UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

TÍTULO: HERRAMIENTA DE AUTOR PARA LA CREACIÓN DE CONTENIDO DE REALIDAD AUMENTADA

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

AUTORES: Yadira Verdecia Álvarez.
Liannet Peña Guerra.

TUTORES: Ing. Ernesto de la Cruz Guevara Ramírez.
Ing. Mileydi Moreno Mirabal.

Ciudad de La Habana, junio del 2010.

DECLARACIÓN DE AUTORÍA

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

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

_____ _____

Yadira Verdecia Álvarez — Autor

Liannet Peña Guerra — Autor

_____

Ing. Ernesto de la Cruz Guevara Ramírez

Tutor

_____

Ing. Mileydi Moreno Mirabal

Tutor

AGRADECIMIENTOS GENERALES

A la Revolución Cubana y a Fidel, por sus enseñanzas, por la oportunidad de formarnos en esta Universidad, porque gracias a ella hoy somos profesionales.

A nuestra familia, por su dedicación y sacrificio durante más de 15 años de estudios.

A nuestros tutores por el tiempo que nos dedicaron y su deseo de vernos triunfar.

Al tribunal por el apoyo y la colaboración en el perfeccionamiento de este trabajo.

A todos nuestros profesores, por contribuir con su ejemplo a nuestra formación.

A los amigos que nos apoyaron continuamente, a los que están, a los que estuvieron y a los que estarán.

Gracias a todos.

AGRADECIMIENTOS

El mayor agradecimiento es a mi mamá y mi papá Daniel por confiar siempre en mí y demostrarme que si podía cuando yo pensaba que no. Gracias a ustedes, por el amor constante; por cada palabra de aliento; por cada segundo de preocupación. Espero que estén muy orgullosos de mí…. ¡Este título es de ustedes!

A mi hermana Yamila. Gracias por estar siempre a mi lado, y en especial por compartir conmigo estos 5 años; por ser mi compañera y amiga; por confiar en mis posibilidades, por apoyar mis logros y criticar mis errores.

A mi hermanito del alma Maikel para que luche siempre en busca de sus sueños, y para que tenga la certeza de tenerme siempre allí para él.

A mi segundos padres, mi tía Leonor y a mi tío William, por todo su amor y apoyo incondicional.

A mis tutores por su asesoría constante y preocupación; por tanta ayuda; por todo el interés y la confianza depositada.

A Liannet mi compañera de tesis, gracias por todo, desde el plano profesional donde siempre trabajamos juntas, hasta nuestra convivencia en estos años.

A mi novio Jorge (Bebe) que aunque sea el último en mis agradecimientos es el primero en muchos aspectos importantes de mi vida.

A mis amigos, con los cuales he compartido desde el primer día que entré a esta universidad y me llevo lo mejor de cada uno. Junto a ustedes aprendí que aceptar a las personas tal y como son, resulta una forma inteligente de convivencia.

Agradezco a todas y cada una de las personas que contribuyeron en mi formación como profesional y como ser humano, a todos los que de una forma u otra aportaron un granito de arena para que este sueño se hiciera realidad.

A todos ustedes muchas gracias… Yadira

Agradecimientos

A mi mamita Miriam por ser la madre más especial del mundo, por el amor constante, el cariño ilimitado y la entrega que siempre ha tenido para conmigo, por apoyarme y darme ánimos cuando más lo necesito, por incitarme a seguir adelante, a no dejarme vencer ante las dificultades y por ser un ejemplo de mujer, de profesional, de madre.

A mi hermana Lianis por siempre estar presente en todo este tiempo, por brindarme tantos momentos de alegría, por su apoyo incesante, por su cariño.

A mis abuelos Raquel y Ángel por su amor y cariño infinito, por estar siempre a mi lado, ayudándome en todo lo que me hizo falta y por el esfuerzo que han hecho para que hoy yo sea una profesional.

A mis tíos Mabel y Eduardo por su cariño, por su ayuda y sus buenos consejos.

A mi papá por apoyarme en cada una de las decisiones que he tomado en mi vida.

A mis tutores por su confianza para la asignación de esta tarea, por su disposición constante sin importar hora o momento, por su ayuda persistente, por todo Gracias!!!

A Yadira mi compañera de tesis, le doy las gracias por haberme dejado compartir esta experiencia junto a ella, por haber aprendido tanto de ella en el plano profesional, porque todo este tiempo ha servido para cultivar una gran amistad.

A mis vecinos Obdulia, Hastie, Yudi, Adis, Héctor y Nolberto por su cariño, por estar siempre presentes en cada despedida y en cada recibimiento en todos estos años, por la amistad y el amor que me han brindado, por el apoyo incondicional.

A mis amigas Lisset, Marisleydis y María Antonia por su presencia en cada momento de mi vida, porque pese a la distancia y a estos 5 años que hemos estado separadas siempre han estado conmigo en los buenos y en los malos momentos, brindándome su apoyo y su hombro cuando ha hecho falta, compartiendo las alegrías y los triunfos de cada una, a ustedes que las quiero mucho de veras Gracias!!!

Agradecimientos

A mi novio Ricardo por su cariño, su paciencia y su comprensión, por enseñarme tantas cosas tanto en lo personal como en lo profesional, gracias por estar a mi lado.

A todas aquellas compañeras de estudio y amigas con las que he convivido desde mi primer año en la UCI, y especialmente a mis compañeras de cuarto, por su amistad, por estar siempre que las he necesitado, por su inmenso cariño, por soportarme todo este tiempo y porque aunque no lo demuestre a veces, no saben cuán importante han sido, son y serán para mí, a todas ellas que las quiero mucho, Gracias!!!

A todas mis amistades, compañeros de estudio y personas que he conocido en estos 5 años de la carrera por la ayuda que me han brindado de una forma u otra, gracias por estar siempre para mí, a los que estuvieron y a los que están.

A todos, muchas Gracias !!! .... Liannet

DEDICATORIA

A mis padres y mi hermana que son todo para mí al igual que yo lo soy para ellos. Que han sufrido mi carrera tanto o más que yo.

Por tanto amor, por enseñarme que la perseverancia, la dedicación y el empeño son las armas claves para el logro de nuestros objetivos.

Por guiarme para ser la persona que soy hoy.

Yadira.

A mi mamita, a mis abuelos Raquel y Ángel, y a mi tío Eduardo por su amor, su cariño y su esfuerzo constante para verme graduada y con un título de ingeniera.

A mis hermanos Lianis, Alcides y Enrique, a mis niños Hamlet, Hansel, Daimy, Dayanis, Jonathan y María Carla para que les sirva de ejemplo en su vida futura.

Y en especial a mi hermanito Leandris aunque ya no esté!!!

Liannet.

RESUMEN

La Realidad Aumentada permite insertar objetos, cuerpos o información senso-perceptual de carácter virtual generados por computadoras en un entorno de la realidad, con la condición de ser interactiva con el usuario en tiempo real. Esta técnica tiene un amplio rango de aplicaciones a nivel mundial. Debido a las potencialidades que brinda, en la línea de Interacción 3D del Departamento de Visualización y Realidad Virtual de la facultad 5, se están realizando estudios relacionados con el desarrollo de componentes de software con el objetivo de obtener una base de conocimientos que posibilite crear un punto de partida para la elaboración de aplicaciones de forma rápida.

Producto a la necesidad de mejorar el proceso de edición de escenas de Realidad Aumentada, se desarrolló una Herramienta de Autor para la creación de contenido de Realidad Aumentada; que reduce el tiempo en que estas escenas son editadas, así como el margen de errores cometidos en el proceso de edición manual.

En la investigación se abordan los conceptos más importantes relacionados con la Realidad Aumentada, la Realidad Virtual y las Herramientas de autor. Se realizó un estudio detallado de los diferentes software para Realidad Aumentada existentes a nivel mundial. Se fundamentó la selección de la metodología, las herramientas y tecnologías que permitieron describir, diseñar e implementar la solución propuesta.

PALABRAS CLAVE: Realidad Aumentada, Herramienta de Autor.

TABLA DE CONTENIDO

Tabla de Contenido

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

La Realidad Aumentada permite insertar objetos, cuerpos o información senso – perceptual de carácter virtual generados por computadoras en un entorno de la realidad, con la condición de ser interactiva con el usuario en tiempo real. El surgimiento de nuevas alternativas tecnológicas para realizar aplicaciones de Realidad Aumentada (RA) permite a la facultad 5 de la Universidad de las Ciencias Informáticas realizar investigaciones relacionadas con la Realidad Aumentada, para crear una base de conocimientos que tribute directamente a la producción, ofreciendo como beneficios la elaboración de productos con valor agregado y de valor comercial en el ámbito nacional e internacional.

Actualmente la línea de Interacción 3D del Departamento de Visualización y Realidad Virtual trabaja en la elaboración de productos para realidad aumentada, durante el desarrollo de los mismos han surgido algunos inconvenientes en la edición y configuración de las escenas de realidad aumentada, debido a que el diseñador tiene que realizar de forma manual los cambios en el código fuente y luego comprobar que la escena tiene la configuración deseada ejecutando el software de visualización. En caso de que no se logre la configuración requerida, el diseñador tiene que repetir el mismo proceso una y otra vez hasta alcanzar el resultado deseado.

Lo anteriormente descrito constituye una de las principales problemáticas que afronta la línea de Interacción 3D, pues todo este proceso trae como consecuencia demoras cuando se va a modificar algún elemento dentro de la escena. Además, en ocasiones se corre el riesgo de que en medio de estos cambios surja una confusión y se pierda el hilo del trabajo, debido a que el diseñador no se enfoca realmente en la tarea que debe realizar. Estas situaciones tienen lugar porque el diseñador tiene que resolver inconvenientes, inherentes al proceso de edición manual, que puedan aparecer en el proceso de edición.

Por la situación problémica expuesta se plantea el siguiente **problema científico**: La necesidad de mejorar el proceso de configuración de escenas para realidad aumentada.

El problema descrito está contenido en el siguiente **objeto de estudio**: las herramientas de autor para la creación de contenido.

El **objetivo general** que se desea alcanzar para darle cumplimiento al presente trabajo es: Implementar un software para la edición escenas de realidad aumentada en tiempo de diseño, mediante el uso de una interfaz gráfica de usuario.

Por consiguiente, el **campo de acción** de la investigación es: las herramientas de autor para la creación de contenido de Realidad Aumentada.

De acuerdo con el problema científico planteado la **Hipótesis** definida es la siguiente:

Si se elaborara un software para mejorar el proceso de edición de escenas de realidad aumentada mediante el uso de una interfaz gráfica de usuario, entonces se logrará un ambiente de trabajo que permita reducir el tiempo en que estas escenas son editadas y reducir el margen de errores cometidos en el proceso de edición manual.

Para darle cumplimiento al objetivo trazado se determinó que las **tareas a realizar** en esta investigación son las que se muestran a continuación:

- Estudio del estado actual del desarrollo de software de edición de contenidos para Realidad Aumentada existentes en el ámbito nacional e internacional, para identificar las características básicas que debe presentar un software de este tipo.
- Análisis de las ventajas y desventajas del estándar XML aceptado internacionalmente en el manejo y divulgación de información persistente, con el objetivo de utilizarlo para la representación de las configuraciones de las escenas.
- Análisis y estudio de los diferentes software existentes en el mercado para la selección de las funcionalidades que no deben faltar en el software.
- Estudio de las herramientas OpenSceneGraph, Qt4, ARToolKit y OSGART para elaboración del software.
- Selección del lenguaje, metodología y entorno integrado de desarrollo para elaborar el software.

Para el desarrollo de las tareas se utilizan como métodos científicos:

A nivel teórico.

- **Métodos de análisis-síntesis e inducción-deducción:** Para el estudio de las concepciones y conceptos empleados en el campo; para analizar las teorías y documentos generados por desarrolladores de otras herramientas de autor. En la revisión y justificación de la metodología de desarrollo de software y tecnologías a utilizar.
- **Análisis histórico-lógico:** Para conocer, con mayor profundidad, los antecedentes y las tendencias actuales referidas a las herramientas de autor existentes, además de conceptos, términos y vocabularios propios del campo como herramienta de autor, realidad virtual, realidad aumentada, entre otros, que contribuyen en gran medida al entendimiento del trabajo.
- **Modelación:** Se aplicará para reflejar la estructura, relaciones internas y características de la solución a través de diagramas.

A nivel Empírico.

- **Entrevistas:** Al jefe del proyecto y otras personas especialistas en el tema de las herramientas de autor y realidad aumentada, que nos permita obtener una guía para el desarrollo del sistema.

El presente trabajo de diploma está estructurado de la siguiente manera:

Un **primer capítulo** que fundamenta todo lo concerniente al objeto de estudio, otras soluciones existentes; así como a los conceptos asociados al problema en cuestión. Además, se hace referencia a las tendencias y tecnologías existentes en la actualidad, que presentan el contenido referido a las fundamentaciones teóricas que sirven de base para el desarrollo de la futura solución a proponer.

Un **segundo capítulo** donde se profundiza en el problema a resolver a través de su descripción y se muestra el modelo de dominio generado. Además, se realiza la propuesta de la solución y se identifican los requisitos funcionales y no funcionales que deben tenerse en cuenta. Por último, se hace una descripción de los Casos de Uso obtenidos a partir de los requisitos.

Un **tercer capítulo** como diseño e implementación, presenta los Diagramas de Clases del Diseño, los Diagramas de Interacción del Diseño, así como el Diagrama de Despliegue y de Componente.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

Introducción.

En este capítulo se exponen algunos conceptos relacionados con las herramientas de autor y realidad aumentada, así como, cuáles son los objetivos, características y aplicaciones de esta última. Se hace además especial énfasis en algunas herramientas de autor para realidad aumentada, destacando de estas sus características, ventajas y desventajas. Además, se profundiza en el estudio de las herramientas, tecnologías y metodología a utilizar para el desarrollo del sistema.

1.1 Realidad Aumentada.

La realidad aumentada es un concepto reciente, establecido originalmente por Pierre Wellner en 1993, donde se define a la realidad aumentada como el opuesto de la realidad virtual en cuanto a la diferencia entre sumergir al usuario en un mundo absoluto de información, contrapuesto a aumentar el mundo real con información adicional. La cual hace posible agregar información a la realidad, es decir, la inserción de elementos digitales a la realidad de un sujeto en particular. En tiempo real la computadora procesa la perspectiva en que el sujeto percibe la realidad, y calcula e inserta elementos predeterminados a la realidad.

Esta técnica ha encontrado diversas aplicaciones como la medicina, entretenimiento y militar, que han sustentado herramientas comerciales a nivel mundial. Lamentablemente para nuestro país, los costos para implementación de esta técnica la han mantenido alejada, debido a que los recursos que se necesitan para el desarrollo de aplicaciones de este tipo son costosos y de difícil adquisición, sin embargo con el objetivo de incursionar en esta tecnología nuestro país y en especial la universidad, han puesto en práctica diversas alternativas con el propósito desarrollar nuestros propios sistemas de realidad aumentada.

El principal objetivo en el diseño de sistemas de realidad aumentada es que éstos lleguen a ser tan portables, ligeros, pequeños y robustos como sea posible y que permitan al usuario explorar cualquier entorno no preparado, ya sea interior o exterior, sin ninguna restricción.

1.1.1 Objetivos de la Realidad Aumentada.

La integración del mundo real con el entorno virtual es lo que trae como resultado la realidad aumentada y cual tiene entre sus objetivos:

- Mejorar la interacción con el mundo real.
- Integrar el uso del ordenador en actividades cotidianas.
- Posibilitar el acceso a usuarios diversos y no especializados.
- Los objetos cotidianos se convierten en objetos interactivos.
- Trasladar el foco de atención del ordenador al mundo real. La información se traslada al mundo real, en lugar de introducir el mundo real en el ordenador (realidad virtual).
- Construir entornos interactivos que aumenten directamente los sentidos de un usuario con material generado por ordenador.

1.1.2 Características de un sistema de Realidad Aumentada.

Se define un sistema de realidad aumentada como aquél con las siguientes propiedades o características:

- Combina objetos reales y virtuales.
- Es interactivo y en tiempo real.
- Se alinean los objetos virtuales y reales unos con otros. **[1]**

1.2 Herramienta de Autor.

Las herramientas de autor (también denominados entornos de autor o lenguajes visuales) son aplicaciones informáticas que permiten elaborar sistemas. Ofrecen un entorno de trabajo que permite una programación basada en íconos, objetos y menús de opciones, los cuales posibilitan al usuario realizar un producto sin necesidad de escribir una sola línea de código en algún lenguaje de programación. Los íconos u objetos se asocian a las exigencias del creador, de tal modo que existen íconos para reproducir

sonidos, mostrar imágenes (gráficos, fotografías, videos), controlar dispositivos y/o tiempos, activar otros programas, crear botones interactivos, etc. **[2]**

La selección de una herramienta de autor dependerá fundamentalmente de 2 factores:

- Las características particulares de la aplicación a desarrollar y la formación y experiencias del propio desarrollador.
- Es importante tener en cuenta las posibilidades del autor en materia de programación y la portabilidad de la aplicación hacia los sistemas operativos de los usuarios.

1.2.1 Ventajas de las Herramientas de Autor.

Las principales ventajas de estos sistemas de última generación son dos:

- Reducen el tiempo de desarrollo de aplicaciones hasta 1/8 del tiempo requerido con las formas de trabajo anteriores.
- Resultan más fáciles y rápidos de aprender que lenguajes de programación tradicionales.

Al ser diseñados para un propósito específico, muchas de las necesidades más habituales de los creadores de software son previstas de antemano. Además, muchos de los programas de autor disponibles en el mercado actualmente son multiplataforma, capaces de funcionar con distintos tipos de sistemas operativos y ordenadores, lo que facilita su utilización en, prácticamente, todas las circunstancias.

1.2.2 Herramientas de Autor para Realidad Aumentada.

Existen varias herramientas de autor para la creación de aplicaciones de escritorio para Realidad Aumentada. Las cuales están ampliamente organizadas en dos tipos: **[3]**

- Herramientas de creación de contenido de Realidad Aumentada para los programadores.
- Herramientas de creación de contenido de Realidad Aumentada para los no programadores.

Las herramientas de autor para los programadores suelen ser las bibliotecas de código que requieren conocimientos de programación, mientras que las herramientas para no programadores permiten crear aplicaciones sin escribir una sola línea de código mediante interfaces que permiten el trabajo con íconos, objetos y menús de opciones.

Las categorías antes mencionadas se pueden organizar además en herramientas de bajo nivel las cuales requieren de habilidades de programación y en herramientas de alto nivel que utilizan bibliotecas de alto nivel o técnicas visuales de autoría.

En la siguiente tabla se muestran algunas de las herramientas más usadas por cada una de las categorías antes mencionadas.

	Programadores.	**No Programadores.**
Bajo nivel.	ARToolkit. ArTag.	DART. ComposAR.
Alto nivel.	Studierstube. OsgART.	AMIRE. MARS.

Tabla 1. Algunos tipos de Herramientas de Autor. **[3]**

Herramientas de Autor para Programadores.

Existen un sin número de bibliotecas de programación que permiten a los desarrolladores crear aplicaciones de Realidad Aumentada. Estas bibliotecas tienen como características comunes que normalmente requieren de capacidad de programación en lenguajes como C o C + +, además son herramientas de desarrollo para producir contenido de Realidad Aumentada y por último demora un tiempo relativamente largo utilizarlas para crear aplicaciones de Realidad Aumentada. **[3]**

Herramientas de Autor para No Programadores.

Hay otro conjunto de herramientas de autor que han sido desarrolladas para los no programadores como artistas o diseñadores. Una característica común de estas herramientas para no programadores es que utilizan técnicas de programación visual o de scripting simple para apoyar los prototipos rápidos, que son interpretativas y no compilados, permitiendo el rediseño rápido de ideas, y que se integren en otras herramientas de diseño. Algunas de estas herramientas de autoría basadas en computadoras también fueron diseñadas para crear aplicaciones móviles de Realidad Aumentada, otras como el BuildAR permite a los usuarios asociar modelos virtuales con marcadores de seguimiento visual desde un nivel más básico. **[3]**

1.3 Herramientas de Autor para Realidad Aumentada existentes en el mercado.

Con el propósito de fomentar el desarrollo y la investigación de esta tecnología, diversas instituciones y personas han facilitado para distribución gratuita software base para generar aplicaciones de Realidad Aumentada. En general se hace disponible el código fuente del programa, o una versión de demostración del programa. Algunos requieren de software adicional que funcione como base para poder ejecutarse y la mayoría de estos programas se encuentran protegidos por licencias contra su uso comercial.

1.3.1 ARToolkit.

ARToolkit es un conjunto de bibliotecas para C/C++ que sirve para la creación de aplicaciones de Realidad Aumentada. No es directamente un software de creación como Flash, Dreamweaver o Director. Según el criterio de algunos autores es el principal software de Realidad Aumentada, y el primero en haber sido desarrollado y hacerse disponible.

Utiliza las capacidades de seguimiento de video, a través de técnicas de visión por computadoras, con el fin de calcular, en tiempo real, la posición de la cámara y la orientación relativa a la posición de los marcadores físicos. Una vez que la posición de la cámara real se conoce, la cámara virtual se puede colocar en correspondencia y los modelos 3D son superpuestos exactamente sobre el marcador real. **[4]**

Así ARToolkit resuelve dos de los principales problemas en la Realidad Aumentada, el seguimiento de objetos y la interacción con el objeto virtual. También proporciona una serie de ejemplos y utilidades de gran ayuda al programador que quiera realizar este tipo de aplicaciones.

Disponibilidad y distribución.

ARToolkit está siendo continuamente desarrollado por el Dr. Hirokazu Kato de la Universidad de Osaka, en Japón y es apoyado por el Human Interface Technology Laboratory de la Universidad de Washington y el Human Interface Technology Laboratory de la Universidad de Canterbury en Nueva Zelandia.

Periódicamente se lanzan nuevas versiones que se distribuyen a través de la página Web de HITLab (**http://www.hitl.washington.edu/artoolkit**). Su distribución es gratuita, y se entrega bajo la licencia GNU General Public License (**http://www.gnu.org/licenses/gpl.html**), lo que permite su uso comercial.

Pre-requerimientos y dependencias de ARToolkit en el entorno Windows.

ARToolkit puede compilarse en los entornos Windows, Linux, SGI, Irix y MacOS. Si bien las funciones que ofrece en las distintas plataformas son las mismas, la instalación del programa varía en cada una de ellas.

El equipo, sistema operativo y plataforma deben satisfacer algunos requisitos mínimos básicos. El hardware utilizado debe ser capaz de recibir un input de video en vivo y tener suficiente capacidad de procesamiento disponible para cumplir con las tareas de proceso del video y display final. Como estas operaciones ocurren en tiempo real, es necesario un procesador capaz de manejarlo. De lo contrario, surgirían problemas como dropped frames y bajos frames por segundos en el aumento.

El software posee también varias dependencias. Estas dependencias son softwares adicionales que de forma accesoria cumplan funciones específicas en pro de ARToolkit para que éste pueda desarrollar su función final.

En esta investigación se decidió utilizar esta herramienta gracias a la flexibilidad, facilidades legales y su posición como el estándar establecido en cuanto a aplicaciones de Realidad Aumentada, en la que

presenta muy buena aceptación. La mayoría de los softwares mencionados en este capítulo se basan en dicha biblioteca.

1.3.2 Mr.Planet.

Mr.Planet es una aplicación de Realidad Aumentada que establece un diálogo sencillo con el usuario. El usuario es capaz de relacionar modelos 3D realizados con una herramienta de diseño (AutoCad, 3D Studio, entre otros) con una gran variedad de patrones de Realidad Aumentada.

Permite con un sencillo menú algunas opciones como rotar, escalar y trasladar el modelo 3D con respecto al patrón. Se incluye en la aplicación un sistema multipatrón que permite relacionar un mismo modelo 3D a varios patrones de tal forma que el modelo se dibujaría en su posición con tal solo verse un patrón de su lista. Por otro lado, cada patrón puede tener relacionados varios modelos y al poder trasladarse se podrá dibujar en sitios separados. **[5]**

Mr.Planet se sirve de las bibliotecas de OGRE para dibujar las imágenes tridimensionales en el mundo de la Realidad Aumentada, OGRE es un motor gráfico 3D de código libre utilizado en Mr.Planet, que le permite no depender del sistema operativo, ni subsistema de render. **[5]**

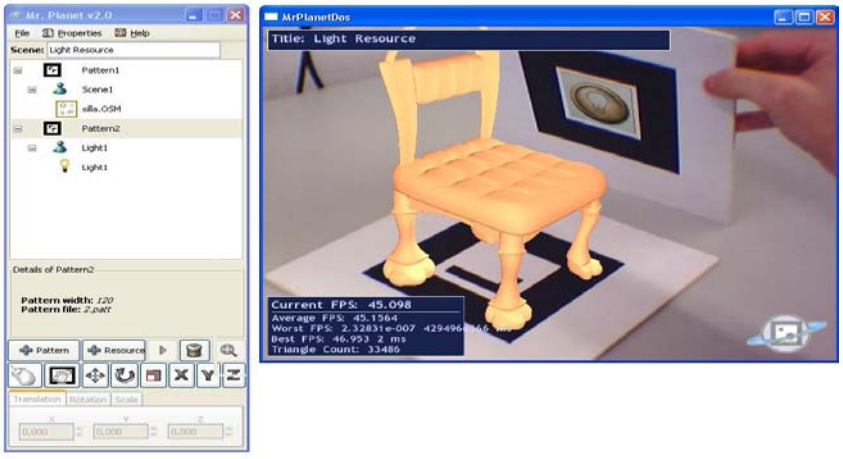

Figura 1. Interfaz de Mr.Planet.

Mr. Planet presenta algunos problemas a la hora de exportar más de un archivo comprimido (formato ZIP) y no es una herramienta de código libre; es por ello que se decide tener en cuenta solo aquellas características y funcionalidades que se consideran indispensables para la herramienta.

1.3.3 DART- Designers' Augmented Reality Toolkit.

La Herramienta de Realidad Aumentada para Diseñadores (DART por sus siglas en inglés) está diseñada para dar soporte a la construcción rápida de prototipos de aplicaciones de Realidad Aumentada que usan dispositivos de visión. DART está constituido por una colección de extensiones del entorno de desarrollo de multimedia Macromedia Director. Su diseño está pensado para apoyar la potencia de Director y asume que los desarrolladores están familiarizados con esta aplicación. El seguimiento de los marcadores en el video en tiempo real se hace a través de la ARToolKit. **[6]**

Inicialmente el desarrollo de la herramienta DART estuvo motivado por el interés de sus autores en las experiencias para la educación informal, arte digital y entretenimiento. Sin embargo, puede ser usado para crear aplicaciones de Realidad Aumentada en cualquier dominio (en la industria, en lo militar y en aplicaciones científicas) y es especialmente útil en la exploración. Su diseño hace que pueda ser usado por todo el que lo desee, tanto diseñadores de oficio como aficionados, artistas e investigadores.

Está liberado bajo una licencia pública, que impone una única restricción en la que se expresa que no se puede vender DART en sí mismo o con pequeños cambios, pero se pueden crear todas las aplicaciones deseadas con cualquier propósito. Este modelo de licencia, aunque parece factible, no lo es, porque para poder utilizar DART en la elaboración de cualquier tipo de aplicación es necesario tener las herramientas propietarias sobre las que él está sustentado.

Se decidió no utilizar esta herramienta, debido a los requerimientos de software y hardware que exige, el modelo de licencia sobre las que está sustentada, además de que no es multiplataforma.

1.3.4 BuildAR.

BuildAR es una herramienta para crear escenas de Realidad Aumentada. Proporciona una visión de video en vivo y permite crear e imprimir sus propios diseños de patrones.

La interfaz de usuario de BuildAR permite cargar archivos de Patrón-Marcador, añadir objetos 3D, posición, orientación y la escala de los objetos. Además se puede guardar y cargar las escenas. BuildAR viene con dos patrones por defecto, HIRO y KANJI. Actualmente BuildAR sólo está disponible en la plataforma Windows, se mantiene bajo desarrollo activo y no ha sido liberado a alguna comunidad de software libre. **[7]**

Se decidió no utilizar esta herramienta debido a que como ya se ha expuesto se prohíbe su uso comercial, además de que no es multiplataforma.

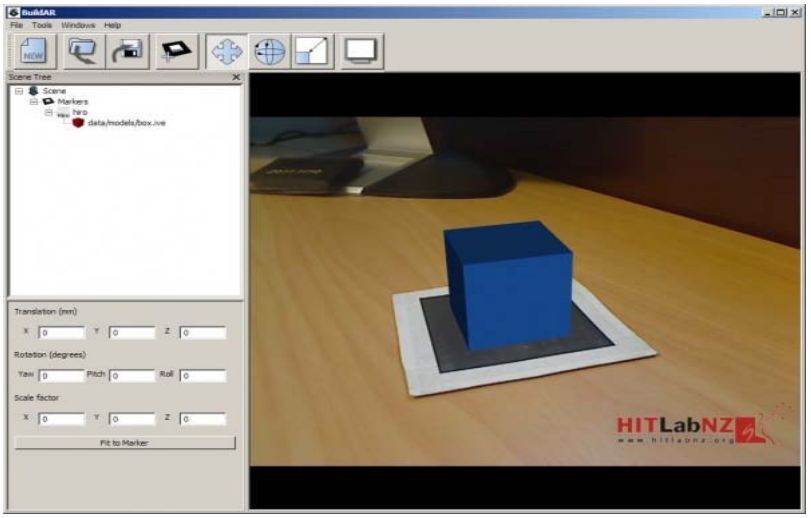

Figura 2. Interfaz del BuildAR.

1.3.5 Atomic.

ATOMIC es una herramienta que permite la creación de aplicaciones de Realidad Aumentada, desarrollada especialmente para no-programadores. Fue creado como un Front end (Interfaz Gráfica) para usar la librería ARToolkit sin tener que saber programar. **[8]**

Fue escrito en el lenguaje de programación Processing y está licenciado bajo la Licencia GNU GPL. Es Multi-plataforma lo que permite que se pueda usar en los sistemas operativos Microsoft Windows, Ubuntu y Mac OS X. **[8]**

La primera versión experimental de ATOMIC fue desarrollada el 7 de septiembre del 2008 y la primera versión estable fue la 0.6 desarrollada el 6 de marzo del 2009.

La principal motivación de ATOMIC es proporcionar a la comunidad una herramienta de código abierto que se pueda modificar con facilidad y no requiere demasiados conocimientos técnicos para acceder a la tecnología de la Realidad Aumentada.

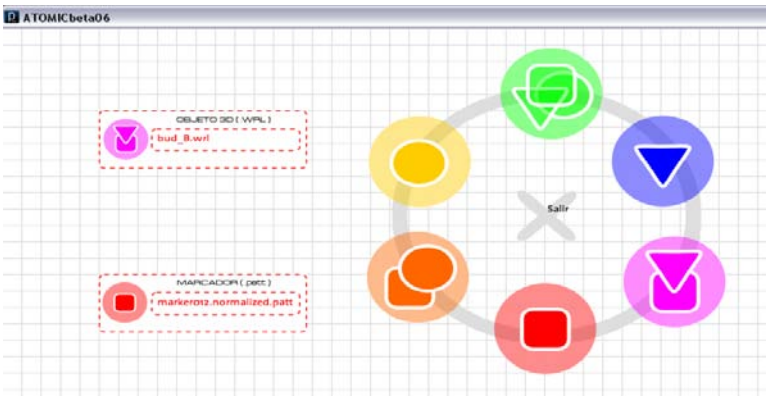

Figura 3. Entorno gráfico del ATOMIC.

Se decidió no utilizar esta herramienta ya que es una aplicación que cuanta con pocas funcionalidades, fue creada para realizar pequeñas y simples aplicaciones de Realidad Aumentada.

1.3.6 ComposAR.

La herramienta de autoría llamado ComposAR se basa en OSGART. ComposAR está dirigido a los usuarios con ningún o poco conocimiento de programación. Es Multi-plataforma lo que permite que se pueda usar en distintos sistemas operativos.

ComposAR imita la apariencia y funcionalidad de una herramienta de modelado 3D. Está escrito en Python lo que significa que la interfaz de usuario y el comportamiento en tiempo de ejecución puede ser fácilmente personalizados. Constituye una base para aplicaciones Realidad Aumentada en la educación, el diseño y actividades orientadas a la investigación. **[9]**

Se llegó a la conclusión de no utilizar esta herramienta debido a que la misma se encuentra en fase de desarrollo, no está disponible y existe muy poca documentación.

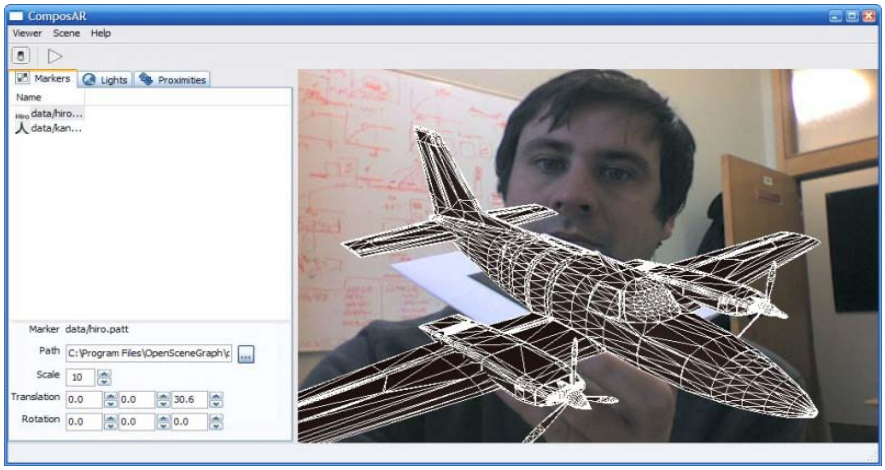

Figura 4. Entorno gráfico de ComposAR.

1.3.7 ARTag.

Es un sistema de reconocimiento de marcadores para Realidad Aumentada inspirado en ARToolkit, fue desarrollado por el National Research Council de Canadá. ARTag consiste en una biblioteca de tramas predefinidas que han sido programadas para ser fácilmente reconocidas por una cámara al ser aplicada en superficies planas. **[10]**

Su objetivo es resolver los problemas que surgen en la detección de marcadores entre los que se encuentran los falsos positivos: cuando el sistema muestra la presencia de un marcador, pero no existe; el problema de los falsos negativos: cuando el sistema no muestra la presencia de un marcador, pero existe; y el problema de la confusión: cuando el marcador en el medio ambiente es un sistema y lo identifica como otro.

Su sistema de reconocimiento de marcadores es más robusto que el de ARToolkit, pero al mismo tiempo considerablemente más exigente. Se distribuye como demo o como un SDK con previo acuerdo de su licencia. Se prohíbe su uso comercial. **[10]**

A pesar de que esta herramienta permite trabajar con costos mínimos, de no emplear otro software y ser aparentemente más robusto que ARToolkit se decidió no utilizarla debido a que se prohíbe su uso comercial.

Resultado del estudio de las herramientas para Realidad Aumentada.

En la **tabla 2** se realiza una comparación de las herramienta de autor para Realidad Aumentada estudiadas en cuanto a la portabillidad y licencia comercial. En la **tabla 3** se expone una comparación de las herramientas de autor para RA en cuanto a sus funcionalidades.

Herramientas de Autor para RA	Multiplataforma	Licencia comercial
Mr.Planet	No	Si
DART	No	Si
BuildAR	No	Si
Atomic	Si	No
ComposAR	Si	__
ARTag	Si	Si

Tabla 2. Comparación entre las herramientas de autor para Realidad Aumentada en cuanto a su portabilidad y licencia comercial.

Se pudo constatar que solo tres de las herramientas estudiadas se encuentran disponibles para todos los sistemas operativos, que la mayoría de ellas están bajo licencia comercial y que existe una herramienta como es el caso de CamposAR que aún está en fase de desarrollo por lo que no se puede determinar el tipo de licencia que esta posee.

Funcionalidades	Mr. Planet	DART	BuildAR	Atomic	ComposAR	ARTag
Cargar modelos 3D.	X			X		
Cargar marcador.			X	X		
Tipo de marcador estilo ARToolkit.	X	X	X	X	X	Codif. binaria
Cargar multimarcadores.						
Asociar modelo-marcador.	X	X	X	X	X	X
Escalar modelo.	X		X		X	
Rotar modelo.	X		X		X	
Trasladar modelo.	X		X		X	
Reproducir animaciones.	X					
Guardar escena.	X	X	X	X	X	X
Cargar escena.	X	X	X	X	X	X
Pantalla completa.			X			
Generar Patrones.			X			
Calibración de la cámara.						
Fuentes de captura de video.						

Tabla 3. Comparación de las HA para Realidad Aumentada en cuanto a sus funcionalidades.

Al identificar las características más prominentes de este tipo de herramientas, se pudo constatar de que una por sí sola no reúne todas las funcionalidades deseables en la herramienta desarrollada. La afirmación anterior se apoya en que cada una de estas herramientas en particular cuenta con funcionalidades que la diferencian de las demás, que en muchas ocasiones son complementarias entre sí y que son de gran utilidad si se encuentran todas reunidas en una sola solución. Además se identificaron otro número de funcionalidades que no presentan ninguna de las anteriores herramientas y que se consideran indispensables para la solución. Es por ello que se decide elaborar una herramienta que agrupe todas las funcionalidades expuestas en la **tabla 3.**

1.4 Metodología de Desarrollo de Software: RUP.

El proceso unificado de desarrollo (RUP) es una metodología para la ingeniería de software que va más allá del mero análisis y diseño orientado a objetos para proporcionar una familia de técnicas que soportan el ciclo completo de desarrollo de software. RUP es un proceso de desarrollo de software que junto al Lenguaje Unificado de Modelado (UML) constituyen la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos. **[11]**

RUP divide el proceso de desarrollo en una serie de ciclos que constituyen la vida de un sistema. Cada ciclo está compuesto por cuatro fases, cada fase finaliza con un hito y dentro de cada una, el trabajo se puede descomponer en iteraciones. Los hitos tienen muchos objetivos, entre ellos se encuentran: la toma de decisiones por parte de los directivos antes de que el trabajo pueda continuar en la siguiente fase, y además, permiten controlar el progreso del trabajo a la dirección del proyecto y a los mismos desarrolladores. **[11] [12]**

1.4.1 Ventajas que aporta RUP: [11]

- Permite desarrollar aplicaciones sacando el máximo provecho de las nuevas tecnologías, mejorando la calidad, el rendimiento, la reutilización, la seguridad y el mantenimiento del software mediante una gestión sistemática de los riesgos.
- Permite la producción de software que cumpla con las necesidades de los usuarios, a través de la especificación de los requisitos, con una agenda y costo predecible.
- Permite llevar a cabo el proceso de desarrollo práctico, brindando amplias guías, plantillas y ejemplos para las actividades críticas.
- No solo garantiza que los proyectos abordados serán ejecutados íntegramente sino que además evita desviaciones importantes respecto a los plazos.
- Permite una definición acertada del sistema en un inicio para hacer innecesarias las reconstrucciones parciales posteriores.

RUP constituye uno de los estándares internacionales que más aceptación tiene. Varias herramientas CASE soportan dicha metodología, permitiendo generar código en distintos lenguajes de programación a partir de un diseño UML. RUP genera una amplia documentación necesaria para el desarrollo del software y muy útil para el caso en que se quiera estudiar el software e implementar nuevas versiones.

1.5 Herramienta CASE: Visual Paradigm.

Visual Paradigm es una de las herramientas CASE, fácil de usar, con soporte multiplataforma y que proporciona excelentes facilidades de interoperabilidad con otras aplicaciones. Permite graficar los diferentes diagramas de UML, revertir y generar código fuente para Java, C++, PHP, DotNet Exe/dll, XML, XML Schema, Python y Corba IDL.

Visual Paradigm ofrece soporte para Rational Rose, integración con Microsoft Visio. Está diseñada para usuarios interesados en sistemas de software de gran escala con el uso del acercamiento orientado a objeto. Una de sus principales ventajas es que incorpora el soporte para trabajo en equipo, permite que varios desarrolladores trabajen a la vez en el mismo diagrama y vean en tiempo real los cambios hechos por sus compañeros. **[13]**

1.5.1 Beneficios:

- Soporte para toda la notación UML.
- Capacidades de ingeniería directa (versión profesional) e inversa.
- Modelo y código que permanece sincronizado en todo el ciclo de desarrollo.
- Disponibilidad de integrarse en los principales IDEs.
- Disponibilidad en múltiples plataformas.

Se llegó a la conclusión de que Visual Paradigm es el más indicado para desarrollar el sistema que se desea implementar, además la Universidad de las Ciencias Informáticas cuenta con una licencia para su uso y funciona perfectamente sobre la plataforma GNU Linux.

1.6 Lenguaje de modelado: UML.

Para el desarrollo de la aplicación se utilizará al Unified Modeling Lenguaje (UML), como el lenguaje con que se modelarán los artefactos que se creen en el proceso de desarrollo del software. UML es un lenguaje de construcción de modelos para visualizar, especificar, construir y documentar los artefactos de un sistema que involucra una gran cantidad de software, con los que se construyen mayormente sistemas orientados a objetos.

Constituye una forma de modelar elementos conceptuales como los procesos de negocio y funciones de sistema, así como de cosas concretas entre las que se encuentran escribir clases en un lenguaje determinado, esquemas de base de datos y componentes de software reutilizables. Intenta solucionar el problema de propiedad de código que se da con los desarrolladores, al implementar un lenguaje de modelado y crea una documentación común, que cualquier desarrollador con conocimientos de UML será capaz de entender. Su utilización es independiente del lenguaje de programación y de las características de los proyectos, ya que UML ha sido diseñado para modelar cualquier tipo de proyecto.

1.7 Framework de Desarrollo: QT-Designer.

Qt es un framework de desarrollo multiplataforma con una biblioteca de más de 400 clases, las cuales encapsulan toda la infraestructura necesaria para desarrollar aplicaciones robustas. Posee una utilidad para la rápida construcción de interfaz gráfica de usuario con apariencia y aspectos nativos de las plataformas soportadas (QT-Designer). Facilita el flujo de trabajo de internacionalización mediante QT Linguist. Está separado en 13 módulos y la API que brinda incluye funciones para conectividad a bases de datos, programación en red, integración con gráficos 3D, desarrollo multi-hilos y lectura/escritura de ficheros XML.

Introduce una alternativa innovadora para comunicación entre objetos, las llamadas señales y ranuras (signals y slots), que reemplazan la antigua e insegura técnica callback. Provee un modelo de eventos convencional para manipular los eventos del mouse, teclas presionadas y demás entradas del usuario.

1.7.1 Características:

- **Características generales:** El diseño modular de la biblioteca Qt proporciona un rico conjunto de aplicaciones que ofrecen todas las funcionalidades necesarias para la construcción de potentes interfaces gráficas de usuario. Qt es una biblioteca fácil de aprender y usar. Permite crear código altamente legible y fácil de mantener **[14]**
- **Entorno de desarrollo:** Qt cuenta con entorno desarrollo integrado (IDE, por sus siglas en inglés) y permite el diseño de las ventanas a través de una GUI, entre otras funcionalidades. **[14]**
- **Multi-plataforma:** Qt está disponible para las plataformas Mac OSX, Windows, Linux/X11, Windows CE y S60. **[14]**
- **Lenguajes de programación:** Qt proporciona una intuitiva interfaz de clases para el desarrollo de aplicaciones en C++. Incluso va un poco más allá del lenguaje C++ en lo referido a la comunicación entre objetos y la flexibilidad para el desarrollo de interfaces gráficas de usuario agregando un poderoso mecanismo de comunicación entre objetos (signals/slots).**[14]**

1.8 Bibliotecas de desarrollo.

Desde el punto de vista de la Informática (computación), una biblioteca es un conjunto de procedimientos y funciones (subprogramas) agrupadas en un archivo con el fin de que puedan ser aprovechadas por otros programas. Existen dos tipos de bibliotecas, estáticas o de enlace y las compartidas o de enlace dinámico. De estas, la primera hace enlaces entre ellas, o sea, arregla las referencias a rutinas en el programa para que apunten a su localización en la biblioteca al momento de la compilación, mientras que el segundo tipo de bibliotecas se enlaza en tiempo de ejecución.

La denominación de biblioteca compartida hace énfasis en que, comúnmente, los procesos que la enlazan comparten una única parte de la memoria donde se encuentran las instrucciones de los subprogramas. En el mundo de la Realidad Aumentada podemos encontrar muchas de estas bibliotecas, unas más avanzadas que otras, algunas de las cuales son más fáciles de usar. Dentro de estas bibliotecas vinculadas a la Realidad Aumentada, se puede distinguir que la mayoría brindan al usuario ficheros tanto

de enlace dinámico, como de enlace estático, con el objetivo de hacer más eficiente el proceso de compilación o ejecución según sea el caso.

Cada una de estas bibliotecas destinadas al desarrollo de sistemas de Realidad Aumentada, se diferencian, ya sea por el tipo de licencia o la plataforma donde es utilizada. Las mismas presentan características especiales como son las diferentes áreas dentro de la Realidad Aumentada sobre la que actúan, los diferentes formatos de fichero con información gráfica que son capaces de leer, entre otras.

A continuación se exponen las características esenciales de biblioteca OsgART que es la que se va a utilizar el de desarrollo de nuestro sistema.

1.8.1 OsgART.

OsgART es una biblioteca de C++ que ofrece la posibilidad de crear aplicaciones de Realidad Aumentada mediante la combinación de ARToolKit y OpenSceneGraph. Dispone de tres funciones principales:

- Alto nivel de integración de entrada de video (video objeto, shaders).
- Soporte de múltiples marcadores basados en características físicas o tecnologías de seguimiento (marcador basado en múltiples rastreadores)
- Registro fotométrico, basado en técnicas para el renderizado de sombras y oclusión.

Características.

- Soporta el renderizado de sombras.
- Soporta varias entradas de video. **[15]**

1.9 Motor gráfico: Open Scene Graph.

OpenSceneGraph (OSG) es un motor gráfico de código abierto, utilizado para el desarrollo de aplicaciones gráficas de alto rendimiento tales como, simuladores de vuelo, juegos, aplicaciones de realidad virtual y visualización científica. Esta herramienta está basada en el concepto de grafos de

escena, provee una plataforma orientada a objetos que utiliza OpenGL como API gráfica. OSG está disponible tanto para uso comercial como no comercial.

1.9.1 Características:

- **Rendimiento:** Soporta view-frustum culling, occlusion culling, small feature culling, LOD, vertex arrays, vertex buffer objects, OpenGL Shader Language y display lists. Todo esto hace de OSG uno de los motores gráficos de más alto rendimiento disponible. **[16]**
- **Productividad:** El núcleo de OSG encapsula la mayoría de las funcionalidades de OpenGL incluyendo las últimas extensiones de este, provee optimización de la visualización, y un conjunto de bibliotecas adicionales las cuales hacen posible desarrollar aplicaciones gráficas de alto rendimiento muy rápidamente. Combinando las experiencias de otros motores gráficos con modernos métodos de ingeniería de software como patrones de diseño, se ha logrado el diseño de una biblioteca robusta y extensible. **[16]**
- **Extensiones Soportadas:** Para la lectura y escritura de archivos la biblioteca (osgDB) proporciona una amplia variedad de formatos por medio de un mecanismo extensible de plugins dinámicos. Actualmente incluye cincuenta y cinco plugins para cargar formatos 3D y formatos de imágenes. **[16]**
- **Portabilidad:** El núcleo de OSG está diseñado para tener una dependencia mínima de cualquier plataforma. Esto ha permitido que OSG fuese rápidamente portado a un amplio rango de plataformas. Originalmente fue desarrollado en IRIX, luego portado a Linux, Windows, FreeBSD, Mac OSX, Solaris, HP-UX, AIX y hasta PlayStation2. **[17]**
- **Soporte multi-lenguaje:** OSG está disponible además en los lenguajes Java, Lua y Pyhton. **[17]**

1.10 Formato para el almacenamiento de datos: XML.

XML (eXtensible Markup Language o lenguaje de marcas extensible) es un metalenguaje extensible desarrollado por el World Wide Web Consortium (W3C). XML es un formato basado en texto, fue diseñado específicamente para almacenar y transmitir datos. Su principal novedad consiste en compartir los datos

con los que trabajan todos los niveles, por todas las aplicaciones y soportes. Permitiendo compartir la información de una manera segura, fiable y fácil.

1.10.1 Principales objetivos: [18]

- Los documentos XML deben ser legibles y claros.
- El diseño de XML debe ser preparado rápidamente.
- El diseño de XML debe ser formal y conciso.
- Los documentos XML deben ser de fácil creación.
- La concisión en las marcas XML es de mínima importancia.

1.10.2 Características:

Entre las características de XML que han logrado que se convierta en un formato universal, se encuentran las siguientes**: [18]**

- Es una arquitectura más abierta y extensible. No se necesita versiones para que pueda funcionar en futuros navegadores.
- Datos compuestos de múltiples aplicaciones. La extensibilidad y flexibilidad de este lenguaje permite agrupar una amplia variedad de aplicaciones, desde páginas web hasta bases de datos.
- Gestión y manipulación de los datos desde el propio cliente Web.
- Se permitirá un comportamiento más estable de las aplicaciones Web.
- Puede exportar a otros formatos de publicación. El documento maestro de la edición electrónica podría ser un documento XML que se integraría en el formato deseado de manera directa.

1.10.3 Ventajas:

Entre las principales ventajas que ofrece XML se pueden citar las siguientes**: [18]**

- Estandarización de la información.

- Integración de aplicaciones.
- Portabilidad de información.
- Mejora el acceso a la información.

1.11 Lenguaje de programación: C++.

El C++ es un lenguaje de programación, diseñado a mediados de los años 1980, por Bjarne Stroustrup, como extensión del lenguaje de programación C.

Se puede decir que C++ es un lenguaje que abarca tres paradigmas de la programación: la programación estructurada, la programación genérica y la programación orientada a objetos. Actualmente existe un estándar, denominado ISO C++, al que se han adherido la mayoría de los fabricantes de compiladores más modernos.

1.11.1 Características.

- **Programación orientada a objetos:** La posibilidad de orientar la programación a objetos permite al programador diseñar aplicaciones desde un punto de vista más cercano a la vida real. Permite la reutilización del código de una manera más lógica y productiva para el uso de plantillas.
- **Portabilidad:** Un código escrito en ANSI C++ puede ser compilado en casi todo tipo de ordenadores y sistemas operativos sin hacer apenas cambios, siempre que exista el compilador.
- **Brevedad:** El código escrito en C++ es muy corto en comparación con otros lenguajes, sobretodo porque en este lenguaje es preferible el uso de caracteres especiales que las "palabras clave".
- **Programación modular:** Un cuerpo de aplicación en C++ puede estar hecho con varios ficheros de código fuente que son compilados por separado y después unidos.Esta característica permite unir código en C++ con código en otros lenguajes de programación como Ensamblador o el propio C.
- **Velocidad:** El código resultante de una compilación en C++ es muy eficiente, gracias a su capacidad de actuar como lenguaje de alto y bajo nivel.

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

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

C++ es uno de los lenguajes más potentes, debido a que permite trabajar tanto a alto como a bajo nivel, sin embargo es a su vez uno de los que menos automatismos trae (obliga a hacerlo casi todo manualmente al igual que C) lo que "dificulta" mucho su aprendizaje.

Conclusiones parciales del capítulo.

En este capítulo se refleja el estudio realizado por las autoras, con el objetivo de identificar las herramientas de autor destinadas al desarrollo rápido de escenas de RA. La búsqueda bibliográfica realizada permitió conocer los principales conceptos relacionados con las herramientas de autor, así como la clasificación de este tipo de herramientas relacionadas con el campo de la RA en dos grupos: orientadas a programadores y orientadas a no programadores. Al identificar las características más prominentes de este tipo de herramientas, se pudo constatar que muchas son propietarias y que una por sí sola no reúne todas las funcionalidades deseables en la herramienta que se propone desarrollar como resultado de la presente investigación.

La afirmación anterior se apoya en que cada una de estas herramientas en particular cuenta con funcionalidades que la diferencian de las demás, que en muchas ocasiones son complementarias y que son de gran utilidad si se encuentran todas reunidas en una sola solución. Se identificó la metodología de desarrollo, marco de trabajo, el lenguaje de programación, las bibliotecas y el motor gráfico a utilizar para el desarrollo de la herramienta, brindando las definiciones necesarias y precisas para comprender el porqué del uso de las mismas. Todo esto encaminado a obtener los elementos necesarios para dar solución al objetivo de esta investigación.

CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA

Introducción.

En el presente capítulo se obtiene una visión más general de la herramienta a desarrollar. Se detallarán los distintos pasos de la metodología propuesta en el capítulo anterior (RUP) para el desarrollo de la herramienta. Esto incluirá la elaboración del modelo del dominio, la captura de los requerimientos tanto funcionales como no funcionales de la herramienta; así como la definición de Actores y la relación entre ellos, los Diagramas de Caso de Uso y las descripciones textuales de los mismos.

2.1 Solución Propuesta.

Como solución al objetivo propuesto en la presente investigación se propone desarrollar una herramienta de autor para creación de contenidos de Realidad Aumentada, que funcionará como una herramienta de diseño independiente para la creación de escenas de Realidad Aumentada. La cual será capaz de reducir el tiempo en que estas escenas son editadas, así como el margen de errores cometidos típicos del proceso de edición manual al ser este reemplazado por un proceso de edición semiautomático.

Para el desarrollo de la herramienta se utilizó la biblioteca Qt y el motor gráfico osgART para el manejo de todos los procesos que se llevan a cabo para la representación visual de la escena y los recursos dentro de ella. Después de realizar un estudio de la metodología RUP, se decidió utilzarla debido a que es una de las metodologías ágiles más usadas actualmente en el mundo, es un proceso de ingeniería de software para producir software de calidad, flexible y en plazos. Como lenguaje de modelado se utiliza el UML y la herramienta para representar los artefactos es el Visual Paradigm por ser de libre distribución, código abierto y brindar facilidades en materia de desarrollo colaborativo.

Hasta aquí se han brindado elementos generales del presente trabajo, ahora se considera necesario abordar los aspectos técnicos de la misma.

2.2 Modelo de Persistencia.

2.2.1 Formato del fichero para almacenar la configuración de la escena.

Luego de realizar la configuración de la escena, la información persistente que contiene la misma necesita ser guardada en un fichero, para ello se hace uso del formato XML. La principal razón por la que se determinó usar XML para salvar la configuración de la escena es la extensibilidad de XML, o sea, uno puede adicionar nuevas etiquetas sin que esto afecte a las aplicaciones que utilizaban la versión que no contaba con las nuevas etiquetas, o sea, que si en versiones posteriores al fichero que guarda la configuración de la escena se le agregan otras etiquetas para almacenar otros datos de la configuración, las aplicaciones que cuenten con una versión anterior a la de los cambios podrán hacer uso de ese fichero, aunque como es lógico no usarán la nueva información.

Estructura del fichero.

La principal etiqueta del fichero se denominará ***Scene***, esta tendrá anidada toda la información referida a la configuración de la escena, dígase, el nombre de la escena y las asociaciones recurso-marcador que se encuentran dentro de la escena, dentro de la etiqueta Asociación se almacena la dirección del modelo y del marcador, así como el tamaño del marcador y la posición que ocupa el mismo dentro de la escena.

```
<?xml version="1.0" ?>
<Scene>
    <Association name_id="AsocID0" activa = "1"> <!--1 V, 0 F-->
        <Marker single="1"> <!--1 V, 0 F-->
            <FileDir MarkerDir = "AsocID0" />
            <Size value="0" />
            <Center x="0" y="0" />
        </Marker>
        <ModelRV fileDir = "">
    </Association>
    <!-- Aquí se comienza a iterar por las asociaciones -->
</Scene>
```

Figura 5. Estructura del fichero XML de la escena.

```
<?xml version="1.0" ?>
<Scene>
    <Asociation name_id="OtraBike" activa="1">
        <Marker single="1">
            <FileDir MarkerDir="patt.kinetic16" />
            <Size value="160" />
            <Center x="0" y="0" />
        </Marker>
        <ModelRV fileDir="C:/bike01.3DS" />
    </Asociation>
    <Asociation name_id="SpaceStation" activa="1">
        <Marker single="1">
            <FileDir MarkerDir="patt.MB16" />
            <Size value="160" />
            <Center x="0" y="0" />
        </Marker>
        <ModelRV fileDir="C:space-station.3DS" />
    </Asociation>
</Scene>
```

Figura 6. Ejemplo concreto del fichero para el almacenamiento de datos de la escena.

2.3 Reglas del Negocio.

- Una asociación solo se podrá realizar si la cantidad de patrones y modelos 3D es mayor que cero.
- Los formatos de modelos 3D que se cargarán serán del tipo .3ds . osg y .obj.
- La información generada en la aplicación se guardará en archivos de tipo .xml.
- Los formatos de imágenes que se cargarán serán del tipo .jpg .png .
- La interacción con la escena se hará por medio del mouse y teclado.
- Se usará solo una cámara.
- La cámara se conectará a la computadora a través de puerto USB.

2.4 Modelo Dominio.

Existen por lo menos dos aproximaciones para documentar un sistema de manera que para los desarrolladores de software sea de gran utilidad: modelo del dominio y modelo del negocio. El modelo del dominio describe los conceptos más importantes del contexto como objetos del dominio, y enlaza estos objetos unos con otros. La identificación y asignación de un nombre para estos objetos ayuda a desarrollar un glosario de términos que permitirán comunicarse mejor con los que trabajen en el sistema.

A continuación se representa el modelo del dominio referente a la herramienta propuesta:

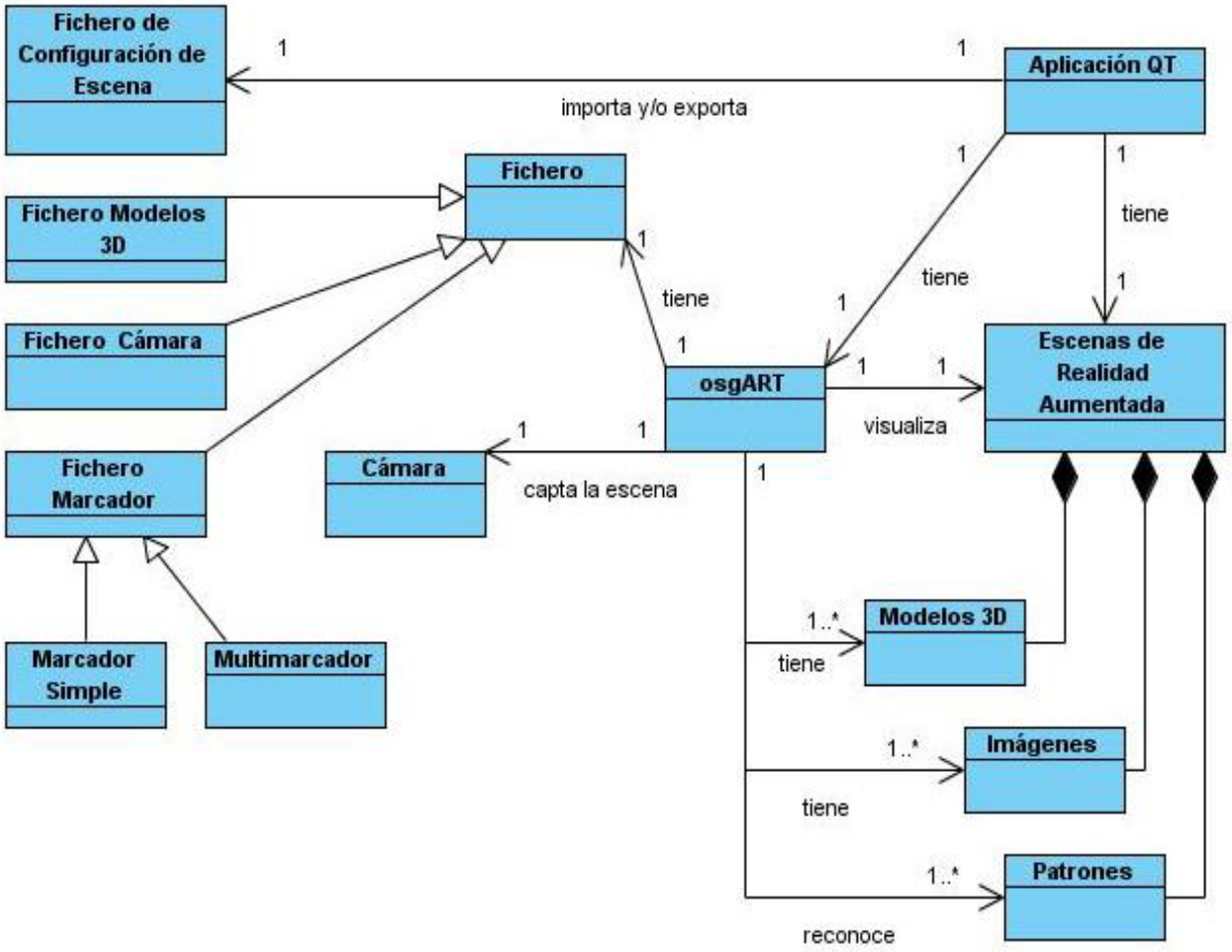

Figura 7. Modelo de Dominio.

2.4.1 Definición de las clases del modelo de dominio.

- **Fichero de Configuración de Escena:** Fichero que contiene la información referente a la escena 3D.
- **Fichero Modelos 3D:** Fichero que contiene la información referente a un modelo 3D.
- **Fichero de la Cámara:** Fichero que contiene la información referente a la escena capturada por la cámara.
- **Fichero Marcador:** Fichero que contiene la información referente al marcador.
- **Marcador Simple:** Son las plantillas de orientación que utiliza la biblioteca ARToolkit. Tienen una forma cuadrada con un marco de color negro que contiene un cuadrado blanco en su interior y un pequeño dibujo negro dentro de este. Esta plantilla es lo que se reconoce en las aplicaciones de Realidad Aumentada como patrones de marca, las cuales contienen un identificador, tamaño y las coordenadas del centro, todos estos atributos sirven para su correcta identificación.
- **Multimarcador:** Conjunto de marcadores relacionados entre sí.
- **OsgART:** Motor gráfico para la visualización de las escenas de Realidad Aumentada.
- **Cámara:** Dispositivo de captura de video, genera un video con la escena capturada. Se utiliza para capturar la imagen del entorno real que es enviada a la computadora, a partir de ese momento la aplicación comienza la búsqueda de patrones en el marco de video capturado.
- **Aplicación QT:** Interfaz gráfica de usuario basada en Qt que posibilita que se ejecuten las distintas funcionalidades de la aplicación de una forma sencilla e intuitiva.
- **Escenas de Realidad Aumentada:** Escena del mundo real enriquecida con elementos virtuales.
- **Modelos 3D:** Modelos virtuales de objetos reales en 3D.
- **Imagen:** es cada fotograma del video.
- **Patrones:** Mapa de bits generado a partir de la región que ocupa un marcador determinado en cada fotograma del video capturado por la cámara.

2.5 Captura de requisitos.

El flujo de trabajo Requerimientos perteneciente al Proceso Unificado de Desarrollo (RUP), tiene como propósito principal guiar el desarrollo hacia el sistema correcto donde se describan con claridad y sin ambigüedades el comportamiento del mismo. Así como lograr una comunicación efectiva entre el cliente, los usuarios y los desarrolladores sobre qué debe y qué no debe hacer el sistema.

La captura de requisitos es una de las principales tareas en el ciclo de desarrollo de un sistema. Los requerimientos se pueden clasificar en: funcionales y no funcionales.

2.5.1 Requerimientos Funcionales.

Los requerimientos funcionales son capacidades o condiciones que el sistema debe cumplir. Su meta principal es identificar y documentar las acciones que debe ejecutar la herramienta para que cumpla con el objetivo de la investigación. Todas estas acciones se convierten en requisitos funcionales y de acuerdo con el objetivo propuesto el sistema debe ser capaz de:

RF 1. Importar la configuración de la escena.

RF 2. Reconocer más de un marcador al mismo tiempo en una escena.

RF 3. Visualizar varios elementos virtuales.

RF 4. Exportar la configuración de la escena.

RF 5. Asociar recursos a un marcador.

RF 5.1 Asociar modelos 3D a un marcador.

RF 6. Transformar Recursos.

RF 6.1 Transformación de rotación.

RF 6.2 Transformación de escala.

RF 6.3 Transformación de traslación.

RF 7. Permitir la entrada a través del ratón y teclado.

RF 8. Inicializar escena de Realidad Aumentada.

RF 8.1 Inicializar Video.

RF 8.2 Inicializar Tracking.

RF 8.3 Inicializar Escena.

2.5.2 Requerimientos no Funcionales.

Los requerimientos no funcionales son propiedades o cualidades que el producto debe tener. Debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable, rápido y confiable.

Requisitos de apariencia.

- El sistema debe contener una interfaz sencilla e intuitiva y que sea agradable al usuario.
- El usuario debe tener acceso por varias vías a una determinada funcionalidad.
- La aplicación debe utilizar como idioma principal el español, excepto aquellas palabras técnicas que no puedan ser traducidas.
- Utilizar botones que expresen su función, ya sea a través del ícono o el texto que lo identifica.
- La aplicación debe verse en una pantalla con una resolución de 1024 x 768 o superior.

Requisitos de usabilidad.

- La aplicación deberá poseer una interfaz funcional, tanto para usuarios expertos, como para los que no tienen conocimientos profundos de Informática.

Requisitos de software.

- Las computadoras que utilizarán el software deben tener instalado: Windows XP Professional Service Pack 2 o Superior.
- Se debe tener instalado el Driver de la cámara.

Requisitos de hardware.

- Las computadoras que utilizarán el software a desarrollar deberán tener 512 MB de Memoria RAM como mínimo (1 GB de Memoria RAM recomendado).
- Contar con una Tarjeta de Video (ATI o NVidia) de al menos 256 MB de memoria.

Requisitos de diseño e implementación.

- El contenido se cargará desde archivos XML.
- Se hará uso del lenguaje de programación ANSI C++.

Requisitos de rendimiento.

- La herramienta propuesta debe ejecutarse con una velocidad de refrescamiento de 30 fps (Frame Per Second).
- Haga un correcto uso de la memoria.

2.6 Modelación del Sistema.

A continuación se identifica el actor del sistema, y partir de los requisitos funcionales obtenidos anteriormente se conciben los casos de uso del sistema y sus respectivas descripciones.

2.6.1 Actor del Sistema.

Los actores del sistema definen el comportamiento y responsabilidades (rol) de un individuo, grupo de individuos, sistema automatizado o máquina, que interactúan con el mismo intercambiando información con este.

A continuación se detalla el actor del sistema:

Actor.	Descripción.
Usuario.	Es la persona que usará la herramienta de autor para configurar escenas de Realidad Aumentada, puede ser un desarrollador, un diseñador o bien una persona que no tenga ningún tipo de conocimientos sobre herramientas de Realidad Aumentada.

Tabla 4. Actor del Sistema.

2.6.2 Diagrama de Caso de Uso del Sistema.

Los Casos de Uso del sistema (CUS) son un conjunto de secuencia de acciones que un sistema ejecuta y que produce un resultado observable para un actor, es decir, son “fragmentos” de funcionalidad que el sistema ofrece al actor que interactúa con el mismo, reportándole algún que otro beneficio.

A continuación se muestra la **figura 8** correspondiente al diagrama de casos de uso del sistema:

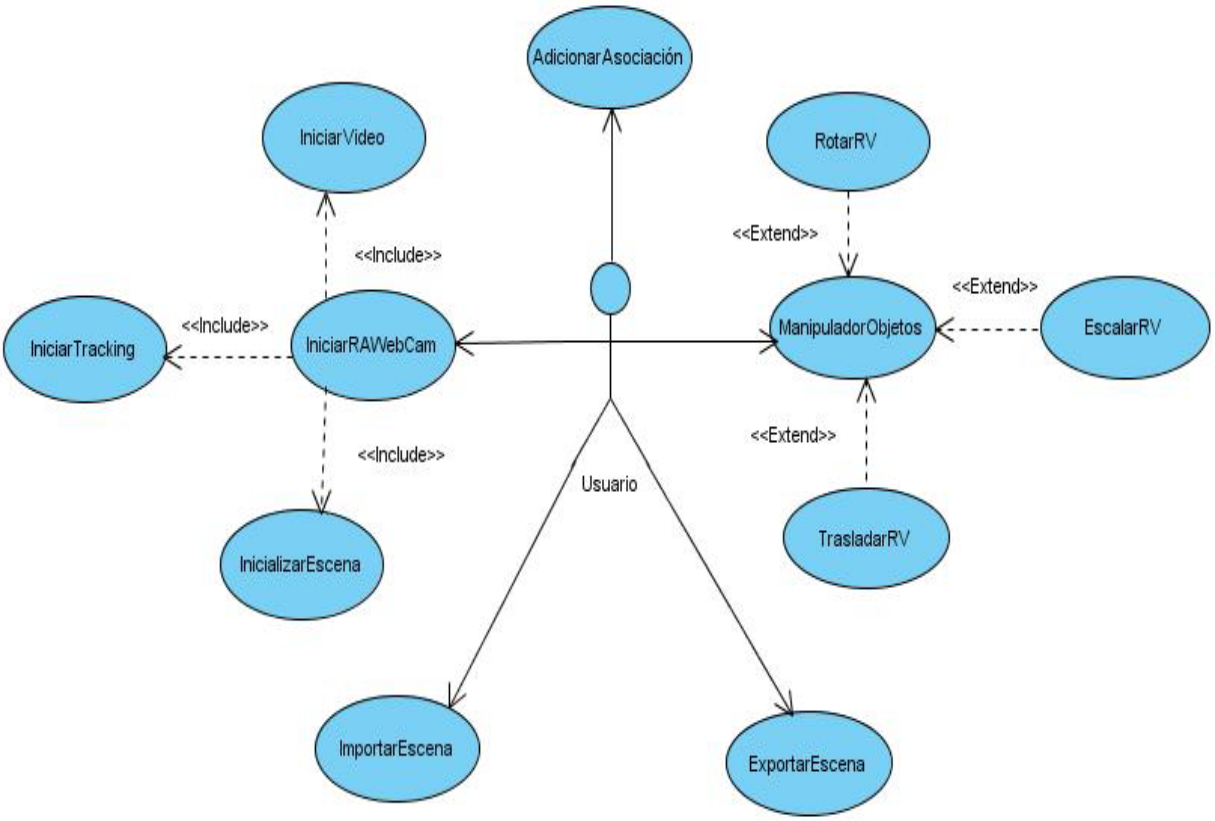

Figura 8. Diagrama de Caso de Uso del Sistema.

2.6.3 Descripción de los Casos de Uso del Sistema.

La descripción de los casos de uso del sistema tienen como fin entender la funcionalidad asociada a cada caso de uso, detallando las acciones que tienen lugar durante la interacción actor-sistema, es decir, describe el flujo de actividades que realiza el actor al hacer uso del sistema y las correspondientes respuestas del mismo.

Caso de Uso	**Importar la configuración de la escena.**
Actores	Usuario.
Resumen	El caso de uso inicia cuando el actor escoge la opción Importar Escena, especificando la escena que desea importar, finalizando así el caso de uso.
Precondiciones	Para poder importar una escena, es necesario que esta exista previamente.
Referencias	**RF 1, 2, 3.**
Prioridad	**Crítico.**
Poscondiciones	Se importa y visualiza la escena.

Figura 9. Caso de Uso “Importar Escena”.

Caso de Uso	**Exportar la configuración de la escena.**
Actores	Usuario.
Resumen	El caso de uso inicia cuando el actor escoge la opción Exportar escena, especificando la extensión que tendrá el fichero y la dirección donde será guardada, finalizando así el caso de uso.
Precondiciones	
Referencias	**RF 4.**
Prioridad	**Crítico.**
Poscondiciones	Se salva la escena

Figura 10. Caso de Uso “Exportar Escena”.

Caso de Uso	**Adicionar asociación.**
Actores	Usuario.
Resumen	El caso de uso inicia cuando el usuario escoge la opción Nueva en el menú Asociación, el sistema le muestra un formulario donde le brinda la posibilidad de crear una nueva relación de patrón y recurso virtual, al usuario realizar la selección el sistema lleva a cabo la asociación y se añade a la escena, finalizando así el caso de uso.
Precondiciones	La escena tiene que estar inicializada.
Referencias	
Prioridad	**Crítico.**
Poscondiciones	

Figura 11. Caso de Uso “Adicionar Asociación”.

Caso de Uso	**Manipular Objetos.**
Actores	Usuario.
Resumen	El caso de uso inicia cuando el usuario accede al sistema seleccionando la opción Transformar Recurso, aquí se le da la posibilidad al usuario de escoger que tipo de transformación desea realizar, ya sea rotar, trasladar o escalar, luego el sistema responde a cada una de estas transformaciones finalizando así el caso de uso.
Precondiciones	Debe de estar seleccionado un recurso.
Referencias	**RF 6, 7.**
Prioridad	**Crítico.**
Poscondiciones	Es trasformado el recurso.

Figura 12. Caso de Uso “Manipular Objetos”.

Caso de Uso	**Inicializar escena de realidad aumentada.**
Actores	Usuario
Resumen	El caso de uso inicia cuando el usuario selecciona la opción Iniciar Escena RA a partir de un dispositivo de captura. Para la realización completa del mismo primeramente se inicializa el vídeo, luego se inicializa el tracking, y por último se inicializa la Escena.
Precondiciones:	Debe haberse inicializado el vídeo, el tracking y la escena.
Referencias	**RF 8. Inicilizar Vídeo, Iniciar Tracking, Inicializar Escena**
Prioridad	**Crítico.**
Poscondiciones	Se muestra una ventana de interfaz de usuario que visualiza el contenido del grafo de la escena que incluye la reproducción del vídeo.

Figura 13. Caso de Uso “Inicializar escena de Realidad Aumentada”.

Caso de Uso	**Inicializar Vídeo<Inclusión>**
Actores	Usuario
Resumen	El caso de uso comienza como un proceso interno de la aplicación cuando el usuario inicializa la escena de realidad aumentada, aquí se le especifica que la captura va a ser desde una cámara y que biblioteca debe utilizar para acceder al hardware de captura de vídeo mediante la cual se obtiene la configuración de vídeo, es decir la velocidad de captura, la resolución del vídeo, espacio de color y compresión.
Precondiciones:	Debe existir una cámara conectada a la computadora. Deben existir los archivos de la biblioteca de captura de vídeo.
Referencias	
Prioridad	**Crítico.**
Poscondiciones	Se abre el flujo de vídeo y queda listo para acceder a sus datos de vídeo.

Figura 14. Caso de Uso “Inicializar Video”.

Caso de Uso	**Iniciar Tracking<Inclusión>**
Actores	Usuario
Resumen	El caso de uso inicia al cargar la biblioteca de análisis de imágenes encargada de detectar los patrones que hay distribuidos en la imagen capturada de una escena con el fin de devolver por cada uno su información de transformación. Además establece los parámetros intrínsecos de la calibración de la cámara mediante la lectura de un archivo y se le establece la fuente de vídeo de la que se va a alimentar.
Precondiciones:	Deben existir los archivos de la biblioteca de reconocimiento de patrones(tracking). Debe existir el archivo de calibración de la cámara que especifica los parámetros intrínsecos de la cámara. Debe estar activo el flujo de vídeo del cual se va a alimentar.
Referencias	
Prioridad	**Crítico.**
Poscondiciones	El módulo de reconocimiento queda activo y listo para realizar el análisis de los fotogramas del vídeo en busca de los patrones y sus transformaciones.

Figura 15. Caso de Uso “Iniciar Tracking”.

Caso de Uso	**Inicializar Escena<Inclusión>**
Actores	Usuario
Resumen	El caso de uso inicia acondicionando una escena básica de realidad aumentada para comenzar a reproducir el vídeo capturado desde la cámara.
Precondiciones:	El flujo de vídeo debe estar abierto. La biblioteca de reconocimiento de patrones (tracking) debe estar activa.
Referencias	
Prioridad	**Crítico.**
Poscondiciones	Se establece un grafo de escena que incluye las texturas del vídeo de fondo que se va a reproducir, una cámara virtual con los mismo parámetros de la cámara real y se comienza a reproducir el vídeo.

Figura 16. Caso de Uso “Inicializar escena”.

Conclusiones parciales del capítulo.

En este capítulo se mostraron las principales clases del dominio para una mayor comprensión de los conceptos relacionados con la herramienta, se seleccionó el actor que interviene; se analizaron las características y funcionalidades fundamentales que se deben incluir en la solución. Se agruparon los requisitos funcionales en casos de uso y estos últimos se describieron para un mejor entendimiento de los procesos que tienen lugar en la herramienta desarrollada. Una vez realizado esto es posible comenzar a realizar el diseño e implementación de la aplicación.

CAPÍTULO 3: DISEÑO E IMPLEMENTACIÓN

Introducción.

En este capítulo se presentan los diagramas de clases del diseño, constituyendo estos unos de los pasos más importantes dentro del desarrollo del sistema, ya que es en esta fase donde se crea una entrada apropiada y un punto de partida para actividades de implementación, capturando los requisitos o subsistemas individuales, interfaces y clases. Se muestran los diagramas de secuencia de la realización de los casos de uso y los componentes físicos por los que estará compuesto el sistema (ficheros .h y .cpp correspondientes a la implementación en C++), y se elabora el diagrama de despliegue para el sistema.

3.1 Diseño.

El diseño tiene el propósito adquirir una comprensión de los aspectos relacionados con los requisitos no funcionales y restricciones relacionadas con los lenguajes de programación, componentes reutilizables, sistemas operativos, tecnologías de distribución y concurrencia y tecnologías de interfaz de usuario. El diseño debe ser suficiente para que el sistema pueda ser implementado sin ambigüedades.

3.1.1 Diseño Arquitectónico.

La arquitectura de software es la estructura general de un sistema. Esta comprende los componentes de software, las propiedades de esos componentes visibles externamente, y las relaciones de ellos.

Importancia de la arquitectura de software:

- Las representaciones de la arquitectura de software facilitan la comunicación entre todas las partes interesadas en el desarrollo de un sistema basado en computadoras.
- La arquitectura destaca decisiones tempranas de diseño que tendrán un profundo impacto en todo el trabajo de ingeniería de software que sigue, y es tan importante en el resultado final del sistema como una entidad operacional.

- Constituye un modelo pequeño e intelectualmente comprensible de cómo está estructurado el sistema y de cómo trabajan juntos sus componentes. **[19]**

Estilo arquitectónico Modelo-Vista-Controlador.

El estilo arquitectónico que se decidió escoger para el diseño de la herramienta autor fue el del Modelo-Vista- Controlador (MVC); el mismo divide una aplicación interactiva en tres áreas: procesamiento, salida y entrada. Para esto, utiliza las siguientes abstracciones:

- **Modelo :** Encapsula los datos y las funcionalidades. El modelo es independiente de cualquier representación de salida y/o comportamiento de entrada.
- **Vista :** Muestra la información del usuario. Pueden existir múltiples vistas del modelo.
- **Controlador :** Recibe las entradas, usualmente como eventos que codifican los movimientos o pulsaciones de botones del mouse, pulsaciones de teclas, etc. Los eventos son traducidos a solicitudes de servicios (“services requests”) para el modelo o la vista.

A lo largo de los años desde la presentación de este patrón a la comunidad científica, se han desarrollado diferentes variantes fundamentales, que se presentan brevemente a continuación.

Variante I: Variante en la cual no existe ninguna comunicación entre el Modelo y la Vista, y esta última recibe los datos a mostrar a través del Controlador.

Variante II: Variante en la cual se desarrolla una comunicación entre el Modelo y la Vista, donde esta última al mostrar los datos los busca directamente en el Modelo, dada una indicación del Controlador, disminuyendo el conjunto de responsabilidades de este.

Variante III: Variante en la cual se diversifican las funcionalidades del Modelo teniendo en cuenta las características de las aplicaciones multimedia, donde tienen un gran peso los medios utilizados en estas.

De las tres variantes expuestas se decidió utilizar la Variante I con el objetivo de lograr una independencia entre la capa de vista y la capa de modelo (representación de los datos), y de lograr un bajo acoplamiento.

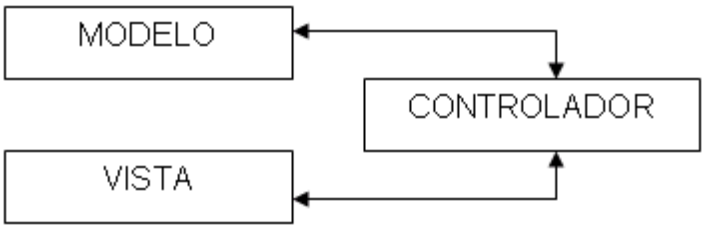

Figura 17. Variante inicial del Patrón MVC.

3.1.2 Patrones de Diseño.

De manera general los patrones constituyen soluciones simples y elegantes a problemas específicos y comunes del diseño orientado a objetos, basados en la experiencia y que se ha demostrado que funcionan. **[20]**

Los patrones son parejas de problema/solución con un nombre, que codifican nuevos principios y sugerencias relacionados frecuentemente con la asignación de responsabilidades. Asignar correctamente las responsabilidades es muy importante en el diseño orientado a objetos. **[20]**

1. **Patrón Singlenton.**

El patrón Singleton se pone de manifiesto en nuestro sistema debido a que es necesario asegurar que exista una única instancia de la clase ***OSGART Manager*** y proveer un punto global para acceder a esta. Una variable global hace un objeto accesible, pero no limita la instanciación de múltiples objetos de un tipo. Una mejor solución es hacer que la clase misma sea responsable de garantizar su única instancia, interceptando solicitudes para crear nuevos objetos y facilitando además una vía para acceder a la instancia de forma compartida.

Aplicabilidad [21]:

- Deba a ver exactamente una instancia de una clase y ésta deba ser accesible a los clientes desde un punto de acceso conocido.

- La única instancia debería ser extensible mediante herencia y los clientes deberían ser capaces de utilizar una instancia extendida sin modificar su código.

Consecuencias [21]:

- Acceso controlado a la única instancia. Puede tener un control estricto sobre cómo y cuándo acceden los clientes a la instancia.
- Espacio de nombres reducido. El patrón Singleton es una mejora sobre las variables globales. Permite el refinamiento de operaciones y la representación.
- Se puede crear una subclase de Singleton. Permite un número variable de instancias.
- El patrón hace que sea fácil cambiar de opinión y permitir más de una instancia de la clase Singleton. Más flexible que las operaciones de clase.

2. Patrón Adapter.

Definición: Convertir la interfaz de una clase en otra interfaz de cliente esperada.

Aplicabilidad:

- Cuando se quiera utilizar una clase ya existente y su interfaz no se corresponda con la interfaz que se necesita.
- Se quiera definir una clase que se pueda utilizar para que pueda colaborar con otras clases cuyas interfaces no se conocen inicialmente.
- Para diseñar un visualizador genérico de estructuras compuestas. **[22]**

Ventajas e inconvenientes:

- Una misma clase adaptadora puede adaptar una clase adaptada y a todas sus subclases.
- La adaptación no siempre consiste en cambiar el nombre y/o los parámetros de las operaciones, puede ser más compleja. **[22]**

El mismo se evidencia en las clases ***ViewerQT*** y ***Adapter Widget*** ya que se realiza una adaptación de la interfaz del visor de OSG a la interfaz de ***QGL Widget***.

3. **Patrón Builder.**

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

La intención de este patrón de diseño es separar la construcción de un objeto complejo de su representación de modo, que el mismo proceso de construcción pueda crear diferentes representaciones. Los objetos que dependen de un algoritmo tendrán que cambiar cuando el algoritmo cambia; por lo tanto, los algoritmos que estén expuestos a dicho cambio deberían ser separados, permitiendo de esta manera reutilizar algoritmos para crear diferentes representaciones.

Aplicabilidad:

- El algoritmo para creación de un objeto complejo debe ser independiente de las partes que conforman el objeto y cómo están ensambladas.
- El proceso de construcción debe permitir diferentes representaciones del objeto que se construye.

Este patrón se pone de manifiesto cuando la clase ***SAsociación*** construye el fichero XML con los datos de la escena que fueron recopilados por la clase ***OSGART_SceneMgr***.

4. **Patrón State (Estado).**

Definición: Permite que un objeto modifique su comportamiento cada vez que cambie su estado interno.

Aplicabilidad:

- Esta apuntado a cuando un determinado objeto tiene diferentes estados y también distintas responsabilidades según el estado en que se encuentre en determinado instante.

- También puede utilizarse para simplificar casos en los que se tiene un complicado y extenso código de decisión que depende del estado del objeto

Ventajas:

- Se localizan fácilmente las responsabilidades de los estados específicos, dado que se encuentran en las clases que corresponden a cada estado. Esto brinda una mayor claridad en el desarrollo y el mantenimiento posterior. Esta facilidad la brinda el hecho que los diferentes estados están representados por un único atributo (state) y no envueltos en diferentes variables y grandes condicionales.
- Hace los cambios de estado explícitos puesto que en otros tipos de implementación los estados se cambian modificando valores en variables, mientras que aquí al estar representado cada estado.
- Los objetos State pueden ser compartidos si no contienen variables de instancia, esto se puede lograr si el estado que representan está enteramente codificado en su tipo. Cuando se hace esto estos estados son Flyweights sin estado intrínseco.
- Facilita la ampliación de estados.
- Permite a un objeto cambiar de clase en tiempo de ejecución dado que al cambiar sus responsabilidades por las de otro objeto de otra clase la herencia y responsabilidades del primero han cambiado por las del segundo.

Desventaja:

- Se incrementa el número de subclases.

Este patrón está presente en la clase Estado, ya que la misma va a cambiar su comportamiento teniendo en cuenta el estado interno en que se encuentre esta, que puede ser estado de rotación, estado de traslación o estado de escalado.

3.1.3 Diagrama de Clases de Diseño.

En el diagrama de diseño que se muestra en la figura se representan las clases de diseño y las relaciones existentes entre cada una de ellas.

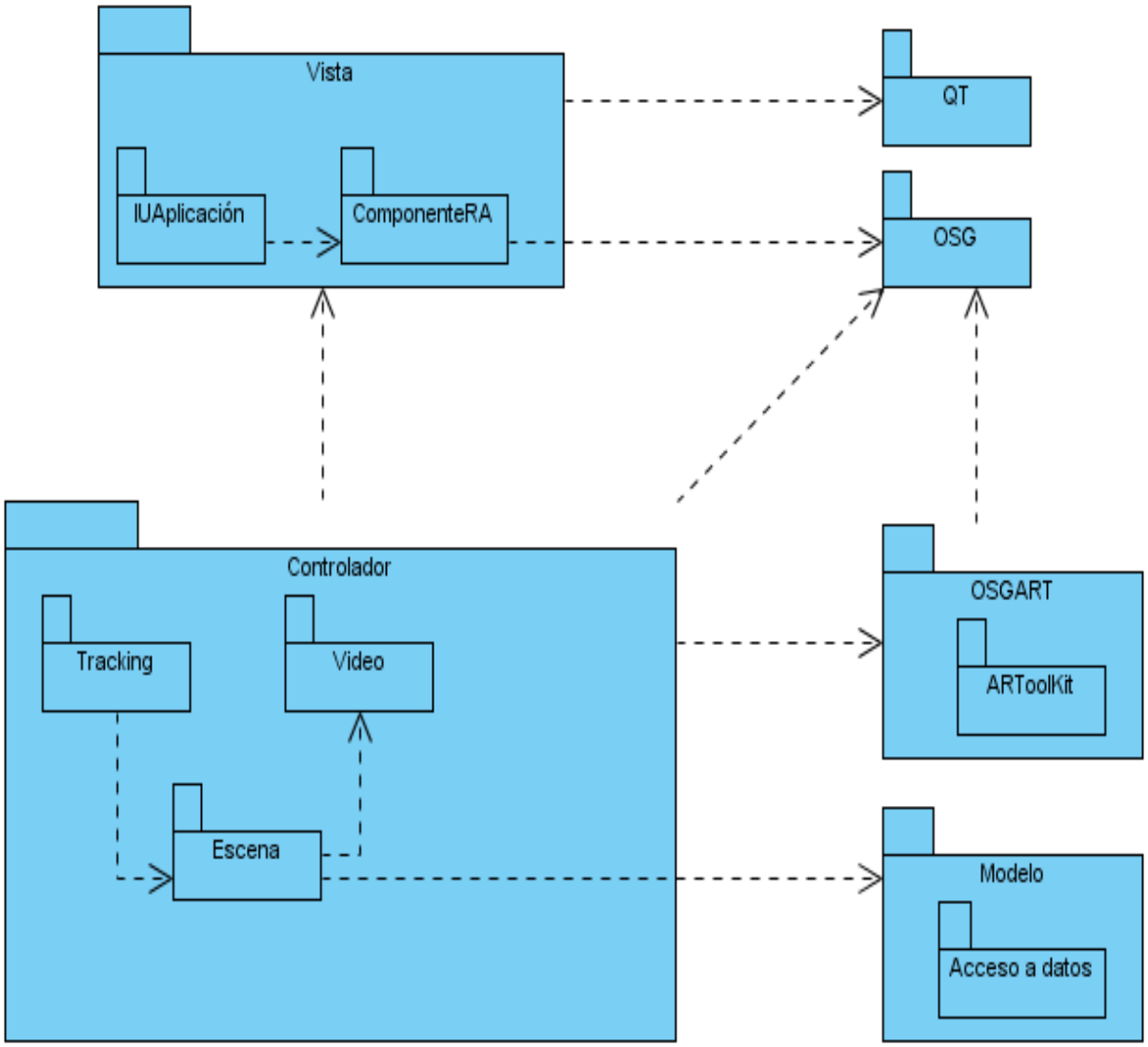

Figura 18. Diagrama de Paquete del Diseño.

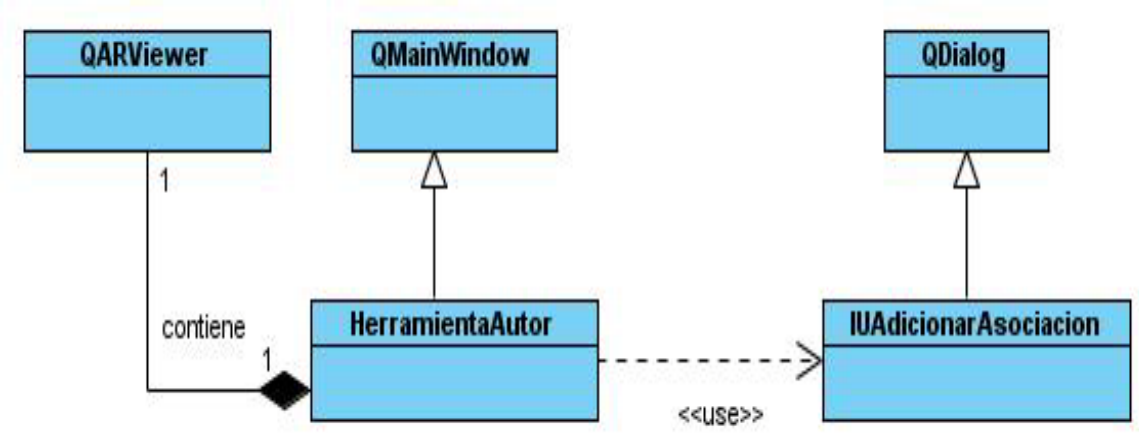

Figura 19. Diagrama de Clases del diseño, paquete “UI_Aplicación”.

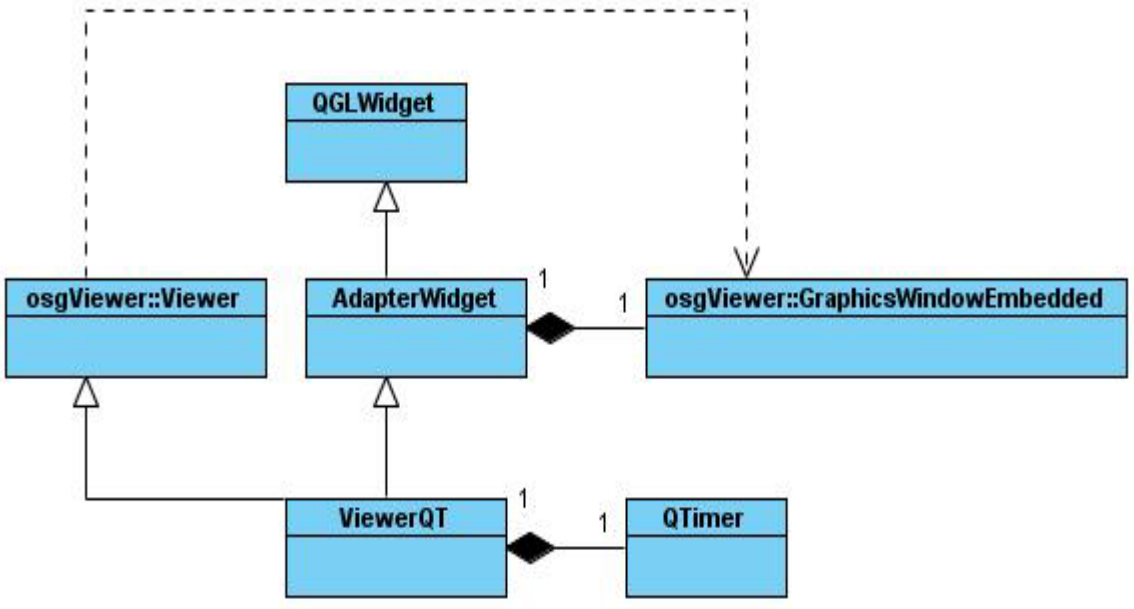

Figura 20. Diagrama de Clases del diseño, paquete “ComponenteAR”.

Figura 21. Diagrama de Clases del Diseño, paquete "Tracker".

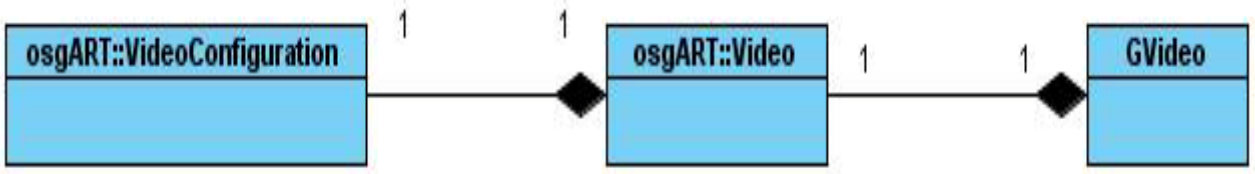

Figura 22. Diagrama de Clases del Diseño, paquete "Video".

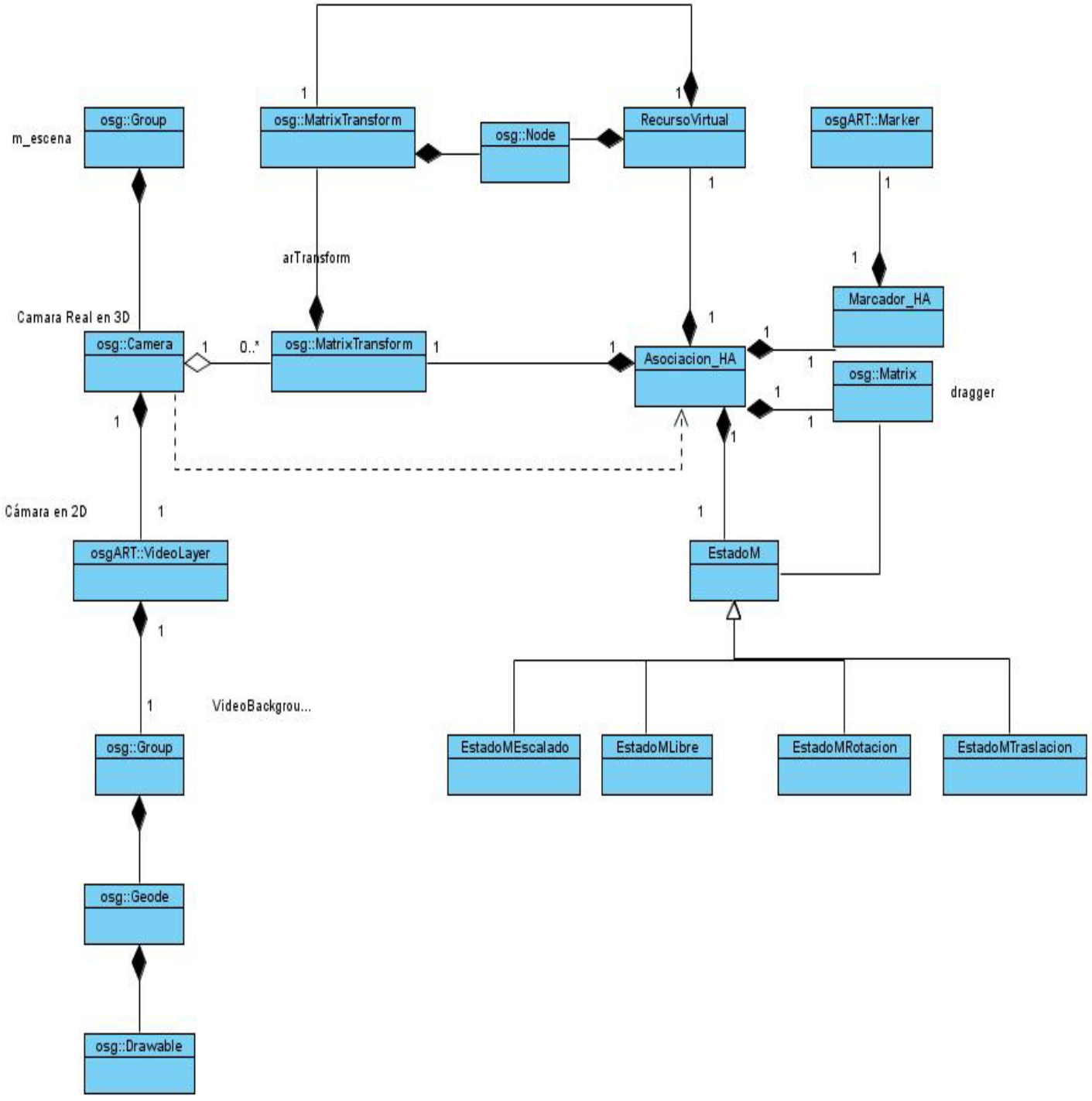

Figura 23. Digrama de Clases del Diseño, paquete "Escena".

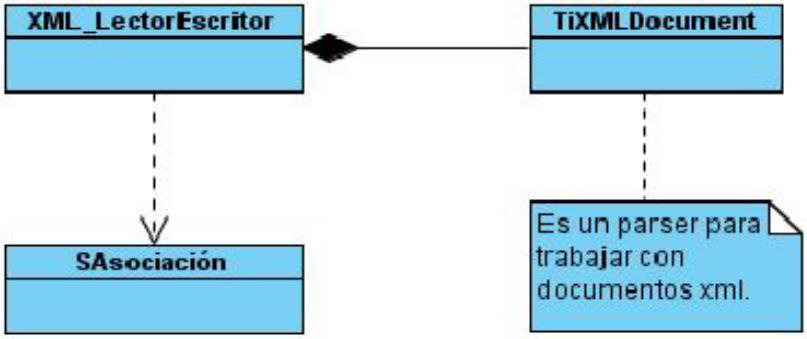

Figura 24. Digrama de Clases del Diseño, paquete "Acceso a Dato".

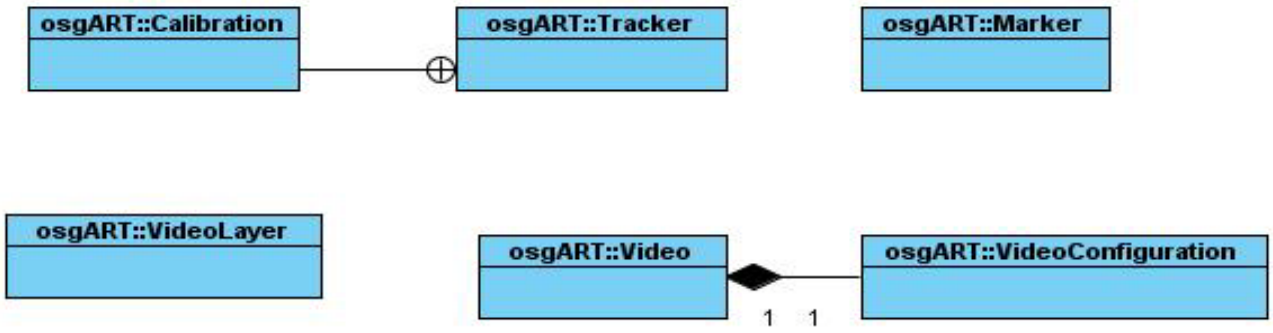

Figura 25. Diagrama de Clases del Diseño, paquete "OSGART".

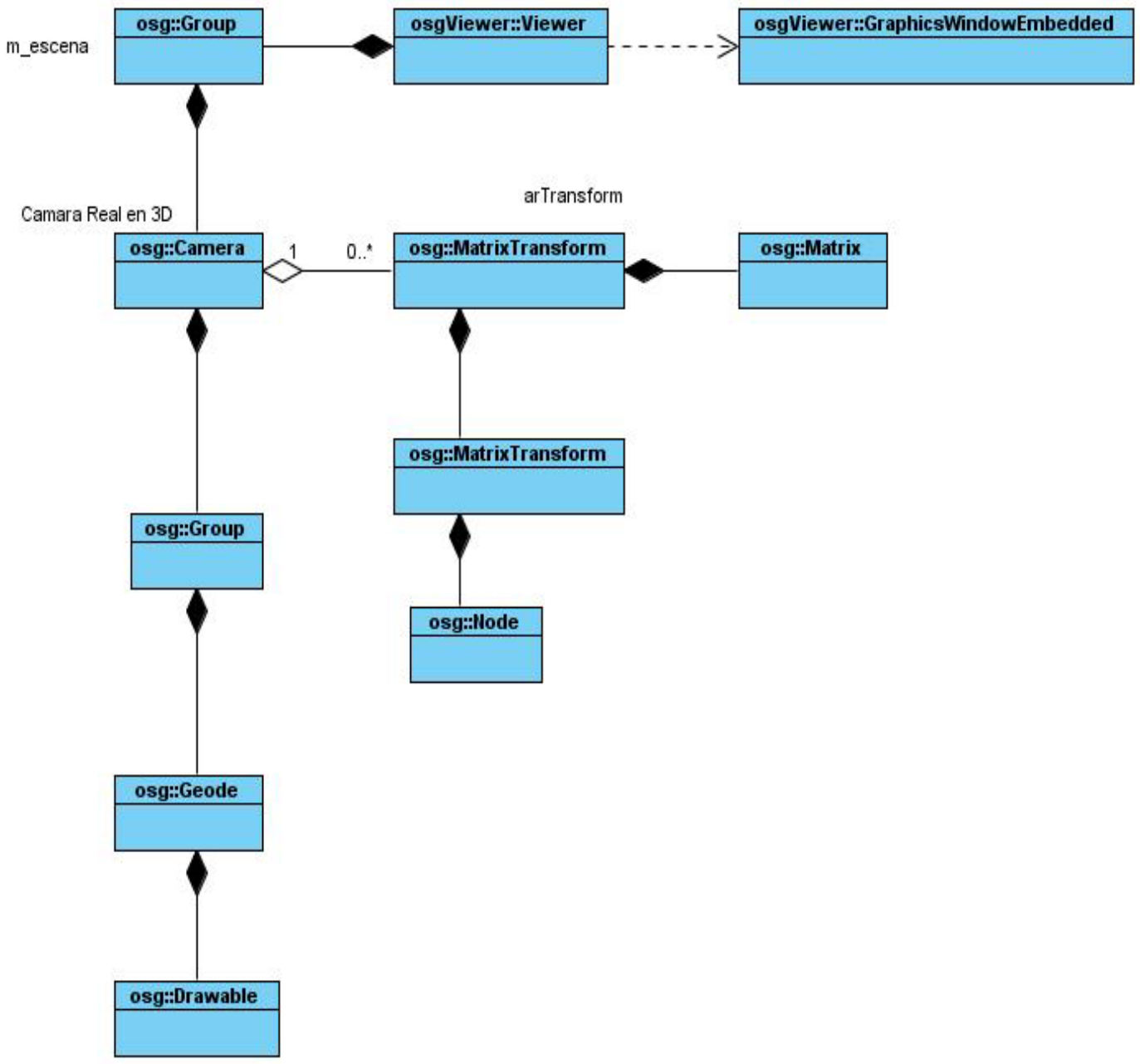

Figura 26. Diagrama de Clases del Diseño, paquete "OSG".

Descripción de Clases de Diseño.

Una vez visto el diagrama de clases, a continuación se proporcionará una descripción de las clases arquitectónicamente más significativas de la solución.

Nombre: Asociacion_HA	
Tipo de clase: Entidad	
Atributo:	**Tipo:**
EstadoM	friend class
m_first_Manipulation	bool
drgMatrix	osg::Matrix
m_mk	Marcador_HA
m_rv	RecursoVirtual
m_nombre_id	std::string
m_seleccionada	bool
m_activa	bool
m_estado	AsocState
m_state	EstadoM
Para cada responsabilidad:	
Nombre:	Asociacion_HA()
Descripción:	Constructor de la clase.
Nombre:	Asociacion_HA(Marcador_HA* mk, RecursoVirtual* rv, bool activa)
Descripción:	Constructor de la clase donde se definen los valores de la asociación, es decir la relación que existe entre el marcador y el recurso.
Nombre:	Asociacion_HA(Tracker_HA* tracker, std::string nombreAsoc, std::string rvFileDir, std::string mkFileDir, int tamano = 80 /*mm*/, int centro_X = 0, int centro_Y = 0 , bool activa = true)
	Constructor de la clase donde se definen los valores de la asociación, es decir la relación que existe entre el marcador y el recurso y se le da a conocer el tracker

Descripción:	para que este maneje internamente la adición del marcador.
Nombre:	Activar(bool valor)
Descripción:	Es para activar el marcador, o sea para que el tracker cada vez que reconozca al marcador visualice el elemento virtual correspondiente al marcador.
Nombre:	cambiarEstado(EstadoM* estado)
Descripción:	Se realiza la transición de estado de la asociación lo cual incluye poner el manipulador correspondiente al estado en el que esté la asociación y en caso que esté libre se le quita el manipulador .
Nombre:	~Asociacion_HA()
Descripción:	Destructor de la clase

Tabla 5. Descripción de Clases de Diseño “Asociacion_HA”.

Nombre: RecursoVirtual	
Tipo de clase: Entidad	
Atributo:	**Tipo:**
m_dir	std::string
m_nombre_id	std::string
m_transformacion	osg::ref_ptr<osg::MatrixTransform>
> m_nodo_rv	osg::ref_ptr<osg::Node
Para cada responsabilidad:	
Nombre:	RecursoVirtual(std::string dir, std::string nombre_id)
Descripción:	Constructor de la clase.
Nombre:	~RecursoVirtual(void)
Descripción:	Destructor de la clase.
Nombre:	getTransformacion()
Descripción:	Retorna una referencia a la matriz de transformación del recurso virtual.
Nombre:	getRecurso()
Descripción:	Retorna una referencia al recurso virtual.

Nombre:	getDir()
Descripción:	Retorna la dirección del archivo que contiene la información del recurso virtual.
Nombre:	setDir(std::string dir_archivo)
Descripción:	Se modifica la dirección del archivo que contiene la información del recurso virtual.

Tabla 6. Descripción de Clases de Diseño "RecursoVirtual".

Nombre: OSGART_SceneMgr	
Tipo de clase: Controladora	
Atributo:	**Tipo:**
_instance	static OSGART_SceneMgr
MapaAsociaciones	typedef std::map< std::string, Asociacion_HA *>
m_asociaciones	MapaAsociaciones
m_escena	osg::ref_ptr<osg::Group>
m_camaraReal	osg::ref_ptr<osg::Camera>
m_tracker	Tracker_HA
m_video	Video_HA
m_sceneTState	SceneMgrState
m_activeAsoc	std::string
Para cada responsabilidad:	
Nombre:	OSGART_SceneMgr()
Descripción:	Constructor de la clase.
Nombre:	~OSGART_SceneMgr()
Descripción:	Destructor de la clase.
Nombre:	instance()
Descripción:	Garantiza una única instancia de la clase.
Nombre:	inicializarVideo(std::string nombrePlugin/* = "osgart_video_artoolkit2"*/, bool desdeCamara /* = true*/, const std::string &deviceConfig/* = ""*/)
Descripción:	Se crea una nueva instancia de la clase video y se inicializa el componente de

	video de osgART con el nombre la biblioteca y se le pone por defecto que es desde la webcam.
Nombre:	inicializarTracking(std::string nombrePlugin /*= "osgart_tracker_artoolkit2"*/, const std::string &calib_file /*= "datos/camera_para.dat"*/)
Descripción:	Inicializa tracking creando la configuración del tracker con el plugin de tracking y se cargan los datos de calibración de la cámara para generar la matriz de proyección de la cámara.
Nombre:	inicializarEscenaRA()
Descripción:	Se le especifica al tracker que recorra todo el grafo de la escena modificando según corresponda, se crea la configuración de la calibración de la cámara, se crea la textura de fondo, se asigna la cámara que mira la escena y se pone a correr el video que aún no se había mandado a visualizar(Acondicionar la escena inicial para adicionar las asociaciones).
Nombre:	inicializarRA(std::string nombrePluginVideo, bool desdeCamara, const std::string &deviceConfig, std::string nombrePluginTracking, const std::string &calib_file)
Descripción:	Se inicializa el video, se inicializa el tracking, y por último se inicializa la EscenaRA.
Nombre:	adicionarAsociacion(std::string nombreAsoc, std::string rvFileDir, std::string mkFileDir , int tamano /*= 80 mm*/, int centro_X /*= 0*/, int centro_Y, bool activa)
Descripción:	Se crea una nueva asociación entre el marcador y el recurso virtual, se añade a la lista de asociación de la clase y se incorpora a la escena.

Tabla 7. Descripción de Clases de Diseño “OSGART_SceneMgr”.

Nombre: EstadoM	
Tipo de clase: Entidad	
Atributo:	**Tipo:**
m_grpMDrg	osg::ref_ptr<osg::Group>
m_dragger	osg::ref_ptr<osgManipulator::Dragger>
m_selection	osg::ref_ptr<osgManipulator::Selection>
m_commandManager	osg::ref_ptr<osgManipulator::CommandManager>
Para cada responsabilidad:	
Nombre:	eRotar(Asociacion_HA * asoc)
Descripción:	Método virtual que se redefine en las clases hijas.
Nombre:	eEscalar(Asociacion_HA * asoc)
Descripción:	Método virtual que se redefine en las clases hijas.
Nombre:	eTrasladar(Asociacion_HA * asoc)
Descripción:	Método virtual que se redefine en las clases hijas.
Nombre:	eLibre(Asociacion_HA * asoc)
Descripción:	Método virtual que se redefine en las clases hijas.
Nombre:	abrirEstado(Asociacion_HA * asoc)
Descripción:	Método virtual que se redefine en las clases hijas.
Nombre:	terminarEstado(Asociacion_HA * asoc)
Descripción:	Método virtual que se redefine en las clases hijas.
Nombre:	EstadoM()
Descripción:	Constructor de la clase.
Nombre:	~EstadoM()
Descripción:	Destructor de la clase.

Tabla 8. Descripción de Clases de Diseño “EstadoM”.

Nombre: EstadoMEscalado	
Tipo de clase: Entidad	
Atributo:	**Tipo:**
m_instance	static EstadoMEscalado
Para cada responsabilidad:	
Nombre:	instance()
Descripción:	Se crea un nueva instancia de la clase.
Nombre:	eRotar(Asociacion_HA * asoc)
Descripción:	Se quita el grafo del estado en que se encuentra la asociación y se pone el grafo del estado al que va a pasar la asociación.
Nombre:	eLibre(Asociacion_HA * asoc)
Descripción:	Se quita el grafo del estado en que se encuentra la asociación y se pone el grafo del estado al que va a pasar la asociación.
Nombre:	eTrasladar(Asociacion_HA * asoc)
Descripción:	Se quita el grafo del estado en que se encuentra la asociación y se pone el grafo del estado al que va a pasar la asociación.
Nombre:	abrirEstado(Asociacion_HA * asoc)
Descripción:	
Nombre:	terminarEstado(Asociacion_HA * asoc)
Descripción:	Se le pregunta a la asociación cual es el estado en el que debe estar, en función de este estado se llama a la instancia de la clase del estado en el que se encuentra la misma, se busca la matriz y se cambia, luego se llama al método correspondiente a la acción.
Nombre:	EstadoMEscalado()
Descripción:	Constructor de la clase.
Nombre:	~EstadoMEscalado()
Descripción:	Destructor de la clase

Tabla 9. Descripción de Clases de Diseño “EstadoMEscalado”.

Nombre: EstadoMLibre	
Tipo de clase: Entidad	
Atributo:	**Tipo:**
m_instance	static EstadoMLibre
Para cada responsabilidad:	
Nombre:	instance()
Descripción:	Se crea un nueva instancia de la clase.
Nombre:	eRotar(Asociacion_HA * asoc)
Descripción:	Se quita el grafo del estado en que se encuentra la asociación y se pone el grafo del estado al que va a pasar la asociación.
Nombre:	eEscalar(Asociacion_HA * asoc)
Descripción:	Se quita el grafo del estado en que se encuentra la asociación y se pone el grafo del estado al que va a pasar la asociación.
Nombre:	eTrasladar(Asociacion_HA * asoc)
Descripción:	Se quita el grafo del estado en que se encuentra la asociación y se pone el grafo del estado al que va a pasar la asociación.
Nombre:	EstadoMLibre()
Descripción:	Constructor de la clase.
Nombre:	~EstadoMLibre()
Descripción:	Destructor de la clase

Tabla 10. Descripción de Clases de Diseño “EstadoMLibre”.

Nombre: EstadoMRotacion	
Tipo de clase: Entidad	
Atributo:	**Tipo:**
m_instance	static EstadoMRotacion
Para cada responsabilidad:	
Nombre:	instance()
Descripción:	Se crea un nueva instancia de la clase.
Nombre:	eEscalar(Asociacion_HA * asoc)
Descripción:	Se quita el grafo del estado en que se encuentra la asociación y se pone el grafo del estado al que va a pasar la asociación.
Nombre:	eLibre(Asociacion_HA * asoc)
Descripción:	Se quita el grafo del estado en que se encuentra la asociación y se pone el grafo del estado al que va a pasar la asociación.
Nombre:	eTrasladar(Asociacion_HA * asoc)
Descripción:	Se quita el grafo del estado en que se encuentra la asociación y se pone el grafo del estado al que va a pasar la asociación.
Nombre:	~EstadoMRotacion(void)
Descripción:	Destructor de la clase.
Nombre:	EstadoMRotacion()
Descripción:	Constructor de la clase.

Tabla 11. Descripción de Clases de Diseño “EstadoMRotacion”.

Nombre: EstadoMTraslacion	
Tipo de clase: Entidad	
Atributo:	**Tipo:**
m_instance	static EstadoMTraslacion
Para cada responsabilidad:	
Nombre:	instance()
Descripción:	Se crea un nueva instancia de la clase.
Nombre:	eRotar(Asociacion_HA * asoc)
Descripción:	Se quita el grafo del estado en que se encuentra la asociación y se pone el grafo del estado al que va a pasar la asociación.
Nombre:	eLibre(Asociacion_HA * asoc)
Descripción:	Se quita el grafo del estado en que se encuentra la asociación y se pone el grafo del estado al que va a pasar la asociación.
Nombre:	eEscalar(Asociacion_HA * asoc)
Descripción:	Se quita el grafo del estado en que se encuentra la asociación y se pone el grafo del estado al que va a pasar la asociación.
Nombre:	EstadoMTraslacion()
Descripción:	Constructor de la clase.
Nombre:	~EstadoMTraslacion()
Descripción:	Destructor de la clase

Tabla 12. Descripción de Clases de Diseño “EstadoMTraslacion”.

Nombre: Video_HA	
Tipo de clase: Entidad	
Atributo:	**Tipo:**
m_video	osg::ref_ptr<osgART::Video>
m_config	osgART::VideoConfiguration*
m_config_disp	std::string
m_nombre_bib	std::string
m_video_id	int
m_camara	bool
Para cada responsabilidad:	
Nombre:	Video_HA(std::string nombre_bib, bool desdeCamara)
Descripción:	Constructor de la clase.
Nombre:	init()
Descripción:	Se inicializa el plugin de video de osgART.
Nombre:	setConfigDispositivo(const std::string config_disp)
Descripción:	Se le da la configuración sobre el archivo de video que se va a reproducir.
Nombre:	open()
Descripción:	Abrir flujo de video.
Nombre:	start()
Descripción:	Se comienza a reproducir el video.
Nombre:	close()
Descripción:	Cierra el flujo de video.
Nombre:	~Video_HA()
Descripción:	Destructor de la clase

Tabla 13. Descripción de Clases de Diseño “Video_HA”.

Nombre: Tracker_HA	
Tipo de clase: Entidad	
Atributo:	**Tipo:**
m_id	int
m_nombre_bib	std::string
m_calib_file	std::string
m_tracker	osg::ref_ptr<osgART::Tracker>
Para cada responsabilidad:	
Nombre:	Tracker_HA(std::string nombre_bib /*= osgart_tracker_artoolkit2"*/, std:: string calib_file /*= "datos/camera_para.dat"*/): m_nombre_bib(nombre_bib), m_calib_file(calib_file)
Descripción:	Constructor de la clase.
Nombre:	~Tracker_HA()
Descripción:	Destructor de la clase.
Nombre:	cargarTracker()
Descripción:	Carga la biblioteca de tracking que se va a utilizar.
Nombre:	esValido()
Descripción:	Para saber si se cargó el tracker.
Nombre:	cargarCalibracion()
Descripción:	Se carga el fichero de calibración de la cámara para generar la matriz de proyección en perspectiva.
Nombre:	actualizar()
Descripción:	Actualiza la información del tracker.
Nombre:	setVideo(osgART::Video* video)
Descripción:	Alimentar el tracker con la imagen de video que se está capturando.

Tabla 14. Descripción de Clases de Diseño “Tracker_HA”.

Nombre: Marcador_HA		
Tipo de clase: Entidad		
Atributo:		**Tipo:**
m_marker		osg::ref_ptr<osgART::Marker>
m_Tipo_Mk		Tipo_MK
m_Biblioteca_Marcador		Bib_Fuente_MK
m_file_Dir		std::string
m_tamano		int
m_centro_X		int
m_centro_Y		int
Para cada responsabilidad:		
Nombre:	Marcador_HA(std::string file_Dir , int tamano = 80 /*mm*/, int centro_X = 0, int centro_Y = 0, Tipo_MK tipo_Mk = SINGLE, Bib_Fuente_MK bib_Marcador = ARTOOLKIT)	
Descripción:	Constructor de la clase.	
Nombre:	~Marcador_HA()	
Descripción:	Destructor de la clase.	
Nombre:	activar(bool valor)	
Descripción:	Activar o desactivar un marcador.	
Nombre:	cargarMarcador(Tracker_HA * tracker)	
Descripción:	Método que sirve para añadir la información del marcador al tracker.	
Nombre:	esValido()	
Descripción:	Para determinar si el tracker cargó la información del marcador.	

Tabla 15. Descripción de Clases de Diseño “Marcador_HA”.

Nombre: ViewerQT	
Tipo de clase: Entidad	
Atributo:	**Tipo:**
_timer	QTimer
Para cada responsabilidad:	
Nombre:	ViewerQT(QWidget * parent = 0, const QGLWidget * shareWidget = 0, WindowFlags f = 0):AdapterWidget(parent, shareWidget, f)
Descripción:	Constructor de la clase.
Nombre:	paintGL()
Descripción:	Se redefine la función de se utiliza para dibujar en el widget de QT con OSG.

Tabla 16. Descripción de Clases de Diseño “ViewerQT”.

Nombre: AdapterWidget	
Tipo de clase: Entidad	
Atributo:	**Tipo:**
_gw	osg::ref_ptr<osgViewer::GraphicsWindowEmbedded>
Para cada responsabilidad:	
Nombre:	AdapterWidget(QWidget * parent = 0, const QGLWidget * shareWidget = 0, WindowFlags f = 0)
Descripción:	Constructor de la clase.
Nombre:	~AdapterWidget()
Descripción:	Destructor de la clase.
Nombre:	getGraphicsWindow()
Descripción:	Retorna el Manipulador de Ventana de OSG.
Nombre:	resizeGL(int width, int height)
Descripción:	Inyectando la redimension en OSGViewer.
Nombre:	keyPressEvent(QKeyEvent* event)
Descripción:	Transfiriendo los eventos de teclado de Qt a osgGA::GUIEventAdapter (Press).

Nombre:	keyReleaseEvent(QKeyEvent* event)
Descripción:	Transfiriendo los eventos de teclado de Qt a osgGA::GUIEventAdapter (Release.)
Nombre:	mousePressEvent(QMouseEvent* event)
Descripción:	Transfiriendo los eventos de Mouse de Qt a OSGViewer (Press).
Nombre:	mouseReleaseEvent(QMouseEvent* event)
Descripción:	Transfiriendo los eventos de Mouse de Qt a OSGViewer (Release).
Nombre:	mouseMoveEvent(QMouseEvent* event)
Descripción:	Transfiriendo los eventos de Mouse de Qt a OSGViewer (Motion).

Tabla 17. Descripción de Clases de Diseño “AdapterWidget”.

3.1.4 Diagramas de Secuencia.

El diagrama de secuencia es uno de los diagramas más efectivos para modelar interacción entre objetos en un sistema. Un diagrama de secuencia muestra la interacción de un conjunto de objetos en una aplicación a través del tiempo y se modela para cada caso de uso del sistema. El mismo contiene detalles de implementación del escenario, incluyendo los objetos y clases que se usan para implementar el escenario, y mensajes cambiados entre los objetos.

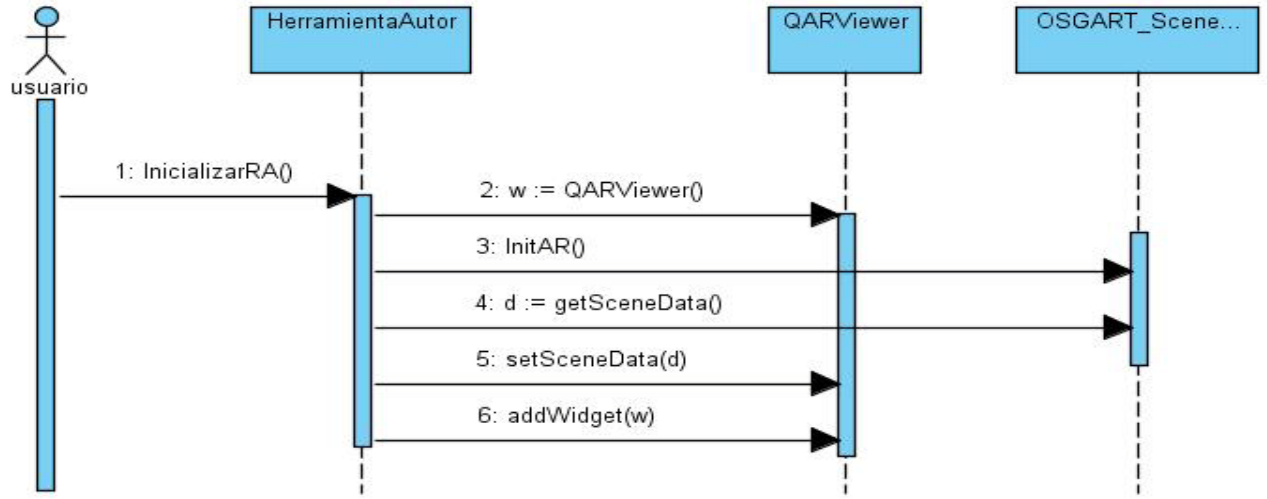

Figura 27. Diagrama de Secuencia "Inicializar escena RA".

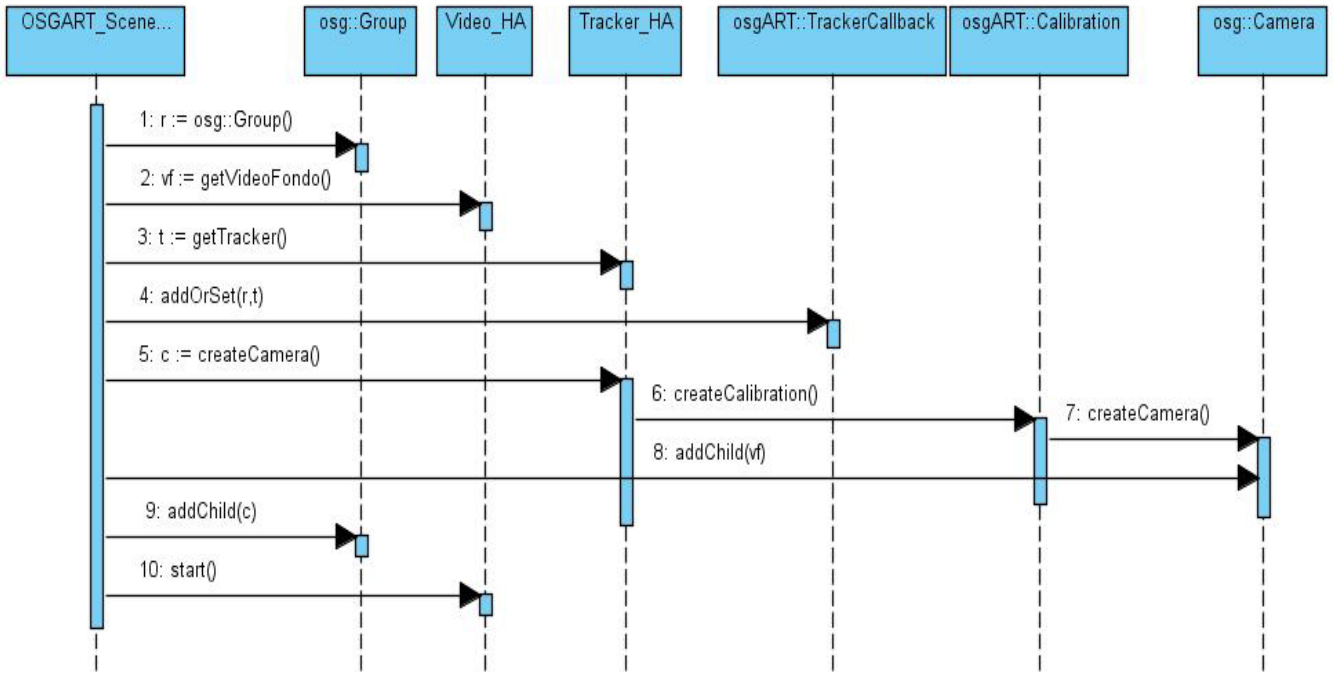

Figura 28. Diagrama de Secuencia "Inicializar escena".

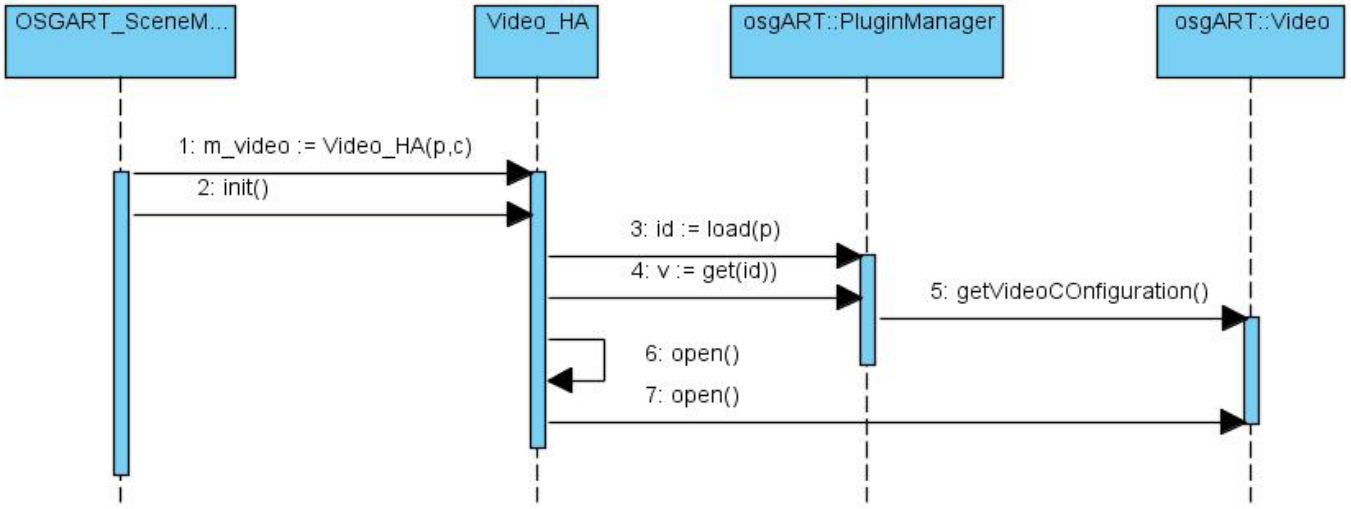

Figura 29. Diagrama de Secuencia "Inicializar Video".

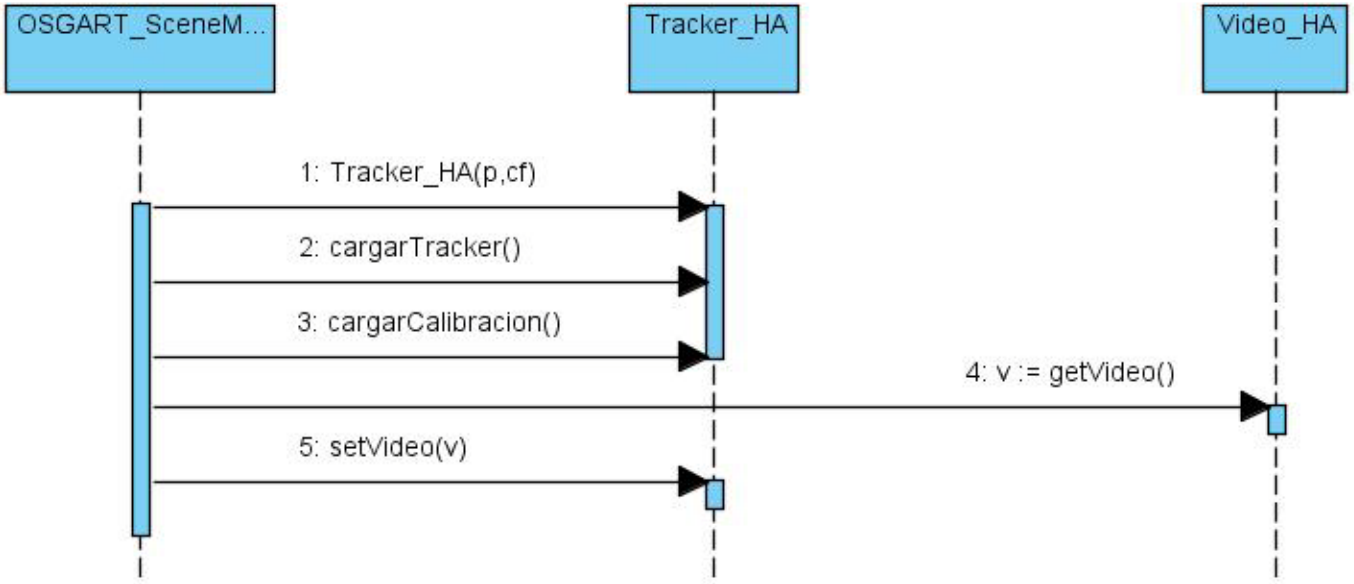

Figura 30. Diagrama de Secuencia "Inicializar Tracker".

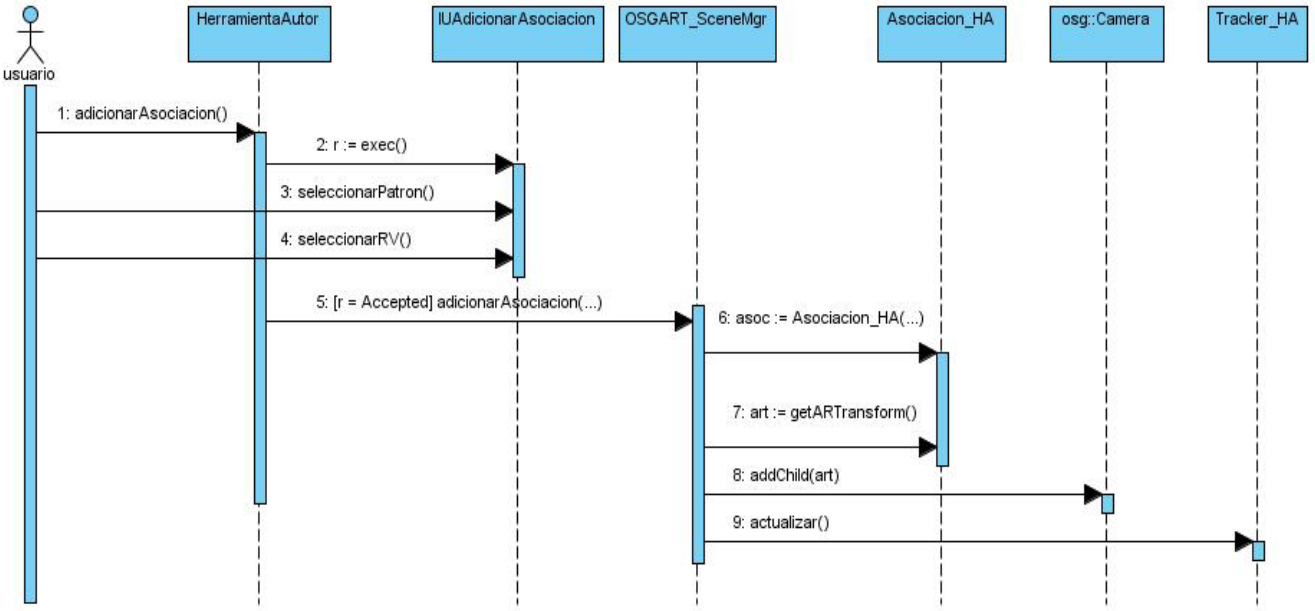

Figura 31. Diagrama de Secuencia "Adicionar Asociación".

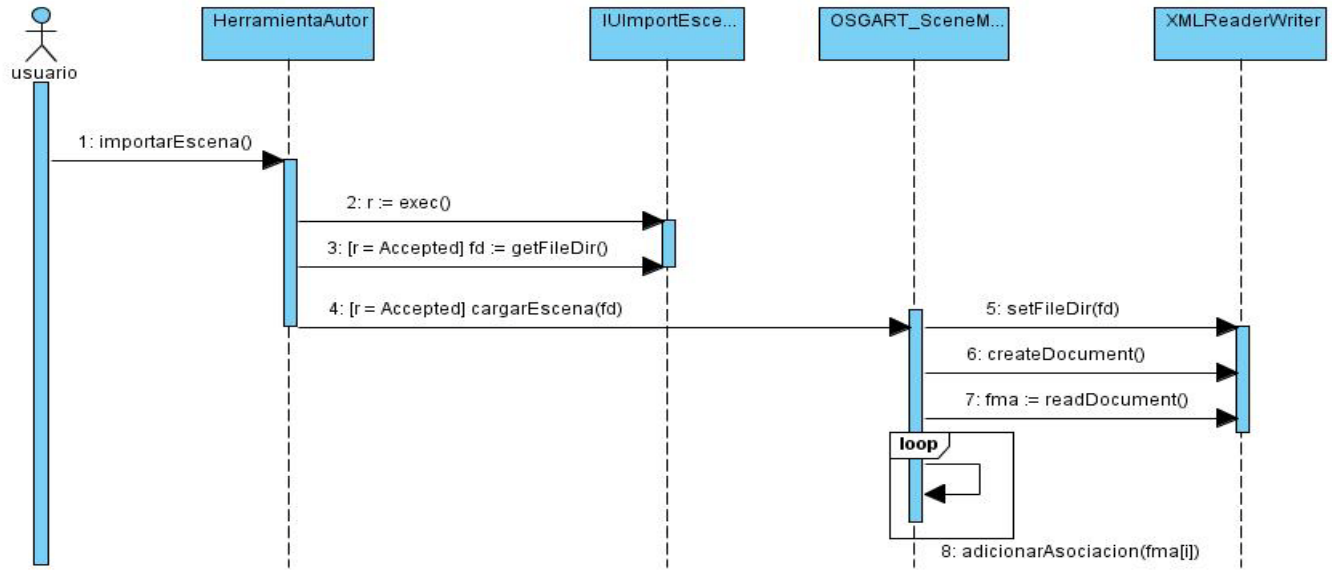

Figura 32. Diagrama de Secuencia "Importar escena".

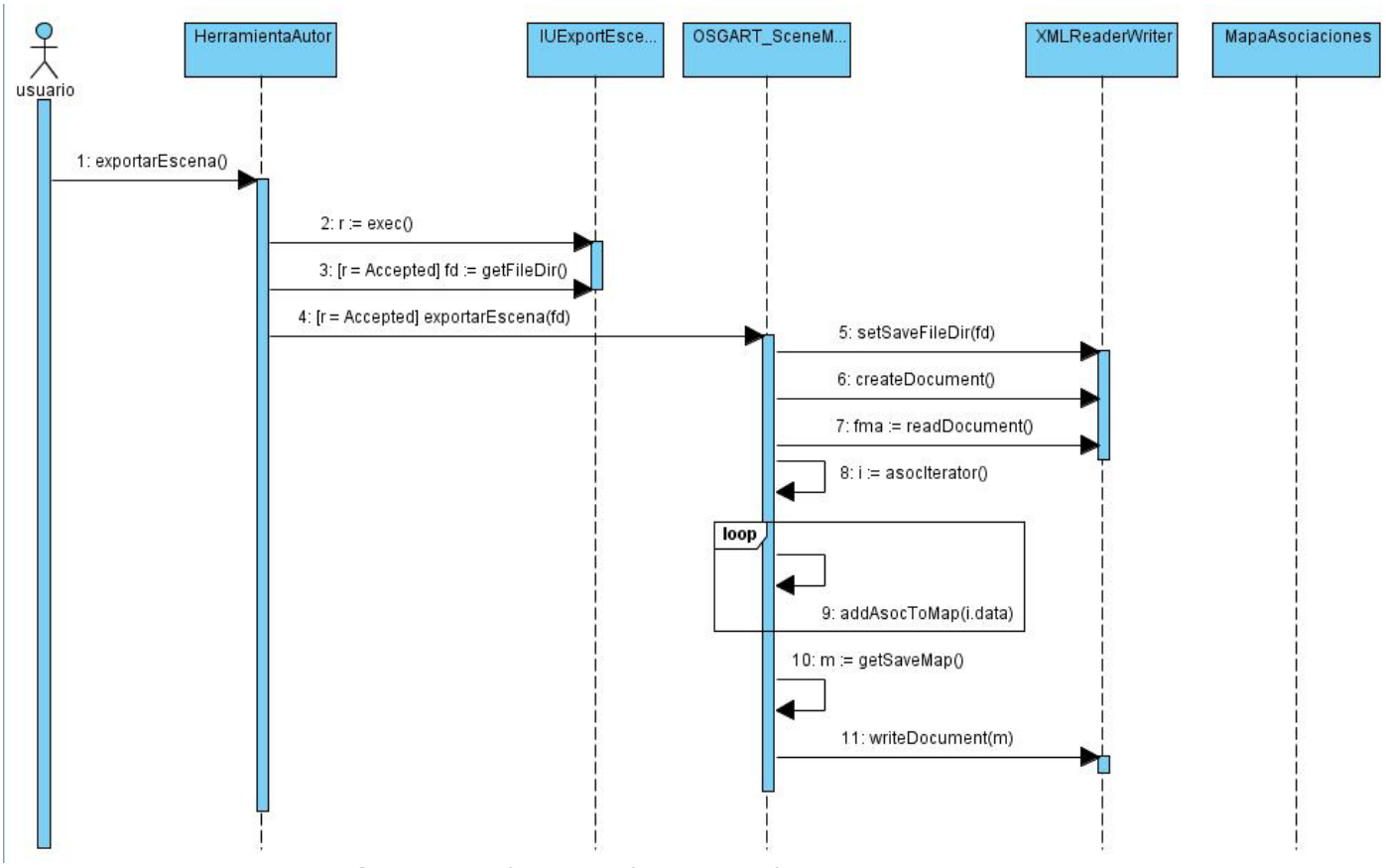

Figura 33. Diagrama de Secuencia "Exportar escena".

3.2 Modelo de implementación.

En el modelo de implementación se describe cómo los elementos del modelo de diseño se deben implementar en términos de componentes, cómo se organizan estos de acuerdo con los mecanismos de estructuración disponibles en el entorno de implementación y en el lenguaje de programación utilizado; así como la dependencia entre los componentes. Esto se hace a través del diagrama de paquetes y del diagrama de componentes.

3.2.1 Diagrama de Despliegue.

El diagrama de despliegue analizado en este documento es muy simple, sólo incluye la representación de una computadora y una cámara conectada por puerto USB, puesto que al ser la aplicación de escritorio todo va a estar en un solo nodo, como se muestra en la figura.

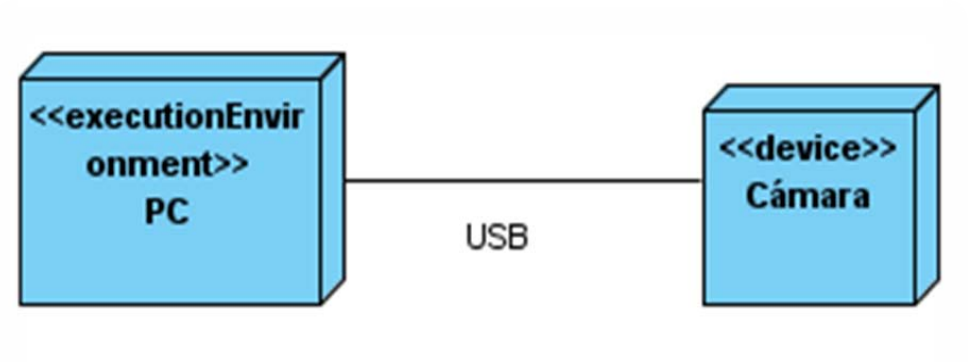

Figura 34. Diagrama de Despliegue.

3.2.2 Diagrama de Componentes.

Los diagramas de componentes son usados para estructurar el modelo de implementación en términos de subsistemas de implementación y mostrar las relaciones entre dichos elementos. Es un grafo de componentes unidos a través de relaciones que pueden ser de compilación o de ejecución, y además se pueden representar las interfaces de esos componentes.

Es otra forma de representar una vista estática del sistema, que muestra la organización y dependencia que existe entre los componentes físicos que se necesitan para ejecutar la aplicación, sean estos componentes de código fuente, librerías, binarios o ejecutables.

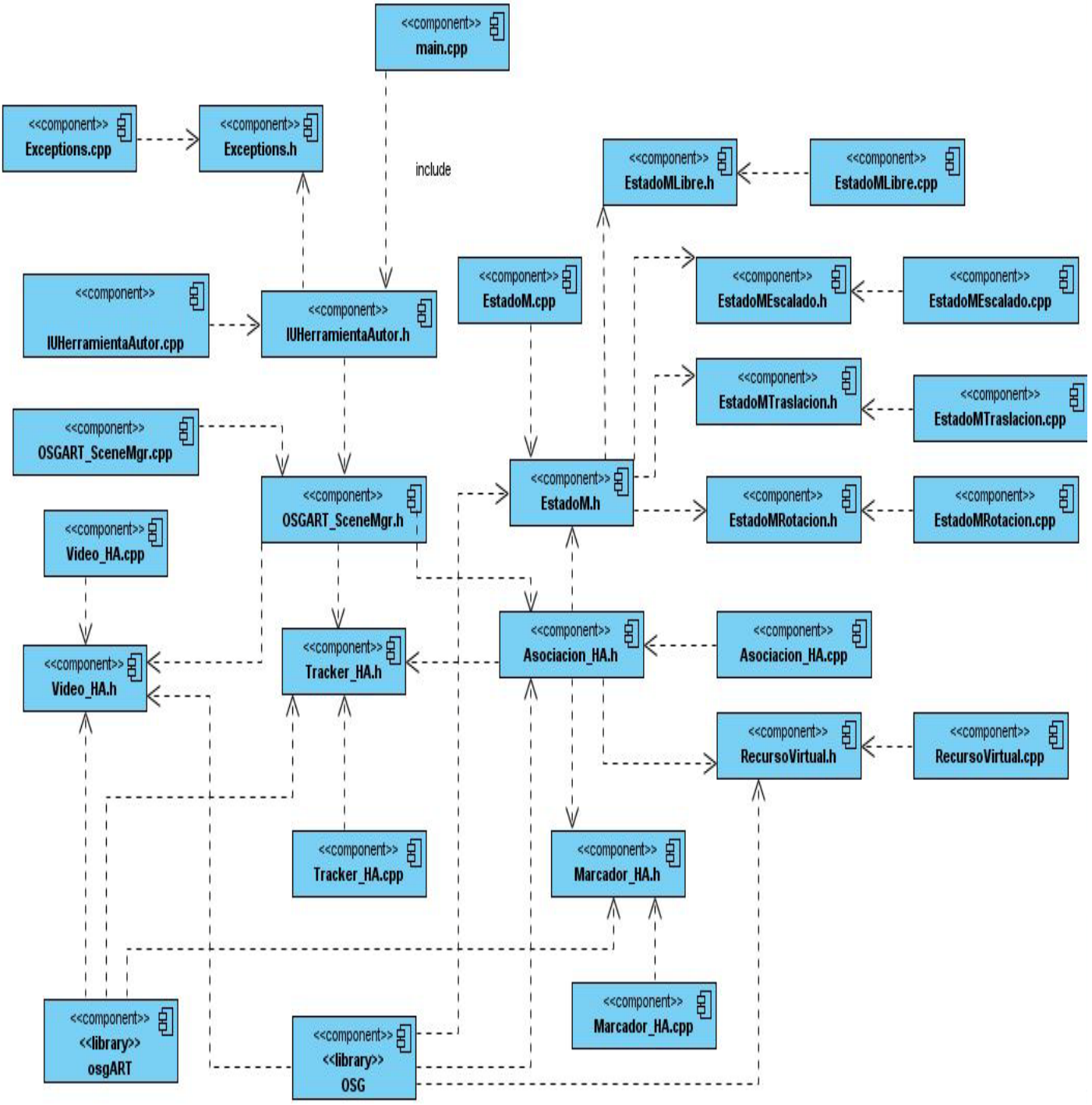

Figura 35. Diagrama de Componente.

Conclusiones parciales del capítulo.

En este capítulo se presentaron los diagramas de clases del diseño y los diagramas de secuencia de la realización de los casos de uso. Permitiendo todo esto que se pase a la fase de implementación del sistema, en la que se dieron a conocer los diagramas más importantes del flujo de trabajo de implementación como son el diagrama de despliegue y el diagrama de componentes, desarrollando la primera iteración de la Herramienta de autor para la creación de contenidos de Realidad Aumentada.

CONCLUSIONES

Durante la presente investigación la búsqueda bibliográfica jugó un papel fundamental, porque permitió conocer los principales conceptos relacionados con las herramientas de autor, así como la clasificación de este tipo de herramientas de creación de contenidos digitales de Realidad Aumentada en dos grupos: orientadas a programadores y orientadas a no programadores. Al identificar las características más prominentes de este tipo de herramientas, se pudo constatar que muchas están bajo licencias propietarias y que una por sí sola no reúne todas las funcionalidades deseables en la herramienta desarrollada como resultado de la presente investigación. La afirmación anterior se apoya en que cada una de estas herramientas en particular cuenta con funcionalidades que la diferencian de las demás, que en muchas ocasiones son complementarias entre sí y que son de gran utilidad si se encuentran todas reunidas en una sola solución.

Con la realización de esta tesis, y en cumplimiento del objetivo inicialmente planteado se implementó una Herramienta de autor para la creación de contenidos de Realidad Aumentada, bajo la clasificación: herramienta orientada a no programadores, que satisface las necesidades de la Línea de Desarrollo Interacción 3D de la Facultad 5. En este momento la utilización de la herramienta permite acondicionar una escena de Realida Aumentada en un tiempo muy reducido en comparación con el método que se utilizaba anteriormente. También se logró reducir el margen de errores cometidos típicos del proceso de edición manual de estas escenas al ser este totalmente reemplazado por un proceso de edición semiautomático.

Para el desarrollo de la solución que aquí se propone se utilizaron un conjunto de bibliotecas que se rigen por licencias libres que ofrecen el valor añadido de la seguridad y estabilidad, avaladas por la amplia comunidad de desarrollo que labora en torno a ellas. Otro factor importante es que la naturaleza multiplataforma de las bibliotecas bases con las que se elaboró la herramienta, permite su fácil migración a plataformas libres GNU\Linux, lo cual es un paso de avance hacia la independencia tecnológica que requiere Cuba en estos momentos, por lo que constituye una solución que en un futuro cercano puede tener un impacto positivo a nivel nacional tanto en la economía como en la sociedad.

RECOMENDACIONES

Como resultado del trabajo realizado se identificaron características y funcionalidades que por su amplitud y complejidad sobrepasan el alcance de la presente investigación. Por tal motivo, para futuras investigaciones y proyectos relacionados con este trabajo y con el objetivo de darle seguimiento al mismo para enriquecer la solución en aras de convertirla en un software informático de nivel internacional, se recomienda:

- Implementar aquellas funcionalidades contenidas en los casos de uso secundario que no fueron incluida en esta primera iteración.
- Realizar análisis de rendimiento de la solución para optimizar el consumo de recursos de hardware en la medida de lo posible.

REFERENCIAS BIBLIOGRÁFICAS

1. Ruiz Aguilar, Alberto, Acién Martínez, Fátima y Vázquez Fernández, José Luis -Baca. *Sistema de posicionamiento en la creación de un libro interactivo. Revista Digital Universitaria, Vol. 8.* **[En línea] [Citado el: 8 de febrero de 2010.]** http://www.revista.unam.mx/vol.8/num6/art49/jun_art49.pdf

2. Valverde Berrocoso, Jesús. *Programas de propósito general o instrumentales: HERRAMIENTAS DE AUTOR.* **[En línea][Citado el 9 de Febrero de 2010]** http://www.unex.es/didactica/Tecnologia_Educativa/info03J.htm

3. Wang, Yuan, Langlotz, Tobias, Billinghurst, Mark y Bell, Tim. *An Authoring Tool for Mobile Phone AR Environments.* **[En línea] [Citado el: 15 de Febrero de 2010.]** http://www.icg.tugraz.at/pub/PhoneAuthoring

4. Software de Realidad Aumentada. *AR Toolkit.* http://www.hitl.washington.eduartoolkit

5. Software de Realidad aumentada. *Mr. Planet.* **[En línea] 2006. [Citado el: 7 de Febrero de 2010.]** http://planet.urv.es/planetrv/media/manual_ES.pdf

6. Software de Realidad Aumentada. *DART: The Designer's Augmented Reality Toolkit.* **[En línea][Citado el: 20 de Febrero de 2010.]** http://www.cc.gatech.edu/projects/ael/projects/dart.html

7. Software de Realidad Aumentada. *BuildAR* **[En línea] [Citado el: 5 de Febrero de 2010.]** http://www.hitlabnz.org/wiki/BuildAR

8. Software de Realidad Aumentada. *Atomic.* **[En línea] [Citado el: 4 de Febrero de 2010.]** http://www.sologicolibre.orgprojectsatomic

9. Seichter, Hartmut, Looser, Julian y Billinghurst, Mark. *ComposAR: An Intuitive Tool for Authoring AR Applications.* **[En línea][Citado el: 5 de Febrero de 2010.]** http://www.hitlabnz.org/publications/2008-ISMAR-ComposARIntuitiveToolAugmentedReality.pdf

10. Software de Realidad Aumentada. *ARTag.* **[En línea][Citado el: 15 de Febrero de 2010.]** http://artag.net/

11. Jacobson, Ivar, Booch, Grady y Rumbaugh, James. *El Proceso Unificado de Desarrollo de Software.* **Madrid: PEARSON EDUCACIÓN, 1999. 84-7829-036-2.**

12. Kroll, Per y Kruchten, Philippe. *The Rational Unified Process Made Easy: A Practitioner's Guide to the RUP.* **s.l.: Addison Wesley, Abril 2003. ISBN 0-321-16609-4.**

13. Company Headquarters. *Visual Paradigm. 10 Reasons to Choose Visual Paradigm.* **[En línea] 2006. [Citado el: 8 de febrero de 2010.]** http://www.visual-paradigm.com/aboutus/10reasons.jsp

14. Qt.Nokia. *Qt: cross-platform rich client development framework*. **[En linea] 2008. [Citado el: 8 de febrero de 2010]** http://qt.nokia.com/products/

15. Realidad Aumentada. [En línea] 2008. [Citado el: 7 de Febrero de 2010.] http://realidadeaumentada.com.br/home/index.php?option=com_content&task=view&id=6&Itemid=28

16. OpenSceneGraph. *Introduction.* **[En línea] [Citado el: 10 de Febrero de 2010.]** http://www.openscenegraph.org/projects/osg/wiki/About/Introduction

17. Encuentros Java. *Grafos de escena.* **[En línea] [Citado el: 13 de Febrero de 2010.]** http://encuentrosjava.uji.es/materiales/workshop_3d.pdf

18. Pérez, Fabienny y Romero, Mario Orlando. *Interacción entre elementos virtuales a través de estándares XML y X3D.* **Ciudad de la Habana: s.n., 2008**.

19. BASS L, P Clements y R.Kazman.1998. *"Software Architecture in Practice", Addison-Wesley.*

20. **Larman, C.** *UML y Patrones.* **Vol. Tomo I.**

21. Welicki, León. *MSDN. El Patrón Singleton.* **[En línea] [Citado el: 20 de marzo de 2010.]** http://msdn.microsoft.com/en-us/library/bb972272(es-es).aspx#EDAA.

22. Gamma, E., et al., *Design Patterns, Elements of Reusable Object-Oriented Software.*

BIBLIOGRAFÍA CONSULTADA

1. Manuela Monsalve Arcila, Adriana Castillón Arango, José David Cuartas. *Exploración teórica de la realidad aumentada para determinar su incidencia en el diseño visual.* **[En línea: Julio 3 de 2007] [Consultado el: 10 de febrero de 2010]** http://sologicolibre.org/jose/archivos/ExploracionTeoricaRealidadAumentada.pdf

2. Lic. Angela Belcastro. *Realidad Aumentada.UNLP.* **[En línea] [Consultado el: 10 de febrero de 2010]** http://www.ing.unp.edu.ar/asignaturas/ias/realaum.pdf

3. Clara Boj, Diego Díaz y Julian Oliver. *Algunas prácticas artísticas relacionadas con la realidad aumentada.* **[En línea: 2009-2010] [Consultado el: 12 de febrero de 2010]** http://htca.us.es/materiales/perezdelama/0910_etsas/0910_rv_ra/20091119_realidad_aumentada.pdf

4. Lizbeth Heras Lara y José Luis Villarreal Benítez. *LA REALIDAD AUMENTADA: UNA TECNOLOGÍA EN ESPERA DE USUARIOS.* **[En línea] [Consultado el: 15 de febrero de 2010]** http://www.revista.unam.mx/vol.8/num6/art48/jun_art48.pdf

5. Roberto Garrido y Alex García-Alonso. *Técnicas de Interacción para Sistemas de Realidad Aumentada.* **[En línea] [Consultado el: 20 de febrero de 2010]** http://www.sc.ehu.es/ccwgamoa/pub/apero/AP-RealidadVirtual/08_JOREVIR_Garrido.pdf

6. *Integración en tiempo real de modelos sintéticos tridimensionales con imágenes reales en movimiento.* **[En línea] [Consultado el: 1 de marzo de 2010]** http://www.alvira-asociados.com/pdf/realidad_aumentada.pdf

7. Manuel Ibáñez Herrero. *Realidad Aumentada: ARToolKit para animación de personajes.* **[En línea] [Consultado el: 5 de marzo de 2010]** http://www.disca.upv.es/magustim/val/pfcs_anteriors/arToolkit/Memoria%20ARToolkit.pdf

8. Dennis Joele. *Development of an Augmented Reality system using ARToolKit and user invisible markers.* **[En línea] [Consultado el: 10 de marzo de 2010]** http://graphics.tudelft.nl/~vrphobia/RA_final_report_Dennis_Joele.pdf

9. Blair MacIntyre, Maribeth Gandy, Steven Dow y Jay David Bolter. *DART: A Toolkit for Rapid Design Explorationof Augmented Reality Experiences.* **[En línea] [Consultado el: 10 de marzo de 2010]** http://www.stanford.edu/~spdow/files/AEL-DART-UIST04.pdf

10. *Rational Unified Process: Best Practices for Software Development Teams.* **[En línea] [Consultado el: 15 de marzo de 2010]** http://www.ibm.com/developerworks/rational/library/content/03July/1000/1251/1251_bestpractices_TP026B.pdf

11. Gonzalo Génova Fuster, Miguel Fuentes Torres, María y Cruz Valiente Blázquez. *Evaluación comparativa de herramientas CASE para UML desde el punto de vista notacional.* **[En línea][Consultado el: 15 de marzo de 2010]** http://www.ie.inf.uc3m.es/ggenova/pub-novatica2006b.pdf

12. Juan Manuel Cueva Lovelle. *Introducción a UML: Lenguaje para modelar objetos.***[En línea] [Consultado el: 20 de marzo de 2010]** http://gidis.ing.unlpam.edu.ar/downloads/pdfs/IntroduccionUML.PDF

13. Salvador Alemany Garrido. *Programación gráfica en C++ con Qt4*

[En línea: 21 de noviembre de 2009] [Consultado el: 22 de marzo de 2010] http://www.polinux.upv.es/drupal/files/IntroduccionQt.pdf

14. Julian Looser, Raphaël Grasset, Hartmut Seichter y Mark Billinghurst. *OSGART: A Pragmatic Approach to MR.* **[En línea] [Consultado el: 24 de marzo de 2010]** http://ismar06.org/data/1b-HITL.pdf

15. Paul Martz. *OpenSceneGraph Quick Start Guide: A Quick Introduction to the Cross-Platform Open Source Scene Graph API*. **[En línea] [Consultado el: 1 de abril de 2010]** http://www.lulu.com/items/volume_51/767000/767629/3/print/OSGQSG.pdf

16. Michael Haller, Werner Hartmann, Thomas Luckeneder y Jürgen Zauner. *Combining ARToolKit with Scene Graph Libraries.* **[En línea] [Consultado el: 5 de abril de 2010]** http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.61.7108&rep=rep1&type=pdf

17. Alfredo Reino Romero.*Introduccion a xml en castellano***. [En línea] [Consultado el: 6 de abril de 2010]** http://sunsite.unam.mx/archivos/xml/IntroXMLc.pdf

18. Jaime E. Villate. *Introducción al XML.* **[En línea] [Consultado el: 10 de abril de 2010]** http://quark.fe.up.pt/cursoxml/curso.pdf

19. Doug Tidwell y Cyber Evangelist.*Tutorial: Introduction to XML.* **[En línea] [Consultado el: 12 de abril de 2010]** http://dblab.ewha.ac.kr/hsyong/document/slide/xmlintro-ibm.pdf

20. Juan J. Martínez. *Programación en C++ (1).* **[En línea] [Consultado el: 15 de abril de 2010]** http://webcache.googleusercontent.com/search?q=cache:AoPjL62CNj4J:www.usebox.net/jjm/docencia/materiales/programacion_cpp1.pdf.gz+caracteristicas+de+C%2B%2B&cd=8&hl=es&ct=clnk&gl=cu

21. Ernesto Bascón Pantoja. *El patrón de diseño Modelo-Vista-Controlador (MVC) y su implementación en Java Swing.* **[En línea] [Consultado el: 10 de mayo de 2010]** http://www.ucbcba.edu.bo/Publicaciones/revistas/actanova/documentos/v2n4/v2.n4.bascon.pdf

22. Juan Pablo Canepa y Matías Recabarren. *Patrones de diseño: Modelo-Vista-Controlador (MVC).* **[En línea: Junio, 5 de 2008] [Consultado el: 5 de mayo de 2010]** http://svn.assembla.com/svn/ImageShockPro/1213061556_aaecheve_sec1_pos0.pdf

23. Daniel Mazzini. *Patrones de Diseño.* **[En línea] [Consultado el: 8 de mayo de 2010]** http://download.microsoft.com/download/d/1/a/d1af3504-adb7-40ce-a460-ac8ba4a8a044/patrones.ppt

24. Pablo Montovani y Alvaro Piñero. *PATRONES DE DISEÑO: Singleton – Factory method*. **[En línea] [Consultado el: 10 de mayo de 2010]** http://dc.exa.unrc.edu.ar/nuevodc/materias/sistemas/2007/Patrones/1181572570/Singleton-FactoryMethod1.ppt.

25. Juan Pablo Canepa y Matías Recabarren. *Patrones de diseño:Singleton, Fabrica, Fachada.* **[En línea] [Consultado el: 15 de mayo de 2010]** http://deadlyjim.googlecode.com/svn-history/r27/trunk/Examen/Clases/Clase25_handout.pdf

26. J. Baltasar García PerezSchofield. *Desarrollo rápido de aplicaciones: Patrones de Diseño.* **[En línea] [Consultado el: 15 de mayo de 2010]** http://trevinca.ei.uvigo.es/~jgarcia/cdRAD/transpas-patrones.pdf

27. Ernesto Pimentel. *Patrones de Diseño.* **[En línea] [Consultado el: 15 de mayo de 2010]** http://kuainasi.ciens.ucv.ve/ideas07/documentos/tutoriales/TUTORIALErnestoPimentel.pdf

28. Javier Garzás.*Patrones de Diseño.* **[En línea] [Consultado el: 18 de mayo de 2010]** http://www.kybele.etsii.urjc.es/docencia/IS4/2009-2010/Material/%5BIS4-0910%5DPatrones%20dise%C3%B1o%20OO.ParteI.pdf

29. Herrera, Mauricio; Navia, Andrés y Villalba, Yenny. *Bridge, Builder & Interpreter.* **[En línea] [Consultado el: 18 de mayo de 2010]** http://www.ldc.usb.ve/~mgoncalves/IS2/sd07/grupo6.pdf

30. *Patrón de diseño Singleton.* **[En línea] [Consultado el: 18 de mayo de 2010]** http://www.lsi.us.es/docencia/get.php?id=608

ANEXOS 1 [Casos de Uso Expandido]

Caso de Uso	**Importar la configuración de la escena.**
Actores	Usuario.
Resumen	El caso de uso inicia cuando el actor escoge la opción Importar Escena, especificando la escena que desea importar, finalizando así el caso de uso.
Precondiciones	Para poder importar una escena, es necesario que esta exista previamente.
Referencias	**RF 1, 2, 3.**
Prioridad	**Crítico.**
Poscondiciones	Se importa y visualiza la escena.

Flujo Normal de Eventos	
Sección "Principal"	
Acción del Actor	**Respuesta del Sistema**
1 El usuario accede al sistema y selecciona la opción Importar Escena.	1.1 El sistema le pide al usuario la ruta desde donde desea importar el fichero de configuración de la escena.
2 Especifica la ruta.	
3 Presiona el botón Aceptar.	3.1 El sistema verifica que el formato sea válido. 3.2 El sistema importa el fichero.
Prototipo de Interfaz	
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
3 El usuario decide cancelar la operación de importar escena.	3.1 El sistema muestra un mensaje de error de formato no valido.
Prototipo de Interfaz	

Caso de Uso	**Exportar la configuración de la escena.**
Actores	Usuario.
Resumen	El caso de uso inicia cuando el actor escoge la opción Exportar escena, especificando la extensión que tendrá el fichero y la dirección donde será guardada, finalizando así el caso de uso.
Precondiciones	
Referencias	**RF 4.**
Prioridad	**Crítico.**
Poscondiciones	Se salva la escena.

Flujo Normal de Eventos	
Sección "Principal"	
Acción del Actor	**Respuesta del Sistema**
1 El usuario accede al sistema y selecciona la opción Exportar Escena.	1.1 El sistema le pide al usuario la ruta donde desea que se guarde el fichero de la configuración de la escena y además la extensión que va a tener el mismo.
2 Especifica la ruta.	
3 Presiona el botón Aceptar.	3.1 El sistema guarda la escena. 3.2 El sistema limpia el espacio de trabajo.
Prototipo de Interfaz	
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
3 El usuario decide cancelar la operación de importar escena.	
Prototipo de Interfaz	

Caso de Uso	**Adicionar asociación.**
Actores	Usuario.
Resumen	El caso de uso inicia cuando el usuario escoge la opción Nueva en el menú Asociación, el sistema le muestra un formulario donde le brinda la posibilidad de crear una nueva relación de patrón y recurso virtual, al usuario realizar la selección el sistema lleva a cabo la asociación y se añade a la escena, finalizando así el caso de uso.
Precondiciones	La escena tiene que estar inicializada.
Referencias	**RF 3, 5.**
Prioridad	**Crítico.**
Poscondiciones	Se muestra en la escena una asociación patrón y recurso virtual.

Flujo Normal de Eventos	
Sección "Principal"	
Acción del Actor	**Respuesta del Sistema**
1 El usuario accede al sistema y escoge la opción Nueva en el menú Asociación.	1.1 El sistema le muestra al usuario un formulario para que seleccione el patrón y el recurso virtual que desea asociarle al mismo.
2 Selecciona el patrón y el recurso virtual.	
3 Presiona el botón Aceptar.	3.1 El sistema visualiza en la escena la asociación realizada.
Prototipo de Interfaz	
Flujos Alternos.	
Acción del Actor	**Respuesta del Sistema**
3 El usuario decide cancelar la operación de realizar una nueva asociación.	
Prototipo de Interfaz	

Caso de Uso	**Manipular Objetos.**
Actores	Usuario.
Resumen	El caso de uso inicia cuando el usuario accede al sistema seleccionando la opción Transformar Recurso, aquí se le da la posibilidad al usuario de escoger que tipo de transformación desea realizar, ya sea rotar, trasladar o escalar, luego el sistema responde a cada una de estas transformaciones finalizando así el caso de uso.
Precondiciones	Debe de estar seleccionado un recurso.
Referencias	**RF 6, 7.**
Prioridad	**Crítico.**
Poscondiciones	Es trasformado el recurso.

Flujo Normal de Eventos	
Sección "Principal"	
Acción del Actor	**Respuesta del Sistema**
1 El usuario accede al sistema y escoge el recurso al cual desea realizar la transformación.	
2 El usuario selecciona la opción Transformar Recursos en el menú Recursos.	2.1 El sistema le muestra al usuario varias opciones de transformación a escoger: a) Rotar. b) Escalar. c) Trasladar.

3 El usuario selecciona la opción deseada.	3.1 El sistema muestra el formulario correspondiente a la opción seleccionada. **A)** Si el usuario selecciona la opción Transformación de rotación, ir a la sección "Transformación de rotación". **B)** Si el usuario selecciona la opción Transformación de escala, ir a la sección "Transformación de escala". **C)** Si el usuario selecciona la opción Transformación de traslación, ir a la sección "Transformación de traslación".
Prototipo de Interfaz	
Flujo Normal de Eventos	
Sección 1 "Rotar**"**	
Acción del Actor	**Respuesta del Sistema**
	1 El sistema muestra un formulario donde le pide al usuario que introduzca los valores de rotación en las coordenadas x, y, z.
2 Introduce los valores de rotación.	
3 Presiona el botón Aceptar.	3.1 El sistema rota el recurso de acuerdo a los valores que se le especificaron. 3.2 El sistema muestra el recurso rotado.
Prototipo de Interfaz	
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
3 El usuario decide cancelar la operación de Transformación de rotación.	
Prototipo de Interfaz	

Flujo Normal de Eventos	
Sección 2 "Escalar**"**	
Acción del Actor	**Respuesta del Sistema**
	1 El sistema muestra un formulario donde le pide al usuario que introduzca los valores de transformación de escala en las coordenadas x, y, z.
2 Introduce los valores de escalado.	
3 Presiona el botón Aceptar.	3.1 El sistema realiza la transformación de escala al recurso, de acuerdo a los valores que se le especificaron. 3.2 El sistema muestra el recurso escalado.
Prototipo de Interfaz	
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
3 El usuario decide cancelar la operación de Transformación de Escala.	
Prototipo de Interfaz	
Flujo Normal de Eventos	
Sección 3 "Trasladar**"**	
Acción del Actor	**Respuesta del Sistema**
	1 El sistema muestra un formulario donde le pide al usuario que introduzca los valores de traslación en las coordenadas x, y, z.
2 Introduce los valores de traslación.	
3 Presiona el botón Aceptar.	3.1 El sistema realiza la transformación de rotación al recurso, de acuerdo a los valores que se le especificaron. 3.2 El sistema muestra recurso trasladado.

Prototipo de Interfaz	
Flujos Alternos 1	
Acción del Actor	**Respuesta del Sistema**
3 El usuario decide cancelar la operación de Transformación de Traslación.	
Prototipo de Interfaz	

Caso de Uso	**Inicializar escena de realidad aumentada.**
Actores	Usuario
Resumen	El caso de uso inicia cuando el usuario selecciona la opción Iniciar Escena RA a partir de un dispositivo de captura. Para la realización completa del mismo primeramente se inicializa el video, luego se inicializa el tracking, y por último se inicializa la Escena.
Precondiciones	Debe haberse inicializado el video, el tracking y la escena.
Referencias	**RF 8. Inicilizar Video, Iniciar Tracking, Inicializar Escena**
Prioridad	**Crítico.**
Poscondiciones	Se muestra una ventana de interfaz de usuario que visualiza el contenido del grafo de la escena que incluye la reproducción del video.
Flujo Normal de Eventos	

Sección “Principal”	
Acción del Actor	**Respuesta del Sistema**
1 El usuario accede al sistema y selecciona la opción Inicializar fuente de video, luego escoge la opción Dispositivo de captura.	1.1 El sistema Inicializa Video. (Ver caso de uso **Inicilizar Video**) 1.2 El sistema Inicializa Tracking.(Ver caso de uso **Inicilizar Tracking**) 1.3 El sistema Inicializa Escena.(Ver caso de uso **Inicilizar Escena**)
Prototipo de Interfaz	
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
	1.1 El sistema lanza una excepción. 1.1.1 Se le da tratamiento a la excepción lanzada. 1.1.2 El sistema muestra una ventana informando al usuario que no se ha podido realizar la acción.
	1.2 El sistema lanza una excepción. 1.2.1 Se le da tratamiento a la excepción lanzada. 1.2.2 El sistema muestra una ventana informando al usuario que no se ha podido realizar la acción.
	1.3 El sistema lanza una excepción. 1.3.1 Se le da tratamiento a la excepción lanzada. 1.3.2 El sistema muestra una ventana informando al usuario que no se ha podido realizar la acción.
Prototipo de Interfaz	

Caso de Uso	**Inicializar Video<Inclusión>**
Actores	Usuario
Resumen	El caso de uso comienza como un proceso interno de la aplicación cuando el usuario inicializa la escena de realidad aumentada, aquí se le especifica que la captura va a ser desde una cámara y que biblioteca debe utilizar para acceder al hardware de captura de video mediante la cual se obtiene la configuración de video, es decir la velocidad de captura, la resolución del video, espacio de color y compresión.
Precondiciones	Debe existir una cámara conectada a la computadora. Deben existir los archivos de la biblioteca de captura de video.
Referencias	
Prioridad	**Crítico.**
Poscondiciones	Se abre el flujo de video y queda listo para acceder a sus datos de video.

Flujo Normal de Eventos	
Sección "Principal"	
Acción del Actor	**Respuesta del Sistema**
1 Especifica captura de video desde la cámara y la biblioteca de captura a utilizar.	1.1 Cargar la biblioteca de captura de video. 1.2 Comprobar si la biblioteca se cargó en memoria con éxito. 1.3 Establecer configuración de video.
2 Selecciona la configuración de video.	2.1 Abrir el flujo de video.
Prototipo de Interfaz	

Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
	1.2 El sistema lanza una excepción. 1.2.1 Darle tratamiento a la excepción lanzada.
Prototipo de Interfaz	

Caso de Uso	**Iniciar Tracking<Inclusión>**
Actores	Usuario
Resumen	El caso de uso inicia al cargar la biblioteca de análisis de imágenes encargada de detectar los patrones que hay distribuidos en la imagen capturada de una escena con el fin de devolver por cada uno su información de transformación. Además establece los parámetros intrínsecos de la calibración de la cámara mediante la lectura de un archivo y se le establece la fuente de video de la que se va a alimentar.
Precondiciones	Deben existir los archivos de la biblioteca de reconocimiento de patrones(tracking). Debe existir el archivo de calibración de la cámara que especifica los parámetros intrínsecos de la cámara. Debe estar activo el flujo de video del cual se va a alimentar.
Referencias	
Prioridad	**Crítico.**
Poscondiciones	El módulo de reconocimiento queda activo y listo para realizar el análisis de los fotogramas del video en busca de los patrones y sus transformaciones.
Flujo Normal de Eventos	
Sección "Principal"	
Acción del Actor	**Respuesta del Sistema**

	1 Especifica biblioteca de reconocimiento de patrones.
	2 Carga la biblioteca de reconocimiento de patrones.
	3 Comprobar que la biblioteca fue cargada con éxito.
	4 Carga archivo de calibración de la cámara.
	5 Comprobar que la calibración se cargó con éxito.
	6 Establecer fuente de video.
Prototipo de Interfaz	
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
	3 Lanza una excepción . 3.1 Darle tratamiento a la excepción lanzada.
	5 Lanza una excepción. 5.1 Darle tratamiento a la excepción lanzada.
Prototipo de Interfaz	

Caso de Uso	**Inicializar Escena<Inclusión>**
Actores	Usuario
Resumen	El caso de uso inicia acondicionando una escena básica de realidad aumentada para comenzar a reproducir el video capturado desde la cámara.
Precondiciones	El flujo de video debe estar abierto. La biblioteca de reconocimiento de patrones (tracking) debe estar activa.

Referencias	
Prioridad	**Crítico.**
Poscondiciones	Se establece un grafo de escena que incluye las texturas del video de fondo que se va a reproducir, una cámara virtual con los mismo parámetros de la cámara real y se comienza a reproducir el video.

Flujo Normal de Eventos	
Sección "Principal"	
Acción del Actor	**Respuesta del Sistema**
	1 Crea un nodo raíz para el grafo de escena.
	2 Se crea otro nodo para contener en las texturas de los fotogramas del video que se está reproduciendo.
	3 Se le añade al nodo raíz de la escena un disparador de actualización del tracking(callBack) .
	4 Se adiciona al grafo de la escena un nodo de cámara virtual con la información de la cámara real del archivo de calibración y se añade al nodo de la escena.
	5 Se reproduce el video.
Prototipo de Interfaz	

Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
Prototipo de Interfaz	

GLOSARIO DE TÉRMINOS

3D: tres dimensiones.

Aumento: Proceso de insertar un elemento digital en una señal de video.

Compilar: Es el proceso por el cual se traducen programas en código fuente a programas ejecutables, se crea software finalizado.

Framework: estructura de soporte definida, en la cual otro proyecto de software puede ser organizado y desarrollado.

GLUT (OpenGL Utility Toolkit): Es un kit que permite el desarrollo de aplicaciones OpenGL.

Herramienta Case: Ingeniería de sistemas asistida por ordenador (Computer-Aided Systems Engineering - CASE) es la aplicación de tecnología informática a las actividades, las técnicas y las metodologías propias de desarrollo de sistemas. Su objetivo es automatizar o apoyar una o más fases del ciclo de vida del desarrollo de sistemas.

Live Feed (Video en vivo): Se refiere al método de distribución directa de una señal de video entre objetos filmados y un usuario final que la recibe en tiempo casi-real.

Motor Gráfico: componente principal de un video juego u otra aplicación interactiva con gráficos en tiempo real.

Multiplataforma: Es un término utilizado frecuentemente en informática para indicar la capacidad o características de poder funcionar o mantener una interoperabilidad de forma similar en diferentes sistemas operativos o plataformas.

Open Source: Código abierto. Manera de nombrar también a las aplicaciones desarrolladas bajo el amparo de un producto con licencia GPL.

OPENGL (Open Graphics Library)(Biblioteca de Gráficos Abierta): OpenGL es una especificación estándar que define una API multi-lenguaje y multi-plataforma para escribir aplicaciones que producen gráficos 3D, desarrollada originalmente por Silicon Graphics Incorporated (SGI).

Plataforma: Combinación de hardware y software usada para ejecutar aplicaciones.

Realidad Virtual: es un sistema o interfaz informático que genera entornos sintéticos en tiempo real, representación de las cosas a través de medios electrónicos o representaciones de la realidad, una realidad ilusoria, pues se trata de una realidad perceptiva sin soporte objetivo, existe sólo dentro del ordenador.

Source Code (código fuente): Es el código de programación computacional con el cual se desarrollan aplicaciones.

Threshold (Umbral): Es el método más simple de segmentación de imagen. A los pixeles individuales en una imagen de escala de grises de les asigna típicamente un valor de “1” si se asume como un pixel de “objeto” (más brillante que el fondo) mientras que para un pixel del “fondo” se le da un valor “0”. El controlador Threshold determinada el valor que se utilizará como criterio para realizar esta segmentación.

UML: Lenguaje Unificado de Modelado (por sus siglas en inglés UML, Unified Modelling Language) es el lenguaje de modelado de sistemas de software más conocido en la actualidad.

XML: Un metalenguaje extensible de etiquetas desarrollado.