Universidad de las Ciencias Informáticas

Facultad 5

COMPONENTE DE MANIPULACIÓN DE VIDEOS PARA LABORATORIOS VIRTUALES

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autor: Andrie Pérez Villamil

Tutores: MSc. Yoander Cabrera Díaz

MSc. Liudmila Pupo Peña

Ing. Yasmany Alfonso Monteagudo

La Habana 2012

"Año 54 de la Revolución"

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de este trabajo y autorizo a la Universidad de las Ciencias Informáticas para que haga el uso que estime pertinente con este trabajo.

Para que así conste firmo la presente a los ____ días del mes de _____ del año 2012.

_____

Andrie Pérez Villamil

Autor

_____

MSc. Yoander Cabrera Díaz

Tutor

_____

MSc. Liudmila Pupo Peña

Tutor

_____

Ing. Yasmany Alfonso Monteagudo

Tutor

DATOS DE CONTACTO

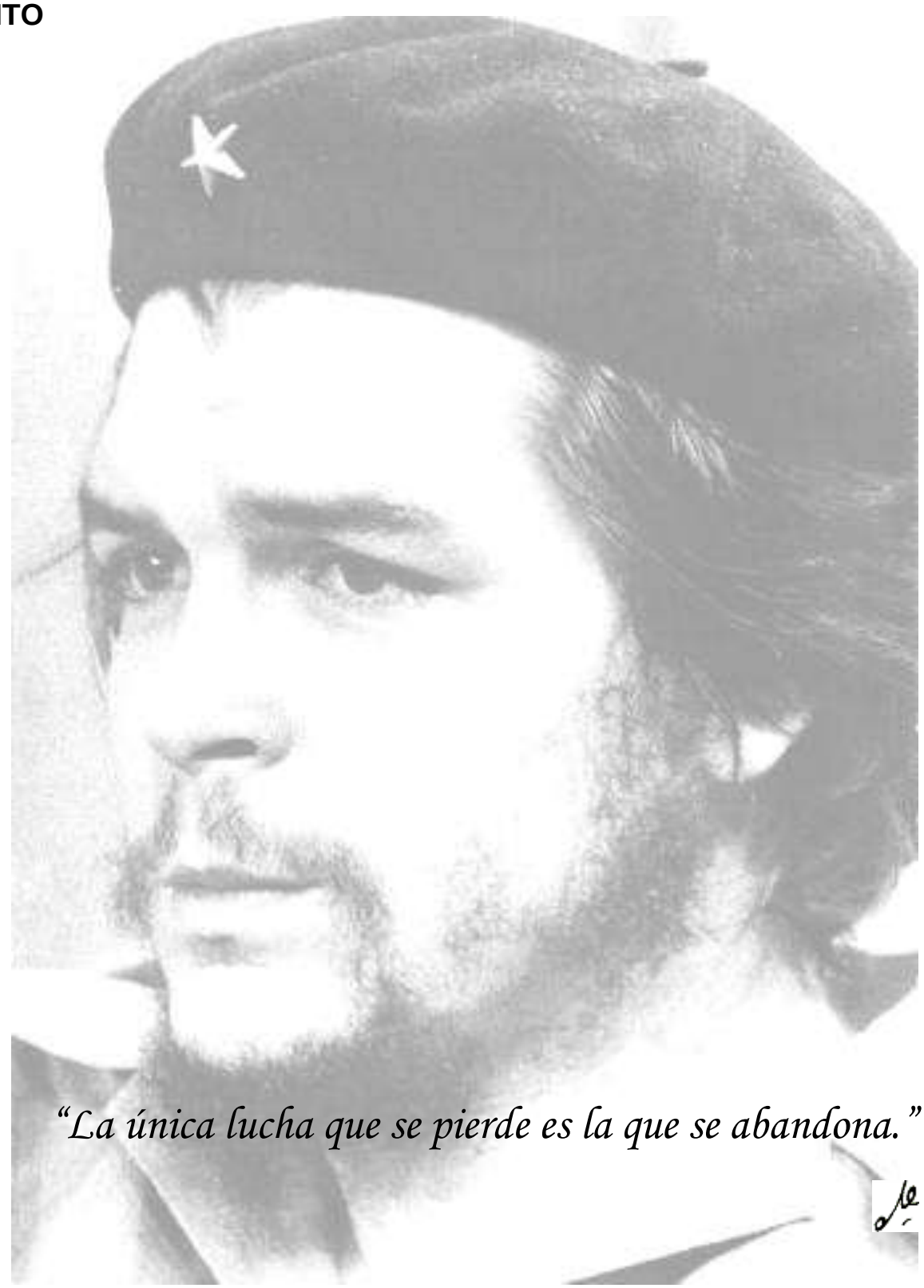
AUTOR:
Andrie Pérez Villamil.
Universidad de las Ciencias Informáticas.
La Habana, Cuba.
E-mail: avillamil@estudiantes.uci.cu

TUTOR:
MSc. Yoander Cabrera Díaz.
Universidad de las Ciencias Informáticas.
La Habana, Cuba.
E-mail: ycabrerad@uci.cu

TUTOR:
MSc. Liudmila Pupo Peña.
Universidad de las Ciencias Informáticas.
La Habana, Cuba.
E-mail: lpupo@uci.cu

TUTOR:
Ing. Yasmany Alfonso Monteagudo.
Universidad de las Ciencias Informáticas.
La Habana, Cuba.
E-mail: yamonteagudo@uci.cu

PENSAMIENTO



DEDICATORIA

A MI PADRE…

Por haberme criado como lo hizo en todo el tiempo que estuvo a mi lado. Forjándome día a día como un hombre. Haciéndome ver los problemas desde su raíz y enfrentarlos para buscarle una solución. Papi, eres el ejemplo de hombre que sigo. Siempre estarás a mi lado.

A MI MADRE…

Por ser como es, por estar siempre en las buenas y en las malas desde mi niñez. Por siempre brindarme una mano cuando lo he necesitado. Espero poder disfrutar más de tu compañía. Te quiero más de lo que te imaginas.

A ambos por traerme a este mundo para disfrutar de sus bellezas. Siempre los querré, no importa que pase, siempre estarán en mi corazón y en mi pensamiento.

A MIS HERMANOS…

Por estar conmigo desde pequeños. No son pocos los momentos de mi infancia que recuerdo con mucha emoción a su lado. Para ustedes un gran beso y sepan que me tienen para lo que necesiten. Siempre pueden contar conmigo.

A MI ABUELA ELISY…

Por ser la mejor abuela del mundo. Por darme todo su cariño en todos estos años.

A MERILEXY…

Por ser la mujer que me ama. La que está conmigo en cada segundo y sabe todo de mí. La que ha sabido soportarme y quererme tanto. La mayor parte de mi madurez mental te la debo a ti. Por ser mi mujer perfecta.

A LELE, A MERI Y A ALE…

Sepan ustedes que en su persona encontré dos padres y un hermano más. Que sé que me apoyarán en todos los pasos que dé en mi vida. A los tres los quiero tanto como a mis padres y hermanos a pesar de los pocos años que hemos convivido juntos. Para ustedes también es este premio.

A MI TÍA ESTHER…

Por estar siempre pendiente de mí. De mis logros estudiantiles. Por comportarse como una madre más conmigo.

A mis demás tíos y tías a los cuales quiero mucho.

A mis primos, con los cuales he compartido tanto.

A mis demás familiares que no se encuentran presentes pero que estuvieron conmigo en algún momento de estos pocos años que tengo de vida.

Para todos es este gran logro.

AGRADECIMIENTOS

Quisiera comenzar diciendo que cuando cursaba mi primer año en este centro no creía que podría con todo el estudio, con toda la programación. Menos mal que siempre tuve a alguien que me reanimara, que me diera esa fuerza para poder continuar la carrera, que me brindara su mano y su sonrisa.

Primeramente quiero agradecer a mi familia natal en general por todo el apoyo que me ha brindado en todo el transcurso de mi carrera.

A mi otra familia también. Gracias por su apoyo y su amor.

A mis tutores, los que me acogieron cuando me veía hundido y me sacaron a flote para lograr esta meta. Gracias por enseñarme que con empeño y sacrificio se obtienen logros.

A los viejos amigos de tantos años los cuales veo poco pero que sé que me apoyan.

A los nuevos amigos que he hecho aquí en esta universidad y que sé que podré contar muy poco de su compañía pero quedan los buenos recuerdos a su lado. Cinco años no pasan por gusto. Gracias por su amistad, su confianza y su apoyo.

A todo el claustro de profesores que me esperaba en las aulas, en los laboratorios o simplemente en el pasillo para enseñarme algo nuevo.

Y para finalizar, doy gracias a esta Revolución por pensar siempre en su pueblo, por darnos mucho de lo que necesitamos y por haber creado este centro que me forjó profesionalmente.

RESUMEN

Este trabajo surge dada la necesidad de incluir secuencias de videos en los laboratorios virtuales desarrollados por el Departamento de Visualización y Realidad Virtual para conocer los objetivos que se presentan en estas aplicaciones.

La investigación se centra en la manipulación de videos digitales utilizando las herramientas *OGRE* (*Object-Oriented Graphics Rendering Engine*) y Qt. Se propone como variante de solución el empleo del *plugin* basado en el formato de video *Ogg Theora* para el trabajo con videos en *OGRE* y el empleo de clases como *MediaObject* y *VideoWidget* para la utilización de videos con Qt.

Como resultado final se obtuvo un componente que brinda las funcionalidades básicas para el trabajo con videos digitales y puede ser utilizado en los laboratorios virtuales que se desarrollan y necesitan la incorporación de estos.

La utilización de estos videos dentro de las aplicaciones virtuales hoy en día presenta gran importancia, pues con su empleo, las aplicaciones se exentan de ser programas monótonos y simplificados para convertirse en aplicaciones agradables a la vista de los usuarios que interactúan con ellas.

PALABRAS CLAVE:

Video Digital, Manipulación de Videos, Realidad Virtual, Laboratorios Virtuales.

Contenido

INTRODUCCIÓN 1
CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA 6
1.1 Video digital 6
1.1.1 Características 7
1.2 Formatos de videos digitales 9
1.2.1 AVI 10
1.2.2 MPEG 10
1.2.3 MOV 11
1.2.4 FLV 12
1.2.5 RM 12
1.2.6 WMV 13
1.2.7 ASF 13
1.2.8 MP4 14
1.2.9 MKV 15
1.2.10 Ogg 15
1.2.11 Comparación de formatos investigados 17
1.3 Los videos digitales en aplicaciones de realidad virtual 17
1.4 Técnicas de manipulación de videos 19
1.4.1 Mediante OGRE 19
1.4.2 Mediante Qt 23
1.4.3 Consideraciones 24
1.5 Arquitectura de laboratorios virtuales 25
CAPÍTULO 2: DESCRIPCIÓN DE LA SOLUCIÓN 27
2.1 Soluciones técnicas 27
2.2 Solución general 28
2.3 Manipulación del video con OGRE 32
2.4 Manipulación del video con Qt 36
2.5 Herramientas y metodologías 38
CAPÍTULO 3: DISEÑO E IMPLEMENTACIÓN DE LA SOLUCIÓN 41

3.1 Modelo de Dominio 41
3.1.1 Glosario de Términos del Modelo de Dominio 42
3.2 Requerimientos del Sistema 42
3.2.1 Requisitos funcionales 42
3.2.2 Requisitos no funcionales 43
3.3 Modelo de Casos de Uso del Sistema 44
3.3.1 Definición del Actor del Sistema 44
3.3.2 Definición de los Casos de Uso del Sistema 44
3.3.3 Diagrama de Casos de Uso del Sistema 45
3.3.4 Descripción de los Casos de Uso del Sistema 45
3.4 Diseño 49
3.4.1 Diagrama de Clases de Diseño 50
3.4.2 Descripción de las Clases de Diseño 51
3.4.3 Diagramas de Secuencia 58
3.4.4 Patrones 59
3.5 Implementación 60
3.5.1 Diagrama de Componentes 60
CAPÍTULO 4: ANÁLISIS DE RESULTADOS 62
4.1 Resultados con OGRE 62
4.2 Resultados con Qt 63
CONCLUSIONES 65
RECOMENDACIONES 66
REFERENCIAS BIBLIOGRÁFICAS 67
GLOSARIO DE TÉRMINOS 70

Índice de Figuras

Figura # 1. Video de presentación de Need For Speed Most Wanted. 18
Figura # 2. Estructura de almacenamiento VideoMap. 29
Figura # 3. Representación del espacio de trabajo. 30
Figura # 4. Representación de los componentes de la solución y sus relaciones. 31
Figura # 5. Interacción entre los componentes de audio, video y lugar de representación. 37
Figura # 6. Modelo de dominio del componente de manipulación de videos. 41
Figura # 7. Diagrama de Casos de Uso del Sistema. 45
Figura # 8. Diagrama de Clases de Diseño. 50
Figura # 9. Diagrama de secuencia del caso de uso Cargar Videos. Sección Cargar video con OGRE. 58
Figura # 10. Diagrama de secuencia del caso de uso Cargar Videos. Sección Cargar video con Qt. 58
Figura # 11. Diagrama de secuencia del caso de uso Manipular Video. 58
Figura # 12. Diagrama de Componentes. 61
Figura # 13. Aplicación de prueba para manipular videos con OGRE. Utilización de videos en uno de los LV desarrollados. 63
Figura # 14. Botones definidos dentro de la aplicación de prueba para manipular los videos. 64
Figura # 15. Aplicación de prueba para manipular videos con Qt. Utilización de videos en lugares específicos dentro del área de trabajo. 64

Índice de Tablas

Tabla # 1. Comparación entre formatos de videos digitales. 17
Tabla # 2. Definición del actor del sistema. 44
Tabla # 3. Definición del caso de uso Cargar Videos. 44
Tabla # 4. Definición del caso de uso Manipular Video. 45
Tabla # 5. Descripción del caso de uso Cargar Videos. 47
Tabla # 6. Descripción del caso de uso Manipular Video. 49
Tabla # 7. Descripción de la clase VideoManager. 51
Tabla # 8. Descripción de la clase Video. 53
Tabla # 9. Descripción de la clase VideoOgre. 54
Tabla # 10. Descripción de la clase VideoQt. 55
Tabla # 11. Descripción de la clase Sound. 56
Tabla # 12. Descripción de la clase Vector3. 57
Tabla # 13. Descripción de la clase Extension. 57
Tabla # 14. Descripción de la clase Exception. 58

INTRODUCCIÓN

El auge de la realidad virtual (RV) ha estado precedido de un largo tiempo de intensa investigación. En la actualidad, la RV se plasma en una multiplicidad de sistemas que permiten que el usuario experimente "artificialmente" [1] en gran cantidad de áreas del conocimiento. Entre los objetivos de la misma está sumergir al usuario en un mundo generado por ordenador, homologando su entorno a la realidad y logrando establecer operatividad e interacción en tiempo real entre la aplicación y el hombre [2].

Sin duda los beneficios de la RV se han puesto en práctica a merced de la necesidad del hombre actual, evidenciándose su uso en esferas como la medicina, la ingeniería, el deporte, la educación, el entretenimiento entre otros.

Dentro de la esfera de la educación se ha incursionado en el desarrollo de laboratorios virtuales (LV). Estos pretenden aproximar a los estudiantes el ambiente de un laboratorio tradicional (LT). En los LV los experimentos se realizan paso a paso, siguiendo un procedimiento similar al de un LT: se visualizan instrumentos y fenómenos mediante objetos dinámicos, como pueden ser *applets* de *Java*, *flash*, imágenes, animaciones, simulación de procesos, entre otros [3] [4]. En ellos se obtienen resultados numéricos y gráficos, tratándose matemáticamente para la obtención de los objetivos perseguidos en la planificación docente de las asignaturas.

Además los LV acercan y facilitan a un mayor número de estudiantes la realización de experiencias. Reducen el coste de montaje y mantenimiento de los LT, siendo una alternativa barata y eficiente, donde el estudiante simula los fenómenos a estudiar como si los observase en un LT. Es una herramienta de autoaprendizaje, donde el alumno altera las variables de entrada, configura nuevos experimentos y aprende el manejo de instrumentos. La simulación en el LV permite obtener una visión más intuitiva de aquellos fenómenos que en su realización manual no aportan suficiente claridad gráfica. Se trata de utilizar la capacidad de procesamiento y cálculo del ordenador, incrementando la diversidad didáctica, como complemento eficaz de las metodologías más convencionales [3] [4].

Los estudiantes aprenden mediante prueba y error, sin miedo a sufrir o provocar un accidente, sin avergonzarse de realizar varias veces la misma práctica, ya que pueden repetirlas sin límite; sin temor a dañar alguna herramienta o equipo.

En la actualidad los videos digitales se utilizan como tendencia para la presentación en las aplicaciones gráficas, sobre todo en videojuegos y simuladores. La visualización de videos constituye sin duda una forma agradable y entretenida de conocer los objetivos de los ambientes virtuales que se presentan. Además es una forma más de cautivar al usuario introduciéndolo en las temáticas a tratar. Estos videos son utilizados en los simuladores sobre todo en las presentaciones y en ocasiones se emplean para dar algún tipo de instrucción. En ambos casos los videos constituyen una manera de introducir a los usuarios en distintos tipos de misiones o situaciones específicas [5] [6].

Actualmente en la Universidad de las Ciencias Informáticas (UCI), en el Centro de Informática Industrial (CEDIN) de la Facultad 5 existe el Departamento de Visualización y Realidad Virtual encargado de la investigación y el desarrollo de aplicaciones de este tipo. En este departamento se desarrollan LV basados en la herramienta *OGRE* (del inglés *Object-Oriented Graphics Rendering Engine*), motor gráfico libre y de código abierto escrito en lenguaje C++. *OGRE* está diseñado para ser usado de forma fácil e intuitiva por los desarrolladores en la producción de aplicaciones usando la aceleración por *hardware* de gráficos 3D [7]. Además se utiliza Qt como biblioteca multiplataforma utilizada para desarrollar aplicaciones con interfaz gráfica de usuario [8].

En el desarrollo de este tipo de aplicaciones se evidencia la **problemática** de que no existe una herramienta diseñada sobre *OGRE* y Qt que tenga integrada la manipulación de videos digitales, que proporcione a los desarrolladores de aplicaciones una interfaz fácil en intuitiva para ser usada, además que se ajuste al modo de visualización impuesto en las aplicaciones para *OGRE* o Qt en cada caso. Al mismo tiempo existe la necesidad de contar con una herramienta de este tipo que deberá ser capaz de manipular videos de baja ocupación de espacio en el disco duro.

Anteriormente en esta facultad se habían desarrollado las investigaciones tituladas “Biblioteca para la manipulación de videos digitales en Sistemas de Realidad Virtual” [5] y “Manipulación de videos digitales en Sistemas de Realidad Virtual para la herramienta gráfica SceneToolKit” [6]. Estas investigaciones propusieron el desarrollo de funcionalidades para la representación de videos digitales en sistemas de RV y surgieron a partir de la necesidad de incorporarle a la herramienta *SceneToolKit (STK)*, que se desarrolla en la UCI, funcionalidades para la manipulación de videos digitales.

Estas investigaciones se desarrollaron alrededor del formato de video digital *AVI* y fueron implementadas para la *STK*.

En su momento esta propuesta cumplió con su cometido, permitiendo la manipulación de videos digitales con formato *AVI* en dicha herramienta gráfica; pero presentaba algunas desventajas para ser usada en la creación de los LV que igualmente son desarrollados en el departamento. Entre las que se pueden mencionar:

- ✓ Los procesos de manipulación de videos utilizando la *STK* se realizan sobre archivos de videos digitales con formato *AVI*, el cual es un formato propietario desarrollado por *Microsoft.*
- ✓ Se utilizó como biblioteca gráfica *OpenGl* para la visualización de los videos en los entornos virtuales, mientras que los LV se desarrollan utilizando *OGRE,* que abstrae el uso de *OpenGl*.

Dada la necesidad de solucionar la situación antes expuesta; el **problema científico** a resolver queda planteado de la siguiente forma: ¿Cómo permitir la manipulación de videos digitales en laboratorios virtuales?

Para dar solución al problema se ha trazado como **objetivo de la investigación**: Desarrollar un componente para la manipulación de videos digitales empleando las herramientas *OGRE* y Qt.

Para cumplir dicho objetivo se planteó como **objeto de estudio** el proceso de manipulación de videos digitales en escenarios virtuales y como **campo de acción** la manipulación de videos con *OGRE* y Qt en laboratorios virtuales.

Para dar cumplimiento al objetivo, se plantearon las siguientes **tareas de investigación**:

- Análisis de las particularidades de los videos digitales para comprender sus características.
- Caracterización de los formatos de videos digitales más utilizados para familiarizarse con el tema.
- Análisis de las tendencias actuales sobre el uso de videos digitales en aplicaciones de realidad virtual.
- Identificación de las técnicas existentes de manipulación de videos digitales usando las herramientas *OGRE* y Qt para desarrollar la solución del trabajo.

- Identificación de los elementos del diseño y la arquitectura de los laboratorios virtuales para entender su funcionamiento.
- Diseño de una arquitectura lógica e implementación de un componente para la manipulación de videos digitales, compatible con la arquitectura de los laboratorios virtuales.
- Diseño e implementación de una aplicación que demuestre el funcionamiento del componente desarrollado.

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

Resumen, Introducción, cuatro Capítulos de contenido, Conclusiones, Recomendaciones, Referencias Bibliográficas y Glosario de Términos. A continuación se hace una breve descripción del contenido de cada uno de los capítulos:

Capítulo 1: Fundamentación Teórica

En este capítulo se hace mención a la importancia del uso de videos en aplicaciones de RV. Se definen algunos formatos de videos digitales y sus características. Se abordan las técnicas que utilizan las herramientas *OGRE* y Qt para manipular videos digitales y se identifican los elementos del diseño y la arquitectura de los laboratorios virtuales que se desarrollan en el centro.

Capítulo 2: Descripción de la Solución

En el transcurso del capítulo se presentan las soluciones técnicas para desarrollar la solución. Se plantean las acciones propuestas para dar cumplimiento a la construcción del componente encargado de manipular los videos digitales en los LV y se destacan las herramientas y metodologías utilizadas en el desarrollo del componente.

Capítulo 3: Diseño e Implementación de la Solución

El objetivo fundamental de este capítulo es mostrar el diseño de la solución que se propone en el capítulo anterior. Para esto se exponen una serie de diagramas y tablas que pretenden explicar detalladamente en términos de ingeniería de *software* el diseño de la aplicación que se quiere obtener. Además se realiza la captura de requisitos para posteriormente definir los casos de uso participantes así como los actores implicados en el proceso. Igualmente contará con la modelación de las clases y sus

interacciones así como los diagramas de secuencia para comprender el sistema que se desea desarrollar. Además se enuncian los patrones existentes en el diseño de clases y se modela el diagrama de componentes del sistema.

Capítulo 4: Análisis de Resultados

Durante el transcurso de este capítulo se tendrán en cuenta los resultados obtenidos durante la aplicación de la solución mediante la representación de imágenes que muestren estos resultados una vez probados en aplicaciones realizadas con *OGRE* y con Qt.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

Para darle solución al problema propuesto es necesario realizar un grupo de acciones investigativas con el objetivo de comprender el campo de acción que se plantea, así se tendrá una base de conocimiento sólida, la cual proporcionará la ayuda necesaria para poder adentrarse en el desarrollo de una solución que posibilite mitigar el problema existente. Para cumplir con este objetivo se divide el presente capítulo en epígrafes los cuales tratarán los puntos del proceso de revisión de esta investigación.

En un momento inicial se expondrá una panorámica sobre videos digitales en la cual quedarán plasmadas sus características generales. Seguidamente se presentan algunos formatos de videos digitales existentes. También se evidencia el empleo de videos digitales en aplicaciones de RV. Acto seguido se abordan las técnicas empleadas por las herramientas *OGRE* y Qt para realizar la manipulación de videos digitales. Finalmente se identifican los elementos del diseño y la arquitectura de los LV que se desarrollan en el departamento con el objetivo de que la solución sea fácilmente adaptable a esta arquitectura.

1.1 Video digital

El video es la reproducción en forma secuencial de imágenes, que al verse con una determinada velocidad y continuidad dan la sensación al ojo humano de apreciar el movimiento natural transmitiendo información al usuario. Además de la imagen, el otro componente esencial es el sonido, aunque no siempre tiene que estar presente [6]. Estos videos pueden ser tomados directamente del medio ambiente o entorno utilizando los equipos necesarios (videocámaras) o pueden ser creados directamente en una computadora mediante la secuencia de imágenes con la ayuda de programas especializados.

El video digital es la conversión de la imagen real al lenguaje binario (0 y 1) para que sea comprensible por las computadoras [9] y equipos similares. Generalmente estos videos digitales pueden ocupar mucho espacio de memoria por lo que se utilizan algoritmos específicos que se encargan de reducir el tamaño del archivo.

Estos algoritmos se denominan códecs (COdificación/DECodificación) y su función no es más que la de reducir el número de *bytes* que ocupa un archivo de video [10].

En el formato de video analógico para dar la sensación de movimiento se tratan de visualizar más de 24 imágenes por segundo. Si se quisiera hacer esto mismo para el formato digital implicaría que si se usaran imágenes con formato imagen de mapa de bits (.BMP), el tamaño del fichero del video digital sería muy grande. Las soluciones más factibles para contrarrestar esta preocupación serían la compresión de *bits* de estas imágenes reduciendo el espectro para solo transmitir las diferencias entre una imagen y la siguiente; y la reducción del tamaño de visualización [11].

En el momento de incluir un video a una aplicación se deben tener en cuenta determinadas características para que el propio video cumpla el objetivo que se aspira ante el usuario que interactúa con la aplicación. La buena selección de los parámetros en las siguientes características conlleva a un mejor aprovechamiento del video dentro de la aplicación.

1.1.1 Características

- **Tamaño del video (ancho x alto)**

El tamaño del video define las dimensiones del ancho y la altura de la imagen. Mientras mayor sea la resolución de la imagen más definición tendrá. Si la resolución es escasa, al ser ampliada se verán "cuadrados" que, mientras más se amplíe la imagen más grande se verán. A este efecto se le conoce como efecto de pixelación o *aliasing*.

Todas las imágenes digitales están compuestas por puntos (píxeles) donde cada punto es la parte más pequeña que un monitor es capaz de representar y ese punto representa un solo color. Por ejemplo: al aumentar una imagen de resolución 640x480 píxeles hasta 800x600 píxeles el ordenador necesita 160x120 píxeles que tiene que inventar pues no están en la imagen original. Aunque mediante técnicas de interpolación el ordenador puede calcular el color más probable para esos píxeles de "relleno" es evidente que cuanto más se amplíe, mayor será el número de píxeles inventados y la imagen se corresponderá menos con la original [12].

- **Flujo de datos (*bit rate*) (Kb/s)**

El flujo de datos de un video es la cantidad de información por segundo que se lee del archivo de video para reproducirlo. En semejanza con el tamaño de imagen, a mayor flujo de datos, mejor calidad de la

imagen. En muchas ocasiones el flujo de datos es más importante que el tamaño y capturas de gran tamaño, pero con poco flujo de datos, pueden llegar a tener mala calidad. Esta característica del video presenta dos variantes en la que puede ser utilizada: el flujo de datos constante y el flujo de datos variable [12].

- ✓ Flujo de datos constante (*CBR - Constant Bit Rate*)

En escenas con poco movimiento y pocos cambios de imagen entre fotogramas, aunque el flujo de datos sea escaso no se tendrán problemas de calidad. No siendo así en las escenas de acción donde los fotogramas son muy diferentes entre sí.

- ✓ Flujo de datos variable (*VBR - Variable Bit Rate*)

No se podrá predecir cuál será el tamaño final exacto del archivo. Todo depende de la complejidad del video, puesto que el flujo de datos varía dependiendo de la complejidad de las imágenes a comprimir.

- **Cuadros por segundo (*FPS - Frames Per Second*)**

Un video resulta de la exposición de imágenes o fotogramas uno detrás de otro. Un parámetro de calidad del video es el número de fotogramas por segundo que muestra durante su reproducción. Este valor oscila entre 15 y 30 [10]. Para distintos tipos de videos están establecidos los valores de esta propiedad [12]:

- ✓ Dibujos animados: 15 fps
- ✓ Cine: 24 fps
- ✓ Televisión PAL: 25 fps
- ✓ Televisión NTSC: 29'97 fps

- **Proporción o Relación de aspecto (*aspect ratio*)**

Es la proporción entre la anchura y altura de un video. Cuando se reproduce un video se suele mantener por defecto esta proporción para evitar deformación de las imágenes. Es habitual una relación 4:3 para los videos domésticos, mientras que en DVD se suele trabajar con proporción de 16:9 [10].

Los televisores actuales son 4:3 ó 16:9 si se divide el televisor con proporción 4:3 en 12 cuadrados iguales, se tendrían 4 cuadros de largo por 3 de alto; mientras que en un televisor 16:9 dividido imaginariamente en 144 partes, tendría 16 cuadros de largo por 9 de alto. Esta es la relación de aspecto para video analógico que se forma a partir de líneas horizontales en los televisores (625 para PAL, 525 para NTSC).

Para video no procedente de videocámaras lo único que hay que seguir es el tamaño del video. Distintos tamaños, por ejemplo 720x576 o 720x480, dan lugar a la misma proporción (4:3). Para mantener una relación de aspecto correctamente el tamaño de las dos dimensiones ha de ser múltiplo de 16 [12].

- **Tecnología de *Streaming***

La tecnología de *streaming* se utiliza para optimizar la descarga y reproducción de los archivos multimedia en la navegación por Internet desde el servidor remoto al cliente local. Esta tecnología funciona de forma tal que el reproductor cliente conecta con el servidor remoto y este comienza a enviarle el archivo que es almacenado en un *buffer* que crea el cliente. Al tener una pequeña fracción inicial del archivo en el *buffer* el reproductor cliente comienza a mostrarlo mientras continúa en segundo plano con el resto de la descarga.

Si la conexión experimenta ligeros descensos de velocidad durante la reproducción, el cliente podría seguir mostrando el contenido consumiendo la información almacenada en el *buffer*, deteniéndose la reproducción si llegara a consumirse toda la información del *buffer* hasta que se volviera a llenar [10].

1.2 Formatos de videos digitales

La tecnología de video fue desarrollada por primera vez para los sistemas de televisión que utilizan formatos de video analógico, pero ha derivado en muchos formatos para permitir la grabación de video de los consumidores y que además pueda ser visto a través de Internet [11].

Un formato de video es usado para agrupar distintos tipos de información, generalmente video y sonido en un único archivo [13]. Actualmente existen varios formatos para videos digitales como: *AVI*, *MPEG*, *MOV*, *WMV*, *FLV*, *RM*, *ASF*, *MP4*, *MKV*, *Ogg*, entre otros. A continuación se presentará una descripción de algunos de los formatos más utilizados.

1.2.1 AVI

Audio y Video Intercalado es un formato contenedor multimedia. Desarrollado por *Microsoft* en 1992, es el formato más común para los datos de audio y video en una computadora [14]. El archivo *AVI* puede llevar cualquier códec o resolución [15]. Puede contener video con una calidad excelente, sin embargo el peso del archivo resulta siempre muy elevado. Admite distintos códecs de compresión. Es ideal para guardar videos originales que han sido capturados de la cámara digital. No es recomendable publicarlos en Internet por su enorme peso [10].

AVI es en realidad un tipo de *RIFF* (*Resource Interchange File Format*) que divide los datos de un archivo en trozos o bloques donde cada paquete se identifica mediante una etiqueta *FourCC*.

En un archivo *AVI*, el primer bloque es el encabezado que contiene los metadatos sobre el video (su anchura, altura, velocidad de cuadro y así sucesivamente), el segundo fragmento contiene el audio real y datos de video que componen la película real y el tercer y último fragmento las direcciones físicas de los fragmentos de datos en el archivo. Las dos primeras partes son obligatorias, mientras que la tercera es opcional.

La variedad de códecs utilizados en los archivos con frecuencia se convierte en un problema. Esto es porque los códecs particulares usados en la producción de los archivos *AVI* pueden ser diferentes de los códecs instalados en el equipo utilizado en la reproducción, haciendo así la reproducción errónea. A pesar de este inconveniente, que puede ser superado mediante la instalación de los códecs adecuados, los archivos *AVI* siguen siendo populares [14].

1.2.2 MPEG

Del inglés *Moving Pictures Expert Group* (Grupo de Expertos en Imagen en Movimiento) es un formato estándar para la compresión de video digital. Son archivos de extensión *.MPG ó *.MPEG. Admite distintos tipos de códecs de compresión en dependencia de su utilización [10].

Existen diferentes estándares de *MPEG*. Algunos de estos son:

MPEG-1: Establecido en 1991, se diseñó para introducir video en un *CD-ROM* [15]. Es el estándar de codificación de imágenes en movimiento y audio asociado para medios de almacenamiento digital de hasta aproximadamente 1,5 *Mbit/s* [16] y resolución a 352x240. Se usa para videoconferencias. Si es usado a mayor velocidad, es capaz de dar más calidad [15].

MPEG-2: Establecido en 1994 para ofrecer mayor calidad con mayor ancho de banda (típicamente de 3 a 10 *Mbit/s*). En esa banda, proporciona 720x486 píxeles de resolución, es decir, calidad TV. Ofrece compatibilidad con *MPEG-1* [15]. Es un estándar que actualmente posee nueve partes. Las tres primeras partes de *MPEG-2* han alcanzado el estado de norma internacional [16].

MPEG-3: Fue una propuesta de estándar para la TV de alta resolución, pero como se ha demostrado que *MPEG-2* con mayor ancho de banda cumple con este cometido se ha abandonado [15].

MPEG-4: Introducido a finales del 1998, toma muchas de las características de *MPEG-1* y *MPEG-2*, así como de otros estándares relacionados, tales como soporte de *VRML* (*Virtual Reality Modeling Language*) extendido para visualización 3D, archivos compuestos en orientación a objetos (incluyendo objetos audio, video y *VRML*) y soporte para la gestión de derechos digitales externos [15].

Es el estándar para multimedia en webs móviles y fijas. Se basa en el éxito probado de tres campos: La televisión digital, las aplicaciones gráficas interactivas y la multimedia interactiva. Además proporciona los elementos estandarizados tecnológicos que permiten la integración de los paradigmas de acceso a la producción, distribución y el contenido de los tres campos [16].

1.2.3 MOV

MOV es una extensión de archivo utilizado por los archivos de *QuickTime*. El formato fue creado por *Apple Computer* para trabajar con archivos multimedia. Aunque los archivos *MOV* se encuentran muy a menudo en la red, para reproducirlos en un equipo con *Windows* es necesario instalar un componente adicional o ser convertidos a otro formato [17]. Utiliza un códec propio que evoluciona en versiones con bastante rapidez. Este tipo de archivos también pueden tener extensión *.QT. Es ideal para publicar videos en Internet por su razonable calidad/peso y admite *streaming* [10].

Es un formato contenedor y pueden contener video, animaciones, gráficos, contenido en 3D o subtítulos. Presenta la ventaja de contener referencias a los datos del archivo [17] o una referencia a los datos ubicados en otro archivo [18], lo que facilita su modificación.

1.2.4 FLV

Es un formato que utiliza el reproductor *Adobe Flash* para visualizar video en Internet. Utiliza el códec *Sorenson Spark* y el códec *On2 VP6*. Las versiones públicas más recientes de *Flash Player* también apoyan video *H.264* y audio *AAC* [10] [18]. Son archivos de extensión *.FLV utilizados para entregar el video a través de Internet usando *Adobe Flash Player* versiones 6-10. Los contenidos *FLV* pueden ser incrustados dentro de archivos *SWF* [18].

Al visualizarse a través del reproductor de *Flash* es accesible desde la mayoría de los sistemas operativos y navegadores web. Se ha establecido rápidamente como el formato de elección para el video integrado en la web. Los usuarios notables del formato *Flash Video* incluyen *YouTube*, *Hulu*, *Google Video*, *Yahoo Video*, *Metacafe*, *Reuters*, y muchos otros proveedores de noticias [18]. Permite configurar distintos parámetros del video para conseguir una aceptable calidad/peso y admite *streaming* [10].

1.2.5 RM

Real Media es un formato contenedor multimedia creado por *RealNetworks*. Se suele utilizar en conjunción con *RealVideo* y *RealAudio*, y es popular para la transmisión de contenido a través de Internet. Normalmente estas corrientes están en *CBR* [19]. Utiliza un códec propio para comprimir el audio. Este tipo de archivos tiene extensión *.RM y *.RAM. Se visualiza con un reproductor específico: *Real Player*. Se puede utilizar para publicar videos en Internet ya que admite *streaming* y presenta una aceptable relación calidad/peso [10].

Recientemente, *RealNetworks* ha desarrollado un nuevo contenedor para *VBR* llamado *RealMedia Variable Bitrate* y son archivos con extensión (*.RMVB) [19].

1.2.6 WMV

Desarrollado por *Microsoft,* utiliza el códec *MPEG-4* para la compresión de video. También puede tener extensión *.ASF. Sólo se puede visualizar con una versión actualizada de *Windows Media 7* o superior. Es ideal para publicar videos en Internet por su razonable calidad/peso. Además admite *streaming* [10].

El aspecto más importante de un archivo *WMV* es que permite que los archivos de video grandes puedan ser comprimidos manteniendo su calidad. Ellos pueden ser de cualquier tamaño y pueden ser comprimidos para adaptarse a cualquier ancho de banda. Muchas veces se encapsulan en un formato contenedor *ASF* (*Advanced Systems Format*). Aunque en otros casos, un archivo *WMV* también se puede almacenar en un formato contenedor como lo es *AVI* o *Matroska*.

Los archivos *WMV* pueden ser reproducidos en el *Microsoft Windows Media* disponible tanto para las plataformas *Microsoft Windows* y *Macintosh*. También pueden ser reproducidos en los dos sistemas a través de navegadores web que tienen *plugins* de *Windows Media Player*. Otras plataformas como *Linux* usan *FFmpeg* para implementar los códecs de *WMV* [14].

1.2.7 ASF

Advanced System Format es el formato propietario de *Microsoft* para contener audio y video, especialmente diseñado para transmisión por Internet bajo demanda. *ASF* es parte del encuadre de *Windows Media*. La tecnología que utiliza no tiene una resolución fija, es compatible con videos de alta y baja calidad y es compatible con diferentes anchos de banda [15].

Es un contenedor capaz de almacenar audio, texto, video, gráficos, animaciones y contenido comprimido con una gran variedad de códecs [14]. Este formato no especifica cómo debe codificarse el audio o el video, en su lugar solo especifica la estructura de la pista de audio/video. Esto quiere decir que los archivos *ASF* pueden contener pistas comprimidas con cualquier códec y aun así seguirían siendo archivos en formato *ASF*. Los formatos más comunes contenidos por *ASF* son *Windows Media Audio* (*WMA*) y *Windows Media Video* (*WMV*) [20].

Un archivo de *ASF* se compone de tres secciones principales: 1- La cabecera, que contiene información básica como el tamaño del archivo, el códec utilizado, y el número de flujos de datos incluido. Los

metadatos también se encuentran en el encabezado y contienen información como el título del archivo, el nombre del álbum, el género de la canción (si es un archivo de audio) y el director del archivo (en caso de un archivo de video). 2- El objeto de datos, que contiene los datos de la secuencia en forma de paquetes. 3- El Índice, contiene una lista de pares de fotogramas clave que facilita a la aplicación reconocer los archivos.

Otra característica importante de *ASF* es su capacidad para ser leído incluso si el archivo completo no se descarga debido a algún problema de datos o error. Esto es posible gracias a los *GUID* (identificadores únicos globales), que se colocan al principio de cada sección, o un objeto, de un archivo *ASF*. Estos *GUID* permiten la transmisión de los objetos en cualquier orden sin que el archivo pierda legibilidad [14].

1.2.8 MP4

MP4 es un formato de archivo contenedor que forma parte del estándar *MPEG-4* parte 14. Se utiliza ampliamente para distribuir video y audio que cumplan el estándar *MPEG-4* (por ejemplo *H.264 AVC* para video o *AAC* para audio), pero también puede almacenar otro tipo de datos como subtítulos, información de capítulos, gráficos 2D y 3D e imágenes fijas. Permite asimismo realizar *streaming* a través de la red [21] [14].

Codifica los datos de audio y video optimizando su calidad de almacenamiento, codificación y distribución en redes. Permite un menor tiempo de descarga y puede ser utilizado para crear archivos de sonido completos.

Un archivo *MP4* utiliza esquemas de compresión independientes para audio y video. Dado que *MP4* es un formato contenedor, la información referente a la composición del archivo se encuentra en la cabecera de este, la cual especifica los tipos de objetos exactos en el contenedor. Los archivos con este formato a pesar de contener audio y video, tienen un tamaño muy pequeño lo que posibilita su envío a través de Internet a gran velocidad [14].

1.2.9 MKV

El formato *MKV* (*Matroska Video*) es un formato de archivos de video flexible, que se ha convertido rápidamente en la extensión de archivo preferida para videos de alta definición (*HD*) en Internet. Reconoce directamente características como pistas de audio alternativas, subtítulos en varios idiomas y puntos de capítulos, así como metadatos enriquecidos, tales como carátulas, valoraciones, descripciones y otros [22].

Es un formato de video totalmente libre licenciado bajo *GNU GPL-L*. Más precisamente, es un contenedor que permite contener video, sonido y subtítulos en el mismo archivo. Gracias al formato *Matroska*, se pueden llevar a cabo funciones de capitulación, crear menús, realizar búsquedas en el archivo, seleccionar una fuente de sonido o elegir un subtítulo [23].

Entre los principales objetivos del proyecto *Matroska* se encuentran crear y documentar un moderno, flexible y multiplataforma formato contenedor de audio y video. Establecer *Matroska* como la alternativa de código abierto a los contenedores existentes como *AVI, ASF, MOV, RM, MP4, MPGE*. Desarrollar un conjunto de herramientas para la creación, edición y aplicación de los archivos *Matroska*. Además de desarrollar bibliotecas y herramientas para desarrolladores de *software* para poder apoyar a *Matroska* en sus aplicaciones y poner en marcha una serie de filtros *DirectShow* para la reproducción y creación de archivos *Matroska* en sistemas operativos *Windows* [24].

Lo mejor de *Matroska* es que permite extraer los datos de una película en *Blu-Ray*, con un peso aproximado de *25Gb* y dejarla en un peso de entre *4Gb* a *10Gb*. Tiene la capacidad de almacenar distintos idiomas de audio y distintos lenguajes de subtítulos. Además permite menús y capítulos sin perder la excelente calidad *HD* [25].

1.2.10 Ogg

Es un formato contenedor multimedia desarrollado por Xiph.org. Al igual que con toda la tecnología desarrollada por esta fundación es un formato abierto, libre para que cualquiera lo use.

Como con la mayoría de los formatos contenedores, *Ogg* encapsula datos no comprimidos y permite la interpolación de los datos de audio y de video dentro de un solo formato convenientemente [14] [26].

Además de la encapsulación y el intercalado de múltiples flujos de datos (capítulos, subtítulos, etiquetas, y datos de usuario adicionales). *Ogg* ofrece la elaboración de paquetes, detección de errores y marcas periódicas de tiempo para realizar búsquedas [26]. Utiliza preferentemente los códecs de audio *Vorbis* y de video *Theora*, desarrollados y mantenidos por la propia fundación.

Los archivos terminados en la extensión ".ogg" pueden ser de cualquier tipo, audio o video, aunque existe la recomendación de renombrarlos con la extensión ".oga" para audio y ".ogv" para video. Ya que su uso está libre de patentes, varios códecs de *Ogg* han sido incluidos en muchos reproductores multimedia existiendo incluso filtros para reproducir los códecs *Ogg* en prácticamente cualquier reproductor que soporte *DirectShow*.

Este formato se consagra como el formato de elección para la difusión de video y audio por Internet gracias a las características anteriormente mencionadas. Con relación a esto La Fundación Mozilla plantea: "Creemos que *Theora* es la mejor ruta disponible para el video abierto, video verdaderamente libre en Internet. También creemos que puede ser mejorado en calidad, en rendimiento, y en calidad de ejecución…" [27].

A diferencia de los formatos propietarios que no tienen la documentación pública disponible el códec de video *Theora* es objeto de una especificación que está disponible para todos en cualquier momento sin restricciones. La naturaleza de código abierto de *Theora* hace muy improbable que simplemente va a desaparecer, lo que puede pasar con códecs propietarios una vez que sus desarrolladores deciden dejar su desarrollo [28].

Ejemplos de algunos juegos que utilizan el formato *Ogg* (*Theora* y *Vorbis*) [28].

- ✓ *Black Prophecy*: Utiliza *Theora* para los videos.
- ✓ *Heroes of Might and Magic V*: Utiliza *Vorbis* para el audio y *Theora* para los videos.
- ✓ *Chronicles of Riddick: Escape from Butcher Bay*: Utiliza *Theora* y *Vorbis* (audio y video en archivos separados).
- ✓ *Serious Sam 2*: Utiliza *Theora* para las escenas de corte.
- ✓ *SimWorld*: Utiliza *Theora* para los videos y *Vorbis* para el audio.
- ✓ *The Book of Unwritten Tales*: Utiliza *Theora* para las escenas de corte.

- ✓ *Legend of Crystal Valley*: Utiliza *Theora* para los videos y *Vorbis* para el audio.
- ✓ *World of Padman*: Utiliza *Theora* para el video introductorio.

1.2.11 Comparación de formatos investigados

Formatos	Características			
	Libre	**Código Abierto**	**Multiplataforma**	***Streaming***
AVI	no	no	si	no
MPGE	no	no	si	no
MOV	no	no	si	si
WMV	no	no	no	si
ASF	no	no	no	si
FLV	no	no	si	si
RM	si	no	si	si
MP4	no	no	si	no
MKV	si	si	si	si
Ogg	si	si	si	si

Tabla # 1. Comparación entre formatos de videos digitales.

En la tabla anterior se exponen características de interés de los formatos investigados para la realización de la solución. En esta comparación se aprecia que los formatos *Ogg* y *MKV* son dignos candidatos para desarrollar la solución pues ambos son formatos libres, de código abierto lo que facilita su empleo a los desarrolladores, se pueden emplear en distintos sistemas operativos y permiten *streaming*. Además poseen la característica de que los archivos de video digitales que presenten estos formatos pueden contener varios flujos de datos, poseen una muy buena calidad y son relativamente pequeños lo que da cumplimiento a parte de la necesidad planteada en la situación problémica.

1.3 Los videos digitales en aplicaciones de realidad virtual

Una práctica en aplicaciones de RV es la inserción de videos con el objetivo de brindarle mayor creatividad y un aspecto más acogedor al *software* que se pretende comercializar.

En los juegos, un uso común para el video es el de crear escenas de corte o video a pantalla completa. Se puede usar una escena de corte como presentación, como unión entre los niveles, o en los créditos. Escenas de corte pueden dar al juego dramatismo y sensación cinematográfica [29].

El uso de videos se evidencia en la mayoría de los videojuegos actuales y son utilizados en las escenas de presentación como es el caso de los juegos *Need For Speed* y *MVP*, de transición de misiones, por ejemplo en el juego *Proyecto IGI*. En otros casos se pueden generar de acuerdo a situaciones específicas como es el caso del videojuego de fútbol *FIFA* [5] [6] y otros juegos deportivos como *NBA* y *Pro Evolution Soccer*. Estos videos son capaces de influenciar al usuario atrayendo su atención y motivándolo a jugar cada día más.

En el video de presentación del juego *Need For Speed Most Wanted*, así como en otros juegos de este tipo, son representadas algunas de las acciones que se pueden realizar durante el juego como lo es el choque contra los demás vehículos de la vía y la activación del óxido nitroso para aumentar la velocidad.

Figura # 1. Video de presentación de Need For Speed Most Wanted.

También en muchos de los juegos deportivos la utilización del video tiene un gran peso para representar las maniobras que jugadores virtuales como Manny Ramírez, Cristiano Ronaldo y LeBron James pueden realizar dentro del terreno virtual. Esto además es una forma muy atractiva de llamar la atención de los consumidores de este tipo de *software* pues en ellos se presentan jugadores reales que pueden ser manipulados por el usuario.

Así mismo los videos pueden ser utilizados en simuladores para realizar la presentación del producto a los usuarios o para realizar las indicaciones necesarias en cada uno de los niveles. Por ejemplo: simuladores de auto como *VirtualGT* y de tiro como *KD* [5] [6].

Otros usos que se le pueden dar a los videos digitales es en aplicaciones de RV dedicadas al área de la medicina. En este sector podemos citar su uso para mejorar la sensopercepción en la evaluación y entrenamiento de funciones visuales de los usuarios [2] [30].

En relación al planteamiento anterior el proyecto Herramientas de Desarrollo para Sistemas de Visión Estereoscópica (HDSVE)[1] crea la herramienta *Graphics Library for Stereoscopic Vision* (*GLSVe*) la cual es una biblioteca estructurada en clases que permite representar objetos a distintas profundidades, que se observen por delante del monitor, en este, o detrás, según los paralajes. Esta herramienta está dedicada principalmente al estudio de funciones visuales, aunque también se puede utilizar en cualquier otro tipo de aplicación que necesite una representación tridimensional estereoscópica. A dicha herramienta se le ha incorporado la funcionalidad de "Visualización de video estereoscópico" en aras de ofrecerle mayores prestaciones.

Utilizando esta herramienta fueron creadas actividades cuyo objetivo es el estudio de la agudeza visual, estereopsis, convergencia y dominancia ocular. En tales actividades se hace indispensable el uso de videos con el objetivo de lograr la acomodación visual del paciente en la aplicación para adaptarse al modo de visualización presentado en los prototipos diseñados, también de esta forma se brinda información sobre las actividades que realizará [2] [30].

1.4 Técnicas de manipulación de videos

1.4.1 Mediante OGRE

OGRE por sí mismo no provee decodificación interna de video mediante sus bibliotecas. Para reproducir videos con *OGRE* es necesario extraer la información referente a las imágenes del video y utilizarlas como texturas que se irán escribiendo en tiempo real para permitir la sensación de movimiento [6] [31]. Existen algunos *plugins* escritos por la comunidad de *OGRE* que permiten la decodificación de varios códecs de video dentro de una textura de *OGRE*. Estos *plugins* son: *DirectShow*, *FFmpeg* y *Theora* [31].

[1] El proyecto *HDSVE* lo conformaban inicialmente un grupo de profesores de la Universidad de Oviedo (España). Como parte de un convenio de colaboración se ha comenzado a trabajar en conjunto con la Universidad de las Ciencias Informáticas (UCI), y para esta nueva línea de investigación colaboran médicos oftalmólogos del Hospital de Arriondas (España) y el Hospital Ramón Pando Ferrer (Cuba).

❖ **DirectShow Plugin (win32)**

Este *plugin* es desarrollado utilizando las bibliotecas de *DirectShow* y solamente se puede utilizar en las plataformas *Windows*. Permite usar un archivo de video como la textura de un material y es capaz de utilizar casi todos los archivos que se pueden reproducir en *Windows Media Player* con este objetivo [6] [31].

Presenta las funcionalidades necesarias que le permiten cargar un video y reproducirlo de esta forma. Dentro de estas podemos encontrar las funcionalidades *loadMovie()* encargada de cargar el video, *pauseMovie(), playMovie(), rewindMovie()* y *stopMovie()* encargadas de las acciones básicas de reproducción del video.

Tiene funcionalidades que permiten conocer algunos datos de interés, por ejemplo: verificar si un video se está reproduciendo o está pausado *isPlayingMovie()* o devolver la textura en la cual *OGRE* está mostrando el video *getMovieTexture()*.

Para lograr la visualización es necesario llamar constantemente a la función *updateMovieTexture()*, donde se actualiza la información de cada *frame* del video.

❖ **FFmpeg Plugin**

No se actualiza pues contiene muchos códecs (*MPEG1 & 2, MP3, DivX*), algunos de los cuales están patentados y puede ser riesgoso su uso pues supone el pago de los derechos de uso [31].

El proyecto *FFmpeg* es una colección de *software* libre que puede grabar, convertir y hacer *streaming* de audio y video. Este incluye la biblioteca de códecs denominada *libavcodec*. Está desarrollado en *GNU/Linux*, pero puede ser compilado en la mayoría de los sistemas operativos, incluyendo *Microsoft Windows*.

FFmpeg está liberado bajo una licencia *GNU Lesser General Public License 2.1*+ o *GNU General Public License 2*+ dependiendo de cuáles bibliotecas estén incluidas.

El proyecto está compuesto por [32]:

- ✓ *ffmpeg*: Herramienta de línea de comandos para convertir un video de un formato a otro. También puede capturar y codificar en tiempo real desde una tarjeta de televisión.
- ✓ *ffserver*: Servidor de *streaming* multimedia de emisiones en directo que soporta *HTTP* (la compatibilidad con *RTSP* está en desarrollo). Todavía no está en fase estable.
- ✓ *ffplay*: Reproductor multimedia basado en *SDL* y las bibliotecas *FFmpeg*.
- ✓ *libavcodec*: Biblioteca que contiene todos los códecs de *FFmpeg*. Muchos de ellos fueron desarrollados desde cero para asegurar una mayor eficiencia y un código altamente reutilizable.
- ✓ *libavformat*: Biblioteca que contiene los multiplexadores/demultiplexadores para los archivos contenedores multimedia.
- ✓ *libavutil*: Biblioteca de apoyo que contiene todas las rutinas comunes en las diferentes partes de *FFmpeg*.
- ✓ *libpostproc*: Biblioteca de funciones de postproceso de video.
- ✓ *libswscale*: Biblioteca de escalado de video. Es la biblioteca principal del proyecto *FFmpeg*, capaz de codificar/decodificar en varios formatos de audio y video, está desarrollada en C.

❖ **TheoraVideoPlugin**

Este *plugin* permite la reproducción de videos *Ogg Theora* en aplicaciones 3D mediante la extracción de los marcos o *frames* para ser copiados directamente en la textura.

El *plugin* es una *DLL* multihilo de forma tal que cada video se decodifica en un hilo por separado. Antes de renderizar un *frame*, *OGRE* chequea si el *plugin* de *Theora* contiene nuevos *frames* que no han sido renderizados y si es así, transfiere la información de los píxeles de ese fotograma a una textura [31].

La clase *TheoraVideoClip* es la encargada de manejar la reproducción del video. Cuando se crea un objeto de esta clase la función *execute()* se ejecuta en un hilo separado decodificando continuamente los *frames* de video y almacenándolos en el *buffer* de memoria.

Una vez comenzada la visualización del *frame* el contenido del *buffer* de memoria es transferido a la textura. Con esto el *buffer* se limpia permitiendo que sea reutilizado.

Los *frames* son decodificados y la información de los píxeles es almacenada en *buffers* de memoria *RGB*. Cuando llega el momento de mostrar el siguiente *frame* la función *blitFrameCheck()* se encarga de copiar el contenido del *frame buffer* a la textura. Mientras más fotogramas sean almacenados más suave será la reproducción.

Theora y otros códecs almacenan la información de los píxeles en formato *YUV*, lo cual es mejor para comprimir las imágenes que usar formato *RGB* pero las texturas donde se mostrarán estos videos solo trabajan con formato *RGB* por lo que es necesario realizar la conversión entre estos dos formatos.

A la hora de realizar esta conversión existen dos variantes:

- ✓ Decodificar *YUV* en el *plugin* por la *CPU*, adecuada para videos pequeños.
- ✓ Decodificar *YUV* en la *GPU* a través de *pixel shader*, adecuada para videos grandes.

Se puede utilizar una técnica u otra independientemente del uso que se le dará al video en la aplicación y mientras sea mayor la resolución del video más tiempo se necesitará para decodificar cada *frame*.

Por su parte la clase *TheoraVideoManager* es la encargada de manipular todas las instancias de la clase *TheoraVideoClip*.

Cuando una nueva textura de video es creada *OGRE* primeramente establece todas las opciones del material en la clase *TheoraVideoManager*. Luego el método *createDefinedTexture()* es llamado creando una nueva instancia de la clase *TheoraVideoClip*.

Presenta las funcionalidades básicas para manipular los videos digitales como *play(), pause(), stop()*.

Como se puede apreciar el *plugin* basado en el formato de video *Ogg* para la representación de videos en escenarios virtuales que empleen la herramienta *OGRE* es el más adecuado. Esta afirmación se sustenta en las características que presenta este *plugin* y el formato de video para el cual ha sido desarrollado. Pues este formato, como se expone en el epígrafe 1.2.10 *Ogg* referente al formato de video digital del mismo nombre, es un formato que es desarrollado por una amplia comunidad encargada de aportarle mejoras continuas a su producto. Es un formato multiplataforma lo cual posibilita su utilización en los disímiles sistemas operativos o plataformas. Presenta una estructura bien definida de

encapsulación de datos en el archivo de video lo cual facilita su manejo. Tanto el código del *plugin* como el del formato de video son abiertos y libres para que pueda ser utilizado por cualquier usuario sin las restricciones de pago y licencias.

1.4.2 Mediante Qt

Por su parte Qt también presenta variantes para el trabajo con videos. Clases como *VideoPlayer*, *MediaObject*, *VideoWidget* y *QAbstractVideoSurface* se encargan del procesamiento y reproducción de videos [33].

- **VideoPlayer**

Clase introducida desde la versión 4.4 de Qt que permite la reproducción de videos mediante las tareas básicas como cargar el video, reproducir, pausar, detener, además de establecer volumen y búsqueda. Para estas tareas se utilizan las funcionalidades *load()*, *play()*, *pause()*, *stop()*, *setVolume()* y *seek()* respectivamente. Adicionalmente presenta también funcionalidades que permiten conocer el estado del video, si se encuentra en reproducción *isPlaying()* o en pausa *isPaused()* entre otras.

- **MediaObject**

Esta clase fue introducida desde la versión 4.4 de Qt con el objetivo de reproducir archivos multimedia. Para lograr este objetivo brinda las funcionalidades básicas *play()*, *pause()* y *stop()* encargadas de reproducir, pausar y detener el objeto multimedia con el cual se esté trabajando. Mediante la función *setCurrentSource()* indicándole la dirección del archivo como parámetro es posible su carga para ser reproducido posteriormente. Además brinda una gran gama de funcionalidades posibles a utilizar como los trabajos con la lista de archivos cargados o la búsqueda mediante las funcionalidades *queue()* y *seek()* respectivamente.

- **VideoWidget**

La clase *VideoWidget* propone un *widget* que se utiliza para mostrar videos, incluida en Qt desde la versión 4.4. Esta clase permite trabajar con las características del video mostrado en el *widget*. Mediante las funcionalidades *aspectRatio()*, *brightness()*, *contrast()*, *hue()*, *saturation()*, *scaleMode()* son

modificadas las características proporción, brillo, contraste, tono, saturación y modo de escalado del video que es reproducido. Además permite establecer el modo de pantalla completa utilizando la función *setFullScreen().*

❖ **QAbstractVideoSurface**

La clase *QAbstractVideoSurface* es una clase base para las superficies de presentación de video. Define la interfaz estándar que los productores de video utilizan para interoperar con las superficies de presentación de videos. Esta clase fue introducida desde la versión 4.6 de Qt.

Dicha clase presenta la funcionalidad *start()* encargada de tomar un formato compatible con la superficie de video y habilitar dicha superficie para comenzar la reproducción de las imágenes. Presenta además la funcionalidad *present()* que se encarga de mostrar en la superficie los *frames* que va recibiendo. Así mismo la función *stop()* es la encargada de deshabilitar la superficie de video donde se muestran las imágenes y elimina las referencias existentes hacia los *buffers* que contienen la información de estas imágenes.

Presenta además funcionalidades encargadas de comprobar si el formato del video es el adecuado para poder representarlo en la superficie. Estas funcionalidades son: *supportedPixelFormats()*, *isFormatSupported()* y *nearestFormat()*.

1.4.3 Consideraciones

Para lograr el manejo de los videos digitales dentro de los LV que se pretendan realizar usando el componente que derivará de esta investigación es necesario tener en cuenta varios factores como son:

- ✓ El (los) formato(s) de videos digitales que se puedan manipular con las técnicas que finalmente se elijan.
- ✓ La posibilidad de uso sin restricciones de las técnicas elegidas, permitiendo la utilización de las mismas por cualquier desarrollador.
- ✓ La posibilidad de que el uso de estas técnicas conlleve el empleo de la solución final en distintas plataformas.

En este sentido el *plugin* para video con formato *Ogg Theora* (*TheoraVideoPlugin*) tiene la delantera cuando sea necesario manipular archivos de videos digitales utilizando *OGRE*.

Por otra parte, la biblioteca de clases Qt brinda la posibilidad de escoger de entre las clases caracterizadas en el epígrafe 1.4.2 las que el desarrollador estime convenientes para manipular los archivos de videos digitales.

1.5 Arquitectura de laboratorios virtuales

La arquitectura del *software* es la organización fundamental de un sistema formada por sus componentes, las relaciones entre ellos y el contexto en el que se implantarán, y los principios que orientan su diseño y evolución [34]. Se refiere a un grupo de abstracciones y patrones que brindan un esquema de referencia útil para guiar el desarrollo de *software* dentro de un sistema informático [35]. Además constituye la plataforma principal para satisfacer la mayor funcionalidad y requerimientos de desempeño de un sistema.

La arquitectura implementada para el desarrollo de LV en el departamento trata de convertir los procesos más comunes utilizados en todos los LV en componentes independientes de manera tal que permita ensamblar un cierto número de componentes para lograr una funcionalidad básica de un LV.

Este desarrollo basado en componentes propone una estrategia de implementación bien definida que representa una gran ayuda a la hora de construir aplicaciones muy parecidas permitiendo la reutilización de elementos.

Con esta estrategia se identifican procesos como la creación e integración de componentes a la arquitectura de *software*. De esta manera se asegura el desarrollo de varios LV de forma paralela y se cumplen las necesidades de uniformidad del departamento. De igual modo se garantiza la reducción de los tiempos de desarrollo de los LV.

Conclusiones parciales

Se investigó las características fundamentales de los videos digitales y sus formatos. Además fueron abordadas las técnicas de manipulación de videos empleadas por las herramientas *OGRE* y Qt lo cual

formará la base para la realización de la solución al problema planteado y que se describirá en el siguiente capítulo de este trabajo.

Igualmente se ejemplificó el uso de los videos digitales en las aplicaciones de RV y se caracterizó la arquitectura actual de los LV con el objetivo de que la solución sea fácilmente integrable a esta arquitectura.

CAPÍTULO 2: DESCRIPCIÓN DE LA SOLUCIÓN

El presente capítulo describe la solución propuesta de forma conceptual. Primeramente se eligen el formato de video y las técnicas de manipulación de videos para el desarrollo de la solución. Se describe la solución a través de las acciones desarrolladas para manipular los videos en cada una de las herramientas definidas: *OGRE* y Qt, así como la integración de estas dos soluciones en una general que da cumplimiento al objetivo de la investigación. Finalmente se plantean las herramientas y metodologías utilizadas para realizar la solución.

2.1 Soluciones técnicas

Durante la realización de la solución que a continuación se propone se define el término manipulación como las acciones que se podrán realizar sobre los videos que se quieran representar mediante la utilización del componente final. Estas acciones son:

- Reproducir el video.
- Pausar el video.
- Detener el video.
- Ocultar el video.
- Mostrar el video.
- Ajustar el volumen del video.
- Ajustar el tamaño del video.
- Ajustar la posición del video.

Como se evidencia en el capítulo 1 existe una amplia gama de formatos de videos digitales. Teniendo en cuenta las indicaciones del proyecto y luego de corroborar las ventajas que brinda frente a los demás formatos caracterizados, el formato de video digital que será utilizado para el desarrollo de la solución es el formato de video digital *Ogg*. Este formato presenta grandes ventajas que fueron descritas en los epígrafes 1.2.10 y 1.2.11.

Proponer la utilización del formato *Ogg* no restringe en ninguna medida la posibilidad de representar los videos en los laboratorios virtuales, si existe la necesidad de incluir algún video digital que presente otro

formato la solución inmediata sería convertir dicho video al formato *Ogg*. Esto podría realizarse utilizando la herramienta *Mobile Media Converter* [36].

Esta herramienta es libre, multiplataforma y presenta una sencilla interfaz gráfica de usuario. Provee un reproductor de audio gratuito y convertidor de video para la conversión de distintos formatos de audio y video como *MP3, WMA, Ogg, WAV, MPEG, AVI, WMV, FLV, MOV* entre otros. Permite editar los clips de audio y video para la creación de tonos o para eliminar las partes no deseadas de la imagen. Además realiza trabajos sobre los subtítulos incrustados e incorpora un copiador de *DVD* para ser transformados a cualquiera de los formatos soportados.

Además se define como técnica a emplear para manipular los videos digitales por la herramienta *OGRE* el *TheoraVideoPlugin* pues es el más adecuado para desarrollar la solución. Además es un *plugin* desarrollado especialmente para el trabajo con el formato de video digital *Ogg* anteriormente seleccionado y que presenta significativas ventajas. Desarrollado para ser utilizado en distintas plataformas y pone a disposición del desarrollador las funcionalidades necesarias para manipular los archivos de videos digitales.

Para manipular los videos digitales mediante Qt se eligen las clases *MediaObject* y *VideoWidget* del módulo *Phonon* el cual es el encargado del trabajo con archivos multimedia.

En este caso se eligen estas clases pues aunque la clase *VideoPlayer* presenta las funcionalidades básicas para manejar los videos y es una clase más compacta en este sentido en su composición presenta un objeto de tipo *MediaObject* encargado de las acciones que se realizan sobre los videos y un objeto *VideoWidget* que representa el *widget* necesario que proporciona Qt para representar archivos de video. Esta elección proporcionará un manejo independiente sobre cada uno de estos objetos y el empleo de forma directa de sus funcionalidades que en algún momento podrían ser de utilidad.

2.2 Solución general

Para facilitar el trabajo de los desarrolladores fue necesario definir dos grupos de acciones que se encargarán de manipular los archivos de videos digitales para poder representarlos de una u otra forma dependiendo del lugar que fuese definido para ello, como una textura para *OGRE* y en un *widget* para Qt.

Dentro de estos dos grupos existen acciones similares como son las relacionadas con la reproducción, la pausa, la detención, el ajuste del volumen, el ajuste del tamaño, el ajuste de la posición, el ocultamiento, la muestra y si la reproducción de un archivo de video digital ha terminado. Por lo que estas acciones similares se declaran de forma unánime y posteriormente son redefinidas en cada una de las soluciones según el lugar donde será representado el video.

Durante la utilización de los LV estos deberán tener previamente almacenados en memoria todos los archivos de videos digitales que serán empleados a lo largo de la experiencia de aprendizaje. Es por eso que se define una estructura de datos encargada de contener los videos para posteriormente realizar las acciones de manipulación sobre ellos a medida que sea necesario.

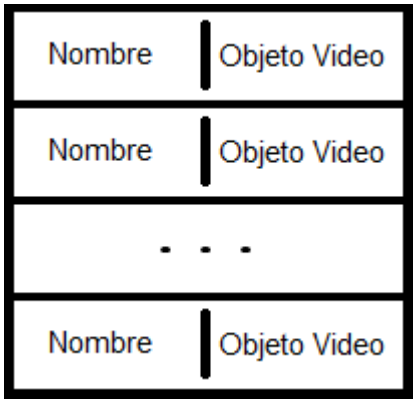


Figura # 2. Estructura de almacenamiento VideoMap.

Esta estructura de datos no es más que un mapa que contiene en su interior un nombre que representa el identificador de un video dentro del mapa y un objeto de video que será el manipulado cuando sea necesario.

Para insertar estos videos digitales dentro del mapa es necesario crear el objeto de video referente a cada archivo de video digital con los parámetros que se definen para cada caso dependiendo de la herramienta con la cual se quiera manipular, con *OGRE* o con Qt. En cualquiera de los dos casos es obligatorio indicar la dirección física donde se encuentra el video digital y el nombre identificador.

Posteriormente cuando se necesite manipular estos videos solamente se tiene que realizar una búsqueda dentro de la estructura de datos atendiendo al nombre indicado, devolviendo el objeto de video relacionado con este identificador y seguidamente indicar la acción de manipulación.

Además se pueden destruir todos los objetos de videos almacenados en el mapa.

Así mismo se define un tipo de datos de uso común en los atributos de tamaño y posición de los videos que se quieran representar. Este tipo de datos se define con los atributos necesarios, una coordenada (X, Y, Z); los cuales representarán: el ancho, el alto y la profundidad para el tamaño y la ubicación en los ejes (X, Y, Z) donde será mostrado el video en el espacio de trabajo.

Esto permite variar los parámetros de tamaño y posición previamente definidos cuando se construye un objeto de video mediante funcionalidades específicas para ello. Además la solución posibilita esconder o mostrar en cualquier momento la reproducción de un video que se encuentre en marcha.

Esta particularidad de la solución de almacenar varios archivos de videos digitales permite representar en el espacio de trabajo tantos videos como sean creados puesto que cada uno se crea con sus propias características.

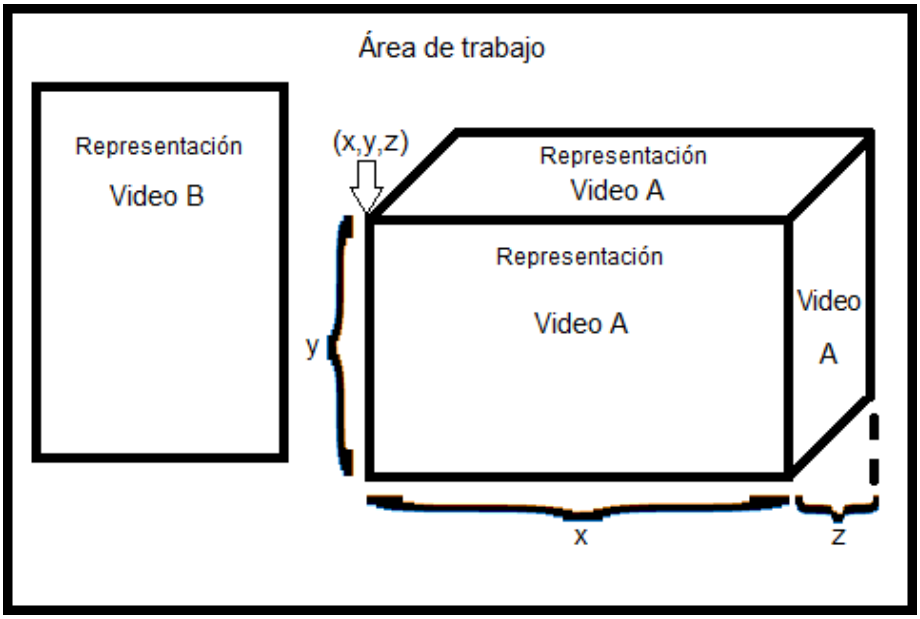


Figura # 3. Representación del espacio de trabajo.

Existe más de una funcionalidad que puede ser utilizada cuando se quiere variar el tamaño del video que se está reproduciendo. Esto se debe a que se puede elegir la proporción con la cual el video puede representarse acompañado al tamaño del área de representación para el mismo. Además se incluye la representación a pantalla completa.

Es necesario resaltar que cuando se quieran hacer transformaciones de tamaño y posición a un video que se está manipulando con las funcionalidades para un video que será representado en un *widget* de Qt el parámetro de la componente Z en la coordenada (X, Y, Z) se le indicará el valor 0.0 puesto que el *widget* no será transformado en esta dirección al ser un objeto de dos dimensiones (2D). Mientras que cuando se esté manipulando el video con las funcionalidades definidas para *OGRE* la componente Z tomará valores distintos de 0.0 cuando se represente el video en un objeto que presente tres dimensiones (3D) o se quiera posicionar en esta dirección.

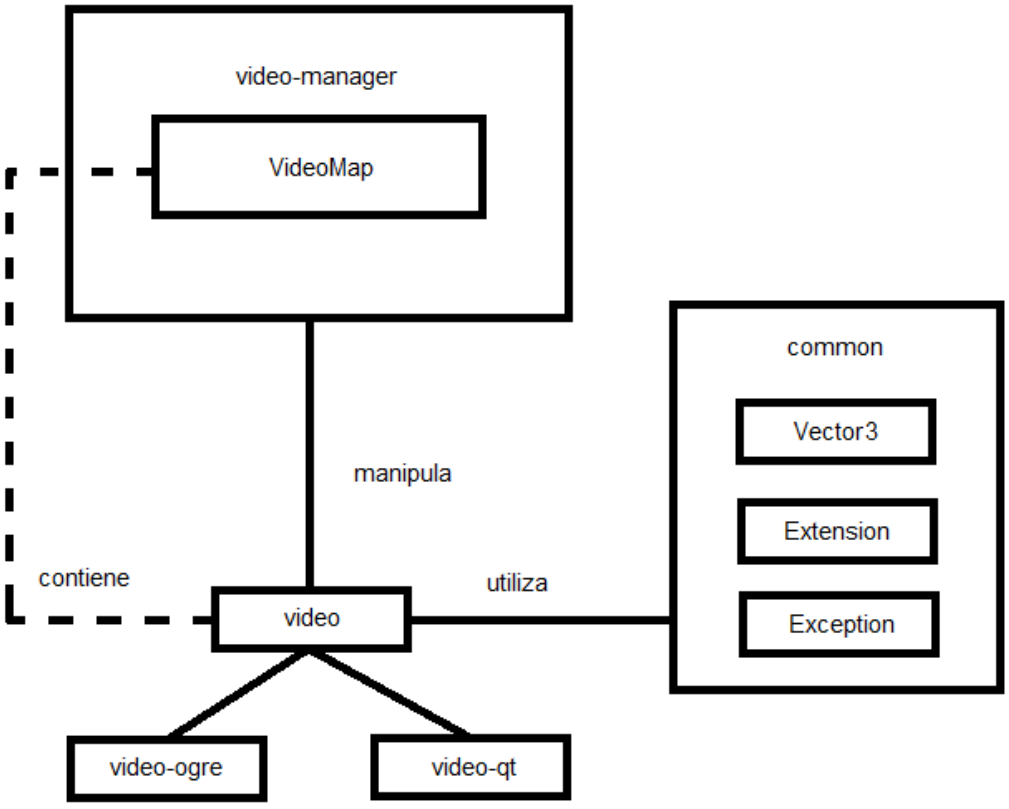


Figura # 4. Representación de los componentes de la solución y sus relaciones.

Otra de las características que presenta esta solución es que permite verificar el tipo de extensión de los archivos de videos digitales que se quieran utilizar en el momento que sean llamados para cargarse, para así comprobar que este sea el correcto y además se utiliza una clase dedicada al manejo de las excepciones.

Para comprender mejor lo anteriormente explicado se presentó la figura # 4 donde se puede observar la relación entre los componentes participantes en la solución que se propone. Se puede observar que *video-manager* manipula los videos previamente almacenados en su mapa. Estos videos pueden ser videos que serán manipulados utilizando las dos variantes indistintamente *video-ogre* y *video-qt* y que utilizan funcionalidades comunes que se encuentran en las clases definidas en el archivo *common*.

2.3 Manipulación del video con OGRE

Actualmente el trabajo con videos en *OGRE* es muy engorroso. Para facilitar las tareas de manipulación de videos que serán representados como textura en un modelo que se visualizará en el escenario de *OGRE*, se redefinen las funcionalidades heredadas desde la clase padre y que permitirán reproducir, pausar, detener y ajustar el volumen del video. Igualmente las concernientes a la posición, el tamaño del video, las definidas para visualizar o no el video convenientemente y saber cuándo finalizó la reproducción de un video.

Para poder manipular y visualizar videos digitales con formato *Ogg* en *OGRE* fue necesario utilizar, como se define anteriormente el *plugin* desarrollado por la comunidad de *OGRE* (*TheoraVideoPlugin*).

Para hacer uso de este elemento fue necesario descargar el proyecto del *plugin TheoraVideoPlugin*[2] y los proyectos para el formato *Ogg* y sus códecs *Theora* y *Vorbis*[3] y compilarlos para generar las bibliotecas necesarias. Además el *plugin* utiliza la biblioteca de tipos de portátiles de C++ *Ptypes*[4] la cual fue necesaria descargar también.

[2] *TheoraVideoPlugin*
http://www.ogre3d.org/addonforums/viewtopic.php?f=18&t=10619
[3] Dependencias (*Ogg, Theora, Vorbis*)
http://www.theora.org/downloads/
[4] Dependencias (*Ptypes*)
http://www.melikyan.com/ptypes/

El orden de compilación para estas dependencias es el siguiente:

- ✓ *Ogg*
- ✓ *Vorbis*
- ✓ *Theora*
- ✓ *Ptypes*
- ✓ *TheoraVideoPlugin*

Luego de compilar ya se encuentran disponibles las bibliotecas referentes al *plugin* para ser utilizadas en la confección de la solución las cuales son incluidas al proyecto.

Es necesario incluir en el fichero de configuración de recursos de *OGRE* la dirección donde se encuentren los archivos de videos digitales que se quieran emplear y definir un nuevo fichero nombrado *"Example_VIDEO.material"* para indicar los materiales que se aplicarán a cada una de las entidades de *OGRE* donde se visualizará un video. Este fichero será colocado en el directorio donde se encuentre instalado *OGRE* en el subdirectorio *"media\materials\scripts".*

Para poder representar más de un video dentro de una misma escena será necesario definir un material para cada archivo de video digital que quiera ser representado. La estructura interna del fichero *Example_VIDEO.material* estará definida de la siguiente manera y se tendrán secciones idénticas a la que a continuación se muestra con la única diferencia de que el nombre definido para cada sección (material) debe ser diferente pues cada una representará un archivo de video digital distinto a utilizar en la aplicación y que será mostrado sobre el cuerpo de la entidad a la cual se le asigne ese material:

```
material NOMBRE_DEL_MATERIAL
{
    technique
    {
        pass
        {
            cull_software none
```

```
                cull_hardware none
                lighting off
                texture_unit
                {
                }
            }
        }
}
```

Se define un objeto de la clase *TheoraVideoClip* y otro de la clase *TheoraVideoManager* para interactuar con el video que se necesite. Estas clases pertenecen al *plugin* seleccionado para la solución y que posibilita la representación de archivos de video con formato *Ogg* (*Theora*) y su manipulación en *OGRE*.

El video que se quiera mostrar será cargado mediante el argumento definido para la dirección física del archivo de video digital en cuanto se cree una instancia de un objeto video que será guardada en el *VideoMap* con el nombre identificador que se indique. Así mismo será necesario definir el material que cubrirá la entidad en la cual se representará el video y el nodo de escena al cual se le realizarán las transformaciones de tamaño (escalado) y posición.

En el caso en que el video se quiera representar ocupando toda el área de la ventana de renderizado de *OGRE* (pantalla completa) será necesario indicar un plano. El cual será construido como un *ManualObject* y será escalado con las dimensiones de esta ventana.

Finalmente será necesario indicar la dirección física del video al objeto de la clase *TheoraVideoManager* anteriormente creado y crear la textura definida que será mostrada sobre la entidad con el nuevo material. Esto se logra indicando el nombre "NOMBRE_DEL_MATERIAL" anteriormente definido con el objetivo de que se pueda visualizar el archivo de video.

Además se inicializa el objeto de la clase *TheoraVideoClip* con el objetivo de que obtenga la información referente a las imágenes del video que se quiera manipular.

Para la reproducción del sonido referente al video que se esté utilizando fue necesario definir un objeto *Sound* e implementar las funcionalidades necesarias de su clase pues el *TheoraVideoPlugin* solamente utiliza la información de las imágenes del archivo de video para ser representadas como textura en las entidades de *OGRE*.

Es por eso que al momento de cargar el archivo de video digital que quiera ser utilizado para obtener la información de sus imágenes, será necesario cargar nuevamente el mismo archivo para obtener su información sonora; la cual será reproducida junto a las imágenes en el momento de representar el video.

Para poder realizar esta acción se utiliza la biblioteca de sonido *OpenAL* la cual se encarga de extraer los datos de audio del archivo de video que se quiera utilizar y permite realizar las acciones de reproducción, pausa y detención de este.

Es necesario señalar que para obtener la reproducción del audio junto a las imágenes será necesario incluir las bibliotecas necesarias de *OpenAL* e inicializar el sistema de sonido mediante la funcionalidad `alutInit()` en la aplicación donde se quiera utilizar el componente que derivará de esta solución.

Es importante destacar que siempre el desarrollador deberá indicar el material que utilizará para cubrir la entidad donde representará el archivo de video digital empleando el nombre definido para dicho material (*"NOMBRE_DEL_MATERIAL"*) una vez creada esta entidad.

Para reproducir, pausar o detener un video utilizando las funcionalidades redefinidas es necesario verificar el estado en que se encuentra el video y en dependencia de cual sea se permite realizar estas acciones sobre el objeto *TheoraVideoClip*. Estas acciones internamente además de efectuar la acción correspondiente sobre el objeto anteriormente mencionado efectúan la acción correspondiente sobre el objeto de sonido que internamente se crea con el objetivo de que se desempeñen simultáneamente las acciones de representación de las imágenes y las acciones de reproducción del audio del archivo de video digital con el que se esté trabajando. Por ejemplo: una vez invocada la acción de reproducción sobre el *TheoraVideoClip* es necesario invocar la acción de reproducción sobre el objeto *Sound*.

El volumen del audio saliente durante la reproducción se podrá variar indicando cualquier valor al objeto entre 0.0 y 1.0.

Además para lograr las operaciones de manipulación referentes al tamaño es posible utilizar cualquiera de las funcionalidades heredadas según el resultado visual que el desarrollador quiera obtener. Para esto es necesario indicar el valor del tamaño definido para la entidad donde será representado el archivo de video digital luego de escoger entre las variantes disponibles y que a continuación se listan.

- ✓ Utilizar las funcionalidades para mostrar un video con tamaño definido dentro del área de trabajo en sus variantes:
  - Escoger la representación con una proporción de 4:3.
  - Escoger la representación con una proporción de 16:9.
  - Escoger la representación utilizando el área definida originalmente para la entidad donde se visualizará.

2.4 Manipulación del video con Qt

Para realizar los procesos de manipulación con el video que se representará en un *widget* de Qt es necesario redefinir las funcionalidades que son heredadas. Estas funciones principales son las necesarias para poder reproducir, pausar, detener y ajustar el volumen del video. Así como las necesarias para la posición, el tamaño del video y las indicadas para mostrar o esconder el video convenientemente y saber cuándo un video terminó de reproducirse.

Para implementar dichas funcionalidades se hace uso del módulo *Phonon* que proporciona la biblioteca Qt para el trabajo con archivos multimedia. Dicho módulo contiene clases como *MediaObject*, *VideoWidget*, *AudioOutput* que fueron empleadas para conformar la solución.

Los objetos de tipo *MediaObject* y *AudioOutput* son los encargados de manipular las informaciones referentes a las imágenes de video y al audio respectivamente.

Para representar el video en el espacio de trabajo es necesario indicarle el *widget* en el cual se visualizará. Este *widget* es un objeto de la clase *VideoWidget* anteriormente mencionada y que es la encargada de proporcionar el elemento específico para mostrar las secuencias de imágenes del archivo de video digital previamente cargado.

Siempre será necesario cargar el video indicado en el argumento definido para la dirección física del archivo de video digital cuando se quiera construir un video de este tipo, indicar un nombre identificador y un *widget* donde será representado. Además es necesario indicar los valores referentes a la posición y el tamaño donde se quiera representar el dentro del área de trabajo.

Por último será necesario realizar las uniones correspondientes para la salida sincronizada del audio y el video y del video con el *widget* donde será representado como se muestra en la figura # 5.

Para reproducir, pausar o detener un video previamente cargado se verifica el estado en que se encuentra y en dependencia de cual sea se permite realizar estas acciones sobre el objeto *MediaObject*.

Además es posible ajustar el volumen del audio saliente durante la reproducción indicando un valor al objeto *AudioOutput* entre 0.0 y 1.0.

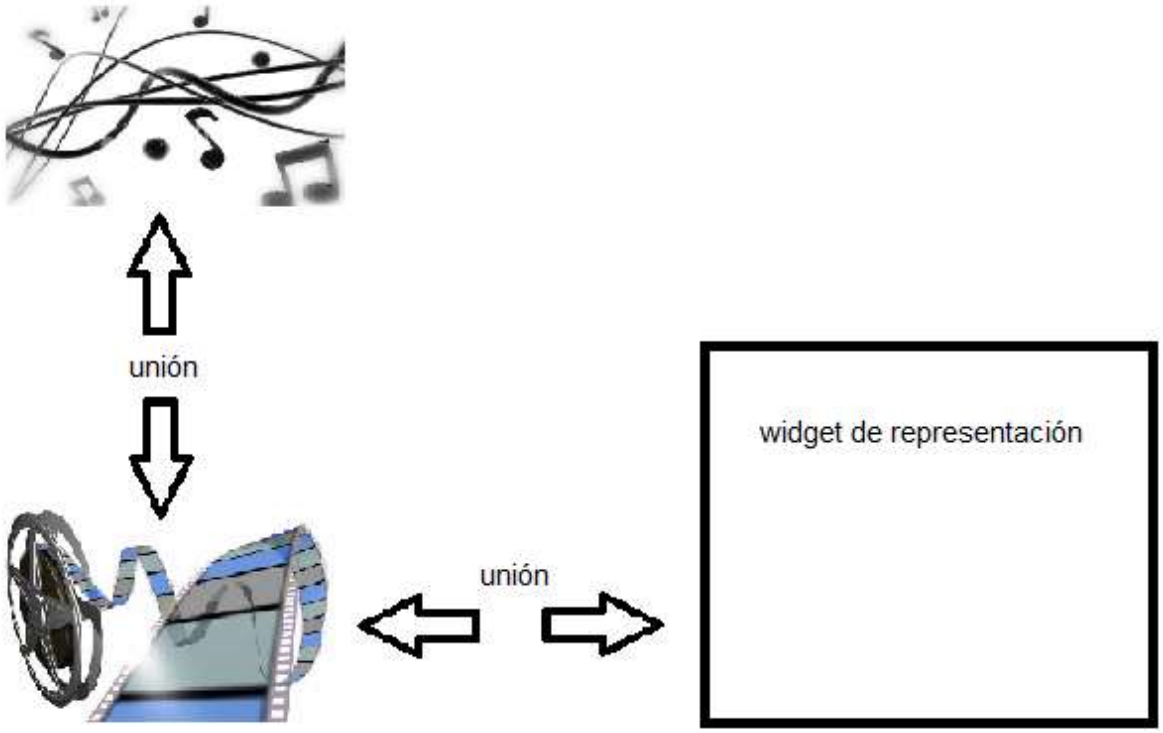


Figura # 5. Interacción entre los componentes de audio, video y lugar de representación.

Cuando se necesite ajustar el tamaño del video, como se menciona en el epígrafe 2.2 es posible utilizar cualquiera de las funcionalidades permitidas para obtener este resultado según las preferencias del desarrollador. Debe indicarse el valor del tamaño definido para el *widget* de representación luego de escoger entre las variantes posibles. El desarrollador puede inclinarse por:

- ✓ Utilizar las funcionalidades para mostrar un video en un widget con tamaño definido dentro del área de trabajo en sus variantes:
  - Escoger la representación con una proporción de 4:3.
  - Escoger la representación con una proporción de 16:9.
  - Escoger la representación utilizando toda el área definida para el *VideoWidget*.

Además puede utilizar la funcionalidad que le permita mostrar un video a pantalla completa y es posible saber cuando un video terminó de reproducirce para luego ejecutar otras acciones. Por ejemplo: al terminar la reproducción de un video introductorio en un LV inmediantamente iniciar el área de trabajo donde el estudiante desarrollará su práctica de laboratorio.

2.5 Herramientas y metodologías

El uso de las siguientes herramientas está presente durante todas y cada una de las fases de desarrollo del componente, así como el lenguaje de programación que se referencia.

Lenguaje de programación: Actualmente el Departamento de Visualización y Realidad Virtual utiliza el lenguaje C++, mostrando significativos resultados. Además es el lenguaje en el que mayor experiencia se tiene por parte del equipo de desarrollo de este departamento. Es uno de los preferidos por la mayoría de los programadores para el trabajo con gráficos por computadora ya que posee diversas características que lo hacen mejor en este sentido. Permite la programación estructurada y con la utilización de este lenguaje el programador tiene el control total de lo que está haciendo. Proporciona facilidades para la programación orientada a objetos y para el uso de plantillas o programación genérica [37]. Permite trabajar tanto a alto como a bajo nivel lo que lo convierte en un lenguaje muy potente. Por estas cualidades es que se elige este lenguaje para el desarrollo de la solución que se desea obtener mediante el transcurso de este trabajo.

Herramienta de modelado: Se elige *Visual Paradigm* para *UML* pues presenta un número de características positivas que sustentan su uso dentro del proceso de desarrollo de *software* utilizando el Lenguaje de Modelado Unificado. Es una herramienta *CASE* (*Computer Aided Software Engineering*) profesional que soporta el ciclo de vida completo del desarrollo de *software*: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Es multiplataforma y posibilita la creación de ingeniería inversa. Cuenta con un editor de Detalles de los Casos de Uso, para la especificación de los detalles de los casos de uso, incluyendo la especificación del modelo general y de las descripciones de los casos de uso y posee una distribución automática de diagramas, lo que se traduce en reorganización de las figuras y conectores de los diagramas *UML* [38].

Entorno de Desarrollo Integrado: *Microsoft Visual Studio 2008* es el *IDE* seleccionado por su alto rendimiento y funcionalidad en el desarrollo de aplicaciones. Soporta varios lenguajes de programación, entre ellos C++. Proporciona a los programadores un lenguaje orientado a objetos de probada eficacia para generar aplicaciones de alto rendimiento [39]. Gracias a plantillas avanzadas, acceso a plataformas de bajo nivel y un compilador compatible con el estándar *ANSI C++*, que optimiza las compilaciones, ofrece funcionalidad para generar componentes sólidos, utilizando el lenguaje de programación C++. Además constituye el *IDE* principal de programación en el Departamento de Visualización y Realidad Virtual por lo que es un requisito la utilización del mismo para la programación de este componente. Además brinda una buena integración con la Biblioteca de Desarrollo de Interfaces Gráficas de Usuario Qt, que es la que se utilizara para el diseño de la interfaz gráfica de los LV.

Motor gráfico: Para el desarrollo de la solución que se pretende presentar se escoge como motor gráfico de código abierto y escrito en C++ a *OGRE*. Pues presenta una colección de *plugins* que lo hacen una herramienta sumamente potente, y se integra fácilmente con bibliotecas externas. Hace un uso óptimo de la aceleración por *hardware* y utiliza las *API*s gráficas *OpenGL* y *Directx*. Este motor presenta una buena y amplia documentación para su empleo y es respaldado por una comunidad de desarrollo extensa. Su uso es bajo licencia *LGPL* [30].

Está diseñado para proveer solamente visualización gráfica aunque con su integración con otras bibliotecas se pueden desarrollar funcionalidades para el trabajo con redes, inteligencia artificial, física y otras utilizadas en el desarrollo de videojuegos aunque no es un motor de juegos [40].

La biblioteca de clases abstrae todos los detalles de la utilización de las bibliotecas del sistema subyacentes como *Direct3D* y *OpenGL*, y proporciona una interfaz basada en objetos globales y otras clases intuitivas. Además presenta funcionalidades para la creación de efectos especiales, funcionalidades de la escena, mayas, animaciones, materiales y soporte de *shaders* [7]. Además es el motor gráfico definido por el departamento para el desarrollo de aplicaciones gráficas 3D como los LV.

Biblioteca de desarrollo: Para proporcionar la portabilidad de la solución que se pretende realizar hacia distintas plataformas se escoge como biblioteca de desarrollo la biblioteca Qt. La cual presenta importantes características que apoyan su utilización en el transcurso de la solución al problema como es la utilización del lenguaje de programación anteriormente definido (C++), su integración con gráficos 3D y su amplia gama de componentes los cuales pueden ser utilizados indistintamente en cada una de las aplicaciones. Además Qt es una biblioteca multiplataforma utilizada para desarrollar aplicaciones las cuales pueden ser con o sin interfaz gráfica. Cuenta con una *API* unificada para la manipulación de archivos, estructuras de datos tradicionales además de brindar funciones para conectividad a bases de datos, programación en red, integración con gráficos 3D, desarrollo multihilos y lectura/escritura de ficheros *XML*. Además de presentar la infraestructura necesaria para desarrollar aplicaciones robustas.

La biblioteca Qt está distribuida bajo los términos de *GNU Lesser General Public License (LGPL)* por lo que es *software* libre y de código abierto. Además Qt provee un rico conjunto de componentes para la construcción de aplicaciones, distribuidas en módulos [33] los cuales facilitan el trabajo del programador y las aplicaciones creadas con Qt tienen una buena respuesta y un uso aceptable de la memoria [41].

Estándar de codificación: El estándar de codificación utilizado es el propuesto por H. Lombera en su trabajo Estándares de codificación para el lenguaje C++ utilizado en el Centro de Diseño y Simulación de Estructuras Mecánicas de marzo del 2011.

Conclusiones parciales

En el transcurso de este capítulo se describió la solución que se propone para darle cumplimiento al objetivo de la investigación. Además fueron elegidas las técnicas de manipulación de videos empleadas por las herramientas *OGRE* y Qt para conformar la solución y se definieron las herramientas y metodologías a utilizar para realizar el componente.

CAPÍTULO 3: DISEÑO E IMPLEMENTACIÓN DE LA SOLUCIÓN

El presente capítulo tendrá como tema fundamental el diseño y la implementación de la aplicación que se quiere desarrollar. Primeramente se propone un modelo de dominio para comprender la solución propuesta. Seguidamente se da paso a la captura de requisitos, que será utilizada posteriormente en la elaboración del modelo de casos de uso del sistema. Se definen los actores del sistema y se especifican los casos de uso obtenidos a partir de los requisitos capturados quedando modelados y descritos. Además se modelan los diagramas de clases y de secuencia del diseño con el objetivo de proporcionar una comprensión más profunda del sistema que se desea desarrollar y se describen las clases modeladas. Además se presenta el diagrama de componentes representando la relación entre estos en la solución final.

3.1 Modelo de Dominio

El modelo de dominio representado a continuación es un acercamiento a la solución propuesta donde se modelan los principales conceptos así como sus relaciones con los que se trabajarán en el componente a obtener.

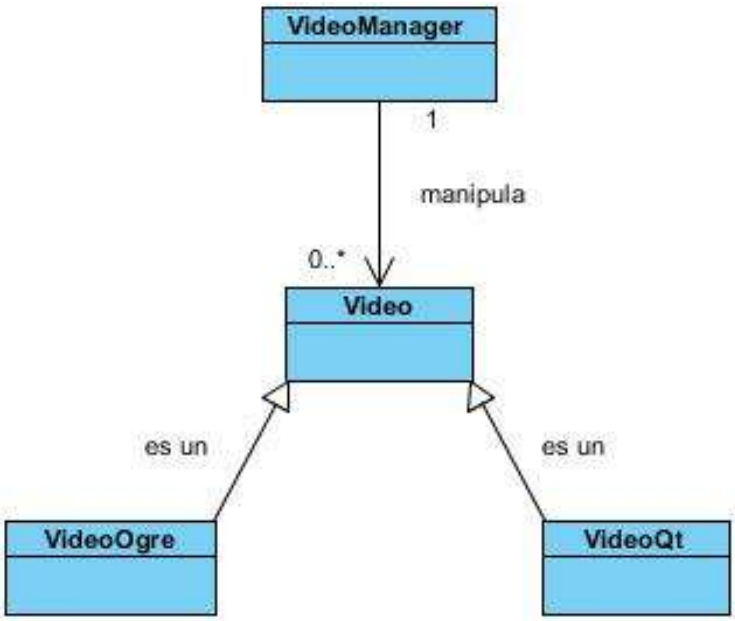


Figura # 6. Modelo de dominio del componente de manipulación de videos.

3.1.1 Glosario de Términos del Modelo de Dominio

VideoManager: Representa la clase que será la encargada de manipular los archivos de video previamente cargados y que serán utilizados durante la aplicación.

Video: Representa un archivo de video.

VideoQt: Representa a un archivo de video creado con las características y que será manipulado utilizando las funcionalidades definidas para Qt.

VideoOgre: Representa a un archivo de video creado con las características y que será manipulado utilizando las funcionalidades definidas para *OGRE*.

3.2 Requerimientos del Sistema

El análisis de requisitos permite al ingeniero de sistemas especificar la función y el rendimiento del *software*, indica la interfaz del mismo con otros elementos del sistema y establece las restricciones que debe cumplir este. El análisis de requisitos proporciona modelos al diseñador del *software* que pueden traducirse en el diseño de datos, arquitectónico y de interfaz. Finalmente, la especificación de requisitos proporciona al diseñador y al cliente los medios para valorar la calidad una vez que se ha construido el *software* [42].

3.2.1 Requisitos funcionales

Los requerimientos funcionales no son más que la determinación clara y concisa de qué debe ser capaz de hacer el sistema, estas se corresponden con opciones que ejecutará el *software*, operaciones realizadas de forma oculta o condiciones extremas a determinar por el sistema.

El sistema que se desarrollará tendrá como requisitos funcionales los siguientes.

RF 1. Cargar videos.

RF 2. Reproducir video.

RF 3. Pausar video.

RF 4. Detener video.

RF 5. Mostrar video.

RF 6. Ocultar video.

RF 7. Ajustar volumen del video.

RF 8. Ajustar tamaño del video.

RF 9. Ajustar posición del video.

3.2.2 Requisitos no funcionales

Los requerimientos no funcionales responden a cualidades que el producto debe tener y a las características para que este sea atractivo, confiable, usable, seguro.

El sistema que se desarrollará contará con los siguientes requisitos no funcionales.

- Restricciones en el Diseño e Implementación.
  - ✓ La aplicación se desarrollará utilizando *OGRE* como motor gráfico.
  - ✓ Se utilizará como biblioteca de desarrollo Qt.
  - ✓ Se utilizará *Microsoft Visual Studio 2008* como *IDE* de desarrollo.
  - ✓ Se empleará C++ como lenguaje de programación.

- Usabilidad

  Fácil para el desarrollador.

- *Hardware*
  - ✓ Procesador *Pentium 4* o equivalente.
  - ✓ 512 Mb de memoria RAM.
  - ✓ *Hardware* de video con 64 Mb de memoria.

- *Software*

Se utilizará como formato de video digital el *Ogg*.

Sistema operativo:

- ✓ *Windows XP* o superior
- ✓ *Linux* en cualquiera de sus distribuciones

❖ Seguridad

El sistema debe realizar el tratamiento de errores para todas las acciones que se realizan en él.

3.3 Modelo de Casos de Uso del Sistema

En esta sección se reconocen los posibles actores del sistema a desarrollar y se conciben, a través de la agrupación de los requisitos funcionales anteriormente hallados, los posibles resultados de valor que les pueda brindar a sus actores, es decir, los casos de uso del sistema.

3.3.1 Definición del Actor del Sistema

Actor	Descripción
Desarrollador	Es el que utiliza el componente desarrollado para emplearlo en los laboratorios virtuales. Para ello hace uso de las funcionalidades implementadas en él y que le permiten interactuar con los archivos de videos digitales que quiera utilizar.

Tabla # 2. Definición del actor del sistema.

3.3.2 Definición de los Casos de Uso del Sistema

CU-1	Cargar Videos
Actor	Desarrollador
Descripción	Se refiere al proceso de carga de todos los videos que se quieran utilizar durante el laboratorio virtual.
Referencia	RF 1

Tabla # 3. Definición del caso de uso Cargar Videos.

CU-2	Manipular Video
Actor	Desarrollador
Descripción	Se refiere a los procesos que el desarrollador puede realizar sobre los videos que se quieran incluir en los laboratorios virtuales, permitiendo la interacción con estos videos a través de las funcionalidades implementadas para ello.
Referencia	RF 2, RF 3, RF 4, RF 5, RF 6, RF 7, RF 8, RF 9.

Tabla # 4. Definición del caso de uso Manipular Video.

3.3.3 Diagrama de Casos de Uso del Sistema

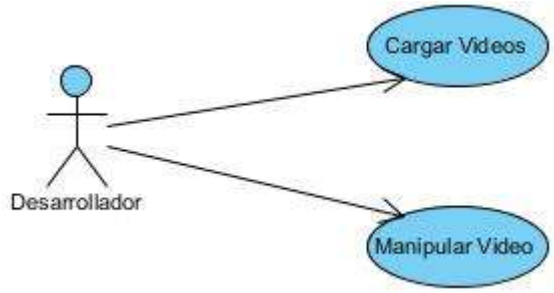


Figura # 7. Diagrama de Casos de Uso del Sistema.

3.3.4 Descripción de los Casos de Uso del Sistema

Para lograr un mayor entendimiento de las funcionalidades asociadas a cada caso de uso, no es suficiente con la representación gráfica del diagrama de casos de uso, es por ello que se realiza la expansión de los mismos, donde se muestra más detalladamente sus funcionalidades, logrando así un mejor entendimiento de los mismos.

Caso de Uso	
CU-1	Cargar Videos
Propósito	Cargar los videos que se utilizarán en el laboratorio virtual.
Actores	Desarrollador

<table>
<tr><td>Resumen</td><td colspan="2">El caso de uso se inicia cuando el desarrollador necesite cargar cada uno de los videos que se utilizarán durante la experiencia de aprendizaje ya sea para ser manipulados con OGRE o con Qt. Sobre estos videos son los que posteriormente el desarrollador podrá indicar las funcionalidades de manipulación. El caso de uso termina cuando se cargan cada uno de los videos digitales para ser utilizados.</td></tr>
<tr><td>Referencias</td><td colspan="2">RF 1.</td></tr>
<tr><td colspan="3">Flujo normal de eventos</td></tr>
<tr><td>Acciones del actor</td><td colspan="2">Respuestas del sistema</td></tr>
<tr><td>1. Elige la opción de cargar un video, especificando la forma en la que se desea cargar:
• Video con Ogre.
• Video con QT.</td><td colspan="2">1.1. En dependencia de la elección del desarrollador se muestran las opciones existentes para cargar los videos.
a) Si la elección es cargar video con Ogre ver sección “Cargar video con Ogre”.
b) Si la elección es cargar video con Qt ver sección “Cargar video con Qt”.</td></tr>
<tr><td colspan="3">Sección “Cargar video con Ogre”</td></tr>
<tr><td colspan="3">Flujo normal de eventos</td></tr>
<tr><td>Acciones del actor</td><td colspan="2">Respuestas del sistema</td></tr>
<tr><td>1. Introduce los parámetros necesarios para cargar un video:
• Dirección física de ubicación del archivo de video en la computadora.
• Nombre con el que se va a almacenar.
• Material para reproducir el video con Ogre.
• Puntero al SceneNode al que se le va a adjuntar el video.
• Puntero al plano donde se va a</td><td colspan="2">1.1. Se verifican los parámetros introducidos.
1.2. Se carga el video a través del parámetro de la dirección definida.
1.3. Termina el caso de uso.</td></tr>
</table>

representar un video para pantalla completa.	
Sección "Cargar video con Qt"	
Flujo normal de eventos	
Acciones del actor	**Respuestas del sistema**
1. Introduce los parámetros necesarios para cargar un video: • Dirección física de ubicación del archivo de video en la computadora. • Nombre con el que se va a almacenar. • Dimensiones para representar el video. • Posición en la cual se representará el video. • Puntero al *VideoWidget* donde se va a representar el video.	1.1. Se verifican los parámetros introducidos. 1.2. Se carga el video a través del parámetro de la dirección definida. 1.3. Termina el caso de uso.
Precondiciones	
Postcondiciones	Se cargan los videos a utilizar en el laboratorio virtual.
Prioridad	Crítico.
Flujo alternativo	
Acciones del actor	**Respuestas del sistema**
	1.1. No se ejecuta ninguna acción de manipulación si el desarrollador indica mal cualquiera de los parámetros. 1.2. Muestra el mensaje de error "*Invalid video format. Use Ogg/Ogv videos*" si el formato de video no es compatible con la solución del sistema. 1.3. Termina el caso de uso.

Tabla # 5. Descripción del caso de uso Cargar Videos.

Caso de Uso	
CU-2	Manipular Video
Propósito	Realizar las acciones de manipulación sobre los videos.
Actores	Desarrollador
Resumen	El caso de uso lo inicia el desarrollador en el momento de querer manipular los videos en el laboratorio virtual que esté desarrollando. Sobre estos videos el desarrollador podrá realizar las siguientes acciones: reproducir, pausar, detener, ajustar el volumen, mostrar, ocultar, cambiar el tamaño y cambiar la posición.
Referencias	RF 2, RF 3, RF 4, RF 5, RF 6, RF 7, RF 8, RF 9.
Flujo normal de eventos	
Acciones del actor	**Respuestas del sistema**
1. Se indica una búsqueda del video que se quiere manipular atendiendo al nombre con el cual fue guardado.	1.1. Busca el video indicado.
2. Se indica una de las acciones a realizar para interactuar con el video:	2.1. Permite la ejecución de una de las siguientes funcionalidades sobre el video buscado en dependencia de la elección del desarrollador:
a) Reproducir el video.	a) Permite reproducir video.
b) Pausar el video.	b) Permite pausar video.
c) Detener el video.	c) Permite detener video.
d) Ajustar el volumen del video indicando el nuevo valor de volumen.	d) Permite ajustar el volumen del video de acuerdo al valor nuevo definido.
e) Mostrar el video.	e) Permite mostrar video.
f) Ocultar el video.	f) Permite ocultar video.
g) Ajustar tamaño del video indicando las nuevas dimensiones del video.	g) Permite ajustar el tamaño del video de acuerdo a las nuevas dimensiones del video indicadas.
h) Ajustar posición del video indicando	h) Permite ajustar la posición del video de

las nuevas coordenadas del video.	acuerdo a las nuevas coordenadas del video indicadas.
	2.2. Termina el caso de uso.
Precondiciones	Debe haberse ejecutado el caso de uso Cargar videos.
Postcondiciones	Se realiza la acción de manipulación sobre el video.
Prioridad	Secundario.
Flujo alternativo	
Acciones del actor	**Respuestas del sistema**
	1.1. No ejecuta ninguna acción si el desarrollador indica mal el nombre identificador en la búsqueda del video que quiere manipular y lanza el mensaje de error *"There is no video loaded with the name ("Nombre")"*. 1.2. Termina el caso de uso.

Tabla # 6. Descripción del caso de uso Manipular Video.

3.4 Diseño

Dentro de los propósitos fundamentales del diseño se encuentra adquirir una comprensión de los aspectos relacionados con los requisitos no funcionales y restricciones definidas para la solución. Además es el encargado de proveer un punto inicial para posteriormente realizar las actividades de implementación. Como artefactos resultantes de este flujo de trabajo aparecen los diagramas de clases de diseño, los cuales exponen las clases del sistema y las relaciones existentes entre ellas quedando expuestas las acciones que el sistema puede realizar y como puede ser finalmente desarrollado el mismo. Además resultan de este flujo de trabajo los diagramas de secuencia y colaboración que describen los pasos en la realización de cada caso de uso.

3.4.1 Diagrama de Clases de Diseño

Para una mejor comprensión a continuación se muestra el diagrama de clases representando solo los nombres de las clases que intervienen en la solución. Seguidamente se describirán en tablas cada una de las clases desarrolladas para la solución propuesta, especificando sus atributos y métodos.

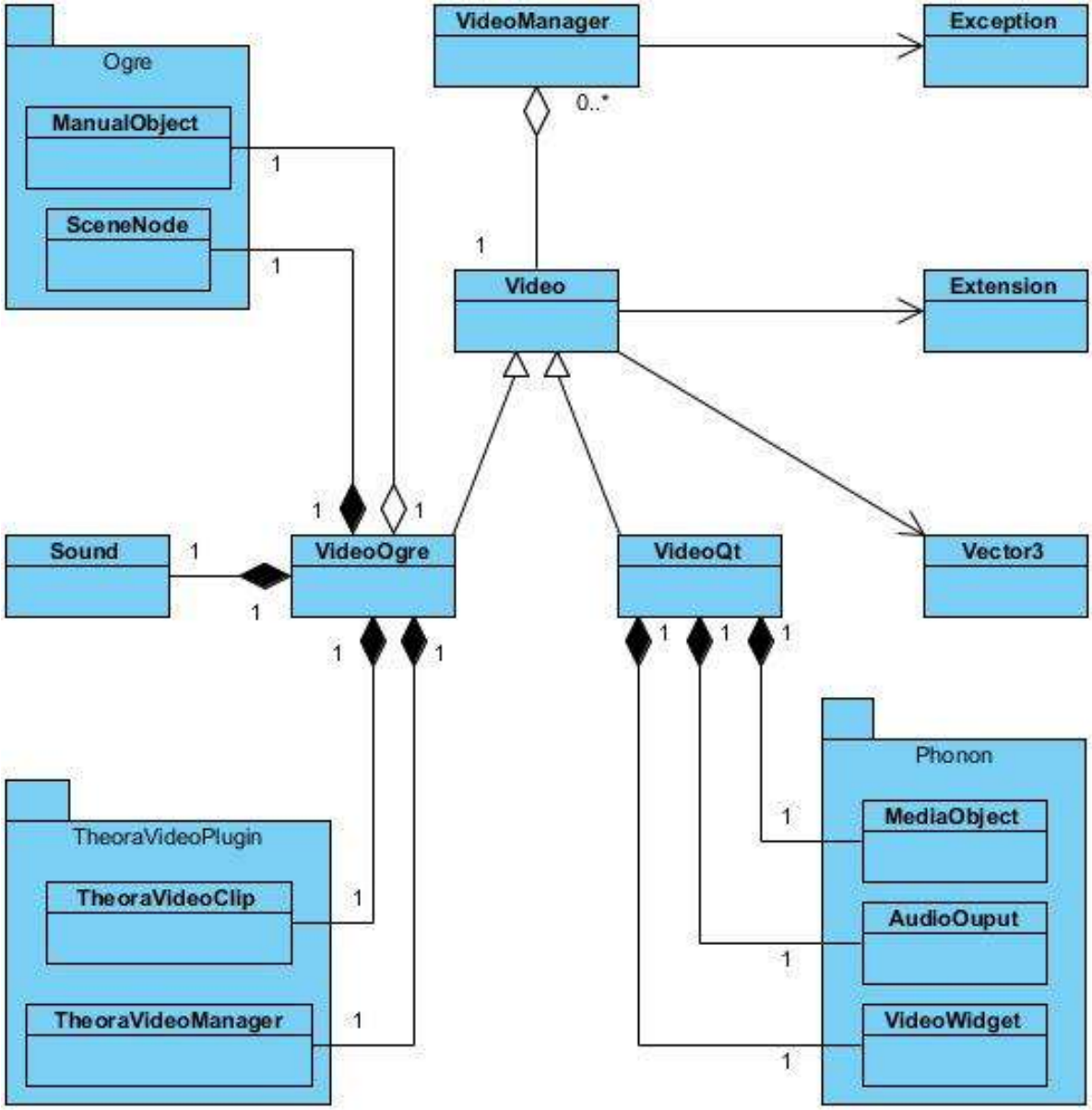


Figura # 8. Diagrama de Clases de Diseño.

3.4.2 Descripción de las Clases de Diseño

Descripción de la clase VideoManager.

Nombre	VideoManager
Tipo de la clase	Controladora
Atributos	**Tipo**
_videoMap	VideoMap
Iter	VideoMap::iterator
Operaciones	**Descripción**
VideoManager()	Constructor de la clase.
~VideoManager()	Destructor de la clase.
createVideo(argVideo : Video*)	Crea un objeto de tipo Video y lo adiciona en el *VideoMap*.
videoSearch(&argIdName : const string)	Busca un video dentro del mapa a partir del nombre identificador indicado.
destroyedAllVideos()	Destruye todos los videos almacenados en el mapa.
destroyVideo(&argIdName : const string)	Busca el video indicado por el identificador y lo destruye.
isEmptyMap()	Devuelve verdadero o falso si el mapa de videos se encuentra vacío o no respectivamente.

Tabla # 7. Descripción de la clase VideoManager.

Descripción de la clase Video.

Nombre	Video
Tipo de la clase	Padre
Atributos	**Tipo**
_idName	string
_path	string
_volume	float
_scale	Vector3

_position	Vector3
_loaded	bool
Operaciones	**Descripción**
Video()	Constructor de la clase.
Video(&argPath : const string, &argIdName : const string, argScale : VideoOgreQt::Vector3, argPosition : VideoOgreQt::Vector3)	Constructor de la clase.
Video(&argPath : const string, &argIdName : const string)	Constructor de la clase.
~Video()	Destructor de la clase.
playVideo()	Reproduce un video.
pauseVideo()	Pausa un video.
stopVideo()	Detiene un video.
volumeVideo(argVolume : float)	Ajusta el volumen de un video mediante el parámetro argVolume.
rewindVideo()	Retrasa un video.
forwardVideo()	Adelanta un video.
showVideo()	Muestra un video.
hideVideo()	Esconde un video.
aspectChanged16_9(argScale : VideoOgreQt::Vector3)	Ajusta el tamaño de un video mediante el parámetro argScale y ajusta la proporción a 16:9.
aspectChanged4_3(argScale : VideoOgreQt::Vector3)	Ajusta el tamaño de un video mediante el parámetro argScale y ajusta la proporción a 4:3.
aspectChangedScale(argScale : VideoOgreQt::Vector3)	Ajusta el tamaño de un video mediante el parámetro argScale y ajusta la proporción al tamaño completo del área de representación.
fullScreen()	Ajusta el tamaño de un video a pantalla completa.
adjustPosition(argPosition :	Ajusta la posición de un video mediante el parámetro

VideoOgreQt::Vector3)	argPosition.
videoFinished()	Devuelve verdadero en caso de que el video haya terminado de reproducirse, falso en caso contrario.
playTime()	Devuelve el tiempo de reproducción de un video.
totalTime()	Devuelve el tiempo total de un video.
removeVideo()	Limpia la memoria ocupada por un video cargado.
getIdName()	Devuelve el nombre identificador de un video.
getPath()	Devuelve la dirección física donde se encuentra el video.
getVolume()	Devuelve el volumen con el que se escucha el video.
getScale()	Devuelve el tamaño del video.
getPosition()	Devuelve la posición del video.
getLoaded()	Devuelve verdadero si un video se encuentra cargado, falso en caso contrario.

Tabla # 8. Descripción de la clase Video.

Dado que las clases *VideoOgre* y *VideoQt* descritas a continuación heredan atributos y métodos de la clase *Video* y estos ya fueron descritos anteriormente solo se plasmarán en estas clases los atributos y métodos diferentes.

Descripción de la clase VideoOgre.

Nombre	VideoOgre
Tipo de la clase	Hija
Atributos	**Tipo**
_sound	Sound *
_ogreMaterial	Ogre::Material *
_ogreSceneNode	Ogre::SceneNode *
_ogrePlane	Ogre::ManualObject *
_theoraVideoManager	Ogre::TheoraVideoManager *
_theoraVideoClip	Ogre::TheoraVideoClip *

AspectChanged	enum
_aspectRatio	AspectChanged
_visible	bool
Operaciones	**Descripción**
VideoOgre()	Constructor de la clase.
VideoOgre(&argPath : const string, &argIdName : const string, &argOgreMaterial : const string, argOgreSceneNode : Ogre::SceneNode *)	Constructor de la clase.
VideoOgre(&argPath : const string, &argIdName : const string, &argOgreMaterial : const string, argOgreSceneNode : Ogre::SceneNode *, Ogre:: argOgrePlane : ManualObject*)	Constructor de la clase.
~VideoOgre()	Destructor de la clase.
loadVideo()	Carga el video.

Tabla # 9. Descripción de la clase VideoOgre.

Descripción de la clase VideoQt.

Nombre	VideoQt
Tipo de la clase	Hija
Atributos	**Tipo**
_mediaObject	Phonon::MediaObject
_audioOutput	Phonon::AudioOutput
_videoWidget	Phonon::VideoWidget *
Operaciones	**Descripción**
VideoQt()	Constructor de la clase.
VideoQt(&argPath : const string,	Constructor de la clase.

&argIdName : const string, argScale : VideoOgreQt:: Vector3, argPosition : VideoOgreQt::Vector3, argVideoWidget : Phonon::VideoWidget *)	
~VideoQt()	Destructor de la clase.
loadVideo()	Carga el video.

Tabla # 10. Descripción de la clase VideoQt.

Descripción de la clase Sound.

Nombre	Sound
Tipo de la clase	Auxiliar
Atributos	**Tipo**
_path	string
_volume	float
_freq	ALsizei
_format	ALenum
_state	ALint
_buffer	ALuint
_source	ALuint
_bufferData	vector<char>
_loop	ALboolean
_bitStream	int
_bytes	long
_array[BUFFER_SIZE]	char
_file	FILE*
_oggInfo	vorbis_info*
_oggFile	OggVorbis_File
Operaciones	**Descripción**
Sound ()	Constructor de la clase.

Sound(&argPath : const string, argVolume : float)	Constructor de la clase.
~Sound ()	Destructor de la clase.
loadSound()	Carga la información del audio desde el archivo de video mediante el parámetro argPath indicado cuando se construye el video.
playSound(argLoop : ALboolean)	Reproduce el audio indicando el valor de repetición (*false* para no repetir, *true* para repetir).
pauseSound()	Pausa el audio.
stopSound()	Detiene el audio.
volumeSound(argVolume : float)	Ajusta el volumen del audio mediante el parámetro argVolume.
setupAL()	Inicializa la biblioteca *OpenAL*.
destroySound()	Libera la memoria ocupada por el sonido.

Tabla # 11. Descripción de la clase Sound.

Descripción de la clase Vector3.

Nombre	Vector3
Tipo de la clase	Auxiliar
Atributos	**Tipo**
_x	float
_y	float
_z	float
Operaciones	**Descripción**
Vector3 ()	Constructor de la clase.
Vector3 (argX : float, argY : float, argZ : float)	Constructor de la clase.
~Vector3 ()	Destructor de la clase.
getX()	Devuelve el valor de la componente x del vector (x, y, z).

setX(argX : float)	Modifica el valor de la componente x del vector (x, y, z).
getY()	Devuelve el valor de la componente y del vector (x, y, z).
setY(argY : float)	Modifica el valor de la componente y del vector (x, y, z).
getZ()	Devuelve el valor de la componente z del vector (x, y, z).
setZ(argZ : float)	Modifica el valor de la componente z del vector (x, y, z).

Tabla # 12. Descripción de la clase Vector3.

Descripción de la clase Extension.

Nombre	Extension
Tipo de la clase	Auxiliar
Operaciones	**Descripción**
Extension()	Constructor de la clase.
~Extension()	Destructor de la clase.
getExtension(&argPath : const string)	Devuelve la extensión del archivo de video a través de la dirección física indicada en el parámetro argPath.

Tabla # 13. Descripción de la clase Extension.

Descripción de la clase Exception.

Nombre	Exception
Tipo de la clase	Auxiliar
Atributos	**Tipo**
_description	string
_source	string
Operaciones	**Descripción**
Exception(&argDescription : string const, &argSource : string const)	Constructor de la clase.
~Exception()	Destructor de la clase.
getDescription()	Devuelve la descripción del error encontrado.
getSource()	Devuelve el lugar (Clase) del error encontrado.

Tabla # 14. Descripción de la clase Exception.

3.4.3 Diagramas de Secuencia

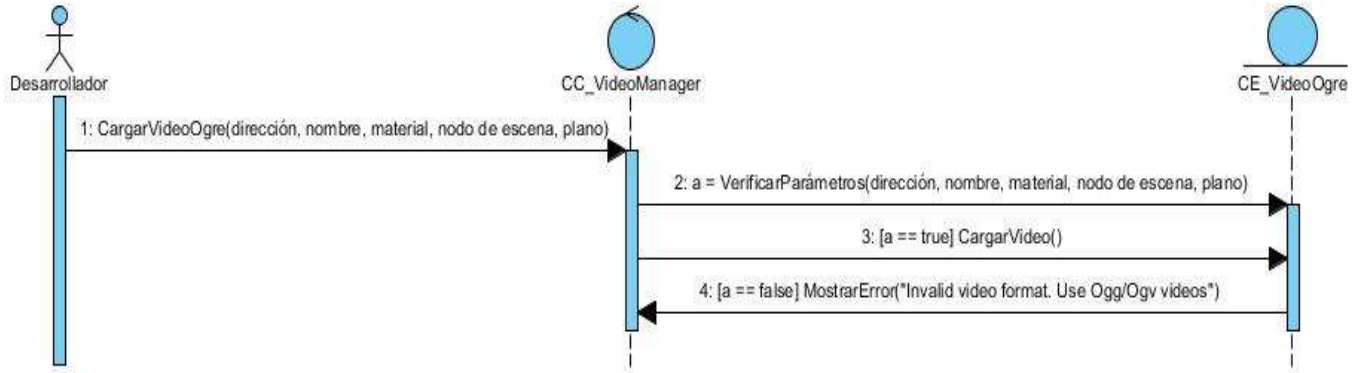


Figura # 9. Diagrama de secuencia del caso de uso Cargar Videos. Sección Cargar video con OGRE.

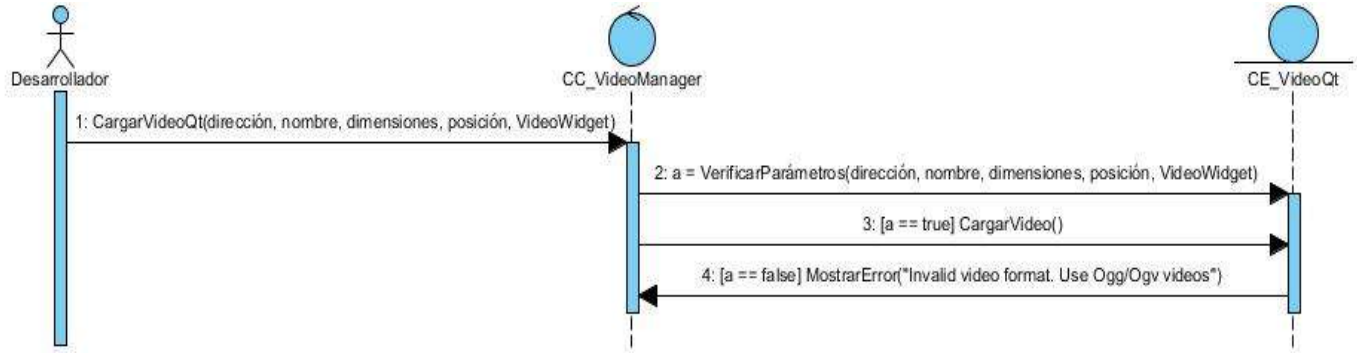


Figura # 10. Diagrama de secuencia del caso de uso Cargar Videos. Sección Cargar video con Qt.

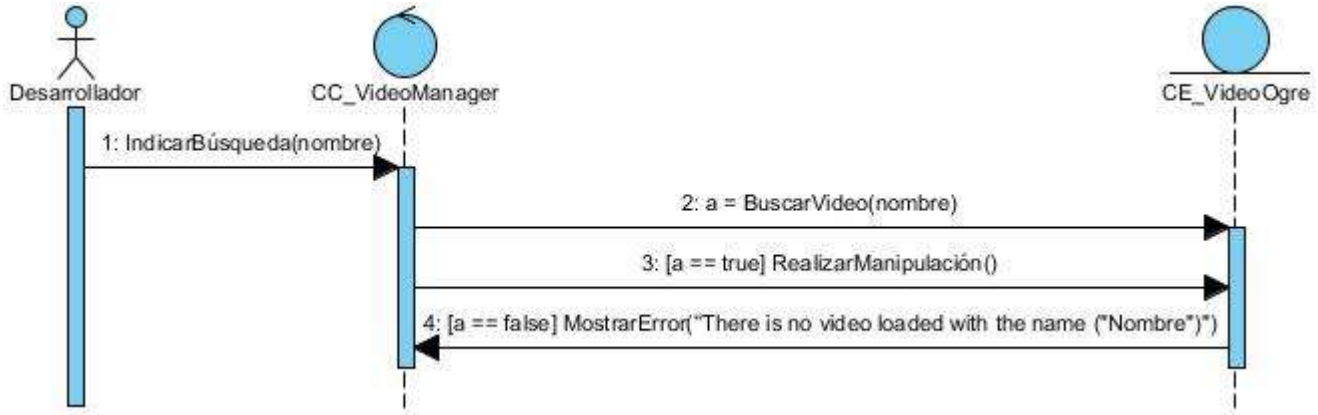


Figura # 11. Diagrama de secuencia del caso de uso Manipular Video.

3.4.4 Patrones

En el diagrama de clases anteriormente presentado se puede definir claramente el uso de los patrones *GRASP (General Responsability Assignment Software Patterns)*. Los mismos son patrones de *software* para asignación de responsabilidades que entre sus principales ventajas se encuentran:

- Mantiene el encapsulamiento.
- Los objetos utilizan su propia información para llevar a cabo sus tareas.
- Se distribuye el comportamiento entre las clases que contienen la información requerida.
- Son más fáciles de entender y mantener.

Dentro de estos patrones se evidencia el uso del patrón creador en las clases *VideoManager* encargada de crear instancias de la clase *Video* y en las clases *VideoOgre* y *VideoQt* encargadas de crear las instancias de las clases necesarias para poder implementar las funcionalidades de manipulación sobre los videos, puesto que presentan relaciones de agregación/composición con las clases necesarias para cada una. Ejemplo de estas son:

Para *VideoOgre*:

- Clase *ManualObject*.
- Clase *SceneNode*.
- Clase *TheoraVideoClip*.
- Clase *TheoraVideoManager*.
- Clase *Sound*.

Para *VideoQt*:

- Clase *MediaObject*.
- Clase *AudioOuput*.
- Clase *VideoWidget*.

Otros de los patrones *GRASP* que se manifiestan son:

Alta cohesión ya que se asignan a las clases responsabilidades que trabajen sobre una misma área de la aplicación y que no tengan mucha complejidad.

Bajo acoplamiento ya que se asignan las responsabilidades de forma tal que las clases se comuniquen con el menor número de clases posible.

3.5 Implementación

El resultado principal del flujo de trabajo implementación es la obtención de componentes, dentro de los que se incluyen ficheros, ejecutables y sus dependencias. En los casos necesarios se incluye también los protocolos que se emplearán para la comunicación entre los componentes.

3.5.1 Diagrama de Componentes

Un componente representa una parte física del sistema, por ejemplo: una biblioteca de clases, un ejecutable, una tabla; que engloba la implementación de un grupo de clases del diseño. Cada componente define una interfaz que describe su funcionalidad y forma de empleo. El diagrama de componentes, permite conocer a los desarrolladores y clientes la estructura física que tiene el sistema y cómo se relacionan sus partes. La figura # 12 representa el diagrama de componentes del sistema propuesto.

El componente resultante de la solución es una biblioteca que será utilizada para manipular videos digitales en aplicaciones de realidad virtual específicamente en los LV que se desarrollan en el departamento de Visualización y Realidad Virtual cuyas salidas serán un archivo Component_Video.dll y un archivo Component_Video.lib que contendrán las funcionalidades necesarias para manipular los videos en estas aplicaciones.

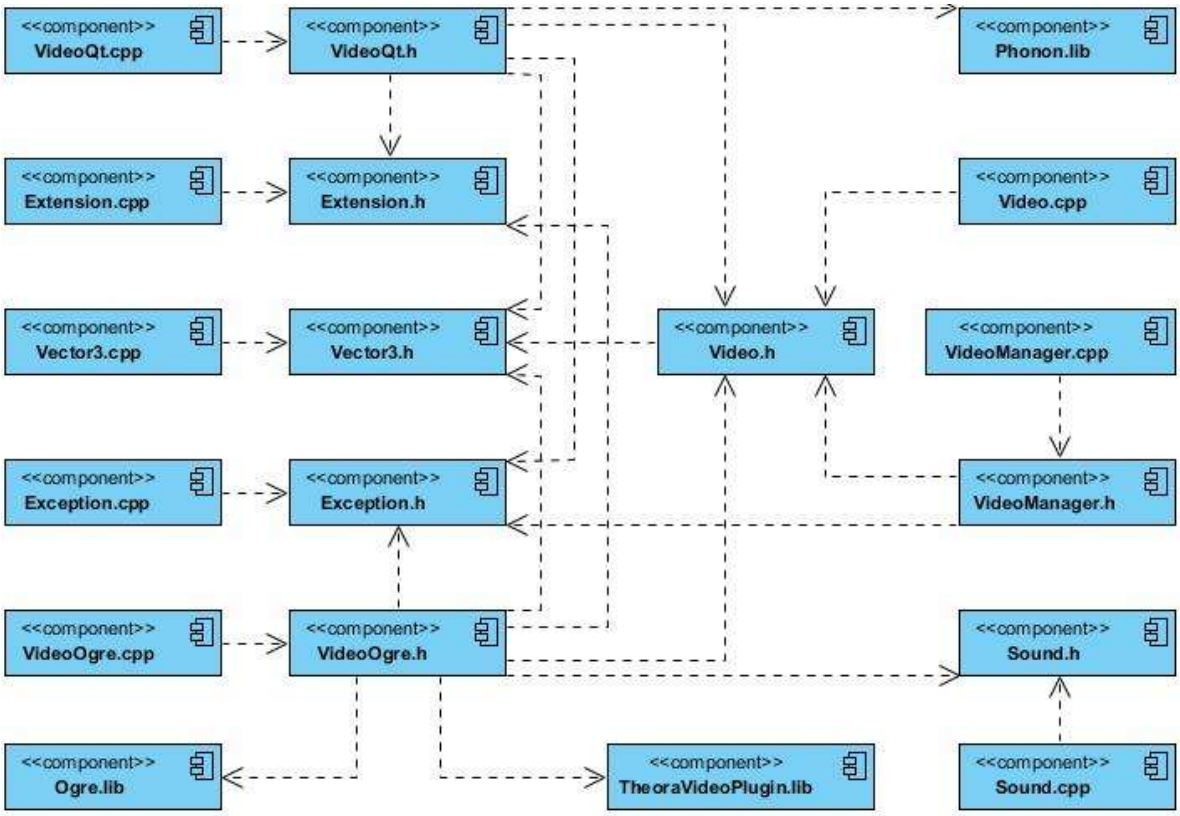


Figura # 12. Diagrama de Componentes.

Conclusiones parciales

Durante el capítulo fueron descritos los procesos de diseño para el desarrollo del *software*. En ellos cuentan los diagramas de modelo de dominio, de clases y de secuencia que brindan una visión sobre el *software* que se quiere desarrollar. Además se plantearon los requisitos funcionales y no funcionales que debe cumplir la solución a desarrollar. Se modelaron los casos de uso del sistema y se describieron estos y las clases de diseño para una mayor comprensión. Además se abordaron los patrones de diseño empleados en la solución y se presenta el diagrama de componentes de la misma.

CAPÍTULO 4: ANÁLISIS DE RESULTADOS

El propósito de este capítulo es mostrar los resultados obtenidos al probar el componente desarrollado comprobando así su correcto funcionamiento y la simplicidad de su uso. Para esto fueron creadas dos aplicaciones de prueba. Cada una para probar las funcionalidades implementadas y que permiten manipular los archivos de videos digitales con *OGRE* y Qt indistintamente.

Las aplicaciones que a continuación se muestran presentan la utilización de videos, primeramente en uno de los LV que se desarrollan y seguidamente en una aplicación sencilla desarrollada con Qt.

Para representar los videos el desarrollador debe realizar las siguientes acciones en la aplicación donde utilizará el componente desarrollado para esta investigación.

- Crear un objeto de la clase *VideoManager*.
- Crear los objetos de video los cuales representarán los videos que serán visualizados.
- Indicar la acción de manipulación sobre un video creado.

Ejemplo:

```
VideoManager *mVM = new VideoOgreQt::VideoManager();

mVM->createVideo(new VideoOgreQt::VideoOgre("konqi.ogg", "video1",
"Example/OGRE_VIDEO2", cubeNode));

mVM->createVideo(new VideoOgreQt::VideoQt("bunny.ogg", "video2",
VideoOgreQt::Vector3 (200, 200, 0), VideoOgreQt::Vector3 (50, 70, 0),
vWidget1));

mVM->videoSearch("video1")->playVideo();
```

4.1 Resultados con OGRE

Primeramente se representa un video sobre un plano y otro sobre un objeto 3D en este caso, un cubo dentro de la escena del LV. Las funcionalidades de manipulación sobre los videos digitales son llamadas mediante las pulsaciones de teclas anteriormente definidas.

Figura # 13. Aplicación de prueba para manipular videos con OGRE. Utilización de videos en uno de los LV desarrollados.

Si es necesidad del desarrollador visualizar, por ejemplo, un video diferente por cada una de las caras del cubo deberá construir seis planos y ubicarlos manualmente para conformar esta figura geométrica y a cada plano asignarle un objeto de video diferente.

4.2 Resultados con Qt

Se representan dos videos diferentes sobre dos *widgets* ubicados espacialmente en lugares distintos y con distintas dimensiones dentro del área de trabajo de la aplicación. Las funcionalidades de manipulación sobre los videos digitales pueden ser aplicadas pulsando con el ratón sobre los botones definidos en la parte superior de la aplicación.

Figura # 14. Botones definidos dentro de la aplicación de prueba para manipular los videos.

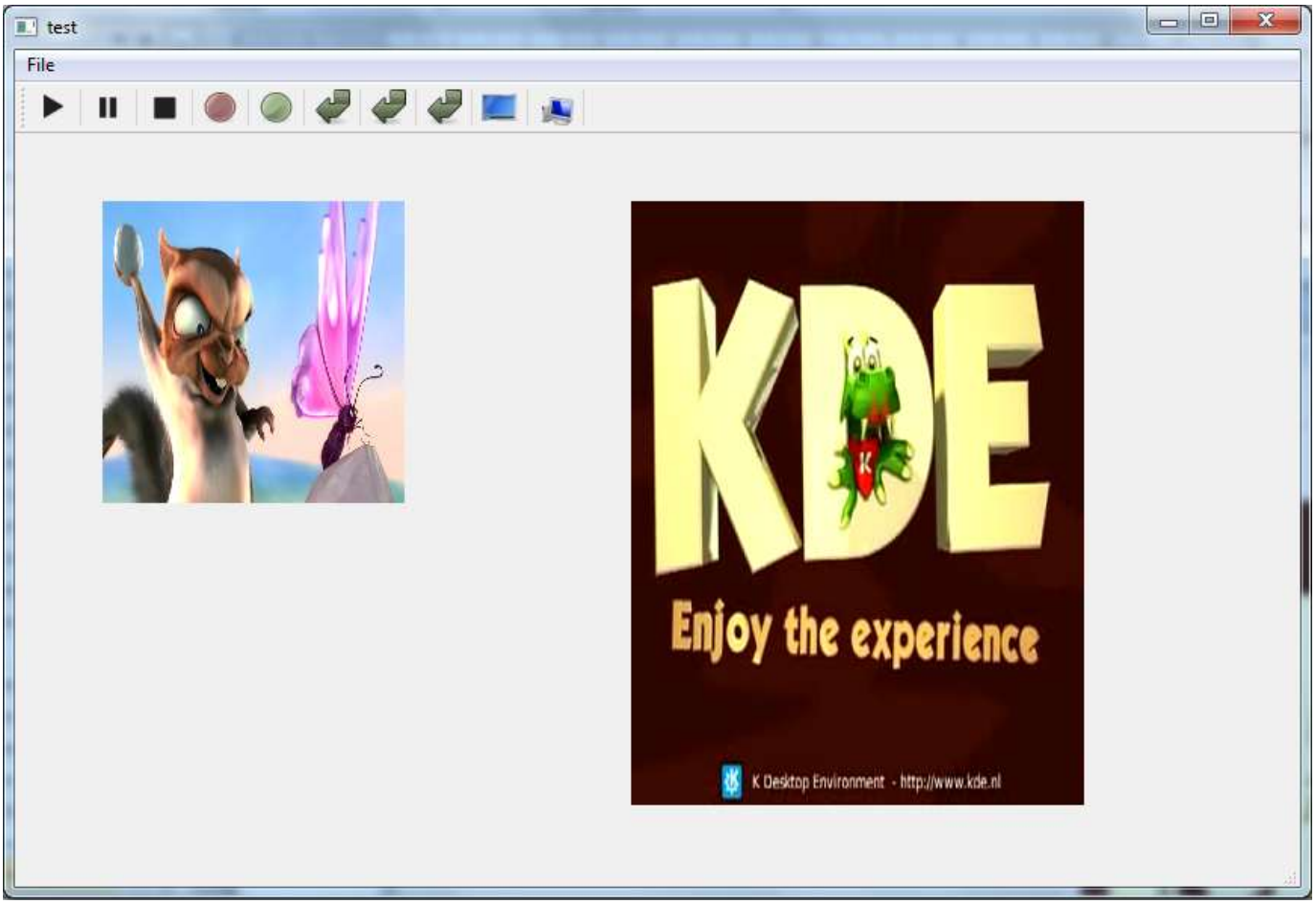


Figura # 15. Aplicación de prueba para manipular videos con Qt. Utilización de videos en lugares específicos dentro del área de trabajo.

CONCLUSIONES

La realización del presente trabajo permite la manipulación de videos digitales en los LV, agilizando y simplificando estos procesos a los desarrolladores del departamento de Visualización y Realidad Virtual. De esta forma se le dio cumplimiento al objetivo de la investigación planteado y se resolvió el problema descrito en la misma, además:

- El componente desarrollado permite una rápida y sencilla manipulación de los videos digitales.
- Se demostró que el formato de video digital *Ogg* resulta apropiado para la inserción de videos en los laboratorios virtuales.
- El componente final es fácilmente integrable a la arquitectura definida para los laboratorios virtuales.

RECOMENDACIONES

Teniendo en cuenta la investigación realizada y los resultados obtenidos, en función de investigaciones posteriores se recomienda:

- Utilizar el componente obtenido en los próximos laboratorios virtuales que sean desarrollados.
- Permitir una mayor utilización del componente agregándole otras funcionalidades de manipulación como pueden ser las concernientes para adelantar y retroceder el video que se encuentre en reproducción.
- Enriquecer el componente con el objetivo de poder manipular archivos de video digital que presenten otros formatos.

REFERENCIAS BIBLIOGRÁFICAS

1. [En línea] www.monografias.com/trabajos4/realvirtual/realvirtual.shtml.

2. **Díaz, Yoander Cabrera.** *Sistema de Sonido 3D para Mejorar la Sensopercepción en el Estudio de Funciones Visuales.* La Habana : s.n., 2011.

3. **L. Rosado, J. R. Herreros.** Nuevas aportaciones didácticas de los laboratorios virtuales y remotos en la enseñanza de la Física. [En línea] 2005. http://www.formatex.org/micte2009/.

4. **M.I. Alberto Pedro Lorandi Medina, M.I. Guillermo Hermida Saba, M.S.I. José Hernández Silva y M.C. Enrique Ladrón de Guevara Durán.** *AcademiaJournals.com Revista Internacional de Educación en Ingeniería.* Veracruz, México : s.n., 2011. ISSN 1940-1115.

5. **Ducongé, Orlay García.** *Biblioteca para la manipulación de videos digitales en Sistemas de Realidad Virtual.* Ciudad de la Habana : s.n., 2007.

6. —. *Manipulación de vídeos digitales en Sistemas de Realidad Virtual para la herramienta gráfica SceneToolKit.* La Habana : s.n., 2011.

7. OGRE. [En línea] 2000 - 2009. http://www.ogre3d.org/.

8. **Blanchette, Jasmin y Summerfield, Mark.** *C++ GUI Programming with Qt 4, Second Edition.* 2008.

9. **Navarro, Juan Ignacio Rodríguez.** desarrollomultimedia.es. *Explicamos la diferencia entre vídeo digital y vídeo analógico.* [En línea] http://www.desarrollomultimedia.es/articulos/diferencia-entre-video-digital-y-video-analogico.html.

10. **Profesorado, Instituto Superior de Formación y Recursos en Red para el.** Diseño de Materiales Multimedia_Web 2.0. [En línea] 2008. http://www.ite.educacion.es/formacion/materiales/107/cd/video/video01.html.

11. **Casamor, Antonio Salavert.** [En línea] http://www.tonet.jazztel.es.

12. **López, Ramón Cutanda.** videoedicion.org. *Conceptos básicos de vídeo digital.* [En línea] 2011. http://www.videoedicion.org/documentacion/article/conceptos-basicos-de-video-digital.

13. Círculo de Maquetadores. [En línea] 2012. http://www.circulodemaquetadores.com/codec-video-en-formatos-html-5/.

14. My Format Factory. [En línea] http://www.myformatfactory.com/.

15. Solo Tips. [En línea] 2009. http://purotip.blogspot.com/2010/08/los-formatos-de-video-mas-populares.html.

16. The MPEG Home Page. [En línea] 2012. http://mpeg.chiariglione.org/.

17. CoolUtils.com. [En línea] 1998 - 2012. http://www.coolutils.com/Formats/MOV.

18. Digiarty Software. [En línea] 2010. http://www.winxdvd.com/.

19. All-Streaming-Media.Com. [En línea] 2003 - 2011. http://all-streaming-media.com/faq/streaming-media/Format-Real-Media-RM-files.htm.

20. DivXLand.org. *Formatos Contenedores Multimedia.* [En línea] 2002 – 2006. http://www.divxland.org/esp/container_formats.php.

21. MundoDivX. [En línea] 2002 - 2012. http://www.mundodivx.com/mp4/index.php.

22. DivX. [En línea] 2012. http://www.divx.com/es.

23. kioskea.net. [En línea] http://es.kioskea.net/contents/video/mkv-matroska.php3.

24. Matroska Media Container. [En línea] 2005 - 2011. http://www.matroska.org/.

25. WordPress.com. [En línea] http://haltadefinicion.wordpress.com/2008/10/03/%C2%BFque-es-el-formato-mkv/.

26. Xiph.org, The xiph open sourse community. [En línea] 1994 – 2012. http://www.xiph.org/ogg/.

27. EcuRed. [En línea] http://www.ecured.cu/.

28. Xiph.org, The xiph open sourse community. [En línea] 1994 – 2012. http://www.theora.org/.

29. **Turcan, Peter y Wasson, Mike.** *Fundamentals of Audio and Video Programming for Games.* [Documento .chm] 2004.

30. **Peña, Liudmila Pupo.** *Sistema de visualización estereoscópica para la evaluación y entrenamiento de funciones visuales.* La Habana : s.n., 2011.

31. Ogre Wiki, Support and community documentation for Ogre3D. [En línea] http://www.ogre3d.org/tikiwiki/.

32. FFmpeg. [En línea] http://ffmpeg.org/.

33. **Corporation, Nokia.** Qt. [En línea] 2008 - 2012. http://qt.nokia.com/.

34. DesarrolloWeb.com. [En línea] 2004. http://www.desarrolloweb.com/articulos/1622.php.

35. MasterMagazine. [En línea] 2004. http://www.mastermagazine.info/termino/3916.php.

36. MIKSOFT. [En línea] http://www.miksoft.net/mobileMediaConverter.php.

37. cplusplus.com. [En línea] 2000 - 2012. http://www.cplusplus.com/.

38. **James Rumbaugh, Ivar y Jacobson, Grady Booch.** El Lenguaje Unificado de Modelado. [En línea]

39. Microsoft.com. [En línea] 2012. http://msdn.microsoft.com/es-es/vstudio/default.aspx.

40. **Junker, Gregory.** Pro OGRE 3D Programing. 2006.

41. **González, Rocío Gutiérrez y González, Raúl Sánchez.** *Comparación entre las bibliotecas graficas GTK y Qt.* [En línea] zarza.usal.es/~fgarcia/docencia/poo/01-02/trabajos/S2T4.pdf.

42. **Ivar, Jacobson, Booch, Grady y Rumbaugh, James.** *El Proceso Unificado de Desarrollo de Software.* Madrid : Addison Wesley, 2000. 8478290362.

GLOSARIO DE TÉRMINOS

A

Agudeza visual: Es la capacidad para discriminar detalles finos de un objeto en el campo visual, es la inversa del ángulo desde el cual los objetos son contemplados. La prueba estándar en el mundo para medir la agudeza visual es denominada *ETDRS* (*Early Treatment Diabetic Retinopathy Study*), que es una extensión del test original de Snellen, que se aplicaba desde 1862 [30].

C

Componente: Componente es aquello que forma parte de la composición de un todo. Se trata de elementos que, a través de algún tipo de asociación o contigüidad, dan lugar a un conjunto uniforme.

Convergencia: Es el movimiento hacia dentro de los dos globos oculares para que sus ejes visuales se junten o converjan en el objeto que se mira [30].

D

Dominancia ocular: Se define generalmente como una preferencia monocular por el uso de uno de los ojos cuando la imagen no puede ser fusionada o cuando se requiere visión monocular (ej. al apuntar con un rifle). Es posible determinar en la mayoría de los casos el ojo dominante de una persona [30].

E

Estereopsis: La estereopsis o estereoagudeza corresponde al mejor nivel posible de las funciones binoculares. Es la consecuencia de una buena fusión sensorial [30].

M

Manipular o manipulación: En esta investigación se refiere a realizar acciones de reproducir, pausar, parar, ajustar el volumen, ocultar, mostrar, ajustar el tamaño y ajustar la posición del video que se cargue para ser mostrado ya sea en la textura como en el *widget*.

Módulo: Un módulo (del latín modŭlus) es una pieza o un conjunto unitario de piezas que, en una construcción, se repiten para hacerla más sencilla. Forma parte de un sistema y mantiene algún tipo de relación o vínculo con el resto de los componentes.

P

Plugin: Un plugin es una aplicación informática que añade funcionalidades específicas a un programa principal. Su nombre procede del inglés y su presencia es muy habitual en los navegadores web, en reproductores de música y en sistemas de gestión de contenidos.

S

Sensopercepción: Conocimiento sensorial de una realidad, basado directamente en la información que el individuo recibe de sus sentidos (audición, tacto, visión, gusto y olfato). Es algo muy complejo que en realidad está compuesto por dos procesos que se encuentran muy relacionados, que son la sensación y la percepción de los estímulos [2].