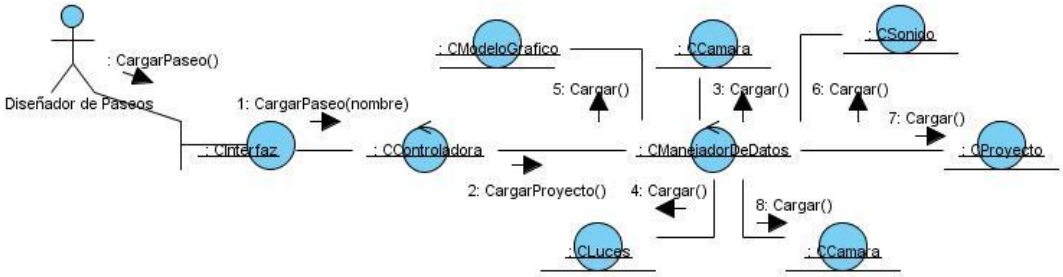



**Diagrama de colaboración CU-Administrar proyecto escenario Cargar proyecto.**

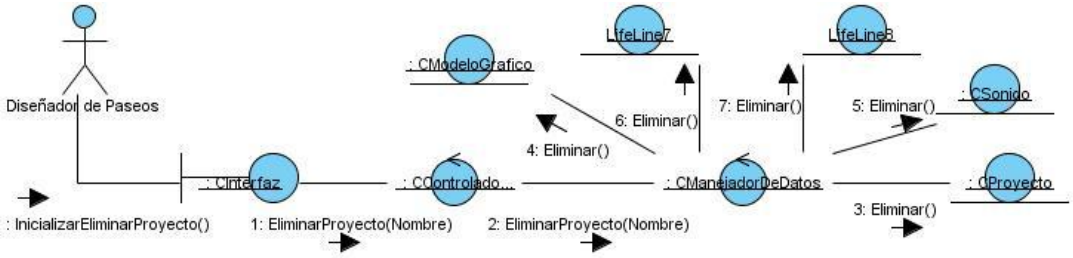



**Diagrama de colaboración CU-Administrar proyecto escenario Eliminar proyecto.**

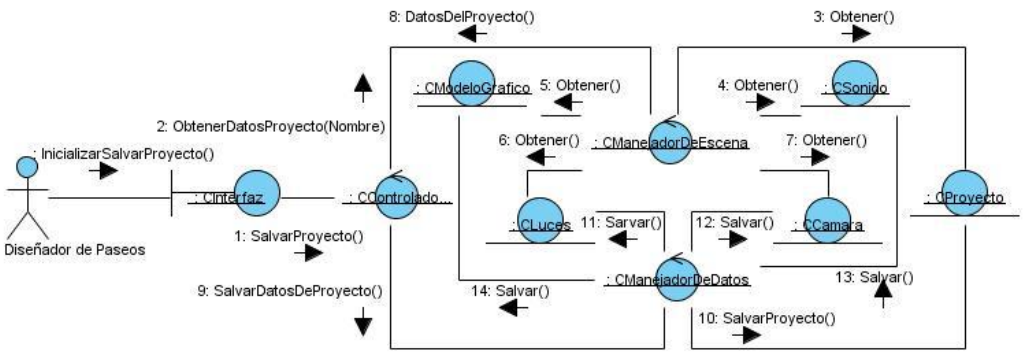



**Diagrama de colaboración CU-Administrar proyecto escenario Salvar proyecto.**

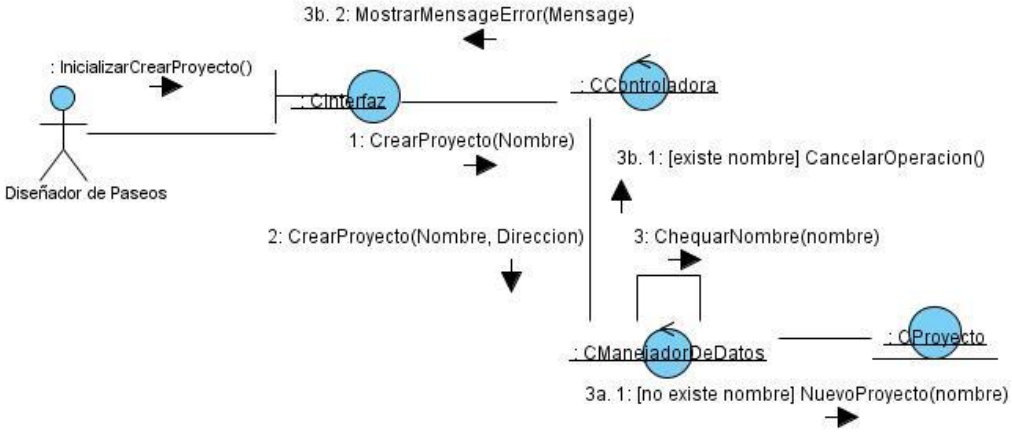



**Diagrama de colaboración CU-Administrar proyecto escenario Crear proyecto.**

## Diagramas de Secuencia.

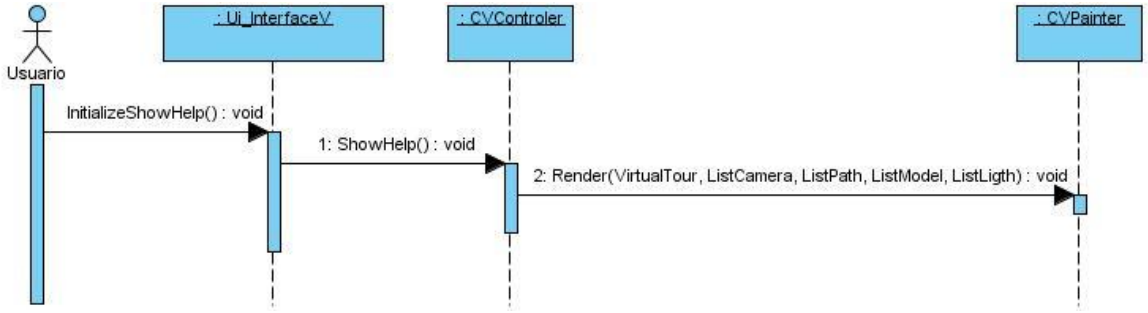



**Diagrama de secuencia CU-Mostrar Ayuda.**

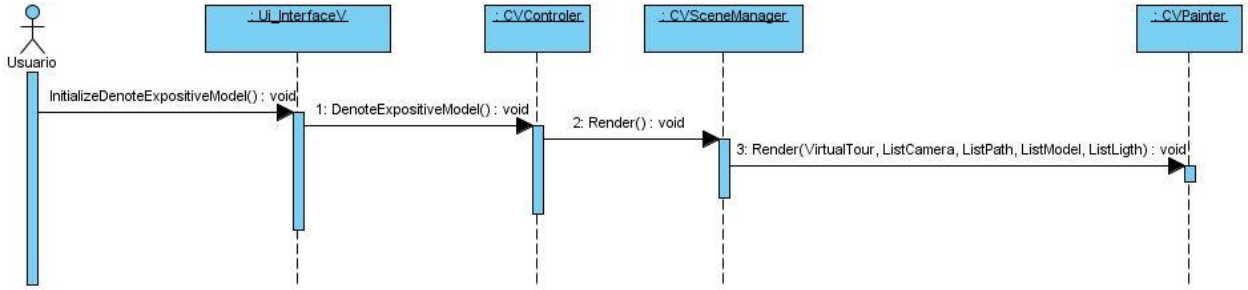



**Diagrama de secuencia CU- Resaltar objetos expositivos.**

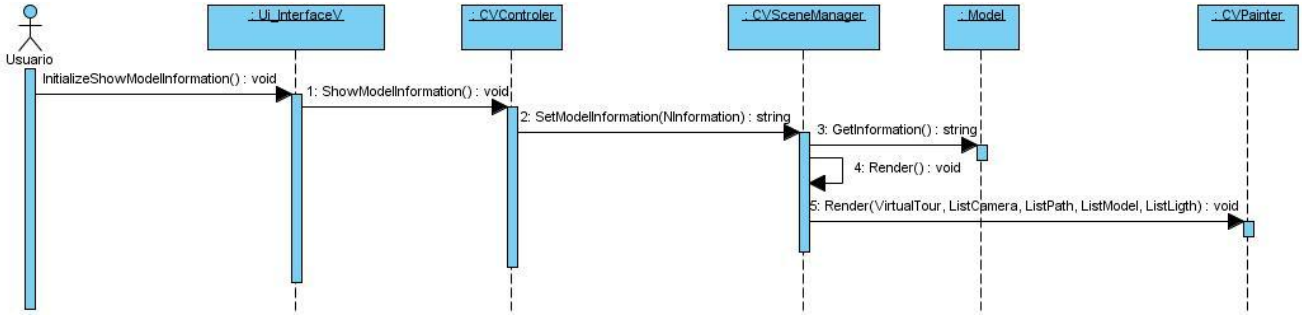



**Diagrama de secuencia CU-Mostrar información asociada a un objeto.**

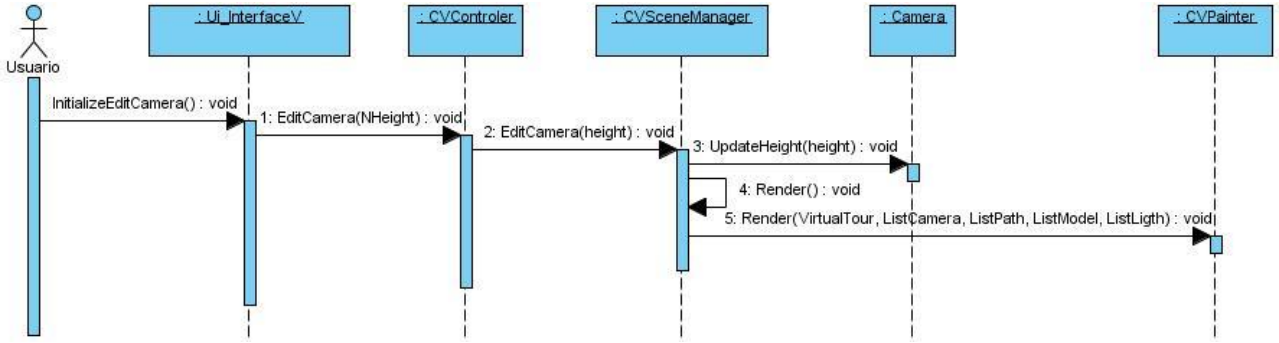



**Diagrama de secuencia CU-Editar cámara.**

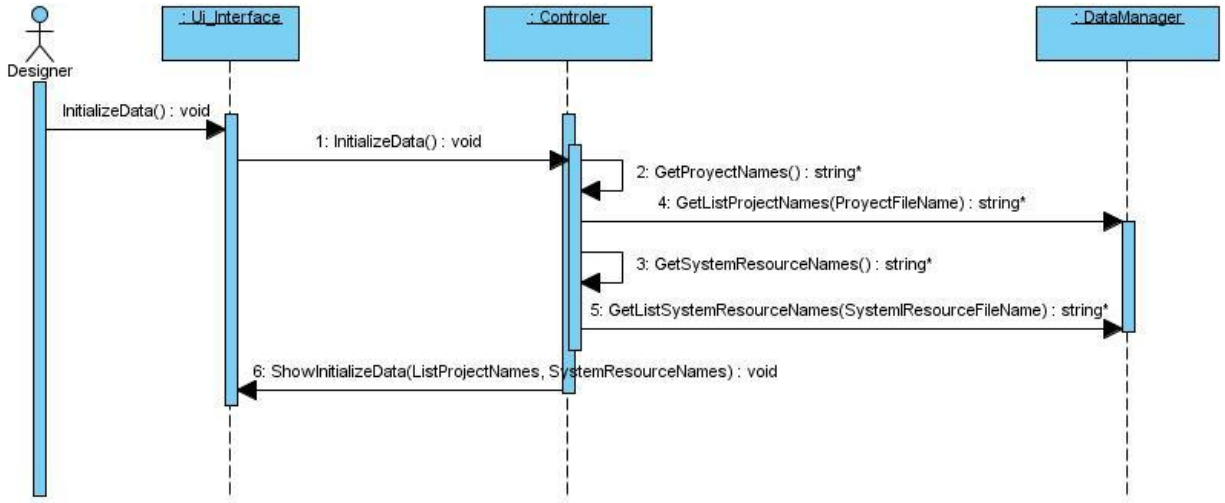



**Diagrama de secuencia CU- Listar Recursos.**

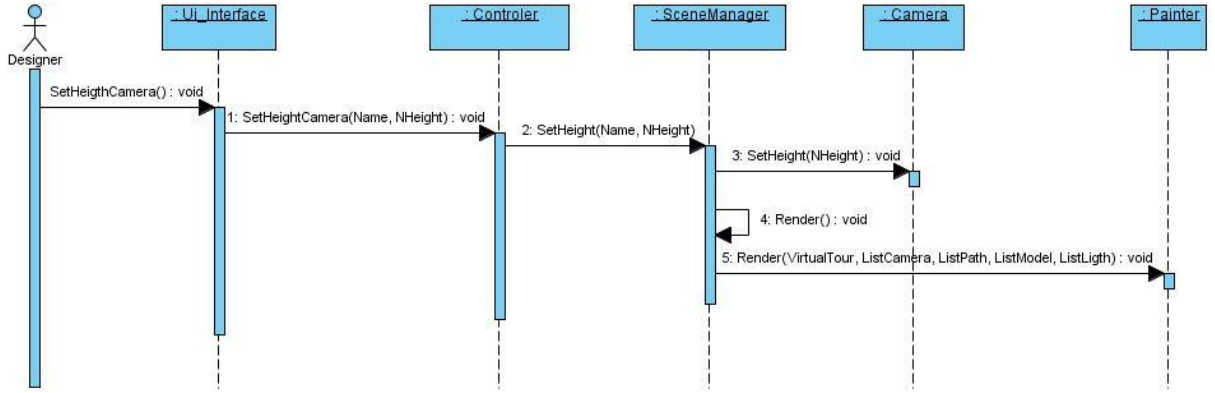



**Diagrama de secuencia CU- Administrar cámara escenario Modificar altura.**

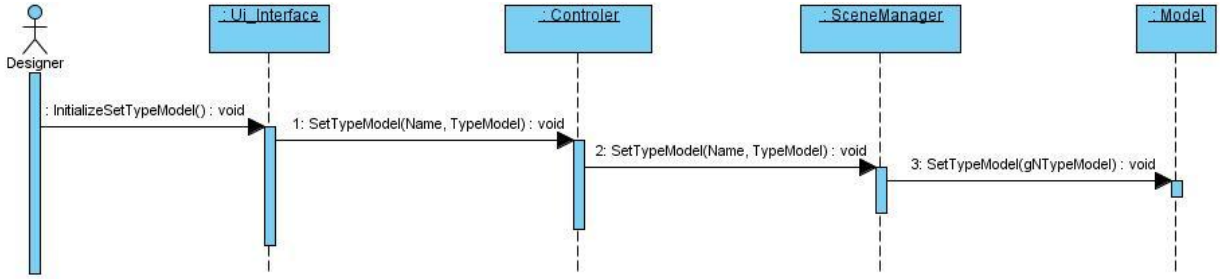



**Diagrama de secuencia CU- Modificar información de los objetos escenario Definir tipo de objeto.**

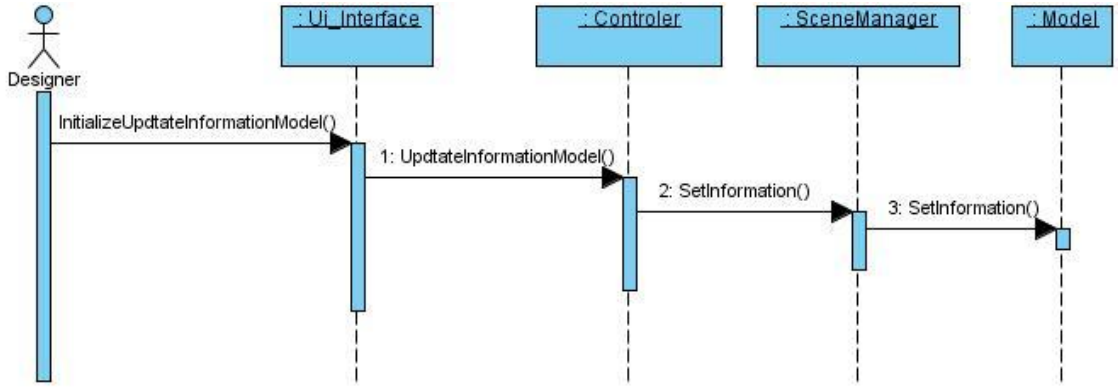



**Diagrama de secuencia CU- Modificar información de los objetos escenario Actualizar información.**

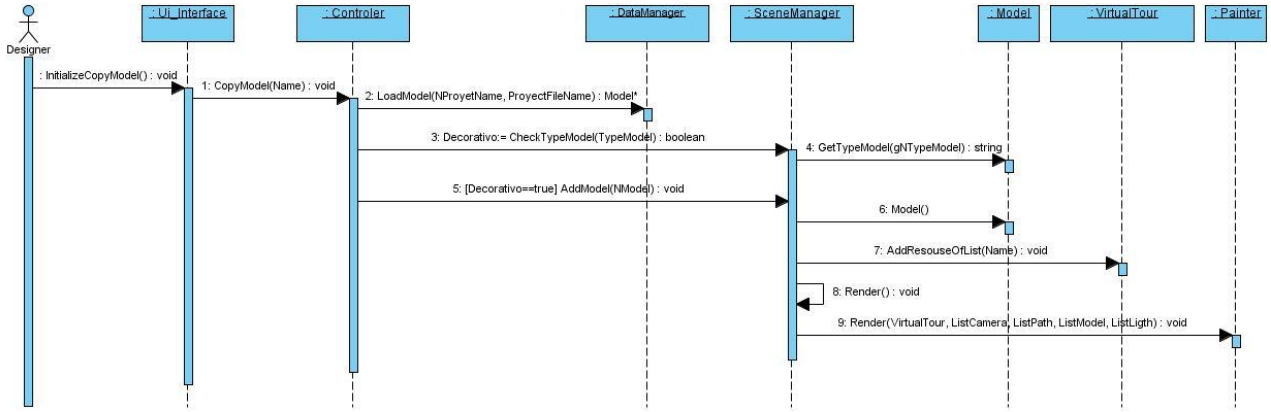



**Diagrama de secuencia CU- Modificar objeto decorativo Duplicar objetos decorativos.**

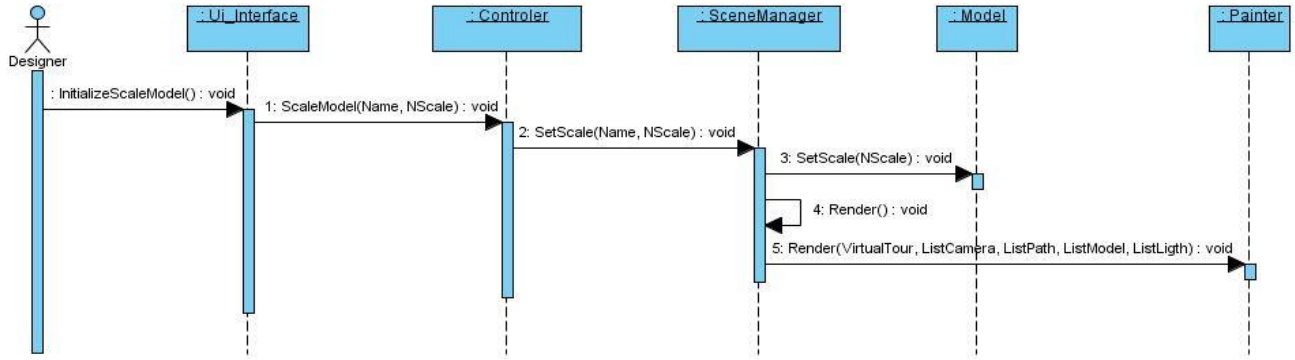



**Diagrama de secuencia CU-Modificar objeto decorativo Escalar objetos decorativos.**

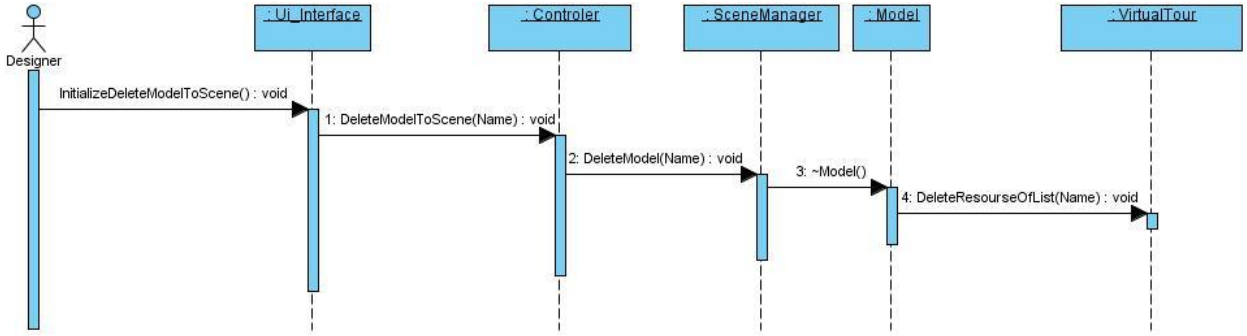



**Diagrama de secuencia CU- Administrar objetos del entorno escenario Eliminar objeto del entorno.**

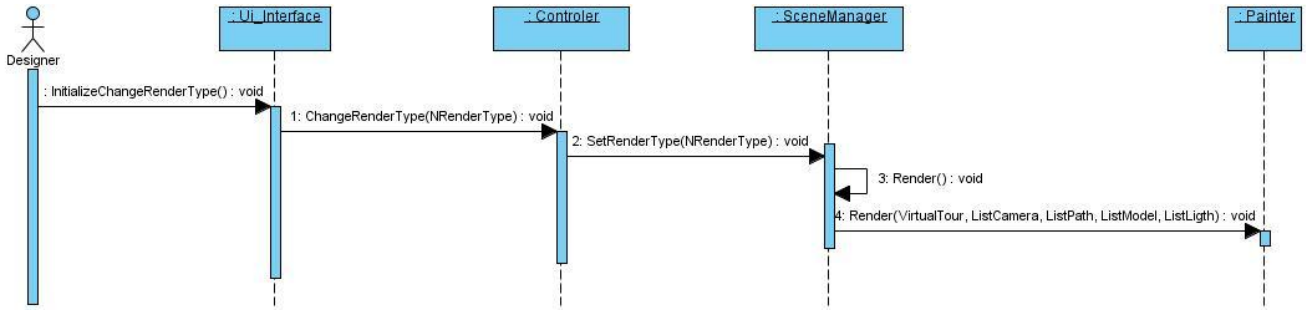



**Diagrama de secuencia CU- Administrar objetos del entorno escenario Cambiar modo de visualización.**

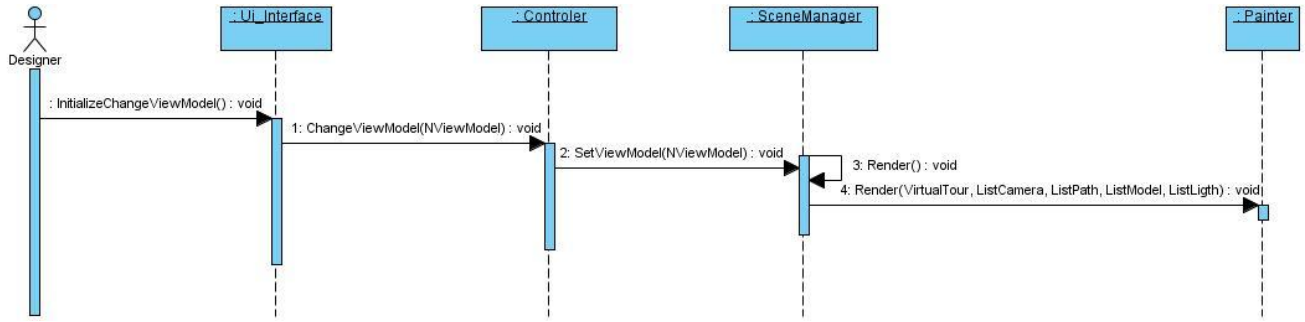



**Diagrama de secuencia CU- Administrar objetos del entorno escenario Cambiar vista del objeto.**

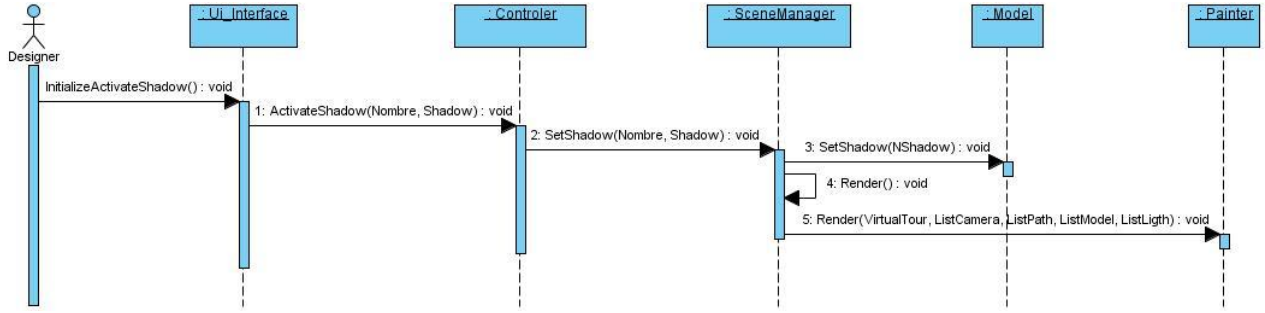



**Diagrama de secuencia CU- Administrar objetos del entorno escenario Activar sombra.**

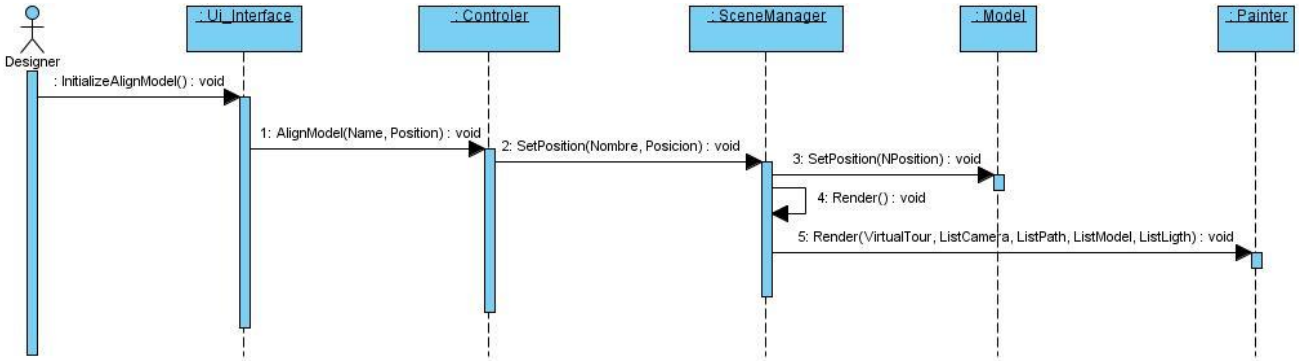



**Diagrama de secuencia CU- Administrar objetos del entorno escenario Alinear objeto.**

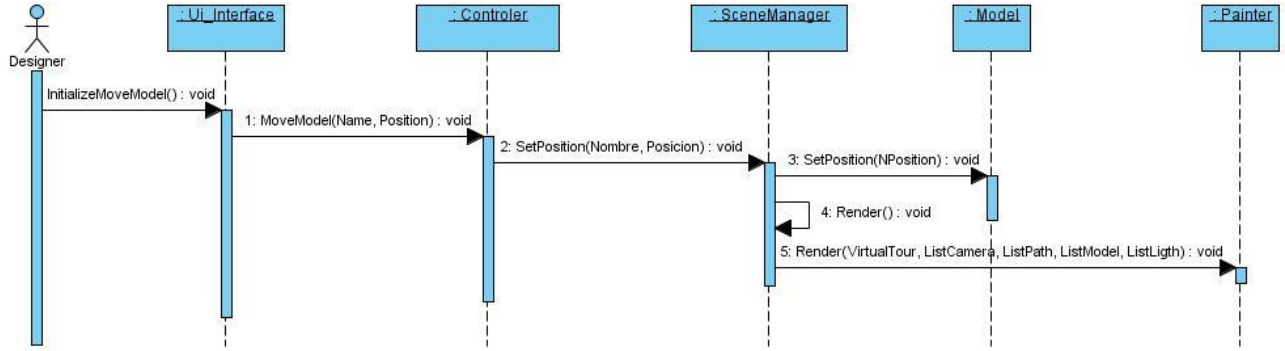



**Diagrama de secuencia CU- Administrar objetos del entorno escenario Mover objeto del entorno.**

## Anexo 6. Clases arquitectónicamente significativas

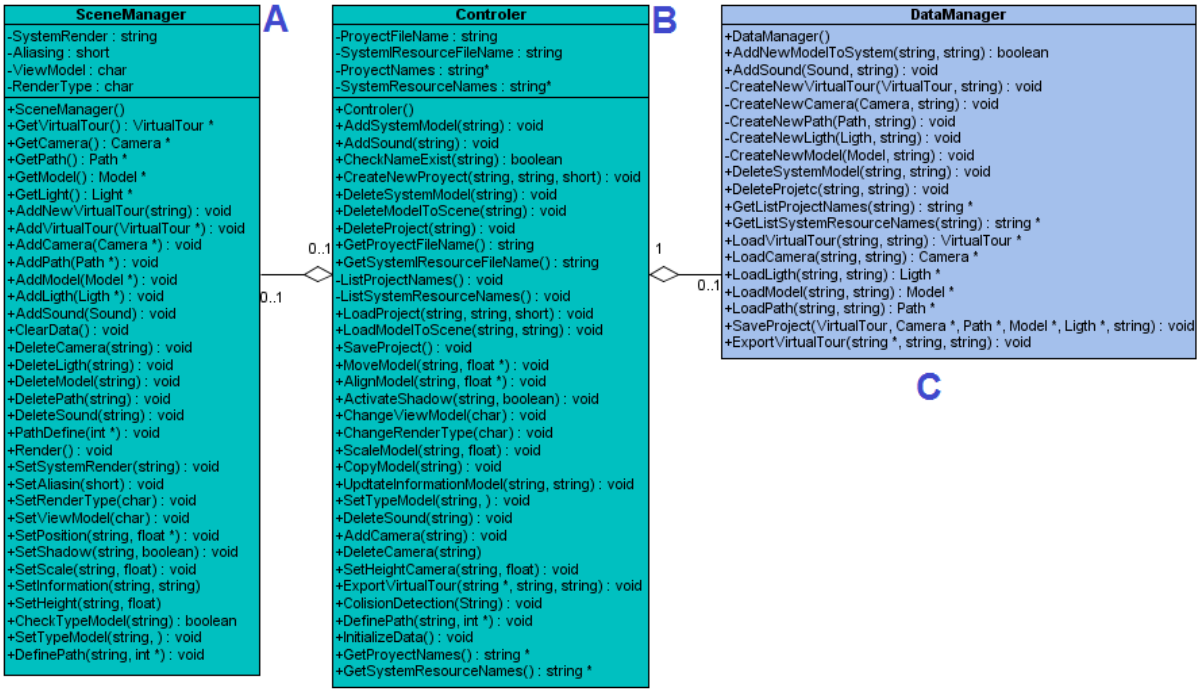



**Ilustración 1. Clases arquitectónicamente significativas del módulo Editor.**

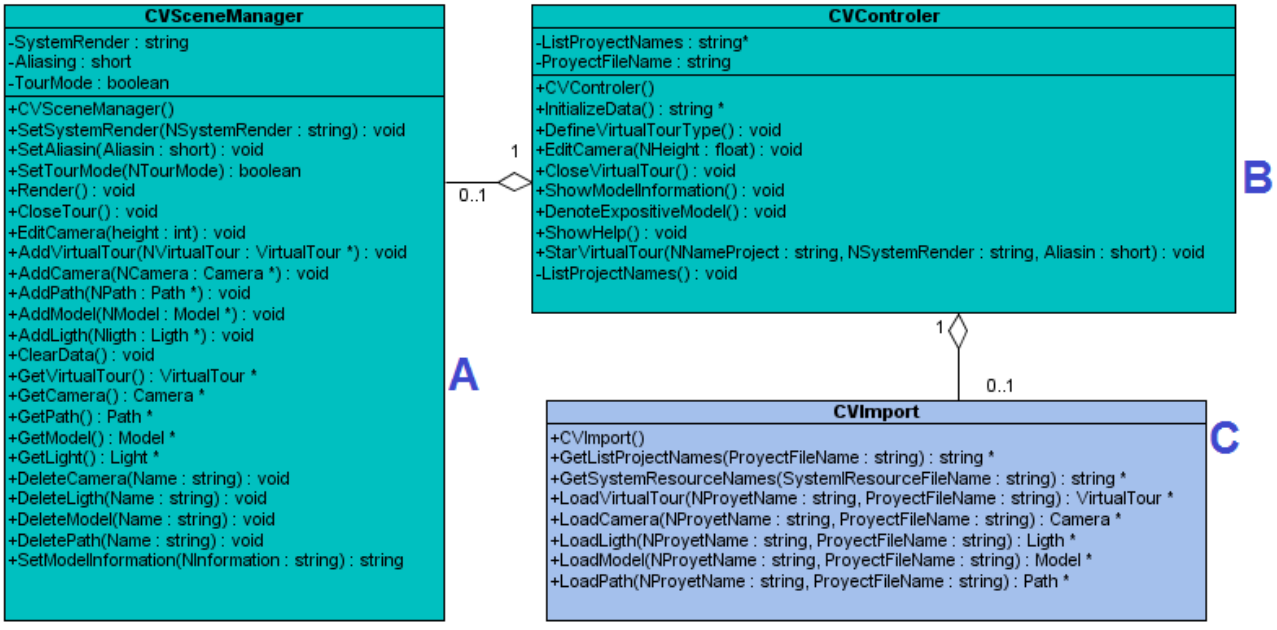



**Ilustración 2. Clases arquitectónicamente significativas del módulo Visualizador.**