UNIVERSIDAD POLITECNICA DE VALENCIA

etsinf

Escola Tècnica Superior d'Enginyeria Informàtica

INSTITUTO DE BIOMECÁNICA DE VALENCIA

PFC: Animación 3D del Raquis Cervical.
Aplicación para la Valoración de Cervicalgias

Febrero 2010

Pablo Silvestre Soler

Tutor en la empresa: Carlos Chirivella Moreno
Tutor en ETSINF: Javier Lluch Crespo

ÍNDICE

AGRADECIMIENTOS

Quiero dar mi más sincero agradecimiento al personal del Instituto de Biomecánica de Valencia (IBV), centro que me ha permitido desarrollar este proyecto final de carrera en sus instalaciones y me ha formado y guiado para llevarlo a cabo con éxito.

También debo agradecer a mis tutores de proyecto, tanto de la Facultad de Informática (ETSINF) de la Universidad Politécnica de Valencia como del IBV, sus consejos y directrices que han permitido trazar una línea de proyecto marcada por la fluidez en la elaboración de dicha tarea.

Gracias a todos los que han hecho esto posible.

RESUMEN

El proyecto definido en la memoria se engloba dentro del ámbito del diagnóstico a través de una valoración funcional y, de forma más concreta, en la valoración de cervicalgias. Se trata de una herramienta de apoyo que otorga, a la persona encargada de realizar un diagnóstico, de un punto de vista alejado de la rigidez de representación de los datos, pero totalmente fiel a los mismos.

Esto se basa en una gran premisa: una imagen vale más que mil palabras. Extrapolar esta afirmación al proyecto actual es inmediato: una animación en tres dimensiones en muchas ocasiones es mucho más útil de cara al usuario de un programa que un gran compendio de datos numéricos. La forma y velocidad en la que la información es recibida es totalmente distinta. Mientras que para sacar conclusiones de unos datos es preciso realizar un procesamiento y una interpretación de los mismos, una animación permite obtener una conclusión de forma prácticamente instantánea. Sin ser un experto cualificado en la materia a través de una animación se puede comprobar si el movimiento realizado por un paciente es relativamente normal o no, hecho que en algunas ocasiones no es posible al contemplar los datos en primera instancia.

Para que la animación tenga una física realista, se emplean ángulos calculados en función del tiempo, a través de unos marcadores que se le colocan al paciente en puntos estratégicos y que permiten describir de forma matemática los movimientos cervicales. Por tanto mediante la interpolación de estos ángulos, se obtiene un movimiento fluido sobre un modelo en tres dimensiones, que dispone de la estructura necesaria para describir de forma precisa los movimientos del raquis cervical.

Otro aspecto a tener en cuenta es la posibilidad de interacción con la animación. A diferencia de un vídeo en el que sólo es posible cambiar el momento de reproducción, la aplicación añade una variable más: la posibilidad de cambiar la orientación y posición espacial de la cámara que permite la visualización de la animación. De esta forma es posible comprobar desde la mejor posición posible un determinado gesto o movimiento realizado por el paciente sin necesidad de que repita la prueba o se coloque una cámara en la posición de visualización deseada.

En consecuencia y tras establecer las bases que describen el proyecto, los contenidos de la memoria están estructurados siguiendo un orden cronológico y causal argumentando los pasos para la realización del mismo.

En primer lugar se describe el ámbito en el que se encapsula el tema del proyecto elegido (cervicalgias y sistemas de valoración funcional) y se analizan las herramientas que permiten desarrollar dicha actividad de la mejor forma posible.

A continuación, una vez elegidas las herramientas para realizar el proyecto (Ogre como motor gráfico y Blender como modelador 3D), se plantea una serie requerimientos que se deben cumplir que conforman la funcionalidad básica de que dispondrá la aplicación a crear.

El siguiente paso es el análisis y la justificación de diseño, formato, nomenclatura y otros aspectos vinculados con el aspecto y la estructura de la aplicación. En este punto se define la arquitectura general del proyecto y cómo se organiza la información, con el fin último de mejorar la usabilidad y hacerla atractiva para el usuario.

Tras establecer un diseño, se analiza detenidamente y de forma lineal todas las estrategias y pasos seguidos para llevar a cabo la aplicación. Se parte de aspectos y opciones relativas al modelado gráfico con Blender, la utilización del modelo en Ogre y, por último, su vinculación con una interfaz funcional.

Los últimos pasos de la memoria defienden, muestran y analizan los resultados que se pueden obtener con la utilización de esta aplicación. Se muestra la validez de dicha aplicación, basándose en ejemplos para detectar posibles anomalías en pacientes con algún tipo de afección cervical. Posteriormente también se incluyen posibles ampliaciones consideradas y conclusiones a las que se ha llegado tras la realización de la misma.

Por último se incluye el conjunto de documentos y enlaces que se han empleado para llevar a cabo el proyecto, así como un anexo con un manual de instalación de la aplicación y un manual de usuario explicando los aspectos básicos para utilizar eficientemente la aplicación.

1.- Introducción

1.1.- Origen del trabajo

Las cervicalgias, dolor en la región cervical, son una afección muy frecuente en la población actual. Alrededor del 50% de la población adulta sufre en algún momento de su vida una cervicalgia.

Son descritas en los códigos CIE-9 en el epígrafe 723 y según el manual elaborado por la Seguridad Social en el año 2008, sobre tiempos estándar de baja laboral, el tiempo medio que un trabajador permanece de baja en España se sitúa en los 43 días, y en relación a las cervicalgias se sitúa en torno a 20 días, lo que repercute sobre el coste económico que deriva de dicha situación tanto a nivel personal como empresarial.

Las causas para su aparición tienen diferentes orígenes, pero las más frecuentes derivan de mantener una postura incorrecta forzada y mantenida durante mucho tiempo que provoca una alteración funcional que se manifiesta mediante dolor.

Sin embargo hay un gran problema para su diagnóstico: no siempre existe una correlación entre los síntomas clínicos y las manifestaciones radiológicas. Esto implica que una persona con una estructura ósea impecable puede presentar dolores y limitación de movilidad cervical. El origen de este problema radica en deficiencias musculares (sobreesfuerzo, estrés, deficiente higiene postural…). Del mismo modo se pueden encontrar alteraciones patológicas en pacientes que no han manifestado síntomas de cervicalgia, generalmente detectadas al realizar pruebas radiológicas a pacientes asintomáticos.

Por este motivo en muchas ocasiones hay grandes dificultades para determinar un diagnóstico correcto y sobre todo en el ámbito de la patología laboral, a la hora de valorar una incapacidad laboral ya sea transitoria o permanente. Con este propósito principal se ha creado NedCervical/IBV. Se trata de una aplicación que permite la valoración funcional del raquis cervical basándose en el estudio de la cinemática de los movimientos que realiza el sujeto durante unas pruebas. De esta forma el personal encargado de realizar la valoración obtiene más información sobre el estado del paciente pudiendo determinar de una forma más eficiente el grado y la etiología de la afección. Además es posible determinar el grado de colaboración del paciente durante el estudio, por lo que se dispone una herramienta muy útil en la medicina del trabajo para apoyar la decisión de otorgar una incapacidad laboral.

Las bajas laborales suponen una gran pérdida económica para las empresas españolas. Se calcula que en España se producen más de tres mil bajas laborales diarias, lo que ha provocado que se endurezcan las penas contra las bajas debidas a presentar una sintomatología simulada o fingida. Además el trastorno musculo-esquelético sustituyó hace unos años al accidente clásico (caídas, cortes, etc.) como primera causa de baja laboral. Por ello es necesario determinar en la medida de lo posible la veracidad o no de estas dolencias.

NedCervical/IBV es, por tanto, una herramienta muy útil para los profesionales de la salud laboral y en especial para las mutuas aseguradoras, que ayuda a obtener un diagnóstico real de pacientes con posibles cervicalgias.

1.2.- Antecedentes generales

Un análisis del mercado referente a todo lo que conlleva el estudio de las cervicales basado en la cinemática y la capacidad de movimiento de las mismas precisa definir dos aspectos fundamentales: su disposición física, que condiciona los tipos de movimientos que se pueden realizar; y la objetivación de dichos movimientos para obtener patrones que se toman de referencia para estudios biomecánicos. De este modo, estos son los dos puntos de partida: describir la anatomía ósea de la región cervical y definir los sistemas de valoración funcional.

1.2.1.- Información anatómica: Cervicalgias

La columna vertebral está compuesta por 33 o 34 vértebras dispuestas a lo largo del esqueleto axial. Se estructura en 5 segmentos:

- Cervical: 7 vértebras
- Dorsal: 12 vértebras
- Lumbar: 5 vértebras
- Sacro: 5 vértebras
- Coxígeo: 3 ó 4 vértebras

En conjunto se encarga de transmitir y amortiguar las cargas, permitir una relación óptima de grado de movilidad y rigidez, y proteger las estructuras neuronales contenidas en el canal.

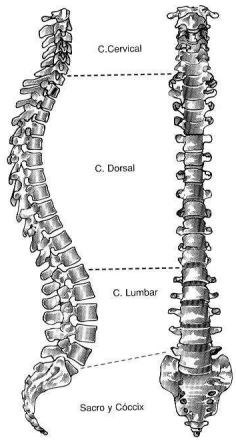


Figura 1: Columna vertebral

Un análisis más profundo de la morfología de las siete vértebras cervicales permite diferenciar entre las vértebras C1, C2 y un grupo con características similares formado por las vértebras de la C3 a la C7. La primera vértebra cervical (C1) o atlas tiene forma de anillo y se articula en su parte superior con el hueso occipital. La segunda vértebra cervical (C2) o axis presenta como característica principal una prolongación ósea vertical que nace del cuerpo del axis y penetra dentro del atlas (apófisis odontoides). Esta particular estructura permite, de manera efectiva, la rotación del cráneo. El resto de vértebras cervicales (C3-C7) poseen todas ellas de un foramen transverso del que carecen las vértebras del segmento dorsal y lumbar.

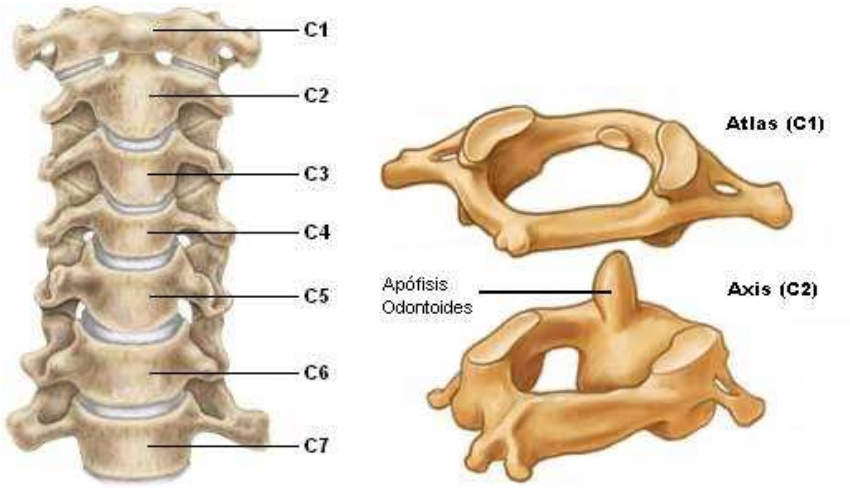


Figura 2: Columna cervical y vértebras C1 y C2

Tras estas básicas nociones anatómicas, se puede definir el concepto de cervicalgia: afección del segmento cervical con síntomas dolorosos. El dolor cervical es una causa frecuente de consulta médica tanto en atención primaria como en asistencia especializada. Sin embargo, la multiplicidad etiológica responsable de los cuadros dolorosos cervicales dificulta en gran medida obtener un diagnóstico correcto de forma eficiente.

Se puede clasificar su etiología en dos ramas atendiendo al estudio del cuadro álgico: dolor mecánico y dolor inflamatorio.

El dolor mecánico se caracteriza por empeorar con la movilización, reproducirse de forma intermitente y mejorar con el reposo funcional. Las causas que pueden condicionar su aparición son diversas: colocarse en una mala postura de forma mantenida, dormir sin un apoyo adecuado de la cabeza, cargar peso con un solo brazo… Todas estas causas generan, en la mayoría de los casos, una contractura muscular que provoca un dolor cervical.

Dentro de este ámbito de la cervicalgia mecánica también se encuentran los traumatismos cervicales. Cabe distinguir entre dos tipos de traumatismos: traumatismos directos y traumatismos indirectos. Los traumatismos directos se producen debido a un golpe producido sobre la región cervical; en cambio los traumatismos indirectos se deben a una sacudida no localizada en la región cervical que genera un movimiento forzado de la misma.

En consecuencia, las lesiones producidas tienen su origen en un movimiento forzado por encima del sector fisiológico de movilidad. Este rango de movilidad está relacionado con la edad del sujeto y con el tipo de movimiento realizado.

Edad	Flexoextensión (flexión + extensión)	Rotación (derecha + izquierda)	Lateralización (derecha + izquierda)
Hasta 30 años	90 (20+70)	90 (45+45)	90 (45+45)
De 31 a 50 años	70	90	90
Más de 50 años	60	90	60

Figura 3: Grados de movilidad normal de la columna cervical

Por tanto, atendiendo al tipo de movimiento realizado se pueden producir lesiones por:

- Extensión
- Flexión
- Compresión
- Rotación
- Cizallamiento

En contraposición, la cervicalgia inflamatoria se presenta en muchas menos ocasiones y requiere un diagnóstico más amplio y complejo. El dolor es generalmente continuo y no cede con el reposo funcional. Las causas de este tipo de cervicalgia son por lo general más graves ya que se deben a una sintomatología inflamatoria originada por un proceso reumático, tumoral o infeccioso.

Sin embargo establecer un diagnóstico de cervicalgia en una de estas dos ramas tiene una dificultad añadida: el dolor cervical puede tener su origen en otras zonas y, del mismo modo, una cervicalgia puede producir dolor en el brazo, región escapular y región pectoral.

1.2.2.- Sistemas de valoración funcional

Los fundamentos conceptuales de la evaluación funcional residen en los "modelos de discapacidad" propuestos por Nagi y Wood. El más conocido es el de la OMS (1980) que distingue entre:

- Deficiencia, toda pérdida o anormalidad de una estructura o función psicológica, fisiológica o anatómica. (*Nivel de órgano*)
- Discapacidad, toda restricción o ausencia de capacidad para realizar una actividad en la forma o dentro del margen considerado normal para el ser humano. Representa, en definitiva, al conjunto de consecuencias motrices o psicológicas de una persona afectadas por una deficiencia. (*Nivel de ámbito global de la persona*)
- Minusvalía, la situación desventajosa para un individuo determinado, consecuencia de una discapacidad, que limita o impide el desempeño de un rol que es normal en función de la edad, el sexo y los factores sociales y culturales. (*Nivel social*)

La evaluación funcional consiste en la medición de las características dinámicas del individuo, incluyendo actividades, habilidades, actuaciones prácticas, condiciones ambientales y necesidades de dicho individuo. El estudio y la interpretación de este conjunto de mediciones, recibe el nombre de valoración funcional y se integra en el bloque de valoración de discapacidades. Este compendio de sistemas y aplicaciones intentan objetivar la valoración de un sujeto concreto realizando una serie de pruebas funcionales o movimientos establecidos. Por tanto su objetivo es servir de apoyo a un experto cualificado para tomar decisiones para establecer un diagnóstico determinado. Su funcionamiento emplea como base métodos estadísticos y conocimientos médicos para realizar comparaciones entre patrones fisiológicos catalogados como normales y los obtenidos en el sujeto de estudio. Esta comparación genera una serie de resultados e informes cuyo destinatario final es el encargado de valorar el diagnóstico del sujeto.

Existen sistemas de valoración funcional aplicados a bastantes tipos de dolencias motrices. Lumbalgias, movimientos articulares de hombro, rodilla y muñeca son algunos entre muchos otros ejemplos para los que se puede encontrar aplicaciones que presentan informes obtenidos a partir de una valoración funcional de los sujetos. A continuación se realiza un análisis de un sistema de valoración funcional aplicado a la región cervical.

NedCerv/IBV
Valoración de Cervicalgias

NedCervical es un sistema diseñado por el Instituto de Biomecánica de Valencia (IBV) que permite la valoración del raquis cervical basado en la cinemática de sus movimientos durante la realización de pruebas funcionales.

Está avalado por el conocimiento de las necesidades de los especialistas clínicos de valoración del daño corporal y tiene el fin principal de proporcionar una prueba complementaria, objetiva, fiable y de fácil interpretación, que permita realizar una valoración más precisa y justa del daño de la columna cervical. Además también permite observar la repercusión funcional en el paciente, así como realizar un seguimiento cercano de su evolución.

Integra la tecnología biomecánica más avanzada de análisis de movimientos humanos para la valoración funcional del raquis cervical y, de forma complementaria, utiliza una base de datos de normalidad a partir de la cual se

permite la comparación de los registros de los pacientes medidos para caracterizar su patrón de movimiento.

Para la obtención de los datos referentes al movimiento de los pacientes se utiliza el sistema de vídeo digital Kinescan/IBV. Este sistema se basa en el registro de muestras de al menos dos cámaras de unos marcadores reflectantes. Aumentar el número de cámaras, mejora la precisión de la obtención de coordenadas 3D y permite evitar carencias ante marcadores que en algún momento se encuentren ocultos a la visión de dichas cámaras.

Los marcadores empleados son simples esferas de corcho cubiertas de un material reflectante. Al tratarse de superficies esféricas, hay que tener en cuenta los ángulos de reflexión de la luz: ante una única luz focal, la zona del marcador que refleja depende de la posición del mismo. Para evitar este problema, las cámaras incorporan en la parte superior luces infrarrojas orientadas con la misma distribución de ejes de la cámara. Así la reflexión siempre se sitúa en el centro del marcador desde el punto de vista de cada cámara. Para obtener resultados óptimos es conveniente controlar las condiciones de luz que afectarán a la sala.

Figura 4: Sistema general Kinescan/IBV

Otra característica a tener en cuenta en el diseño de los marcadores es la posición de los mismos en el paciente. Si se sitúan ceñidos a la cabeza, los movimientos mínimos realizados requieren de cámaras de alta sensibilidad para detectarlos. Sin embargo si se les añade un factor como la distancia, ante el mismo movimiento de la cabeza la posición de los marcadores varía en mayor medida, siendo más sencilla su detección.

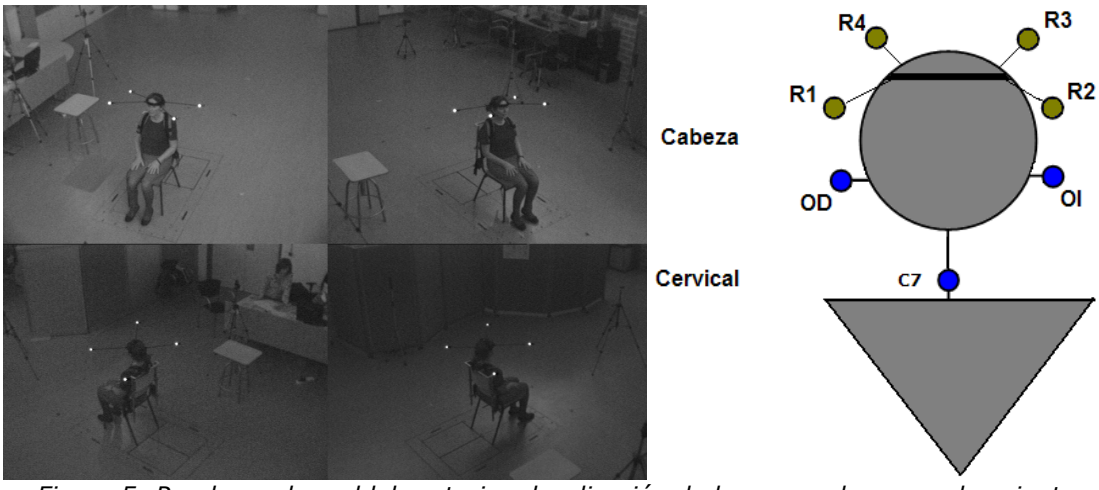
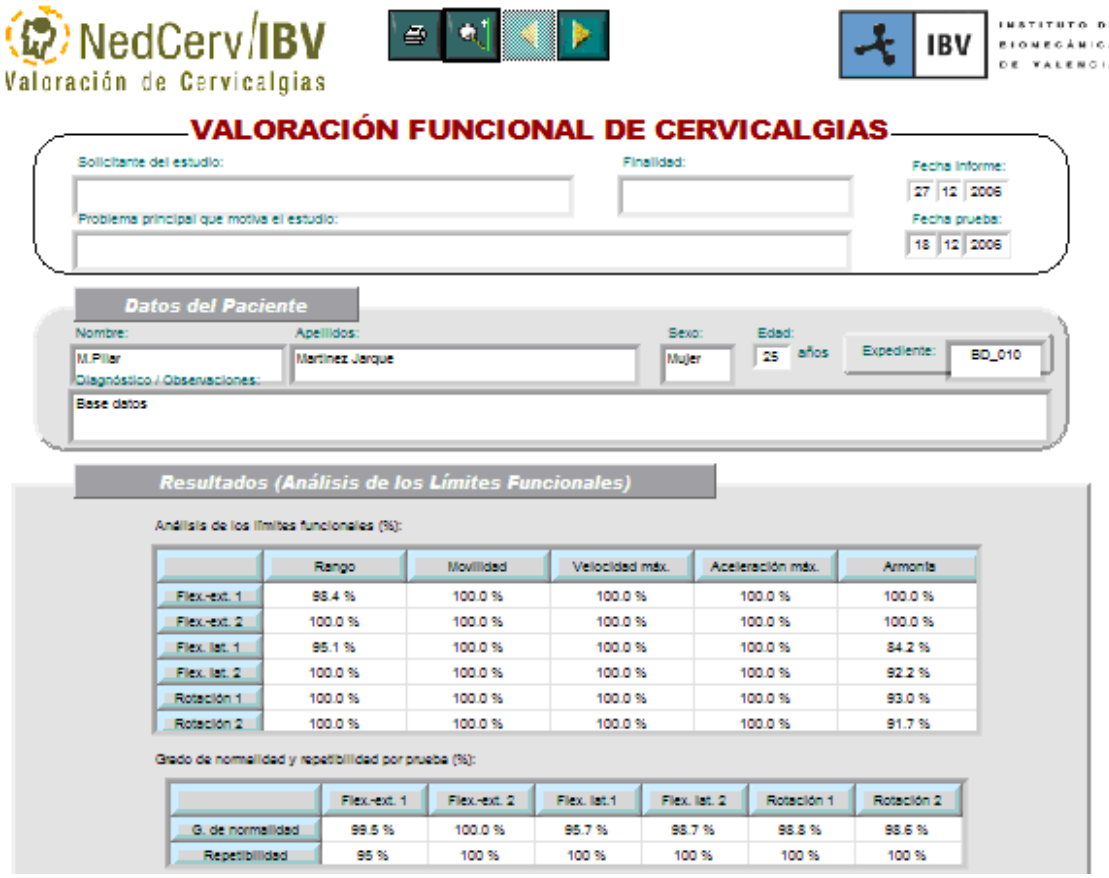


Figura 5: Prueba real en el laboratorio y localización de los marcadores en el paciente

El sistema registra y analiza dos tipos de gestos:

- Los límites funcionales del movimiento en cada una de las direcciones del espacio (flexo-extensión, flexión lateral y rotación).
- Determinados gestos cuyo objetivo es simular movimientos de la vida diaria que comprometen a la columna cervical (en este caso, dichos gestos consisten en mirar hacia arriba hasta observar una figura sobre una lámpara que se ilumina para, posteriormente, cumplimentar un sencillo cuestionario).

Finalmente incorpora un método de clasificación estadístico, basado en regresión logística, que obtiene resultados mediante la comparación de la cinemática de movimientos del paciente con la base de normalidad. De esta forma, es capaz de diferenciar con alta sensibilidad y especificidad entre sujetos normales y patológicos, y entre sujetos colaboradores en la valoración y no colaboradores, con una elevada probabilidad de acierto. Todo el estudio realizado se recoge en un informe personalizado específicamente diseñado.


NedCerv/IBV
Valoración de Cervicalgias
IBV
INSTITUTO DE BIOMECÁNICA DE VALENCIA
VALORACIÓN FUNCIONAL DE CERVICALGIAS
Solicitante del estudio:
Finalidad:
Fecha informe: 27 12 2006
Problema principal que motiva el estudio:
Fecha prueba: 18 12 2006
Datos del Paciente
Nombre: M.Pilar
Apellidos: Martínez Jarque
Sexo: Mujer
Edad: 25 años
Expediente: BD_010
Diagnóstico / Observaciones:
Base datos
Resultados (Análisis de los Límites Funcionales)
Análisis de los límites funcionales (%):

	Rango	Movilidad	Velocidad máx.	Aceleración máx.	Armonía
Flex-ext. 1	98.4 %	100.0 %	100.0 %	100.0 %	100.0 %
Flex-ext. 2	100.0 %	100.0 %	100.0 %	100.0 %	100.0 %
Flex. lat. 1	95.1 %	100.0 %	100.0 %	100.0 %	84.2 %
Flex. lat. 2	100.0 %	100.0 %	100.0 %	100.0 %	92.2 %
Rotación 1	100.0 %	100.0 %	100.0 %	100.0 %	93.0 %
Rotación 2	100.0 %	100.0 %	100.0 %	100.0 %	91.7 %

Grado de normalidad y repetibilidad por prueba (%):

	Flex-ext. 1	Flex-ext. 2	Flex. lat.1	Flex. lat. 2	Rotación 1	Rotación 2
G. de normalidad	99.5 %	100.0 %	95.7 %	98.7 %	98.8 %	98.6 %
Repetibilidad	95 %	100 %	100 %	100 %	100 %	100 %

Figura 6: Informe personalizado que emite la aplicación NedCervical/IBV (parte 1)

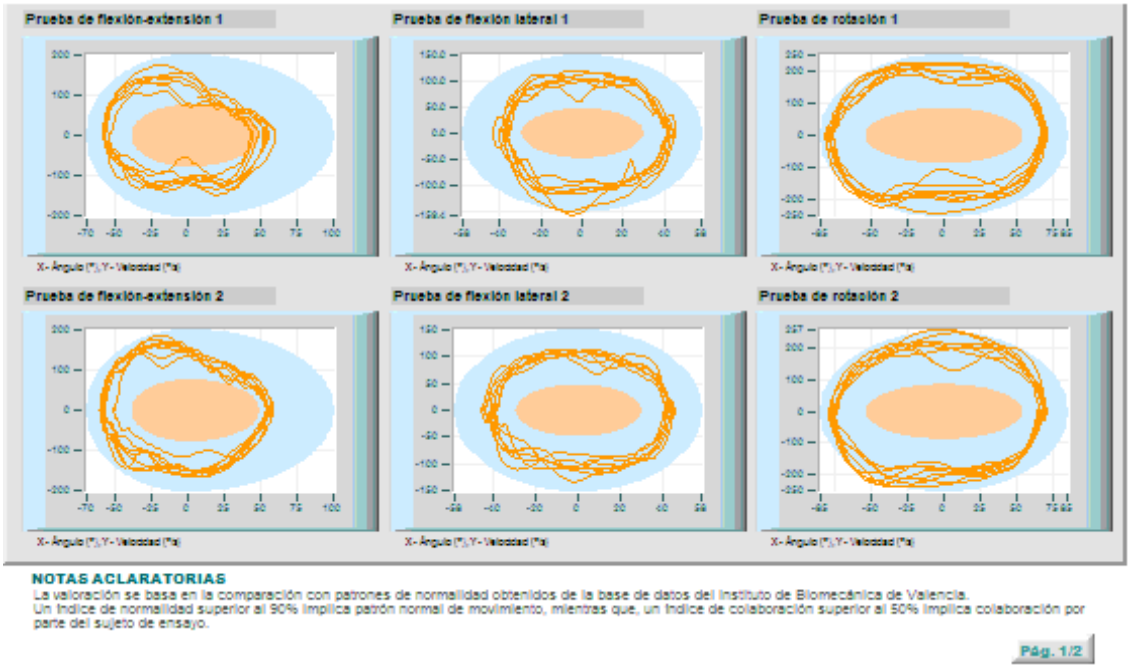


Figura 7: Informe personalizado que emite la aplicación NedCervical/IBV (parte 2)

La utilización del sistema de vídeo digital Kinescan/IBV como tecnología base para el análisis de movimientos de la columna cervical, proporciona gran versatilidad frente a cualquier tipo de estudios de análisis de movimientos que puedan realizarse en un futuro inmediato. Así por ejemplo, estudios de nuevos protocolos de valoración de rodilla o de hombro, en los que el IBV trabaja actualmente, podrán ser llevados a cabo utilizando la misma instrumentación base.

1.3.- Antecedentes técnicos

La aplicación NedCervical, en la cual se apoya el proyecto desarrollado, está programada en C++ y se ha considerado como requisito de partida.

C++ es un lenguaje de programación diseñado a mediados de los años 1980 por Bjarne Stroustrup. Se creó como una extensión del lenguaje de programación C con mecanismos para la manipulación de objetos. En ese sentido, desde el punto de vista de los lenguajes orientados a objetos, el C++ es un lenguaje híbrido.

Una característica particular de C++ es la posibilidad de redefinir los operadores (sobrecarga de operadores) y la posibilidad de crear nuevos tipos que se comporten como tipos fundamentales.

En cuanto a las herramientas de modelado y animación 3D se han analizado las siguientes:

- Modeladores gráficos 3D: crear y diseñar el modelo 3D.
  - Blender
  - Maya
  - 3ds Max

- Motores gráficos 3D: animación del modelo 3D ante unos datos de entrada.
  - Ogre
  - 3D GameStudio
  - Crystal Space

A continuación se va a realizar un estudio de las características principales de cada una de las herramientas de modelado y motores gráficos mencionados.

1.3.1.- Modeladores gráficos 3D

1.3.1.1.- Blender

Blender es un programa informático multiplataforma de software libre, dedicado especialmente al modelado, animación y creación de gráficos tridimensionales.

Las características principales de este programa son:

- Multiplataforma, libre, gratuito y con un tamaño de origen realmente pequeño comparado con otros paquetes de 3D, dependiendo del sistema operativo en el que se ejecuta.
- Capacidad para una gran variedad de primitivas geométricas, incluyendo curvas, mallas poligonales, vacíos, NURBS y metaballs.
- Junto a las herramientas de animación se incluyen cinemática inversa, deformaciones por armadura o cuadrícula, vértices de carga y partículas estáticas y dinámicas.
- Edición de audio y sincronización de video.
- Características interactivas para juegos como detección de colisiones, recreaciones dinámicas y lógica.
- Posibilidades de renderizado interno versátil e integración externa con potentes trazadores de rayos o "raytracer" libres como kerkythea, YafRay o Yafrid.
- Lenguaje Python para automatizar o controlar varias tareas.
- Blender acepta formatos gráficos como TGA, JPG, Iris, SGI o TIFF. También puede leer ficheros Inventor.
- Motor de juegos 3D integrado, con un sistema de ladrillos lógicos. Para más control se usa programación en lenguaje Python.
- Simulaciones dinámicas para softbodies, partículas y fluidos.
- Modificadores apilables, para la aplicación de transformación no destructiva sobre mallas.
- Sistema de partículas estáticas para simular cabellos y pelajes, al que se han agregado nuevas propiedades entre las opciones de shaders para lograr texturas realistas.

Dispone de una interfaz gráfica poco intuitiva y no basada en el clásico sistema de ventanas, por lo que ha sido bastante criticada. Sin embargo permite la personalización absoluta facilitando el trabajo y la disposición de los menús para el usuario.

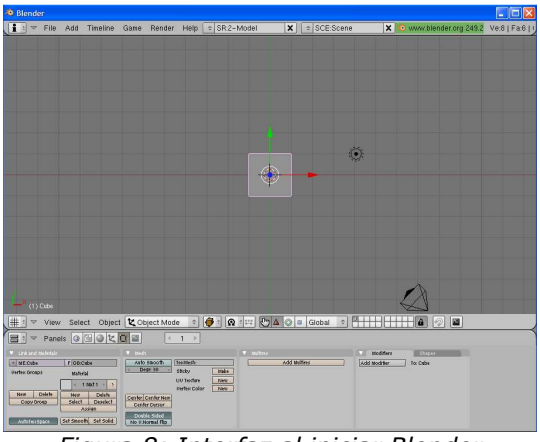
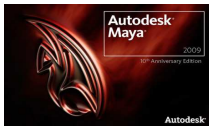


Figura 8: Interfaz al iniciar Blender

Blender posee una herramienta muy útil para aumentar la personalización: incorpora un intérprete de Python totalmente funcional. Esto le permite a cualquier usuario añadir funcionalidades a Blender escribiendo un simple script de Python.

1.3.1.2.- Maya



Maya es un programa que proporciona potente solución integrada para el modelado, la animación, la creación de efectos visuales y la renderización en 3D distribuido por la empresa Autodesk. Está basado en una arquitectura abierta que permite programar o generar archivos de comandos para todo su trabajo con una API (interfaz de programación de aplicaciones) completa y bien documentada, mediante uno de los dos lenguajes de comandos incorporados: Maya Embedded Language (MEL) o Python. Este carácter abierto, combinado con un conjunto de herramientas 3, le ayuda a hacer realidad una visión creativa para cine, televisión, desarrollo de juegos y proyectos de diseño.

Las características principales de la versión 2009 de Maya Autodesk son:

- Programa multiplataforma implementado por Autodesk con una experiencia de 10 años respecto de la salida al mercado de la primera versión.
- Se caracteriza por su potencia y las posibilidades de expansión y personalización de su interfaz y herramientas. Está diseñado para permitir incorporar software de terceros, el cual puede cambiar completamente la apariencia de Maya.
- Se comercializa en dos variantes: "Maya Complete" (una versión básica que incluye los módulos de modelado, animación, render, dinámicas/partículas) y "Maya Unlimited" (la versión más avanzada, que dispone de los módulos de la versión "Maya Complete", más los de Fluids, Hair, Cloth, el nuevo NCloth etc.).
- Maya trabaja con cualquier tipo de superficie NURBS, Polygons, Subdivision Surfaces e incluye la posibilidad de convertir entre todos los tipos de geometría.
- Respecto a anteriores versiones, las novedades incorporadas por la versión estudiada son:
    - Un sistema de selección flexible y un resaltado de preselección que implica un uso más eficiente y sencillo del mismo.
    - Mejor flujo de modelado: ofrece mejoras de modelado intuitivas y productivas, como un modelado simétrico con uniones flexibles, un modo de ajuste para realizar modificaciones rápidas y una nueva fusión de vértices que permite combinar partes de una malla.
    - Maya Assets: es una herramienta que permite organizar, compartir, referenciar y presentar eficazmente los datos complejos.
    - Animación por capas: un nuevo y potente paradigma de animación por capas, basado en tecnología de Autodesk MotionBuilder, aumenta la flexibilidad al crear y editar animaciones de una forma no destructiva.
    - Maya nParticles: basado en el marco de simulación unificado Maya Nucleus, este conjunto de herramientas ofrece un flujo de trabajo intuitivo y eficiente para simular gran variedad de efectos complejos, incluidos líquidos, nubes, humo, aerosol y polvo.
    - Maya Muscle: permite crear movimientos realistas de músculos y piel.
    - Mejoras en la disposición UV: nuevas opciones de despliegue y disposición de UV y optimización de su funcionamiento.
    - Render Proxy: permite sustituir elementos de la escena por una simple malla de baja resolución y cargar sólo los datos reconvertidos cuando hacen falta para renderizar.
    - Mejoras en Render Pass: permite un control exacto de la salida de renderización.

1.3.1.3.- 3ds Max

3ds Max, anteriormente conocido como 3D Studio Max, es un programa de creación de gráficos y animación 3D desarrollado por Autodesk. Aunque creado por la misma compañía que Maya, tienen orígenes distintos y se presumía que se iban a fusionar en un híbrido; sin embargo se ha mantenido como un producto independiente.

Es uno de programas de animación 3D más utilizados. Aunque generalmente se encuentra vinculado a desarrolladores de videojuegos, se emplea también en el desarrollo de proyectos de animación como películas o anuncios de televisión, efectos especiales y en arquitectura.

Las características principales de la versión 2009 de este programa son:

- Sólida capacidad de edición, una omnipresente arquitectura de plugins y una larga tradición en plataformas Microsoft Windows.
- La primera versión data de 1990, lo que contribuye a un grado de perfeccionamiento muy alto por la gran experiencia acumulada.
- Además de las características básicas de cualquier herramienta de modelado, da soporte a sombreadores avanzados (como la oclusión del ambiente y la dispersión de superficies), simulación dinámica, sistemas de partículas, creación y renderizado de mapas…
- Dispone de una interfaz intuitiva y totalmente personalizable y un lenguaje de script propio.
- Está abierto para incorporar plugins de renderización de terceros, tales como V-Ray, Brazil r / s, Maxwell Render y finalRender.
- Las mejoras respecto a versión anteriores son:
    - Renderización con Reveal: este sistema proporciona el control exacto necesario para refinar rápidamente las renderizaciones. Permite renderizar una escena entera excepto un determinado objeto, renderizar un único objeto o incluso una región específica del búfer de fotograma.
    - Mejoras de Biped: esta técnica permite animar de forma más sencilla personajes cuadrúpedos.
    - Mejor compatibilidad con OBJ y FBX: la mayor fidelidad al convertir el formato OBJ, junto con más opciones de exportación, facilitan el intercambio de datos entre 3ds Max y Mudbox, además de otras aplicaciones de esculpido digital. También ofrece mejor gestión de memoria de FBX y nuevas opciones de importación que garantizan la interoperabilidad entre 3ds Max y otros productos de Autodesk, como Maya y MotionBuilder.
    - Edición rápida de texturas UV: incorpora avanzadas herramientas de mapeado fáciles de usar. La nueva función de mapeado de spline permite mapear objetos tubulares y tipo spline, como una carretera sobre un terreno.
    - Compatibilidad con .NET en el SDK que permite utilizar las API de interfaz de usuario de alto nivel de Microsoft para ampliar el software.
    - ProMaterials: 3ds Max cuenta con una nueva biblioteca de materiales basados en la física y fáciles de utilizar.
    - Mejoras de iluminación fotométrica.

1.3.2.- Motores gráficos 3D

1.3.2.1.- Ogre

Ogre (Object-oriented Graphics Rendering Engine) es una librería gráfica de código abierto multiplataforma con un flexible motor gráfico. Está diseñada con el objetivo principal de permitir a los programadores producir aplicaciones utilizando gráficos en 3D acelerados por hardware.

Sin embargo, para aplicar otras características, como sonido, networking, AI, colisión, leyes físicas… se necesita vincular Ogre con otras librerías. Es, por tanto, un motor gráfico base que provee un API muy amigable, apto para integraciones, permitiendo que el programador escoja otras librerías si así lo desea. Está construido de forma que no se compromete con una API en particular, puesto que el motor soporta tanto el uso de DX9 como de OpenGL. Teniendo en cuenta estos aspectos, no existen límites intrínsecos del motor, pudiendo resultar algo confuso en un primer momento.

El equipo programador base encargado del desarrollo de este motor es un grupo pequeño, compuesto por ingenieros de software experimentados. Además la creciente comunidad de usuarios de Ogre, unida a la creación de parches por parte de ésta, permite crear una estructura fuerte entorno a este motor. Cada parche es sometido a una estricta prueba para confirmar su calidad y coherencia con el resto de los componentes del motor.

Otro aspecto a comentar es que Ogre no asume qué tipo de programa (simulación, juego, demo…) se desea crear de antemano. Utiliza una flexible jerarquía de clases, la cual permite diseñar plugins para especializar la organización de escenas con el fin de crear cualquier tipo de programa que se desee.

En definitiva, Ogre posee las siguientes características principales:

- Diseño orientado a objetos. Interfaz simple y fácil de usar, diseñada para que requiera poco esfuerzo el renderizado de escenas en tres dimensiones.
- Arquitectura basada en plugins muy flexible que permite extender las funcionalidades del motor.
- Sistema de Carga/Respaldo. Soporte de zip/pk3 para archivar.
- Diseño limpio y bien documentado de las clases del motor.
- Independiente de la API gráfica, se puede utilizar OpenGL o DirectX.

Además también ofrece soporte en los siguientes elementos de un motor gráfico:

- Renderizado
  - Material LOD.
  - Soporta la gama completa de operaciones de función fija como multitextura y multipass.
  - Blending y coordinación de generación de texturas.
  - Soporte para múltiples técnicas de materiales.
  - Objetos transparentes gestionados automáticamente.
  - Sistema de fuentes con fuentes TrueType y texturas pre creadas.
  - Sistema de GUI 2D con botones, listas, cajas de edición, barras de desplazamiento, etc. (Usando CEGUI).

- Gestión de escena
  - Altamente personalizable. Flexible gestión de escena no vinculada a ningún tipo de escena. Se puede usar las clases predefinidas o crear subclases propias para obtener un control total sobre la organización de la escena.
  - Grafo de escena jerárquico.
  - BSP, Octrees, Occlusion Culling y LOD.

- Texturizado
  - Básico, multi-texturizado, bumpmapping, mipmapping, texturas volumétricas y proyectadas.
  - Texturizado proyectivo automático entre vínculos con texturas unitarias a instancias Frustum.
  - Puede registrar texturas de fuentes externas.
  - Soporta PNG, JPEG, TGA, BMP y DDS como archivos de imagen.

- Iluminación
  - Por vértice, por pixel y Lightmapping.
  - Puede tener un número ilimitado de luces en la escena.
  - Soporte a través de programas de vértices y de fragmentos.

- Sombras
  - Shadow mapping y shadow volume.
  - Técnicas soportadas: modulative stencil, additive stencil y modulative projective.
  - Múltiples esténcils para optimizaciones de sombras, incluyendo programas de vértices de extrusión, luz y sombras inteligentes, integración con la malla LOD, métodos zpass y zfail, esténcils de doble cara y saturación de recorte de región.
  - Textura sombras que se desvanecen a larga distancia.

- Shaders
  - Vertex y pixel shaders de alto nivel.
  - Soporta programas de vértices y de fragmentos de bajo nivel escritos en ensamblador, y programas de alto nivel en Cg, HLSL, y GLSL.

- Animación
  - Cinemática inversa, animación esquelética y animación de mezcla.
  - Animación del esqueleto, incluyendo la mezcla de múltiples animaciones y de peso variable de skinning de huesos.

- Mallas
  - Cargado de malla, skinning y progresivo.
  - Aceleración de skinning por hardware.
  - Flexibilidad en los formatos de malla de datos aceptados.
  - Exportadores para muchas herramientas de modelado incluidas Milkshape3D, 3D Studio, Max, Maya, Blender y Wings3D.

- Curvas y superficies
  - Splines.
  - Caminos Bezier bicuadrados para superficies curvas

- Efectos especiales
  - Cartografía de medio ambiente, billboarding, sistema de partículas, motion blur, cielo, agua y niebla.

  - Sistemas de partículas, incluyendo emisores fácilmente extensibles y affectors (personalizable mediante plugins). Los sistemas pueden ser definidos mediante scripts para un ajuste fácil.
  - Soporte para skyboxes, skyplanes y skydomes.

- Scripting
  - El lenguaje de script permite mantener los materiales fuera del código mediante archivos .material.
  - Scripts de renderizado multipaso.

- Física
  - Física básica, detección de colisiones y cuerpos rígidos.
  - Controladores que permiten organizar fácilmente los valores entre los objetos derivados.
  - Incluye bindings para múltiples sistemas de colisión/física de terceras partes, como ODE o Newton.

1.3.2.2.- 3D GameStudio

3D GameStudio es un kit de desarrollo para la realización de aplicaciones 2D y 3D. Está provisto de un motor 3D, un motor 2D, un editor de niveles y modelos, compilador de scripts y librerías de modelos, texturas, etc. El principal objetivo que esta aplicación persigue es que el creador del juego no necesite ser un programador experimentado. Aboga por reducir al máximo el esfuerzo del creador a costa de perder flexibilidad en el diseño del juego.

Ofrecen tres posibilidades para crear un juego:
- Juegos diseñados a base únicamente de ratón, para usuarios sin conocimientos de programación.
- Juegos o efectos diseñados con algo de programación utilizando C-scripts.
- Juegos o efectos programados en C++ o Delphi, para programadores con experiencia.

Las características de renderizado principales que implementa esta motor son:
- Modelado
  - LOD geométrico.
  - Modelado de terreno.
- Visibilidad
  - Árboles BSP, portales y PVS.
- Iluminación
  - Fuentes de luz estáticas y dinámicas.
  - Sombras estáticas y dinámicas.
- Texturas
  - Texture Mapping y procedural para agua y lava.
  - Mip-mapping trilineal.
- Efectos
  - Partículas, niebla coloreada y transparencia.
  - Transformación y deformación suave de mallas.
- Otros
  - Efecto de movimiento lento/rápido.
  - Detección de colisiones.

Otras características de este motor son:
- Importa multitud de formatos de ficheros gráficos y de audio.
- Requiere DirectX8 o superior para hacer rendering por hardware.
- Requiere DirectX5 o superior para hacer rendering por software.
- No hay planeado soporte para OpenGL.

En cuanto a la posibilidad de incorporar archivos de algún modelador 3D, permite incorporar un plugin para crear modelos animados creados con 3ds Max.

1.3.2.3.- Crystal Space

Cristal Space es un kit de desarrollo de juegos 3D con licencia LGPL y portable escrito en C++. Se puede ejecutar en multitud de sistemas (GNU/ Linux, Windows, Windows NT, OS/2, BeOS, NextStep, OpenStep, MacOS/X Server, DOS, Macintosh…) y da soporte a soporte para visualización a 8-bits, 16-bits y 32-bits, Direct3D, OpenGL, Glide, etc.

Cristal Space, además, soporta seis grados de libertad, luces de colores, mipmapping, portales, espejos, transparencias, superficies reflectivas, sprites 3D (basados en frames o animaciones de esqueleto), texturas procedurales, radiosidad, sistemas de partículas, halos, niebla volumétrica, lenguaje de script (Python y otros).

Las características principales de renderizado que incorpora son:

- Modelado
  - Motor de terreno.
  - Motor de física.
  - Mallas 3D con animación. Conversores desde los formatos Quake MDL and Quake II MD2 a Crystal Space. Importadores de objetos 3DS, MDL, MD2, OBJ, POV, y ASE. Las mallas son multirresolución "progressive meshes" permitiendo LOD dinámicos.
  - Permite representar superficies curvas (nurbs, Bezier…).
  - Modelado LOD general.

- Visibilidad
  - Sistema de visibilidad basado en la combinación de portales, octrees, arboles BSP y c-buffer (coverage buffer).
  - Futuro trabajo en PVS.

- Iluminación
  - Cielo iluminado dinámicamente, sol en movimiento.
  - Espejos.
  - Superficies brillantes y con reflejos con espejos y alpha mapping.
  - Luces estáticas de colores con sombras reales (sombras precalculadas).
  - Luces dinámicas, de colores con sobras suaves.
  - Radiosidad precalculada sobre los ligthmaps.
  - Niebla volumétrica.
  - Luces con halo y lens-flares.

- Texturas
  - Texturas de cualquier dimensión y formatos GIF, TGA, PNG, BMP, JPG y otros.
  - Corrección perspectiva con interpolación cada 16 pixels.
  - Texturas con canal alpha.
  - Mipmapping.
  - Soporte para texturas dinámicas.
  - Multitexuras con OpenGL.

- Efectos
  - 2D sprites y sistemas de partículas utilizando esos sprites.

- Otros
  - Plugins para fuentes.
  - Sistema de detección de colisiones jerárquico.
  - Soporte para sonido 3D (DS3D, EAX, A3D) en varios formatos: WAV, MP3, Ogg/Vorbis, AU, AIFF, IFF y MOD.
  - Movimiento de objetos con un lenguaje de script.
  - Soporte de red simple para Windows, GNU/Linux y Unix (basado en sockets).

En cuanto a la posibilidad de importar modelos realizados con algún modelador 3D, Crystal Space incorpora un conversor de 3ds y permite añadir varios scripts importar mapas y modelos realizados en Blender.

1.4.- Objetivos del trabajo

El objetivo principal del proyecto actual es generar una aplicación, preparada para ejecutarse en un entorno Windows, que otorgue al usuario responsable de detectar la afección de cervicalgia de una visión amigable de la prueba realizada, más allá de datos o gráficas, mediante una animación 3D. El hecho de disponer de un medio con el que mediante un breve vistazo permita recordar o visualizar los movimientos realizados en la prueba del paciente es de gran ayuda para obtener un diagnóstico de una forma rápida y eficaz.

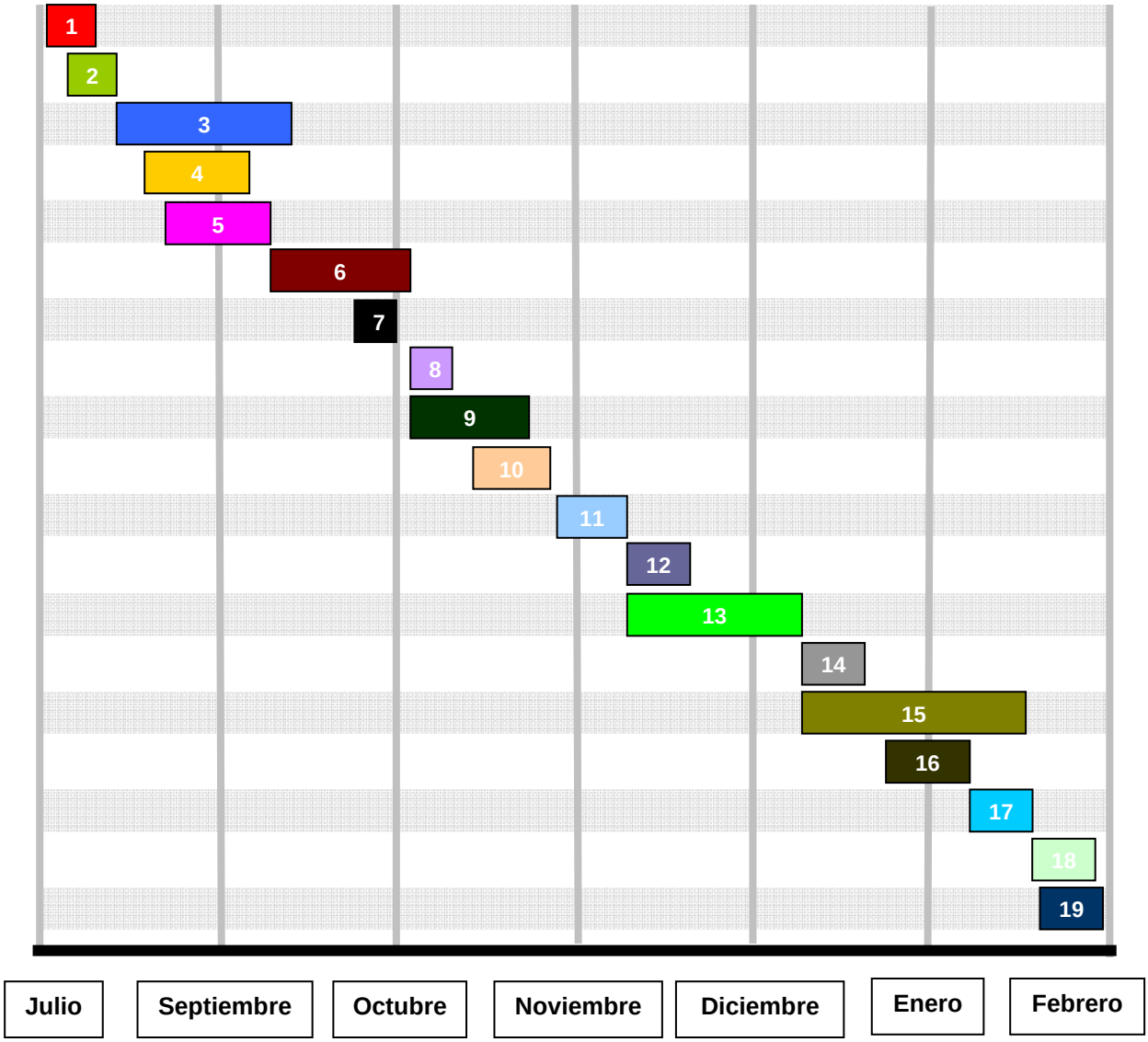
Por ello las herramientas a utilizar deben permitir determinar opciones en cuanto a la visualización de la animación. El usuario elige la posición en que comienza la reproducción de la animación y debe poder situar el cursor en el momento que considere más oportuno, al igual que cambiar la orientación y posición de la cámara que enfoca al modelo 3D para detectar un determinado movimiento de la forma más precisa posible.

La animación 3D estará acompañada de una gráfica que, además de representar los ángulos en los tres ejes principales (x,y,z) en función del tiempo, permita establecer una referencia objetiva a la animación presentada a través de la representación numérica de los resultados de las pruebas. Además es interesante incluir elementos interactivos en la gráfica que afecten al momento de la reproducción de la animación, tanto para facilitar la usabilidad de la aplicación como para hacerla más atractiva y dinámica.

1.5.- Plan de trabajo

Para la realización de este proyecto se ha diseñado un plan de trabajo estructurado respondiendo a las necesidades de estudio de las herramientas a utilizar, la implementación de código de módulos independientes, la integración de los nombrados módulos, el análisis de pruebas de test sobre la aplicación y la elaboración de la documentación para registrar el proceso seguido. El proyecto se plantea entre los meses laborales de julio de 2009 y enero de 2010.

Plan de trabajo



Leyenda

1 Presentación del proyecto a desarrollar y de las herramientas a emplear
2 Establecimiento de los objetivos a alcanzar
3 Estudio del motor gráfico Ogre
4 Estudio del funcionamiento de Blender
5 Adaptación del modelo proporcionado en Blender y exportación del mismo
6 Elaboración de la animación mediante una ventana de Ogre independiente
7 Adición de materiales básicos al modelo 3D
8 Validación de la animación por parte del IBV
9 Estudio de las clases MFC sobre Visual Studio
10 Implementación de la base de la interfaz
11 Integración de la ventana de Ogre en la interfaz base
12 Validación de la interfaz base por parte del IBV
13 Creación de la gráfica y elaboración del aspecto final de la interfaz
14 Validación de la aplicación por parte del IBV
15 Elaboración de la memoria
16 Mejora del modelo de Blender empleado y añadido de texturas
17 Maquetación del proyecto para la entrega
18 Validación final por parte del IBV
19 Entrega del proyecto

2.- Material y métodos

La aplicación NedCervical/IBV en la que se basa el proyecto actual permite al especialista clínico, como prueba complementaria, consolidar su diagnóstico. Para ello proporciona todo tipo de indicadores, parámetros y gráficos relativos a la prueba. Una animación 3D realista de las diferentes pruebas permitiría al especialista observar desde cualquier punto de vista alteraciones en los movimientos realizados y contribuiría a una mejora significativa del valor clínico de la prueba.

De esta forma las herramientas que se han seleccionado en la aplicación son: Blender, para la elaboración física del modelo; Ogre, para obtener una animación del mismo ante unos datos de entrada; y el entorno de desarrollo que proporciona Visual Studio 2005.

Modelado gráfico 3D: ¿Por qué Blender?

Entre los modeladores gráficos estudiados, Blender ofrece una característica muy valorada actualmente: se trata de software libre. Este dato no se refiere únicamente al precio de adquisición del producto, gratuito por supuesto, sino también a la comunidad que lo envuelve. Un programa con código abierto permite que cualquiera tenga la capacidad de añadirle scripts o elementos nuevos y ofrecerlos al resto de interesados con un beneficio común: aumentar las capacidades y perfeccionar el programa para que esté en constante evolución ante nuevas tecnologías.

Además del hecho comentado de presentar una comunidad bastante numerosa, que también implica el disponer de un gran número de tutoriales explicativos, Blender en cuanto a posibilidades de diseño no tiene nada que menospreciar a los otros modeladores analizados.

Otro motivo para su elección como modelador en el proyecto actual son las posibilidades de exportación y personalización que incorpora. Mediante scripts realizados en Python es posible aumentar las funcionalidades que incorpora el programa básico, generando un gran número de posibilidades de utilización alternativas. En el proyecto actual se ha añadido el script *Ogre Meshes Exporter*.

La versión del programa elegida es la 2.49a, a la cual se ha incorporado la versión 2.6.2 de Python con el fin de añadir scripts a Blender. Al ejecutar Blender se puede comprobar la versión de Python instalada en la ventana de comandos adjunta.

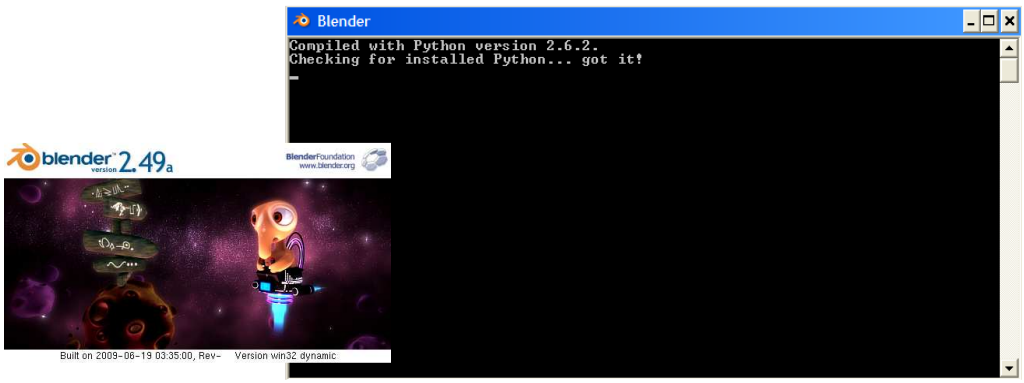


Figura 9: Versión de Blender empleada y ventana de comandos al iniciar Blender

Motor gráfico 3D: ¿Por qué Ogre?

Los motivos por los que se ha elegido este motor gráfico y no otro de los analizados son similares a los de la elección de Blender: se trata de software libre con una comunidad numerosa y activa. Aunque el motor gráfico Crystal Space también pertenece al mundo de software libre, la comunidad que lo representa es bastante más reducida que la de Ogre, lo que implica que exista una menor cantidad de actualizaciones y posibles ampliaciones a incorporar al motor.

Además otra razón para su elección es el hecho de que el modelador elegido permita exportar archivos que puedan ser empleados por el motor gráfico. Blender cumple con este perfil, justificando la elección de ambas herramientas.

Otro aspecto a comentar que ha facilitado emplear Ogre en el proyecto actual es el soporte al lenguaje C++, requisito con el cual parte el proyecto descrito. Además sin en un futuro se deseara ampliar las funcionalidades de la aplicación, la estructura en la que se dispone Ogre permitiría realizar este paso de una forma más o menos gradual, debido en gran medida a su programación abierta a integrar nuevas funcionalidades y librerías.

La versión de Ogre elegida en el proyecto actual es la 1.6.3.

Entorno de desarrollo: ¿Por qué Visual Studio 2005?

Elegidos el modelador y motor gráficos, puesto que la aplicación va a utilizar el lenguaje C++ se ha escogido uno de los entornos de desarrollo más utilizados del mercado ante tales requisitos: Visual Studio 2005. La versión empleada depende de la disposición de programas que ofrece la empresa en la que se ha realizado el proyecto.

Microsoft Visual Studio 2005 es un entorno de desarrollo integrado para sistemas operativos Windows. Soporta varios lenguajes de programación tales como C++, C#, J#, ASP.NET y Visual Basic .NET, aunque actualmente se han desarrollado las extensiones necesarias para muchos otros.

Visual Studio permite a los desarrolladores crear aplicaciones, sitios y aplicaciones web, así como servicios web en cualquier entorno que soporte la plataforma .NET (a partir de la versión net 2002). Así se pueden crear aplicaciones que se intercomuniquen entre estaciones de trabajo, páginas web y dispositivos móviles.

Este entorno permite la creación de proyectos separando cada uno de los recursos de que se dispone: archivos de encabezado (archivos *.h*), archivos de código fuente (*.cpp* para el caso de lenguaje C++) y otros archivos de recursos.

La versión elegida en este proyecto es Microsoft Visual Studio 2005 Professional Edition.

El entorno de desarrollo elegido presenta multitud de herramientas que se le pueden incorporar para aumentar sus funcionalidades. La aplicación a desarrollar requiere dotarla de una interfaz funcional y de la visualización de unos datos en gráficas. Visual Studio ofrece soluciones para ambos aspectos: la utilización de clases MFC y el empleo del paquete de clases Measurement Studio.

- **MFC** (Microsoft Foundation Classes) es un conjunto de clases que provee un acceso más sencillo a las API de Windows. Fueron introducidas por Microsoft en 1992 y desde entonces fueron apareciendo nuevas versiones con las nuevas versiones del entorno de programación Visual C++.

  La utilización de clases MFC permite la creación y manipulación de forma sencilla de una interfaz a la que se le pueden añadir elementos mediante un entorno de ventanas.

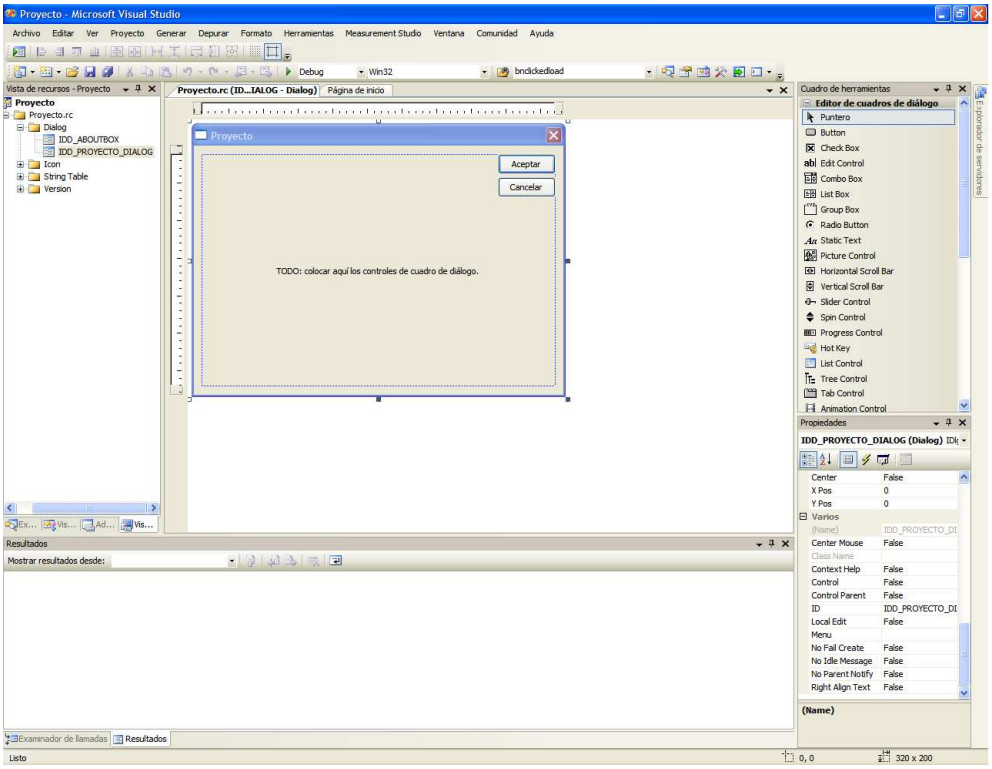


Figura 10: Interfaz de ventanas mediante el uso de MFC en Visual Studio 2005

- **Measurement Studio** es un paquete integrado de clases y controles para aplicaciones de pruebas, medidas y automatización en Microsoft Visual Studio 2008/2005/.NET 2003 y Visual Studio 6.0. Measurement Studio ha sido desarrollado por National Instruments y reduce drásticamente el tiempo de desarrollo de aplicaciones al proporcionar formularios de Windows, formularios Web y componentes de interfaz de usuario ActiveX. Está diseñada para ingenieros, análisis científico avanzado y adquisición de datos y asistentes para control de instrumentos optimizados para pruebas.

  Basándose en más de 20 años de experiencia de medición de programación de National Instruments, Measurement Studio ofrece la posibilidad de una integración sencilla para la adquisición de datos y control de instrumentos en una interfaz.

2.1.- Requerimientos

Los requerimientos para la realización del proyecto actual son coherentes con el tipo de programa a realizar: gestión de una serie de archivos para visualizar una animación con la que es posible interactuar. De esta forma, los requisitos que se han tenido en cuenta se pueden clasificar, atendiendo a su funcionalidad, de la siguiente forma:

- Gestión de ficheros de prueba
- Visualización de la animación
- Modificación del momento de la animación
- Modificación de cámaras

Un análisis de cada uno de estos apartados tan generales permite realizar una clasificación más refinada:

- Gestión de ficheros de prueba
    - Abrir ficheros de prueba
    - Borrar ficheros de prueba
- Visualización de la animación
    - Cargar prueba
- Modificación del momento de la animación
    - Reproducir prueba
    - Rebobinar prueba
    - Pausar prueba
    - Cambiar posición de reproducción de prueba
    - Limitar rango de reproducción
- Modificación de cámaras
    - Cambiar cámara
    - Cambiar posición
    - Cambiar orientación

A continuación se va a realizar un estudio pormenorizado y minucioso de cada uno de estos apartados a través de la realización de los casos de uso correspondientes.

2.1.1.- Gestión de ficheros de prueba

Este módulo debe permitir importar ficheros de prueba a la aplicación, así como eliminar algunos ficheros que ya se encuentren cargados en la misma.

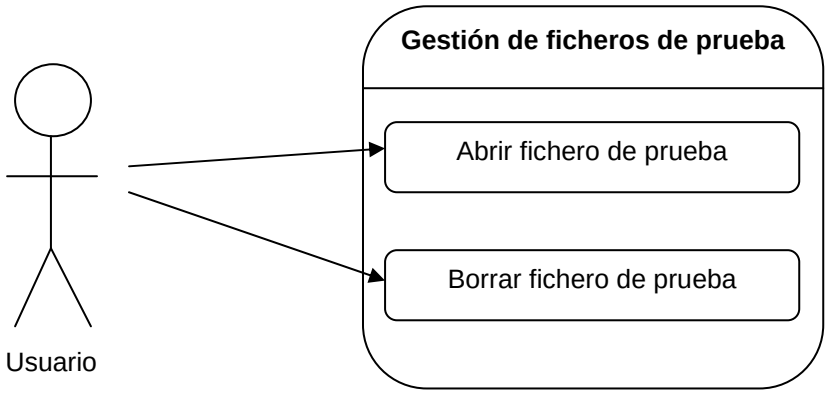


Figura 11: Caso de uso de gestión de fichero de prueba

2.1.1.1.- Abrir fichero de prueba

El usuario selecciona las pruebas a cargar a través de un diálogo y, en caso de tratarse de ficheros válidos, se importan al programa.

Detalles del caso de uso

Nombre: Abrir fichero de prueba.
Actores: Usuario.
Precondiciones: Ninguna.
Postcondiciones: Los ficheros de prueba se cargan en la interfaz del sistema.

Flujo de eventos

	Intención de Usuario	Obligaciones del Sistema
1.	El usuario pulsa el botón de Abrir Fichero.	
2.		El sistema abre un diálogo de tipo "Examinar…".
3.	El usuario selecciona los ficheros a abrir.	
4.		El sistema valida los ficheros seleccionados (nomenclatura y contenido).
5.		El sistema notifica al usuario el número de pruebas válidas.
6.		El sistema extrae la información necesaria de los ficheros válidos.
7.		El sistema incorpora la información de los ficheros a la interfaz.

Extensiones síncronas

#1 En **3** el usuario no selecciona ningún fichero y pulsa Abrir:
El sistema vuelve a mostrar el diálogo.

#2 En **4** el sistema no encuentra ningún fichero válido:
El sistema se lo notifica al usuario y acaba el caso de uso.

2.1.1.2.- Borrar fichero de prueba

El usuario selecciona una prueba cargada en la interfaz y la elimina de la misma.

Detalles del caso de uso

Nombre: Borrar fichero de prueba.
Actores: Usuario.
Precondiciones: El fichero a borrar debe estar cargado en la interfaz.
Postcondiciones: Las pruebas borradas se eliminan de la interfaz.

Flujo de eventos

	Intención de Usuario	Obligaciones del Sistema
1.	El usuario selecciona la prueba a borrar.	
2.	El usuario pulsa el botón de Borrar Prueba.	
3.		El sistema elimina la prueba de la interfaz.
4.		El sistema actualiza la posición del resto de pruebas en la lista.

Extensiones síncronas

#1 En **3** la prueba a borrar está cargada actualmente en la animación:
El sistema muestra un error y acaba el caso de uso.

2.1.2.- Visualización de la animación

Este módulo debe poseer la funcionalidad de interpretar los datos de los ficheros de prueba cargados y transformarlos en una animación que permita una visualización en pantalla.

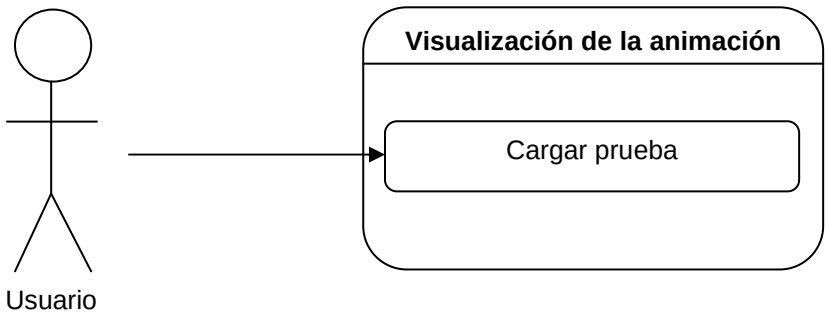


Figura 12: Caso de uso para la visualización de la animación

2.1.2.1.- Cargar prueba

El usuario selecciona una prueba cargada en la interfaz y presiona el botón correspondiente a la carga de la prueba, visualizándose ésta en la ventana de Ogre.

Detalles del caso de uso

Nombre: Cargar prueba.
Actores: Usuario.
Precondiciones: Los ficheros con los datos de las pruebas a cargar deben estar previamente cargados en la interfaz.
Postcondiciones: En la ventana de Ogre se visualiza el modelo y el inicio de la animación.

Flujo de eventos

	Intención de Usuario	Obligaciones del Sistema
1.	El usuario selecciona una prueba de la interfaz.	
2.	El usuario presiona el botón de Cargar Prueba.	
3.		El sistema lee el contenido del fichero de prueba y almacena los datos.
4.		El sistema carga la gráfica con los datos almacenados.
5.		El sistema interpreta los datos y crea la animación.
6.		El sistema activa los elementos y eventos que pueden interactuar con la animación.
7.		Se carga la cámara libre por defecto.
8.		Se inicia el bucle de renderizado.

Extensiones síncronas

#1 En **2** el usuario presiona el botón Cargar Prueba sin tener un fichero seleccionado:
El sistema muestra un error y acaba el caso de uso.

#2 En **2** el usuario presiona el botón Cargar Prueba teniendo el diálogo de apertura de pruebas abierto:
El sistema muestra un error y acaba el caso de uso.

#3 En **3** el sistema tiene animaciones previas cargadas:
El sistema libera recursos relacionados con las mismas y sigue su ejecución.

#4 En **5** se produce algún error al emplear los recursos para la creación de la animación:
El sistema muestra un error y acaba el caso de uso.

2.1.3.- Modificación del momento de animación

Este modulo encapsula toda la interacción que el usuario realiza con la animación para variar la reproducción de la misma.

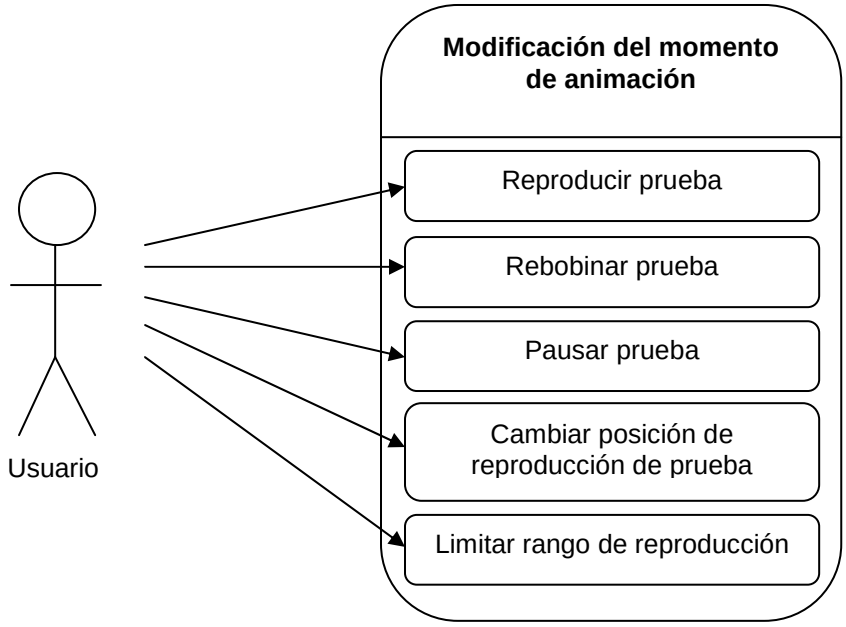


Figura 13: Caso de uso encargado de la modificación del momento de animación

2.1.3.1.- Reproducir prueba

El usuario pulsa en el botón correspondiente a reproducir una prueba y comprueba los movimientos que realiza el modelo en función de la prueba que se encuentre cargada.

Detalles del caso de uso

Nombre: Reproducir prueba.
Actores: Usuario.
Precondiciones: Debe haber una animación cargada en el programa.
Postcondiciones: La ventana de Ogre reproduce la animación.

Flujo de eventos

	Intención de Usuario	Obligaciones del Sistema
1.	El usuario presiona el botón Reproducir Prueba.	
2.		El sistema interpola entre los datos interpretados de las pruebas para generar movimiento en la animación.
3.		El sistema enlaza el momento de reproducción con elementos de la interfaz.

Extensiones síncronas

#1 En **1** el usuario presiona el botón Reproducir Prueba encontrándose el momento de reproducción al final de la animación:
El sistema reinicia el momento de reproducción y vuelve a comenzar.

#2 En **2** ocurre un problema al modificar el estado de la reproducción en Ogre:
El sistema muestra un error y acaba el caso de uso.

2.1.3.2.- Rebobinar prueba

El usuario pulsa en el botón correspondiente a rebobinar una prueba y comprueba los movimientos reproducidos en sentido inverso que realiza el modelo en función de la prueba que se encuentre cargada.

Detalles del caso de uso

Nombre: Rebobinar prueba.
Actores: Usuario.
Precondiciones: Debe haber una animación cargada en el programa.
Postcondiciones: La ventana de Ogre reproduce la animación en sentido inverso.

Flujo de eventos

	Intención de Usuario	Obligaciones del Sistema
1.	El usuario presiona el botón Rebobinar Prueba.	
2.		El sistema interpola entre los datos interpretados de las pruebas para generar los movimientos en sentido inverso en la animación.
3.		El sistema enlaza el momento de reproducción con elementos de la interfaz.

Extensiones síncronas

#1 En **1** el usuario presiona el botón Rebobinar Prueba encontrándose el momento de reproducción al principio de la animación:
El sistema lleva al final el momento de reproducción el momento de reproducción y sigue su ejecución.

#2 En **2** ocurre un problema al modificar el estado de la reproducción en Ogre:
El sistema muestra un error y acaba el caso de uso.

2.1.3.3.- Pausar prueba

El usuario pulsa en el botón correspondiente a pausar la reproducción de una prueba y la animación se detiene.

Detalles del caso de uso

Nombre: Pausar prueba.
Actores: Usuario.
Precondiciones: Debe haber una animación reproduciéndose en el programa.
Postcondiciones: El sistema mantiene fijo el momento de reproducción de la animación.

Flujo de eventos

	Intención de Usuario	Obligaciones del Sistema
1.	El usuario presiona el botón Pausar Prueba.	
2.		El sistema detiene el momento de reproducción si la animación se encuentra en estado de reproducir o rebobinar prueba.
3.		El sistema enlaza esa detención del momento de reproducción con elementos de la interfaz.

Extensiones síncronas

#1 En **2** ocurre un problema al modificar el estado de la reproducción en Ogre: El sistema muestra un error y acaba el caso de uso.

2.1.3.4.- Cambiar posición de reproducción de prueba

El usuario interactúa con la barra de reproducción o con la gráfica modificando el momento de reproducción de la prueba cargada.

Detalles del caso de uso

Nombre: Cambiar posición de reproducción de prueba.
Actores: Usuario.
Precondiciones: Debe haber una animación cargada en el programa.
Postcondiciones: El sistema varía el momento de reproducción de la prueba.

Flujo de eventos

	Intención de Usuario	Obligaciones del Sistema
1.	El usuario pulsa con el botón izquierdo del ratón sobre un punto de la gráfica o la barra de reproducción.	
2.		El sistema calcula la correspondencia entre el punto pulsado y el tiempo de animación correspondiente.
3.		El sistema traslada el momento de reproducción al tiempo calculado.
4.		Se actualizan los elementos de la interfaz atendiendo al momento de reproducción actual.

Extensiones síncronas

#1 En **1** la acción de pulsado se puede sustituir por pulsación y arrastre, ejecutándose el caso de uso por cada variación de pulsación del ratón.

#2 En **1** también se puede modificar la posición de reproducción de la prueba pulsando en los botones correspondientes a ir al comienzo o ir al final.

#3 En **1** el momento de reproducción también se puede variar mediante la interacción con la rueda del ratón.

#4 En **1** el usuario pulsa en una región inactiva de la gráfica:
El sistema traslada el momento de reproducción a la posición correspondiente al cursor límite más cercano.

#5 En **1** se encuentra activo el modo de rebobinar o reproducir prueba:
El sistema da preferencia al caso de uso actual y, al finalizar la interacción con éste, prosigue con el modo de reproducción que se encontraba en ejecución.

2.1.3.5.- Limitar rango de reproducción

El usuario modifica la posición de los cursores límite de la gráfica, variando el rango de reproducción de la animación.

Detalles del caso de uso

Nombre: Limitar rango de reproducción.
Actores: Usuario.
Precondiciones: Debe haber una animación cargada en el programa.
Postcondiciones: El rango de reproducción se limita a los cursores límite de la gráfica.

Flujo de eventos

	Intención de Usuario	Obligaciones del Sistema
1.	El usuario pulsa y arrastra sobre un cursor límite de la gráfica.	
2.		El sistema calcula la correspondencia entre la nueva posición de arrastre y el tiempo de animación.
3.		El sistema actualiza el rango de reproducción de la animación.
4.		Se actualiza el área de la región activa en la gráfica.
5.		Se actualiza el rango entre el que se ejecuta la barra de reproducción.

Extensiones síncronas

#1 En **1** la posición de arrastre del cursor límite supera al cursor de animación:
El sistema llama al caso de uso Cambiar posición de reproducción de prueba para dejarlo siempre dentro de la región activa.

#2 En **1** la posición de arrastre del cursor izquierdo se intercambia con el derecho o viceversa:
El sistema muestra un error, actualiza la posición de los cursores y el momento de reproducción a la posición por defecto y acaba el caso de uso.

2.1.4.- Modificación de cámaras

El usuario debe poder interactuar con las cámaras que permiten visualizar la animación modificando la posición y orientación de visualización.

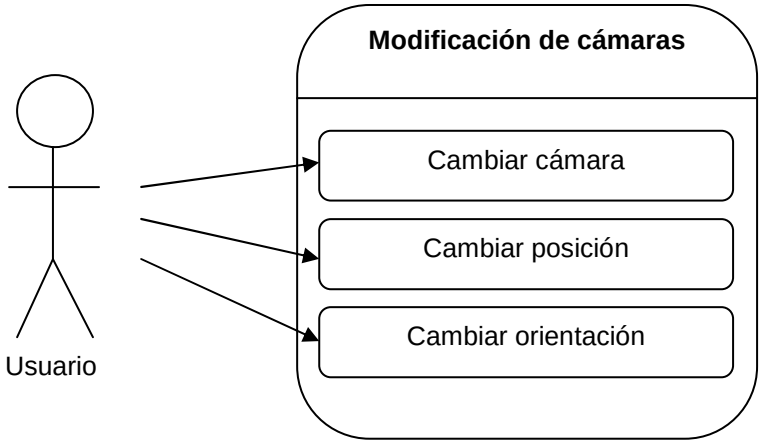


Figura 14: Casos de uso que controlan la modificación de las cámaras

2.1.4.1.- Cambiar cámara

El usuario selecciona la cámara con la que desea visualizar la reproducción de la animación.

Detalles del caso de uso

Nombre: Cambiar cámara.
Actores: Usuario.
Precondiciones: Debe haber una animación cargada en el programa.
Postcondiciones: La visualización de la animación cambia en función de la cámara seleccionada.

Flujo de eventos

	Intención de Usuario	Obligaciones del Sistema
1.	El usuario pulsa sobre alguna de las cinco cámaras disponibles para visualizar la animación.	
2.		El sistema identifica la cámara y la carga en la animación para visualizar su punto de vista.
3.		El sistema desactiva la cámara cargada previamente en la interfaz.

Extensiones síncronas

#1 En **2** la cámara seleccionada es la cámara libre:
El sistema activa los elementos de la interfaz que permiten modificar la posición de la misma.

#2 En **2** la cámara seleccionada es la cámara libre:
El sistema carga la última modificación de posición y orientación que se realizó sobre la misma.

2.1.4.2.- Cambiar posición

El usuario modifica la posición de la cámara libre para visualizar la animación con el fin de observar mejor un movimiento determinado.

Detalles del caso de uso

Nombre: Cambiar posición.
Actores: Usuario.
Precondiciones: Debe haber una animación cargada en el programa y estar activa la cámara libre.
Postcondiciones: La visualización de la animación cambia en función de la nueva posición de la cámara.

Flujo de eventos

	Intención de Usuario	Obligaciones del Sistema
1.	El usuario pulsa sobre alguno de los *spins*.	
2.		El sistema calcula el incremento de posición que supone el pulsar el *spin* seleccionado.
3.		El sistema traslada la posición de la cámara a la posición obtenida a través de la posición previa y el incremento obtenido calculado.

Extensiones síncronas

#1 En **1** es posible modificar la posición de la cámara mediante la utilización del teclado.

#2 En **1** es posible cambiar la posición de la cámara libre en el eje Z mediante el empleo de la rueda del ratón.

#3 En **1** es posible devolver a la cámara libre a su posición por defecto pulsando en el botón *Reset*.

#4 En **2** el spin pulsado llega a su límite de posición:
El sistema mantiene fija la posición de la cámara.

2.1.4.3.- Cambiar orientación

El usuario pulsa sobre la ventana de la animación para modificar la orientación de la cámara libre.

Detalles del caso de uso

Nombre: Cambiar orientación.
Actores: Usuario.
Precondiciones: Debe haber una animación cargada en el programa y estar activa la cámara libre.
Postcondiciones: La orientación de la cámara libre varía en función de la modificación realizada.

Flujo de eventos

	Intención de Usuario	Obligaciones del Sistema
1.	El usuario pulsa el botón izquierdo del ratón y arrastra sobre la ventana de la animación.	
2.		El sistema obtiene las coordenadas de ratón.
3.		El sistema calcula el incremento entre la posición de pulsación y la de arrastre.
4.		El sistema modifica la orientación de la cámara en función del incremento calculado.

Extensiones síncronas

#1 En **1** la posición de arrastre sobrepasa los límites de la ventana de la animación:
El sistema no actualiza la orientación mientras que el puntero no se sitúe de nuevo, en la misma acción de arrastre, sobre la ventana de la animación.

2.2.- Diseño de la aplicación

2.2.1.- Arquitectura del proyecto

El proyecto está organizado en torno a tres módulos principales relacionados: Animación Ogre, Interfaz y Recursos Externos.

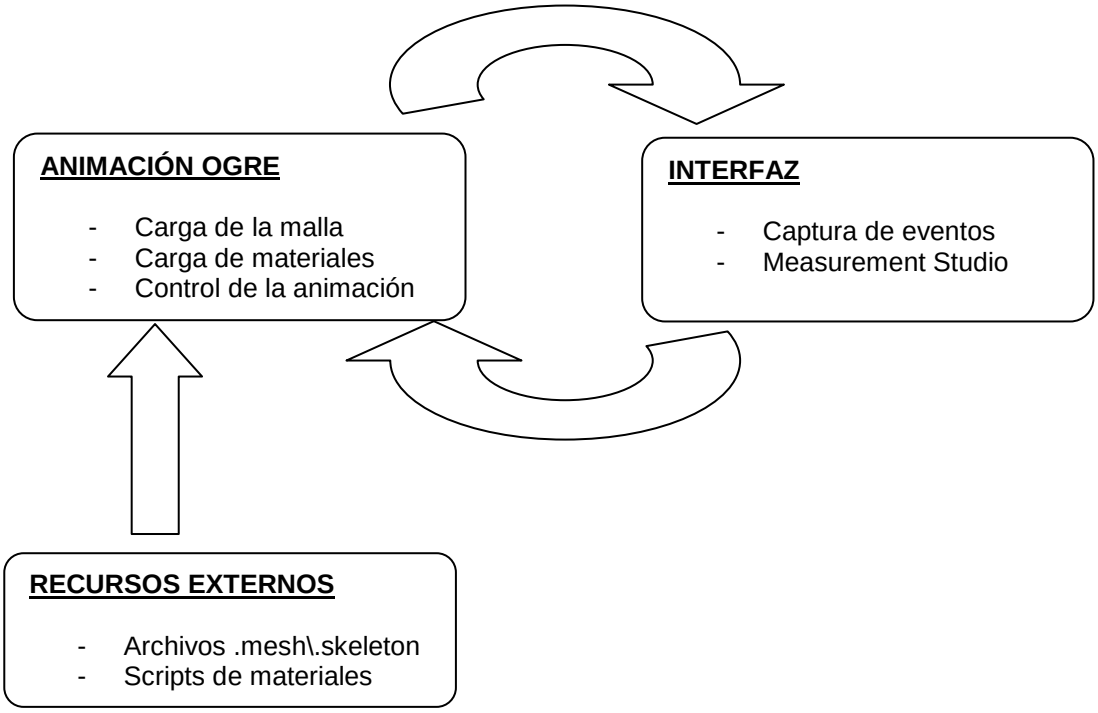


Figura 15: Esquema de la arquitectura del proyecto

2.2.1.1.- Animación Ogre

Este módulo se encarga de todas las funcionalidades del programa que estén relacionadas con la animación 3D. Por tanto su rango de actuación se define:

- Inicialización del motor Ogre con la carga de todos los recursos necesarios para su ejecución.
- Carga de los recursos externos como la malla, el esqueleto y los materiales de los que dispondrá el modelo.
- Lectura de los parámetros enviados por la interfaz y actualización de la animación en función de éstos.
- Captura de eventos de teclado que permiten modificar la animación.

2.2.1.2.- Interfaz

Este módulo es el que está en contacto directo con el usuario de la aplicación. Las funcionalidades principales son:

- Inicializa las librerías necesarias para la ejecución del programa.
- Define el aspecto visual del programa y la disposición de sus elementos.
- Captura eventos producidos por el usuario.
- Envía parámetros al módulo de Animación Ogre en función de la interpretación de estos eventos.
- Permite la interacción entre la gráfica de Measurement Studio y los elementos de la interfaz.
- Actualiza elementos de la interfaz relacionados con la animación 3D.

2.2.1.3.- Recursos Externos

El contenido de este módulo se ciñe a los archivos necesarios para la carga del modelo 3D con el que trabaja el módulo de Animación Ogre:

- Archivos .mesh: modelación del objeto 3D.
- Archivos .skeleton: encargados de definir la deformación de las mallas y sus posibilidades de movimiento.
- Archivos .material: su función es la definición de los materiales a mostrar, así como su variación dependiendo del punto de vista o localización de la cámara.
-

Por tanto se trata de un módulo que únicamente se emplea como fuente y no tiene parámetro alguno.

2.2.2.- Arquitectura de clases

Las clases utilizadas siguen la arquitectura del proyecto y se organizan de manera análoga. Se distingue entre los módulos de Interfaz y de Animación de Ogre.

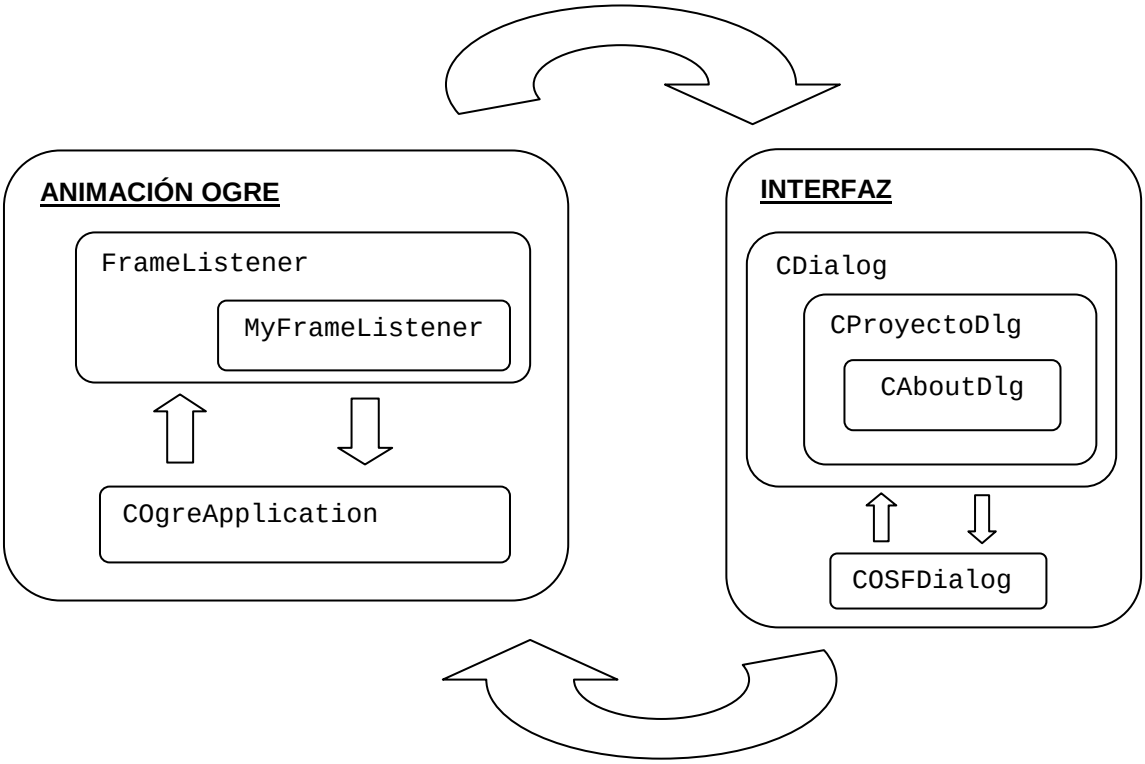


Figura 16: Arquitectura de clases

2.2.2.1.- Clases Ogre

Para realizar toda la funcionalidad el módulo de Animación Ogre se han empleado las siguientes clases:

- `FrameListener`: clase creada por defecto en las librerías de Ogre. Esta clase permite la recepción de notificaciones para eventos generados. Incorpora los siguientes métodos:
  - `frameStarted`: se ejecuta cuando un frame está en situación de ser renderizado.
  - `frameRenderingQueued`: se ejecuta cuando todos los objetivos han emitido una señal de que su renderizado ha acabado pero antes de que se muestre el renderizado en pantalla.
  - `frameEnded`: se ejecuta cuando un frame ha sido renderizado.

- `MyFrameListener`: clase que hereda de `FrameListener` donde se han implementado todos los eventos que pueden modificar el estado de la animación o la visualización de la misma. Incorpora los siguientes métodos seguidos de su funcionalidad:
  - `frameStarted`: método heredado y sobreescrito de `FrameListener`. En este método se ha implementado la funcionalidad para realizar el núcleo de la animación.

  - `resetAnimation`: sitúa el momento de animación en el cursor izquierdo (por defecto en el inicio).
  - `finalAnimation`: sitúa el momento de animación en el cursor derecho (por defecto al final).
  - `playAnimation`: inicia la animación. Si se encuentra en el instante final en el momento de su ejecución vuelve al inicio.
  - `rewAnimation`: inicia la animación en sentido inverso. Si se encuentra en el instante inicial en el momento de su ejecución vuelve al final.
  - `pauseAnimation`: pausa la animación.
  - `sliderControl`: permite la modificación del momento de animación mediante la manipulación del *slider* situado en la interfaz.
  - `spinControl`: permite la modificación de la posición de la cámara mediante la manipulación de los *spins* situados en la interfaz.
  - `wheelMouseControl`: modifica la profundidad de la cámara.
  - `cameraControl`: modifica la orientación de la cámara.
  - `resetCameraPosition`: devuelve la cámara libre a su posición y orientación por defecto (posición frontal).
  - `viewAnimation`: permite visualizar u ocultar las mallas y los ejes asociados a la animación.

- `COgreApplication`: clase donde se inicializan todos los recursos necesarios para la ejecución de Ogre, además de cargar el modelo 3D y obtener los datos necesarios para la realización de la animación. Contiene la declaración de los métodos siguientes:
  - `createRoot`: método donde se crea el elemento raíz que se utiliza como base para configurar Ogre.
  - `defineResources`: encargado de la carga del archivo de configuración y de definir la localización de los recursos mediante la lectura del fichero *resources.cfg* obtenido tras la instalación de la librería Ogre.
  - `setupRenderSystem`: define todas las opciones de visualización que tiene la ventana Ogre a mostrar (pantalla completa, resolución, nivel de detalle, tarjeta gráfica que cargará la información…).
  - `createRenderWindow`: creación de la ventana de Ogre con las opciones de configuración descritas.
  - `initializeResourceGroups`: inicialización de recursos.
  - `setupScene`: creación de elementos básicos de la escena: cámara principal, punto de vista y director o gestor de la escena.
  - `setupInputSystem`: inicialización de eventos de entrada. En este caso sólo se manejaran eventos de teclado puesto que los de ratón se manejan desde la interfaz.
  - `createScene`: carga de las mallas del modelo, de una luz inicial e inicialización de las cámaras de las que constará la animación.
  - `readFile`: lectura de un fichero de pruebas y rellenado de la estructura con los datos de los ángulos en función del tiempo.
  - `createAnimation`: creación de la animación mediante los datos obtenidos en la estructura.
  - `destroyAnimation`: liberación de los recursos generados al crear una animación.
  - `destroyAllAnimations`: liberación de los recursos generados por todas las animaciones creadas.
  - `createFrameListener`: llamada al constructor de la clase `MyFrameListener` para interactuar con la animación creada.
  - `startRenderLoop`: comienzo de la renderización y llamada al método `frameStarted` de la clase `MyFrameListener`.

2.2.2.2.- Clases Interfaz

La definición de la interfaz gráfica en este proyecto incorpora las siguientes clases:

- `CDialog`: clase básica para la creación y muestra por pantalla de diálogos mediante MFC. Existen dos tipos de diálogos: *modal* y *modeless*.
  - Un diálogo de tipo *modal* debe ser cerrado por el usuario antes de que la aplicación continúe su ejecución.
  - Un diálogo de tipo *modeless* permite al usuario mostrar el diálogo y seguir con otra tarea sin cancelar o cerrar el diálogo mostrado.

  En este proyecto se ha empleado el tipo de diálogos *modeless* debido a su versatilidad de utilización.
  Sin embargo, un objeto de tipo `CDialog` no se puede definir por sí mismo. Un objeto de tipo `CDialog` es una combinación de una plantilla de diálogo y una clase derivada de `CDialog`.

- `CProyectoDlg`: clase derivada de `CDialog` que crea el diálogo principal de la interfaz. Sobre esta clase se ha implementado toda la funcionalidad que incorpora la interfaz de usuario. Se han controlado todo tipo de eventos, excepto los eventos de teclado que se han controlado en el módulo de Ogre, que permiten al usuario interactuar con la aplicación. Los métodos implementados los podemos dividir en:
  - Eventos de la interfaz: métodos ejecutados cuando el usuario interactúa de alguna forma especificada con los elementos situados en la interfaz (pulsación de botones, movimiento del ratón, temporizadores…). Estos métodos poseen una cabecera por defecto.
  - Funciones propias: métodos creados para obtener algunos datos necesarios que proporcionan algunos elementos de la interfaz. En algunas ocasiones se implementan para facilitar la lectura del código y obtener funciones más diversificadas.
  - Eventos de Measurement Studio: métodos ejecutados cuando el usuario interactúa de alguna forma con la gráfica mostrada. Los métodos disponibles dependen del modo de recepción de eventos en el que se encuentre la gráfica generada.
- `CAboutDlg`: clase creada en el contenido de `CProyectoDlg` y que también deriva de `CDialog`. Es una clase creada por defecto al emplear MFC y se encarga de mostrar un diálogo con información relativa a las herramientas utilizadas.

2.2.2.3.- Clases externas

Las clases externas corresponden a clases no implementadas en el proyecto actual que se han descargado para incorporar la funcionalidad de la que disponen a este proyecto.

- `COSFDialog`: esta clase incorpora la funcionalidad necesaria para utilizar los diálogos tipo de apertura y guardado de ficheros. En este proyecto sólo se ha empleado la funcionalidad de apertura con el fin de cargar las pruebas seleccionadas. Además incorpora las siguientes opciones:
  - Modificación del aspecto de los diálogos mostrados para la apertura y salvado de ficheros.
  - Posibilidad de selección múltiple.
  - Filtrado de las extensiones de fichero a mostrar.
  - Obtención de las rutas de los ficheros seleccionados.

2.2.3.- Ficheros de prueba de pacientes

Los ficheros que se muestran en la aplicación son obtenidos de pacientes reales y están proporcionados por el Instituto de Biomecánica de Valencia (IBV). Estos ficheros se obtienen fruto del proceso que realiza el programa NedCervical de los datos muestreados con el sistema de vídeo digital Kinescan/IBV. Así mediante las coordenadas 3D de cada uno de los marcadores obtenidas por este sistema, es posible realizar el cálculo de los ángulos en los ejes X, Y, Z en función del tiempo para establecer los movimientos realizados por el paciente en cuestión.

2.2.3.1.- Tipo y nomenclatura

Los ficheros generados son simples archivos de texto con extensión *.txt* para facilitar la lectura e interpretación de los datos. La lectura de los datos incluidos en los ficheros se realiza mediante el uso de librerías que incorporan utilidades para la manipulación de ficheros (véase `fstream` para el caso del lenguaje C++ en el que se ha realizado el proyecto).

Sin embargo y según un convenio obtenido con personas vinculadas a la empresa donde se ha realizado el proyecto, estos ficheros deben recibir una nomenclatura específica según la prueba de la que se trate. De esta forma se distinguen 9 tipos de pruebas:

- Prueba de flexo-extensión (1): el paciente realiza movimientos variando fundamentalmente el eje X (similar a una afirmación continuada). Este fichero puede recibir los nombres: *angulos_1.txt* o *v2_1.txt*.

- Prueba de flexo-extensión (2): el paciente realiza los mismos movimientos que la prueba anterior para corroborar su cinemática. Este fichero puede recibir los nombres: *angulos_2.txt* o *v2_2.txt*.

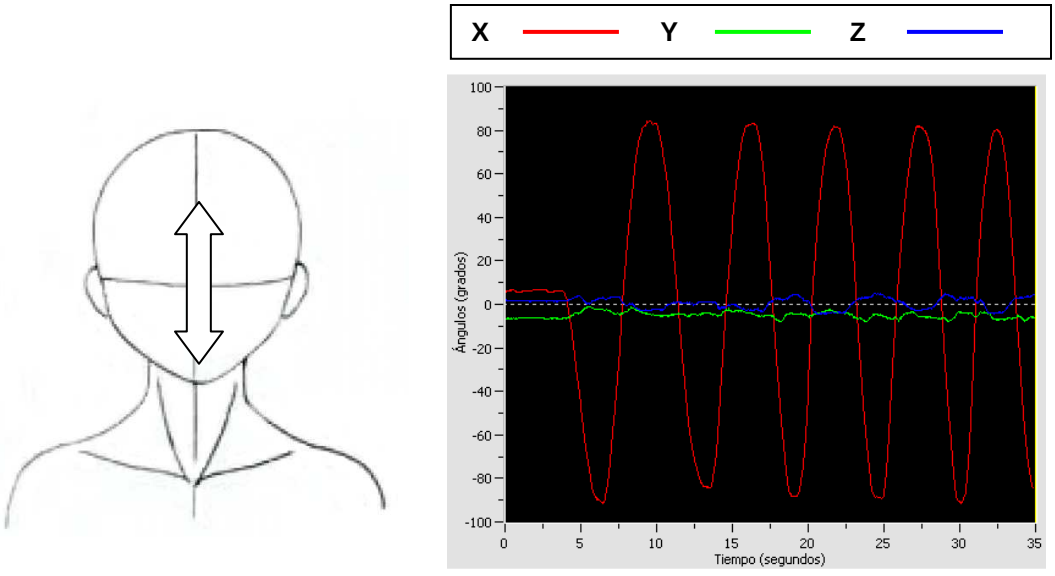


Figura 17: Movimiento de flexo-extensión en un paciente sano

- Prueba de flexión lateral (1): el paciente realiza movimientos variando fundamentalmente el eje Z (balance de la cabeza a los lados manteniendo hombros rectos y mirada al frente). Este fichero puede recibir los nombres: *angulos_3.txt* o *v2_3.txt*.

- Prueba de flexión lateral (2): el paciente realiza los mismos movimientos que la prueba anterior para corroborar su cinemática. Este fichero puede recibir los nombres: *angulos_4.txt* o *v2_4.txt*.

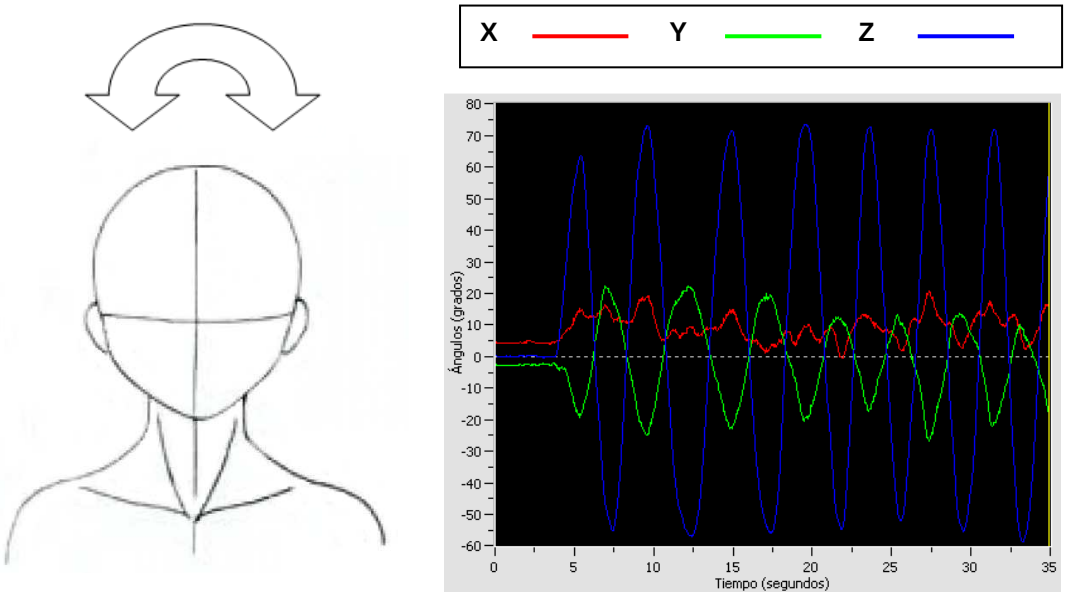


Figura 18: Movimiento de flexión lateral en un paciente sano

- Prueba de rotación (1): el paciente realiza movimientos variando fundamentalmente el eje Y (similar a una negación continuada). Este fichero puede recibir los nombres: *angulos_5.txt* o *v2_5.txt*.

- Prueba de rotación (2): el paciente realiza los mismos movimientos que la prueba anterior para corroborar su cinemática. Este fichero puede recibir los nombres: *angulos_6.txt* o *v2_6.txt*.

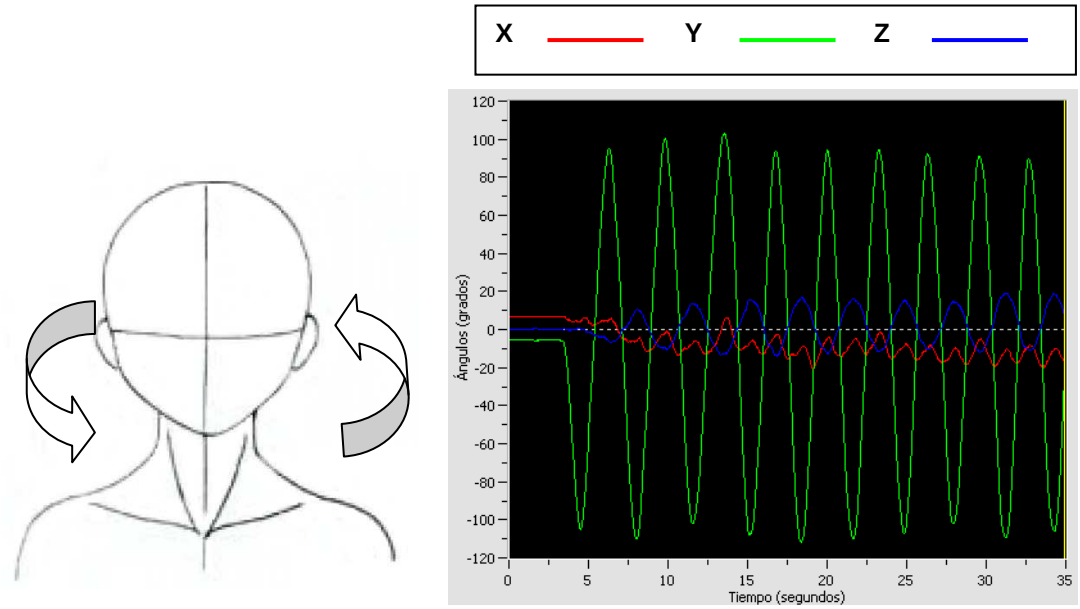


Figura 19: Movimiento de rotación en un paciente sano

- Prueba funcional (1): el paciente mira a una lámpara situada arriba a su izquierda, después mira un cuestionario situado en sus rodillas y por último mantiene la mirada al frente. Este fichero puede recibir los nombres: *angulos_9.txt* o *v2pf_1.txt*.

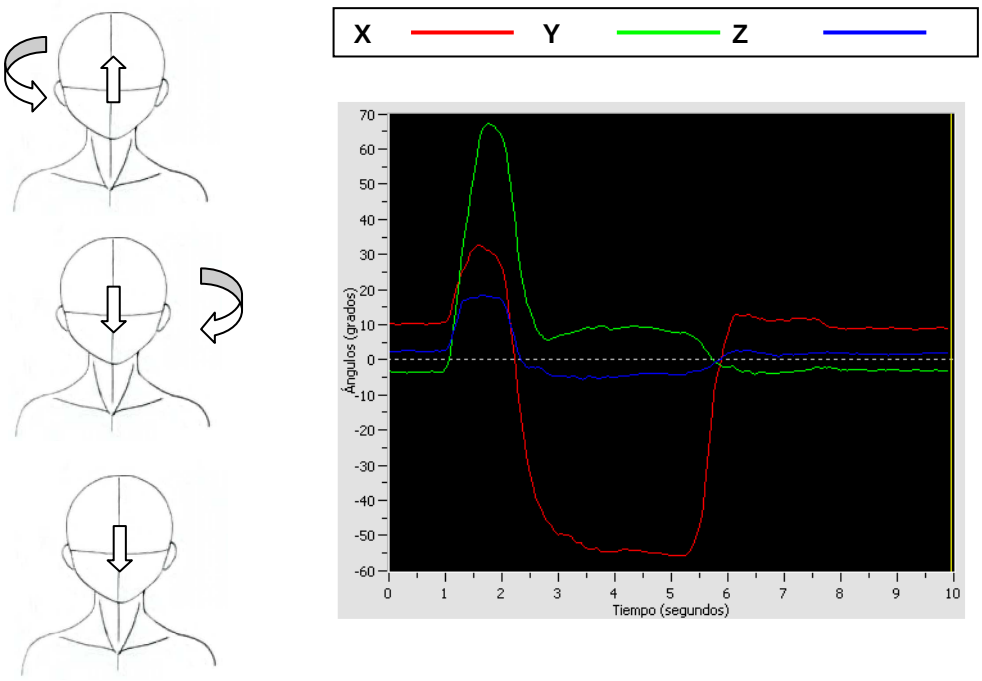


Figura 20: Movimiento de la prueba funcional 1 en un paciente sano

- Prueba funcional (2): el paciente mira a una lámpara situada encima de su cabeza, después mira un cuestionario situado en sus rodillas y por último mantiene la mirada al frente. Este fichero puede recibir los nombres: *angulos_8.txt* o *v2pf_2.txt*.

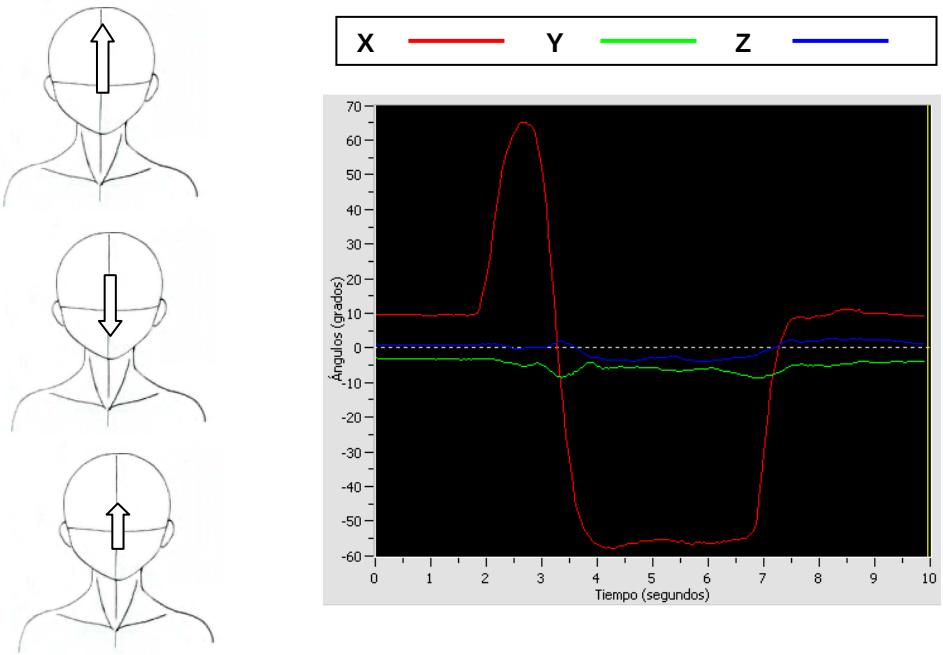


Figura 21: Movimiento de la prueba funcional 2 en un paciente sano

- Prueba funcional (3): el paciente mira a una lámpara situada arriba a su derecha, después mira un cuestionario situado en sus rodillas y por último mantiene la mirada al frente. Este fichero puede recibir los nombres: *angulos_7.txt* o *v2pf_3.txt*.

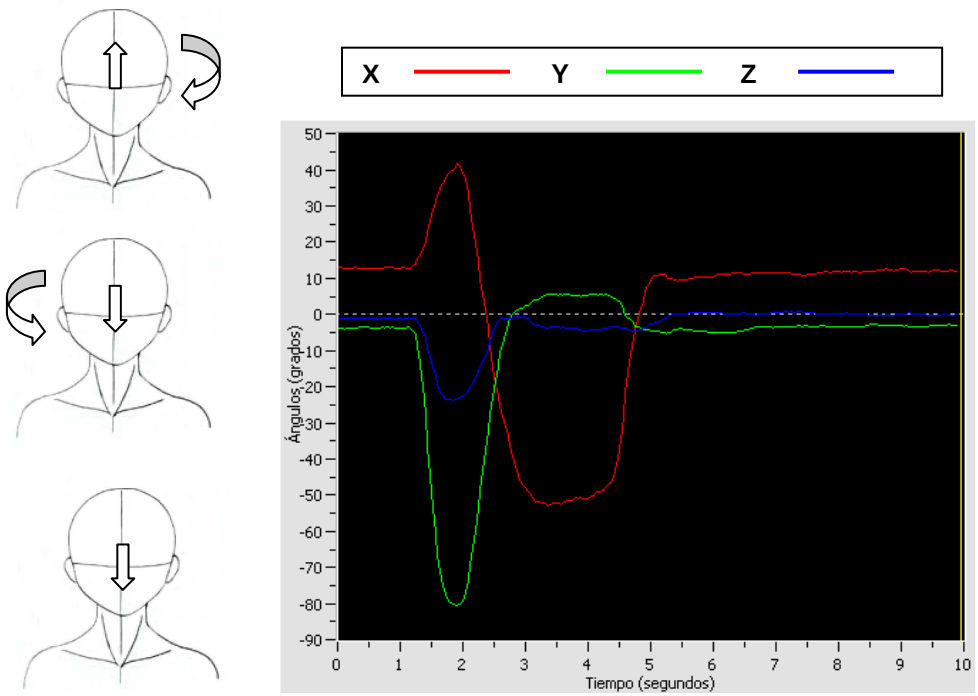


Figura 22: Movimiento de la prueba funcional 3 en un paciente sano

2.2.3.2.- Estructura del contenido

Todos los ficheros responden a un mismo patrón en cuanto a la disposición de los datos. Este patrón se obtiene fruto de un convenio con la empresa que proporciona los datos de estas pruebas. De esta forma el patrón y orden en que se obtiene es el siguiente:

- Tiempo de obtención de los ángulos (segundos).
- Valor del ángulo X (grados).
- Valor del ángulo Z (grados).
- Valor del ángulo Y (grados).

La posición de los marcadores se actualiza con una frecuencia de 25 Hz (cada 0.04 segundos), ya que es la velocidad a la que trabajan las cámaras de Kinescan/IBV. Por tanto, en las pruebas se obtiene un valor de los ángulos en los tres ejes cada intervalo de tiempo comentado.

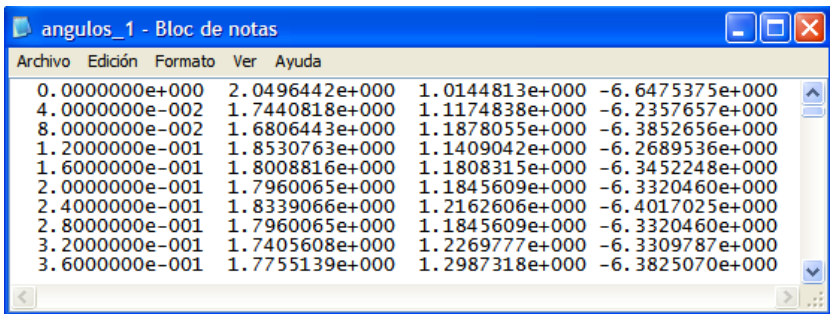


Figura 23: Ejemplo del contenido de un fichero de pruebas

2.2.4.- Interfaz de usuario

La interfaz gráfica de usuario ha sido diseñada teniendo en cuenta, fundamentalmente, aspectos relacionados con la usabilidad. La aplicación desarrollada en este proyecto va a ser utilizada generalmente por personas que no están en la obligación de poseer conocimientos informáticos avanzados y, partiendo de esta base, la interacción con el usuario debe ser simple, intuitiva y clarificadora. El usuario recibe en todo momento una respuesta del programa en torno a operaciones consideradas como inválidas, indicándole cómo llevarlas a cabo de manera correcta mediante mensajes.

Considerar la usabilidad como pilar fundamental a partir del cual construir la capa que interactúa con el usuario atiende, sin lugar a dudas, a la calidad de uso. El hecho de utilizar la aplicación debe favorecer el objetivo que el usuario busca con su ejecución y fomentar su uso; en ningún caso debe ser una experiencia ardua o que requiera una iniciación previa. Debe permitir alcanzar objetivos específicos con efectividad, eficiencia y satisfacción.

Además y partiendo del hecho de considerar la usabilidad como primer factor para el desarrollo de la interfaz, la estética no se ha descuidado. Se ha utilizado un estilo de ventanas "tipo" habitual en las aplicaciones ejecutadas en Windows con el fin de que el usuario se sienta cómodo con su visualización. El hecho de mantener este estilo permite dar cierta confianza y seguridad al considerar los elementos que contiene la interfaz como cotidianos.

2.2.4.1.- Diseño y elementos

La disposición y elección de los elementos a utilizar está estudiada para fomentar también la facilidad de uso de la aplicación. El simple hecho de colocar los botones generales en la parte superior derecha de la ventana, los archivos cargados en la parte superior o la posibilidad de interactuar con los elementos mostrados de forma directa no atiende a una razón trivial: todo se basa en facilitar la labor de uso de la aplicación.

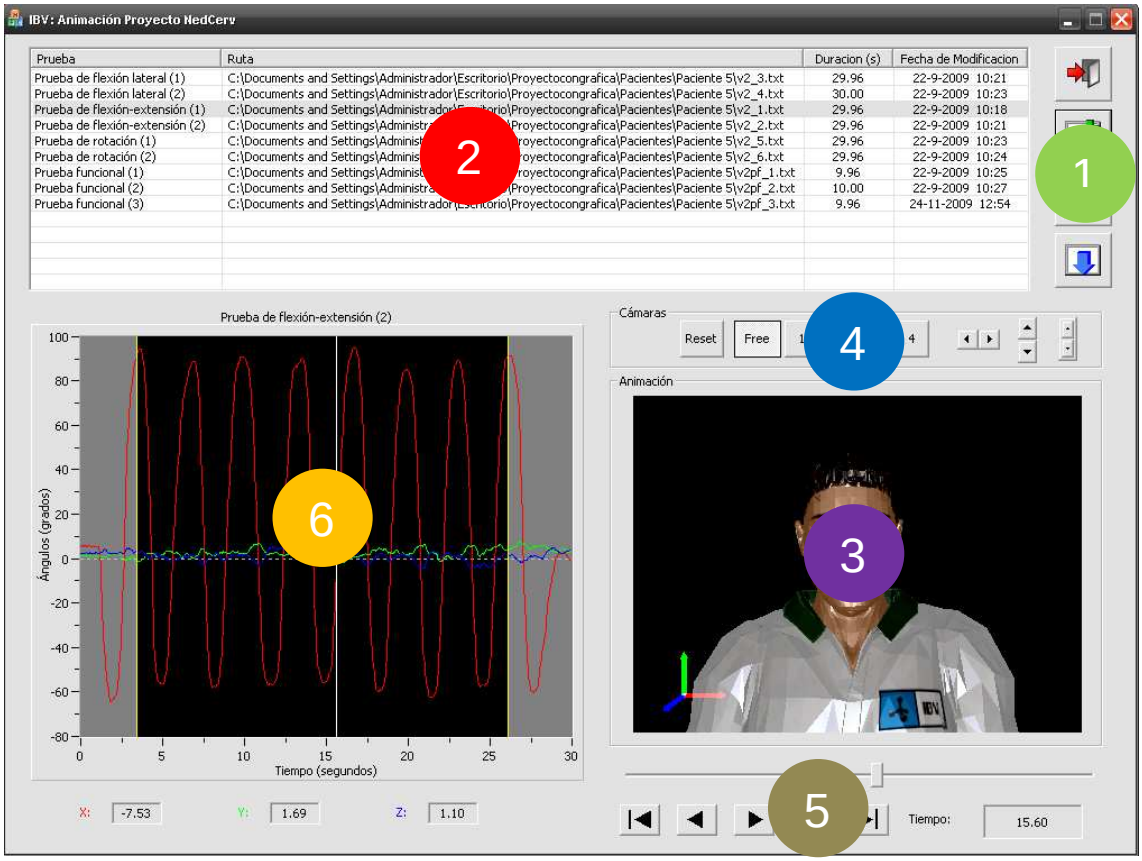


Figura 24: Disposición y señalado de elementos de la interfaz

Se distinguen las siguientes zonas en la representación de la interfaz:

- Zona de botones generales (**1**): estos botones son los iniciadores de las funciones que se pueden realizar en la interfaz. Son los elementos principales que condicionan los datos mostrados por el resto de elementos. Permiten la apertura de ficheros de prueba, el borrado de los mismos, cargar la animación correspondiente a un fichero seleccionado y salir de la aplicación.

- Zona de ficheros importados (**2**): se trata de un *List Control* que contiene información de las pruebas importadas de los pacientes. La información a mostrar se divide en: nombre de la prueba, ruta de la que se obtiene, duración de la misma (en segundos) y fecha de modificación del fichero (día y hora). Este elemento permite la selección de los ficheros importados para realizar su carga o su borrado.

- Ventana de animación (**3**): esta ventana incorpora toda la funcionalidad vinculada con el motor Ogre. Muestra la animación del fichero cargado actualmente y con ella pueden interactuar las zonas marcadas como 4, 5 y 6. Además permite la modificación de la orientación de la cámara o su profundidad mediante eventos de ratón.

- Zona de control de cámaras (**4**): incorpora la funcionalidad relacionada con la posición de las cámaras de la ventana de animación. Permite la variación de la posición de una cámara libre mediante el control de *Spins*, así como la selección de 4 cámaras fijas para detectar de forma adecuada la cinemática de los movimientos representados.

- Zona de control de reproducción de la animación (**5**): esta zona permite la variación de la reproducción mostrada en la animación. A través de un *Slider* o barra de reproducción, se puede seleccionar el momento de reproducción adecuado para observar un movimiento determinado. Además se incluyen botones habituales para la visualización de una animación: iniciar reproducción, reproducción inversa, pausa de la animación, colocar momento de reproducción al principio y colocar momento de reproducción al final.

- Gráfica del fichero cargado (**6**): esta zona muestra los datos del fichero cargado en una gráfica con la variación de los valores de los ángulos en función del tiempo. El cursor de animación está vinculado al momento de reproducción de la misma. Además la gráfica permite interactuar con la reproducción limitando la zona de representación (mediante la pulsación y arrastrado de los cursores límite) y colocar el momento de animación en un valor de tiempo determinado (mediante la pulsación de una zona activa de la gráfica).

Diseño de la gráfica

La gráfica mostrada en la zona 6 (figura 25) se ha realizado mediante las clases incorporadas en el paquete Measurement Studio de National Instruments. Así se han representado los datos en dos ejes en función del contenido de los ficheros de prueba.

En el eje X se representan los valores de tiempo en segundos (primera columna de los ficheros de prueba); mientras que el eje Y contiene los valores de los ángulos (en grados) del paciente en cada uno de sus tres ejes (X, Y, Z). El rango de valores que pueden delimitar los ejes es autoescalable y depende de los datos a mostrar.

Por tanto se obtienen tres trazos: uno referente a la variación del eje X (señalada en color rojo), uno con la variación del eje Y (señalada en color verde) y otro la variación del eje Z (señalada en color azul).

Otro aspecto a destacar es la utilización de cursores y regiones. En la gráfica hay tres cursores: cursor de animación, cursor de inicio de reproducción y cursor de final de reproducción. El cursor de animación (línea vertical de color blanco) marca el momento de reproducción de la animación; mientras que los cursores de inicio y fin de reproducción (líneas verticales de color amarillo) marcan los valores mínimo y máximo de la región activa de la reproducción. La región inactiva se marca con un fondo grisáceo.

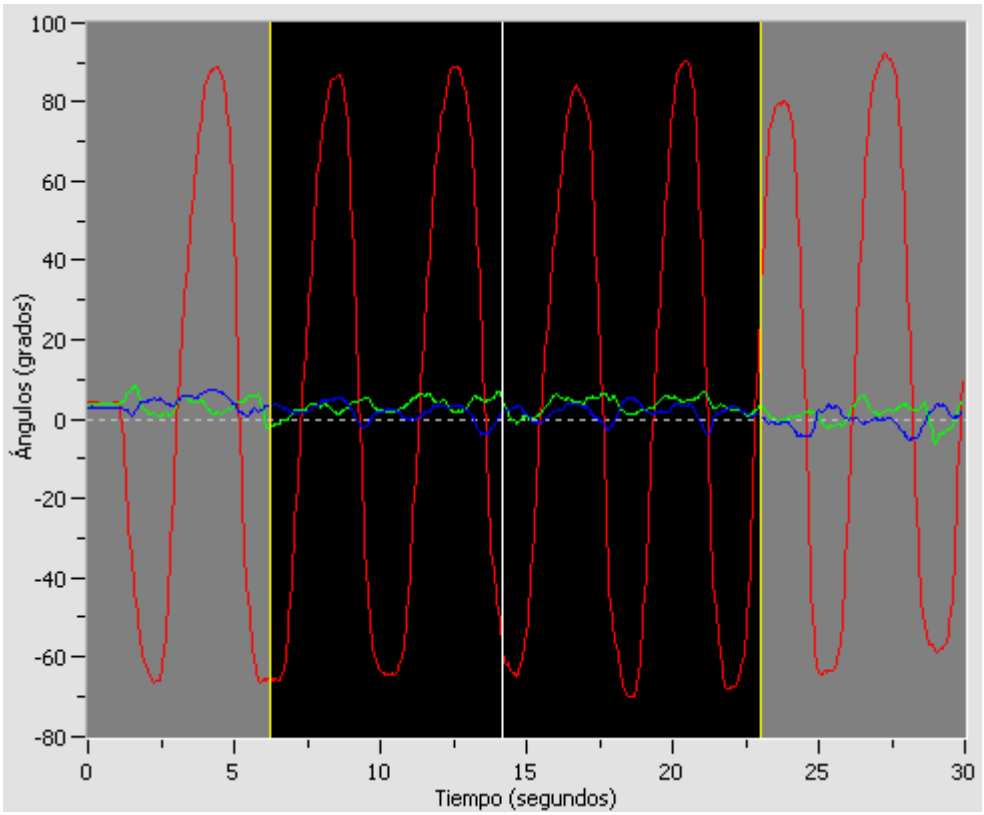


Figura 25: Ejemplo de gráfica con cursores y región activa

Estos cursores se pueden modificar mediante su pulsación y arrastrado o, en el caso del cursor de animación, mediante un clic en una zona activa de la gráfica. Estas modificaciones afectan al momento y rango de la reproducción de la animación en tiempo real.

Iconos

Los iconos que representan la funcionalidad de cada uno de los botones se han seleccionado con el fin de, con un simple vistazo, tener una noción de la función que realiza la pulsación de cada botón. De esta forma se han supuesto ciertos criterios estándar para la elección de algunos iconos y, de forma más concreta, para los que modifican la reproducción de la animación.

Figura 26: Iconos principales de la aplicación

2.3.- Desarrollo de la aplicación

El desarrollo de la aplicación ha requerido la utilización y estudio de tres herramientas de forma secuencial: Blender para el modelado gráfico, Ogre para la interpretación y representación de este modelo y Visual Studio para la creación de una interfaz que permita al usuario interactuar con éste último.

2.3.1.- Blender

El uso de un programa de modelado como Blender permite la creación de gráficos tridimensionales de una manera eficiente y sencilla. Sin embargo, para la utilización correcta del programa se requiere una iniciación previa, ya que su manejo requiere muchos de los denominados "atajos de teclado" y su interfaz no es muy intuitiva. Quizá este sea el escollo más difícil de superar a la hora de realizar modelos tridimensionales con Blender, pero una vez superado los beneficios que se obtienen con su utilización superan a estos defectos iniciales.

En primer lugar es necesario describir el sistema de ventanas básico que utiliza Blender. Aunque el sistema de ventanas es completamente personalizable al gusto que le quiera dar el usuario, se distinguen tres zonas fundamentales:

- Visor 3D: permite ver y manipular objetos en una escena 3D. Es la ventana base del programa ya que el resto de ventanas reflejan en ésta los efectos aplicados al modelo en cuestión.

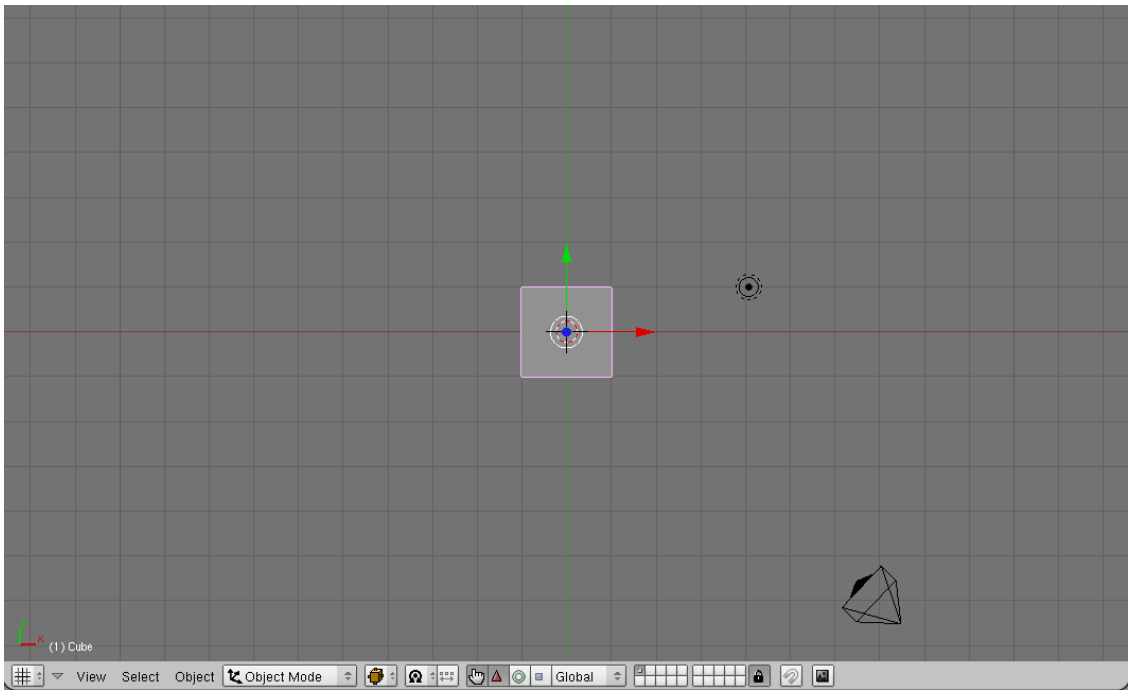


Figura 27: Ventana Visor 3D con la escena por defecto

- Ventana de botones: permite realizar operaciones sobre mallas, cámaras, luces y cualquier objeto representado en la escena. Su funcionalidad es muy amplia y es la ventana secundaria por defecto (normalmente se sitúa en la parte inferior de la ventana total del programa).

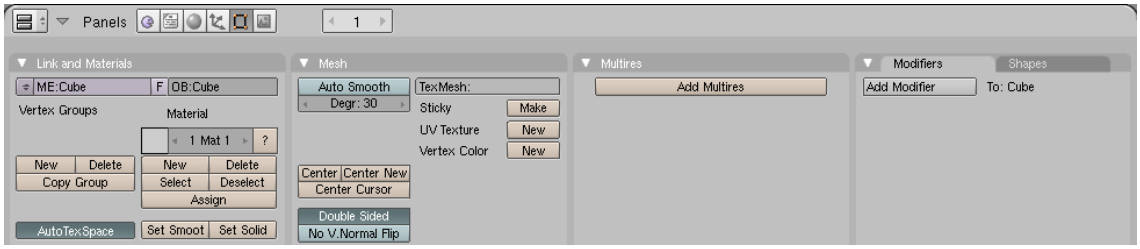


Figura 28: Ventana de botones

- Barras de menú: por defecto (*User Preferences*) incorporan la funcionalidad típica de cualquier programa con posibilidades de guardado y gestión de archivos, así como las funcionalidades más generales.

Figura 29: Barra de menú general (User Preferences)

Sin embargo, Blender incorpora un elemento muy útil: la posibilidad de elección de la ventana a mostrar. Esto facilita enormemente el trabajo realizado en varias capas sin la necesidad de la utilización de menús salientes o *pop-ups* que no permiten la gestión simultánea de distintas ventanas.

Las funcionalidades que se aplican a cada tipo de ventana son muy diversas y permiten realizar cualquier modificación sobre los gráficos 3D. Aspectos como la animación, materiales aplicados a las mallas, estructura de los elementos presentes en la escena y otros utilidades son tratadas en la ventana correspondiente. Por tanto es importante conocer de antemano qué posibilidades incorpora cada tipo de ventana y qué funcionalidades se pueden implementar en ella.

La descripción general de cada uno de los tipos de ventanas, excepto los de 3D View (Visor 3D), Buttons Window (ventana de botones) y User Preferences que ya se comentaron anteriormente, es la siguiente:

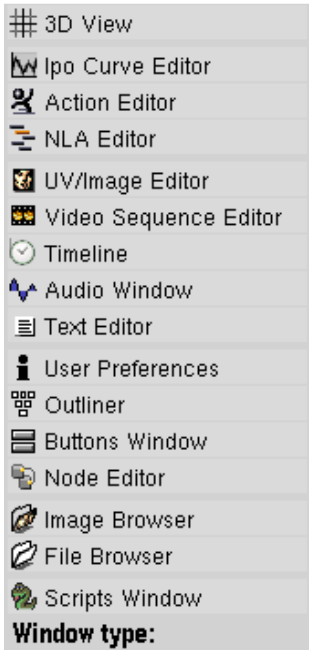


Figura 30: Tipos de ventanas

- Ipo Curve Editor: permite la edición de curvas 2D que definen una animación en Blender.
- Action Editor: define una línea de tiempos en la que se definen acciones a realizar por el modelo 3d. El funcionamiento es a través del establecimiento de KeyFrame entre los que Blender interpola para conseguir una animación fluida.
- NLA Editor: define una línea de tiempos en la que se definen KeyFrames referentes a las acciones declaradas en Action Editor. Blender interpola entre estas acciones para conseguir la animación.
- UV/Image Editor: permite la carga, manipulación y creación de texturas en el modelo.

- Video Sequence Editor: incorpora un sencillo compositor de vídeo que permite el enlazado de imágenes y audio mediante una línea de tiempos.
- Timeline: línea de tiempo que permite la edición de animaciones. Incorpora tanto controles para manipular la animación como botones clásicos para su reproducción.
- Audio Window: representa las ondas referentes a los sonidos añadidos en la aplicación y permite su reproducción.
- Text Editor: se trata de un editor de texto incorporado. Se utiliza sobretodo en conjunción con lenguaje Python para la programación avanzada de la lógica que pretende un modelo que pueda interactuar con el usuario.
- Outliner: define un diagrama jerárquico de los elementos utilizados y sus dependencias entre ellos. Su creación se va haciendo a medida que se incorporan elementos pero también permite su manipulación directa.
- Node Editor: permite el establecimiento de una jerarquía entre los materiales utilizados por un nodo determinado.
- Image Browser: permite la apertura de archivos de imágenes a través de un explorador de carpetas.
- File Browser: permite la apertura de archivos mediante la introducción de la ruta determinada.
- Scripts Windows: incorpora todos los scripts cargados en el programa. Estos scripts pueden venir por defecto instalados en Blender o pueden descargarse dependiendo de la funcionalidad que el programador quiera obtener con los mismos.

Además del conocimiento del sistema de ventanas que proporciona Blender, es fundamental tener una comprensión espacial adecuada. Es básico el conocer y comprender adecuadamente los tipos de transformaciones espaciales:

- Escalado: variación del tamaño del objeto en función de un factor de escala.
- Traslación: variación de la posición que ocupa el objeto en un entorno.
- Rotación: variación de la orientación del objeto. En ámbitos de programación gráfica se distingue entre dos tipos de sistemas de coordenadas:
    - Local o referente al objeto: el sistema de coordenadas tiene su origen en el centro del propio objeto y permite la rotación sobre sí mismo.
    - Global: el sistema de coordenadas tiene su origen fuera del objeto y al realizar una rotación el objeto gira sobre dicho origen.

Cabe comentar que estas transformaciones espaciales se pueden realizar sobre el eje considerado en cada momento con el fin de obtener un resultado adecuado.

El conocimiento, tanto del entorno de ventanas como de las transformaciones espaciales, facilita la explicación para la creación del modelo resultante que se dispone en la aplicación. Para su obtención se han seguido tres pasos:

- Modelado de la malla
- Adición de esqueleto al modelo
- Exportación íntegra del modelo

2.3.1.1.- Modelado de la malla

Blender propone una escena inicial estándar al ejecutar el programa. Esta escena está formada por un cubo, una cámara y una fuente de iluminación. El objetivo que se persigue es "deformar" o modificar ese cubo para convertirlo en el modelo resultante.

El proyecto actual parte de un modelo básico generado por la empresa con el objetivo de perfeccionarlo y mejorarlo. Sin embargo, cabe comentar una técnica

que facilita en gran medida el obtener un modelo inicial: la rotoscopia. La rotoscopia consiste en basarse en una imagen 2D vista desde distintas perspectivas para generar una modelo 3D. Las vistas más empleadas para modelar un objeto 3D suelen ser la de alzado y perfil, aunque su elección depende en gran medida del tipo y la complejidad del objeto a modelar.

De esta forma mediante la variación de la posición de vértices y la generación y deformación de las caras que se forman es posible obtener el objeto 3D deseado. Esta labor es básicamente artística ya que la línea a seguir para obtener un modelo más detallado es equivalente a la realizada por un alfarero al moldear una vasija de barro: depende en gran medida de la experiencia y las horas dedicadas.

Figura 31: Comparación entre modelo inicial (izquierda) y modelo final (derecha)

Una vez realizado el modelado, es posible mejorar su aspecto añadiendo un suavizado de la malla a través del algoritmo de Catmull-Clark que permite aplicar Blender. Este algoritmo se basa en la subdivisión de superficies para conseguir un aspecto más redondeado de la malla y evitar la visualización tan poligonal que presenta. Cuantas más iteraciones del algoritmo se realizan sobre la malla, más redondeada se obtiene la superficie sobre la que actúa. Estas iteraciones del algoritmo en Blender tienen la nomenclatura de niveles de Catmull-Clark.

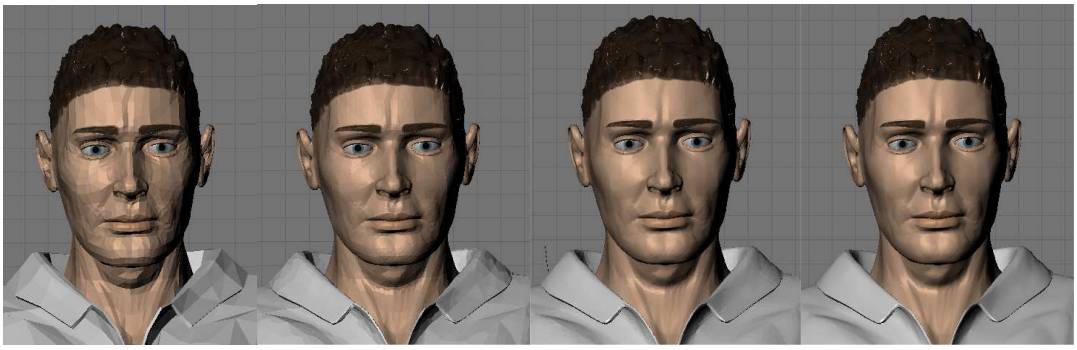


Figura 32: Niveles del algoritmo de Catmull-Clark ordenados de izquierda a derecha

Sin embargo, esta mejora de la visualización conlleva un inconveniente: a mayor nivel de redondeado, mayor número de polígonos a cargar. Por tanto, la visualización y manipulación del modelo requiere mayor carga tanto de procesador como de tarjeta gráfica. Este es el motivo principal que explica la no aplicación de dicha técnica sobre el modelo, ya que el objetivo de la aplicación desarrollada es la utilización en prácticamente cualquier ordenador con requisitos hardware medios.

2.3.1.2.- Esqueleto

Una vez se ha completado el modelado de la malla, habrá que dotarla de un esqueleto para que pueda realizar los movimientos deseados. A este proceso se le conoce normalmente como *skinning* y se fundamenta en la creación de una estructura jerárquica de componentes con el fin de obtener articulaciones que poder manipular. Por tanto el proceso a seguir a grandes rasgos es el siguiente:

- Creación de un *armature* o esqueleto mediante una jerarquía de huesos o *bones*.
- Asignación de vértices de la malla a los huesos del modelo.
- Comprobación de movimientos realizados por las articulaciones.

Cabe comentar que los dos últimos pasos del proceso constituyen un bucle que únicamente finalizará en el caso de que el movimiento realizado sea el correcto.

A continuación se describen cada uno de los pasos de forma más detallada:

Creación de un *armature* o esqueleto mediante una jerarquía de huesos o *bones*

El primer aspecto a tener en cuenta es la posibilidad de movimientos que hay que otorgar al modelo. En este caso y puesto que los movimientos se reducen a la zona del cuello y las cervicales, únicamente necesitamos dos huesos: un hueso que haga referencia a la cabeza y otro referente al resto del cuerpo. Por tanto el hueso de la cabeza es el que realiza las rotaciones sobre el hueso que trabaja como punto de referencia, es decir el referente al resto del cuerpo, que permanece fijo.

De esta forma al añadir un hueso en Blender en la ventana del Visor 3D apreciamos un polígono que representa el hueso añadido. Se puede modificar su orientación y posición para colocarlo en el lugar adecuado de la malla. En este caso se coloca, con la ayuda de las cámaras que por defecto presenta el programa, el hueso de la cabeza de forma vertical pasando por el centro de la cabeza, mientras que el hueso del cuerpo justo debajo con la misma orientación.

Figura 33: Posición de los huesos en el modelo

Una vez colocados se modifica su jerarquía asignando el hueso del cuerpo como "hueso padre" del hueso de la cabeza. Con ello se consigue que los movimientos realizados por el hueso de la cabeza dependan siempre de la posición y orientación del hueso del cuerpo.

Asignación de vértices de la malla a los huesos del modelo

Con los huesos ya colocados estratégicamente, se procede a vincularlos a los vértices que forman la malla. Con ello, además del movimiento del modelo se consigue el efecto denominado "deformación de la malla" que otorga un mayor realismo a la cinemática del movimiento. Las variaciones producidas por el movimiento de unos vértices tendrán un reflejo gradual en sus vértices próximos, dependiendo siempre de la proximidad de éstos y de la estructura ósea que presente el modelo en cuestión.

Por tanto y tras seleccionar un hueso, Blender posee una herramienta para enlazar vértices de la malla al hueso seleccionado. Valiéndose de la ayuda de las cámaras se realiza la selección de los vértices afectados por el movimiento y que, por tanto, formarán parte de la articulación.

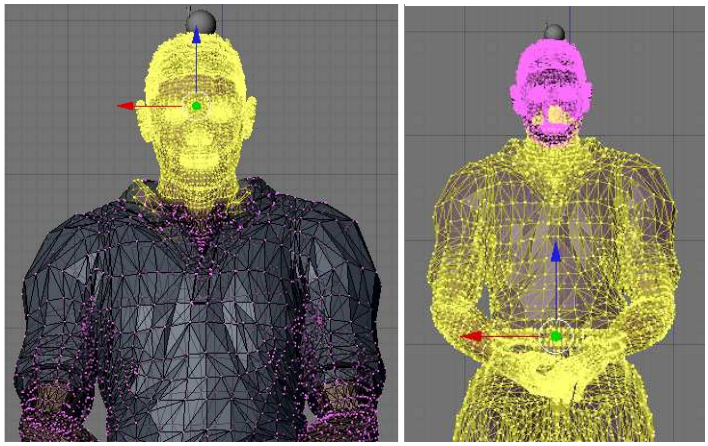


Figura 34: Vértices asociados al hueso de la cabeza (izquierda) y al cuerpo (derecha)

En el modelo implementado, se han distribuido los vértices en grupos en función de la posición de los mismos en el modelo. Así los grupos de vértices son los que se han enlazado a los huesos del esqueleto correspondientes para obtener una animación correcta.

Para evitar futuros problemas de exportación hay que tener en cuenta:

- Todos los vértices de la malla deben estar enlazados a algún hueso. En caso de que esta norma no se siga, se producirán mensajes de advertencia durante la exportación del modelo. En este modelo todos los vértices de la malla que no forman parte de la cabeza o cuello se han asignado al hueso del cuerpo para evitar este problema.

- Generalmente no es problemático tener un mismo vértice asignado a ambos huesos; estará afectado por los movimientos de ambos. En este caso y puesto que un hueso permanece fijo, a los vértices asignados a ambos huesos sólo les afectarán los movimientos realizados por el hueso de la cabeza. Esta característica facilita en gran medida la asignación de vértices ya que no exige un proceso tan minucioso cuando, como es el caso, se dispone de una enorme cantidad de vértices.

Comprobación de movimientos realizados por las articulaciones

La asignación de los vértices de la malla a los huesos es un proceso vinculado a procesos más bien artísticos y que, al igual que la mayor parte del modelado, dependen de la pericia del diseñador más que de procesos sistemáticos. De esta forma este paso consiste en la visualización del movimiento realizado por la articulación y la detección de errores que se hayan podido producir: vértices mal enlazados por una deformación incorrecta de la malla, posibles deficiencias en la colocación general de los huesos, movimientos imposibles debidos a una mala jerarquía de huesos…

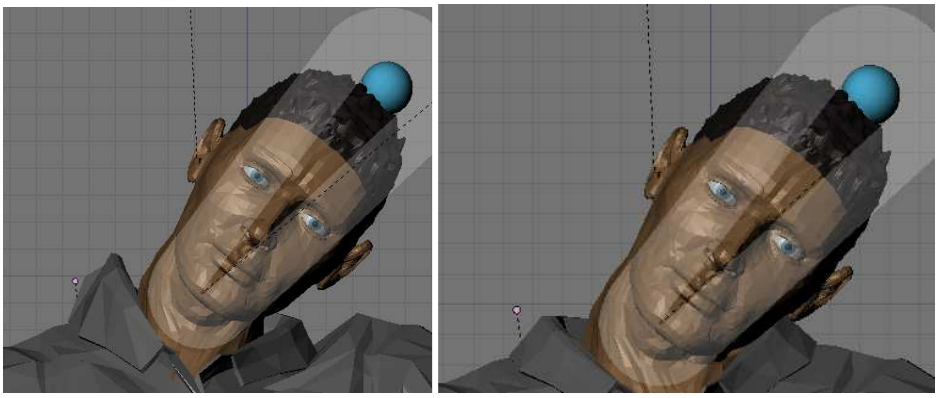


Figura 35: Ejemplo de un movimiento incorrecto (izquierda) y correcto (derecha)

Estos son algunos de los errores más comunes que se pueden detectar en este apartado y que obligan a volver al paso anterior, mediante un proceso similar a "prueba y error", para conseguir un movimiento lo más fidedigno posible a una física realista.

2.3.1.3.- Mapeado de texturas

El siguiente paso para obtener un modelo realista es añadirle las texturas necesarias con el fin de poder incorporar un mayor grado de detalle que el dispuesto mediante la adición de materiales planos.

Para ello se realiza el mapeado de texturas del modelo o también conocido como mapeado UV. Esta técnica consiste, a grandes rasgos, en la adición de una textura 2D dividida en función de la malla, con el fin de adaptarla lo más fielmente posible al perfil del modelo. La función principal de este tipo de técnica es la de otorgar al modelo los detalles que la geometría en sí no puede proporcionarle.

A continuación se realiza una descripción de cada uno de los pasos para poder aplicar dicha técnica al modelo en Blender.

En primer lugar, se selecciona la malla de la cual se desea obtener su mapeado UV y se crea una imagen sobre la que se va a colocar el desplegado de la malla. Con la malla seleccionada se emplea la función Unwrap y ya se obtiene la malla desplegada sobre la imagen creada.

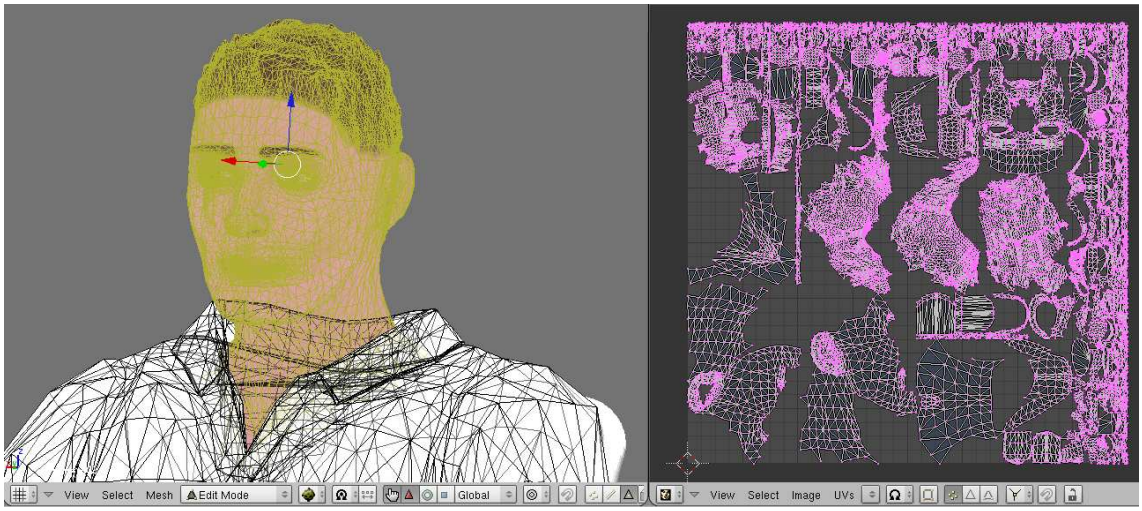


Figura 36: Desplegado de la malla correspondiente a la cabeza del modelo

El siguiente paso consiste en colorear la textura para dotarla de los detalles que se quieren añadir al modelo. Para esta función existen multitud de programas de dibujado avanzados que se pueden emplear; sin embargo Blender incorpora un modo llamado *Texture Paint* que permite pintar la textura UV sobre la malla, de forma equivalente a como en la realidad se pintaría una escultura. Así a medida que se va coloreando sobre el modelo, se hace la correspondencia con las coordenadas de la textura desplegada y se va obteniendo la textura final.

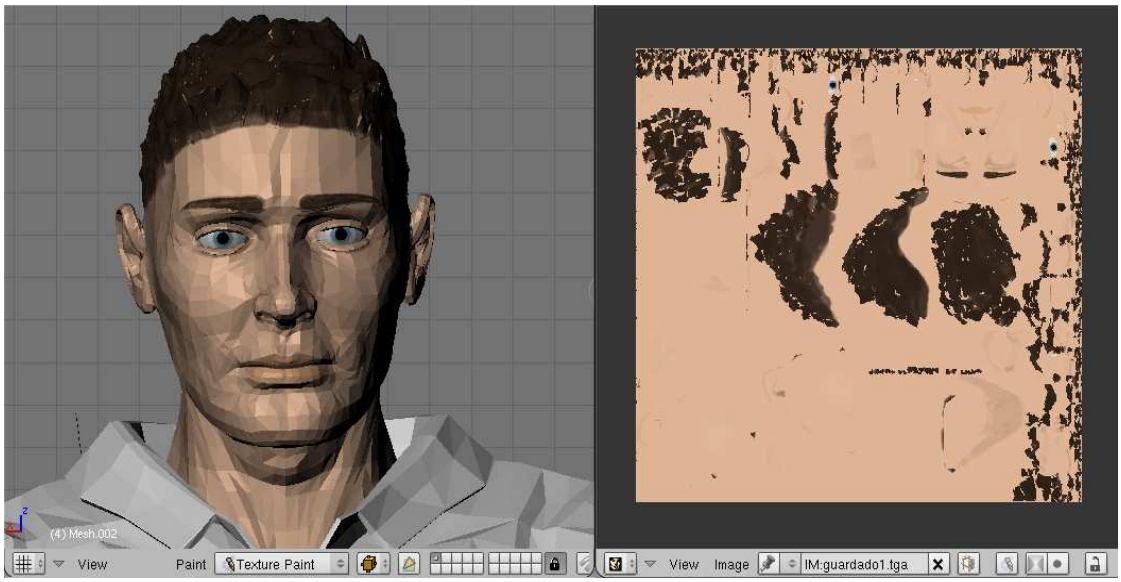


Figura 37: Textura UV coloreada de la malla correspondiente a la cabeza del modelo

2.3.1.4.- Exportación del modelo

En este momento se dispone de un modelo completo texturizado con todo el abanico de posibilidades de movimiento que se la ha querido otorgar; sin embargo, hay que prepararlo para una exportación satisfactoria.

Para ello y puesto que el fin es utilizar el modelo junto con el motor gráfico Ogre, se ha incorporado el script *Ogre Meshes Exporter* que permite exportar el modelo realizado en Blender a archivos reconocibles por Ogre. Se instala en una carpeta de Blender y de inmediato se puede acceder a su funcionalidad a través de Blender.

Ogre Meshes Exporter

Este script permite la exportación de "*meshes*" (mallas) y "*armatures*" (esqueletos) a un fichero en formato Ogre (XML / binary mesh). Además también permite exportar:

- Mallas con vértices coloreados, múltiples materiales y texturas UV.
- Una gran cantidad de propiedades referidas a los materiales tales como: ambiental, difusa, especular, emisivo...
- Animaciones basadas en keyframes enlazadas al esqueleto del modelo.
- Animaciones basadas en keyframes asociadas a la colocación y deformación de mallas.

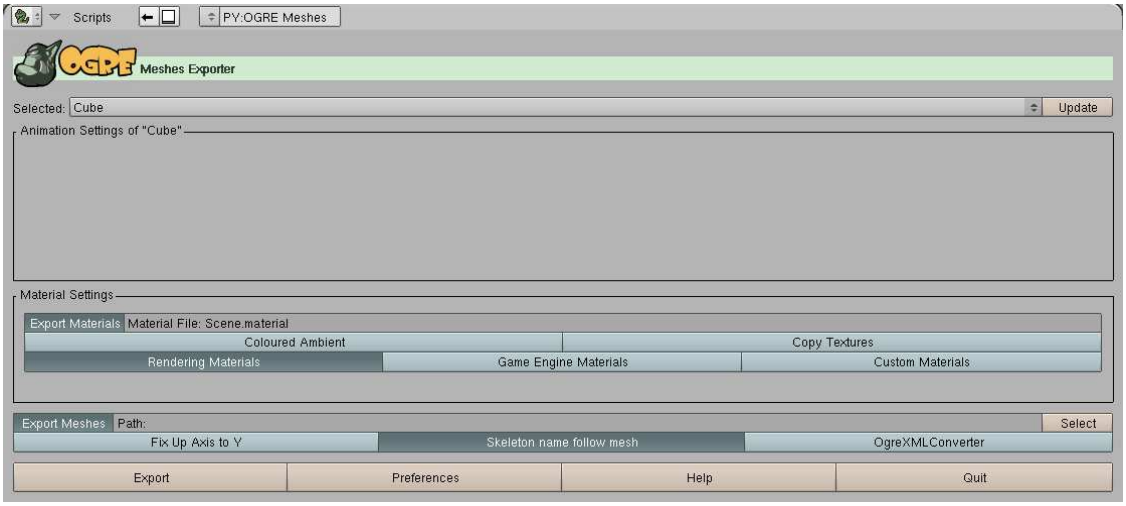


Figura 38: Script Ogre Meshes Exporter

Al visualizar la *Figura 38* se puede comprobar que se exporta únicamente las mallas seleccionadas en Blender, pudiendo presionar el botón *Update* para actualizar las mallas que se exportarán.

Otras opciones que proporciona la ventana principal son la exportación de un fichero de materiales. Para esta exportación se pueden seleccionar las siguientes propiedades: seleccionar color ambiente del modelo o copiar texturas del modelo y, a elegir una entre tres, renderizar materiales, obtener materiales del Game Engine (materiales asociados al motor gráfico) o materiales por defecto.

Por último nos da la opción de intercambiar ejes X e Y, nombrar los esqueletos exportados de la misma forma que las mallas y la posibilidad de usar conjuntamente a la exportación el script *OgreXMLConverter*. Cabe tener muy en cuenta esta última acción: los archivos exportados no se pueden usar directamente en Ogre, necesitan de otro script para convertirlos a archivos .mesh.

En este punto existen dos opciones: exportar el modelo seleccionado a .xml y posteriormente usar el script *OgreXMLConverter* desde una línea de comandos; o seleccionar en *Preferences* la ruta del script y permitir que al hacer la exportación se conviertan de forma automática.

Si se sigue el primer caso y se exportan conjuntamente materiales, esqueletos y mallas se dispone de tres tipos de archivos: archivos ".material", archivos ".skeleton.xml" y archivos ".mesh.xml".

Figura 39: Ejemplos de tipos de archivo obtenidos al exportar

Los archivos .material se pueden enlazar de forma automática con la librería gráfica Ogre, mientras que los otros tipos de archivo obtenidos requieren una transformación previa realizada por *OgreXMLConverter*.

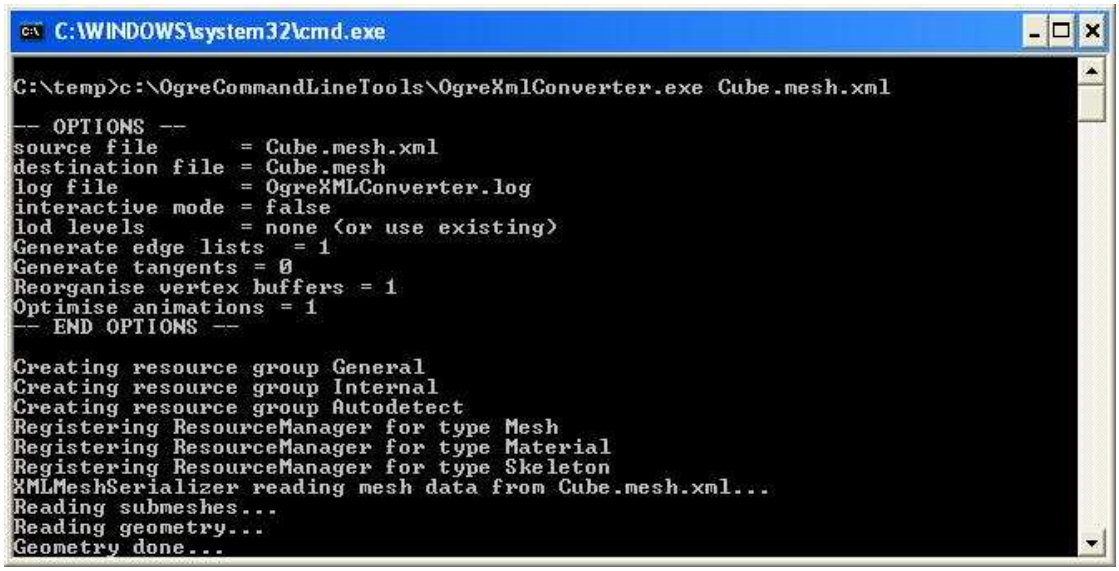


Figura 40: OgreXMLConverter desde línea de comandos

Como resultado de esta acción, que sería el mismo si se selecciona la segunda opción comentada al exportar, se obtienen los siguientes tipos de archivos de archivo .skeleton y .mesh que ya se podrán utilizar con la librería Ogre.

Figura 41: Archivos .mesh y .skeleton

Una vez se ha realizado una inicialización al funcionamiento básico del script de exportación a Ogre, se han contemplado tres estrategias de exportación de mallas:

- Exportación íntegra de las mallas y esqueleto del modelo.
- Exportación por partes o zonas afectadas por las articulaciones.
- Exportación por mallas con distintas texturas.

A continuación se analiza cada posibilidad comentando aspectos favorables y adversos para su elección como forma de exportación idónea:

<u>Exportación íntegra de las mallas y esqueleto del modelo</u>
Para realizar esta exportación basta con seleccionar todo el modelo y exportarlo con Ogre Meshes Exporter.

- Aspectos favorables:
  - Forma más simple de exportación.
  - No se producen más archivos que los necesarios para exportar el modelo.

- Aspectos adversos:
  - Requiere un paso previo de asignación de materiales a las mallas.
  - Los materiales exportados por Blender son muy simples y no responden a aspectos como las sombras o el punto de vista de la cámara; se reducen a colores planos. Se requiere un postproceso de los materiales.
  - El fichero generado que representa a la malla del modelo tiene un tamaño demasiado grande para ser interpretado por el script Ogre Meshes Exporter, ya que genera problemas de memoria.

Exportación por partes o zonas afectadas por las articulaciones

Esta exportación se divide en dos secciones: exportar en primer lugar la zona del cuello y la cabeza, partes que están afectadas por el movimiento reproducido; y en segundo lugar exportar el resto de elementos, zonas que permanecen fijas. Para ello basta con seleccionar la zona de la cabeza, actualizar en *Ogre Meshes Exporter* la zona a exportar y, posteriormente, realizar el mismo proceso para el resto del cuerpo.

- Aspectos favorables:
  - División clara de mallas en ficheros diferenciando los que estarán afectados por el movimiento y los que no.
  - Se producen pocos ficheros de mallas (uno para la cabeza y otro para el resto del cuerpo).
- Aspectos adversos:
  - Mismos problemas que los comentados en el tipo de exportación anterior, a pesar de que el fichero de malla tiene un tamaño menor al estar dividido.

Exportación por mallas con distintas texturas

Esta forma de exportar el modelo se basa en exportar cada malla que disponga de una textura distinta a sus vecinas en un fichero distinto. Para ello se selecciona cada vez la zona que se desea exportar y se actualiza dicha selección mediante Ogre Meshes Exporter.

- Aspectos favorables:
  - División por mallas con distintas texturas que permite realizar un postproceso de los materiales ajeno a la exportación.
  - Creación, asignación y tratamiento sencillo de los scripts de materiales de las mallas.
  - Enlace de los ficheros .skeleton únicamente a las mallas necesarias que reproducen movimiento.
- Aspectos adversos:
  - Generación de bastantes ficheros de mallas.
  - No se distingue entre las mallas que tienen vinculadas un esqueleto y las que no.
  - Enlace por partes de los materiales al modelo (un enlace por cada malla creada).

Por tanto y comprobadas las ventajas e inconvenientes de cada tipo de exportación, la estrategia elegida para exportar el modelo es una combinación del segundo y tercer tipos comentados. Con esta elección, se obtiene una malla que representa a la cabeza y tendrá asociada un esqueleto, y una malla por cada mapeado de textura distinto realizado sobre los elementos fijos del modelo. De esta manera se obtienen cinco archivos *.mesh* asociados a la cabeza, los brazos, la camiseta, los pantalones y los zapatos respectivamente, y un archivo *.skeleton* asociado a la malla de la cabeza que contiene los elementos necesarios para permitir el movimiento deseado en el modelo.

Teniendo en cuenta la estrategia de exportación comentada, únicamente queda definir el resto de parámetros de exportación en el script *Ogre Meshes Exporter*, asociados en su gran mayoría a la forma de exportación de los materiales. Los parámetros que permiten exportar las texturas UV implementadas se resumen en seleccionar la opción de copiar texturas y utilizar el modo de visualización Game Engine como base para el renderizado de materiales.

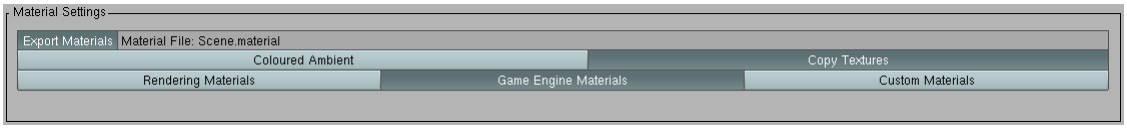


Figura 42: Opciones de exportación

2.3.2.- Ogre

El motor gráfico Ogre permite la manipulación y tratado de gráficos 3D que, generalmente, se han creado mediante modeladores como Blender. Por tanto, para continuar con este apartado se dispone ya de un modelo con una estructura ósea predefinida para realizar determinados movimientos que impliquen deformar las mallas creadas. Sin embargo cabe indicar que, aunque no sea el caso del proyecto actual, Ogre también otorga la posibilidad de definir el esqueleto del modelo como un paso posterior y ajeno al programa encargado del modelado del mismo.

2.3.2.1.- Cargar el modelo

La información que contiene todos los datos de la malla del modelo generado en Blender se almacena en dos tipos de archivo: archivos *.mesh* y archivos *.skeleton*. Estos tipos de archivo permiten la carga de mallas y manipulación de esqueletos mediante la utilización de la librería proporcionada por Ogre. Si se observa su estructura, se trata de simples archivos de texto con una estructura definida que Ogre puede interpretar para obtener los datos necesarios de la misma.

La estructura general de estos dos tipos de archivo es la siguiente:

Un archivo ***.mesh****.xml* tiene las siguientes propiedades asociadas a la malla:

- Definición de caras (<face>).
- Definición de vértices (<vertex>) con su posición, normal, color y coordenadas UV.
- Asignación de vértices a huesos (<vertexboneassignment>).
- Enlace a un esqueleto (<skeletonlink>).

```
<mesh>
	<submeshes>
		<submesh material="Material" usesharedvertices="false">
			<faces count="16875">
				<face v1="0" v2="1" v3="2"/>
				<face v1="3" v2="4" v3="5"/>
					(...)
			</faces>
			<geometry vertexcount="50625">
				<vertexbuffer positions="true" normals="true" colours_diffuse="true" texture_coords="1">
					<vertex>
						<position x="-51.373001" y="-118.497002" z="-300.509003"/>
						<normal x="0.130238" y="-0.267648" z="-0.954674"/>
						<colour_diffuse value="1.000000 1.000000 1.000000 1.000000"/>
						<texcoord u="0.000000" v="1.000000"/>
					</vertex>
					(...)
				</vertexbuffer>
			</geometry>
			<boneassignments>
				<vertexboneassignment vertexindex="0" boneindex="1" weight="1.000000"/>
				<vertexboneassignment vertexindex="1" boneindex="1" weight="1.000000"/>
					(...)
			</boneassignments>
		</submesh>
	</submeshes>
	<skeletonlink name="Mesh.005.skeleton"/>
</mesh>
```

Figura 43: Estructura general de un archivo .mesh.xml

Por otro lado, en un fichero ***.skeleton**.xml* se distinguen los siguientes campos:

- Definición de huesos (<bone>) con su posición y ángulo de rotación.
- Jerarquía de huesos (<bonehierarchy>).

```
<skeleton>
	<bones>
		<bone id="1" name="resto">
			<position x="-16.815542" y="-194.056992" z="-413.352692"/>
			<rotation angle="3.141592">
				<axis x="-0.000000" y="0.719685" z="-0.694301"/>
			</rotation>
		</bone>
		<bone id="0" name="Bone.004">
			<position x="1.521393" y="199.527191" z="-5.474000"/>
			<rotation angle="3.105691">
				<axis x="-1.000000" y="0.000000" z="0.000000"/>
			</rotation>
		</bone>
		<bone id="2" name="cabeza">
			<position x="-0.000001" y="80.504387" z="0.000012"/>
			<rotation angle="0.046251">
				<axis x="0.776884" y="0.628507" z="-0.037812"/>
			</rotation>
		</bone>
	</bones>
	<bonehierarchy>
		<boneparent bone="Bone.004" parent="resto" />
		<boneparent bone="cabeza" parent="resto" />
	</bonehierarchy>
</skeleton>
```

Figura 44: Estructura general de un archivo .skeleton.xml

Nota: Cabe recordar que estos archivos deben ser transformados mediante el script *OgreXMLConverter* para poder utilizarlos en Ogre de manera directa.

Para poder cargarlos, estos archivos se deben incluir en el directorio adecuado de la librería ("Directorio_de_Ogre\Samples\Media\models"). En este directorio también se encuentran otros archivos *.mesh* y *.skeleton* creados por defecto al instalar la librería.

Con los archivos almacenados en las rutas comentadas, hay que describir la estructura de elementos que permiten la manipulación de escenas en Ogre. En Ogre existen tres elementos fundamentales a considerar: *Entity*, *SceneNode* y *SceneManager*.

- *Entity* o entidad: corresponde a todos los elementos que se pueden dibujar en pantalla. Cualquier modelo, malla u objeto que carguemos es considerado una entidad; sin embargo no son entidades las cámaras, las luces… Se puede considerar a las entidades como la abstracción que representa un objeto en pantalla. Cabe comentar que la posición y la orientación geométrica de las entidades no es una característica de las mismas, estos parámetros son controlados por otra estructura; sin embargo, por ejemplo el material o la textura que cargan sí lo es.

- *SceneNode* o nodo de la escena: se encargan de las manipular las características de las entidades. Se trata de una especie de contendores en los que se engloban las entidades, cámaras y luces para modificar sus propiedades. Es posible crear una jerarquía de nodos de la escena ya que un nodo puede ser padre de otro. Como es coherente, la posición siempre es relativa a la de los nodos padre.

- *SceneManager* o controlador de nodos: es el nodo raíz de todo lo que se dibuja y tiene que ver en la escena. Siempre se sitúa como nodo raíz en la jerarquía de nodos y, por tanto, a través de él siempre es posible acceder a cualquier entidad o nodo creado en la escena.

Por tanto y tras la descripción de estos tres elementos fundamentales, es simple realizar de forma teórica una jerarquía de los nodos presentes en la escena del proyecto actual.

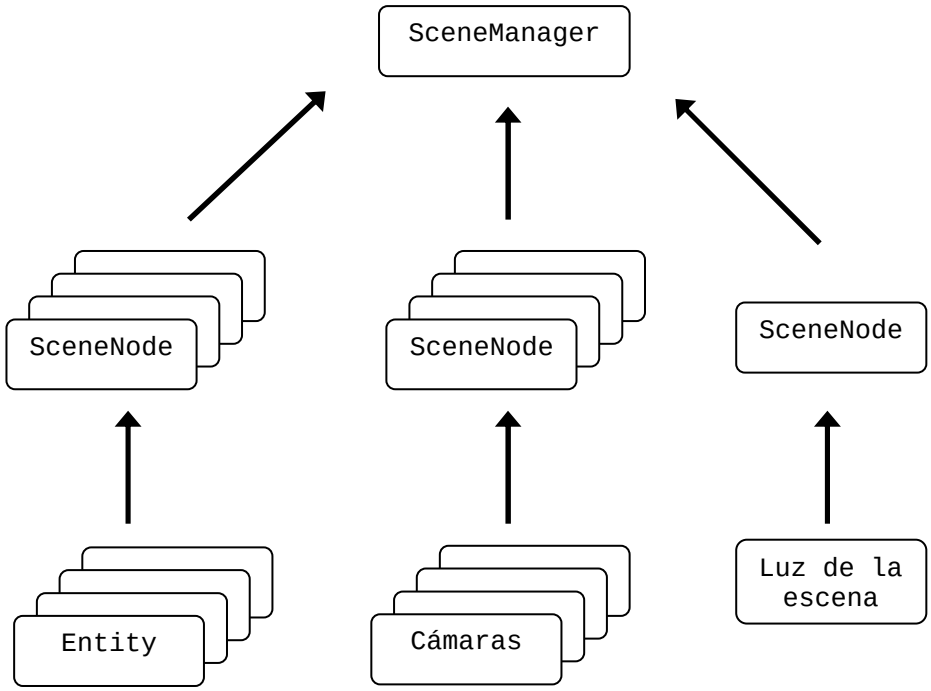


Figura 45: Jerarquía teórica de nodos de la escena

Se dispone de tantas entidades como mallas cargadas, teniendo cada una enlazada a un nodo de la escena (puede haber más de una entidad enlazada al mismo nodo si comparten el mismo patrón de orientación y posición geométrica). Del mismo modo se sigue el mismo procedimiento para las cámaras y para la luz presente en la escena.

Otro punto a comentar es el posicionamiento de los objetos en la escena. Este se realiza a través de la definición de coordenadas en el espacio de los ejes X, Y, Z. El eje X se define de izquierda a derecha de la pantalla, ubicando su parte positiva a la derecha; el eje Y de abajo hacia arriba de la pantalla, con su parte positiva situada arriba; mientras que el eje Z desde dentro hacia fuera de la pantalla, teniendo su zona positiva hacia fuera.

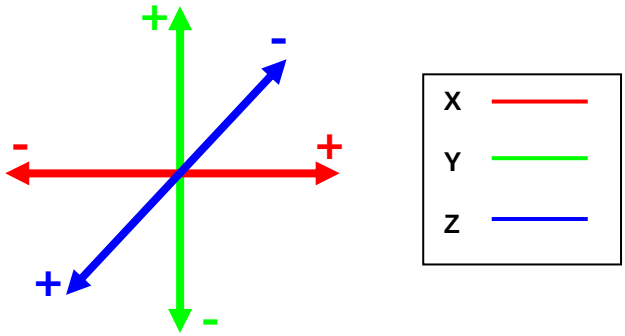


Figura 46: Distribución y signo de los ejes X, Y, Z

Ogre, por tanto, emplea álgebra tridimensional para definir la posición de los objetos. Esto se realiza a través de vectores, que utilizan la clase interna `Vector3`, con la definición de las coordenadas de los tres ejes.

Una vez definidos los aspectos previos esenciales para entender la carga de objetos, únicamente se debe utilizar el método `createEntity("Nombre de la entidad", "archivo.mesh")` para definir la *Entity* que representa a una malla y, posteriormente, enlazar dicha entidad a un *SceneNode* mediante el método `attachObject(entidad_creada)`. Así ya se dispone de toda la infraestructura necesaria para manipular la posición, orientación y demás características referentes al objeto en la escena.

Para no confundir al usuario con las distintas orientaciones que puede tomar la cámara libre de la escena, se ha incorporado unos ejes cuya orientación está vinculada a la de la cámara de la escena. Estos ejes se encuentran superpuestos a la escena, manteniendo siempre fija su posición respecto de la ventana de Ogre. Para ello se ha utilizado un elemento llamado `Overlay` que incorpora Ogre, a la que se le ha añadido la malla con los ejes indicados.

Finalmente la carga del modelo completo para el proyecto actual genera la siguiente jerarquía de nodos en la escena:

- SceneManager
- SceneNode: nodo del cuerpo
    - Entity: malla de los brazos
    - Entity: malla de los pantalones
    - Entity: malla de los zapatos
    - Entity: malla de la camiseta
    - Entity: malla de la cabeza
- SceneNode: nodo para el resto de elementos de la escena
    - Entity: malla de la parte inferior de la silla
    - Entity: malla de la parte superior de la silla
- Entity: ejes superpuestos vinculados con la orientación de la cámara

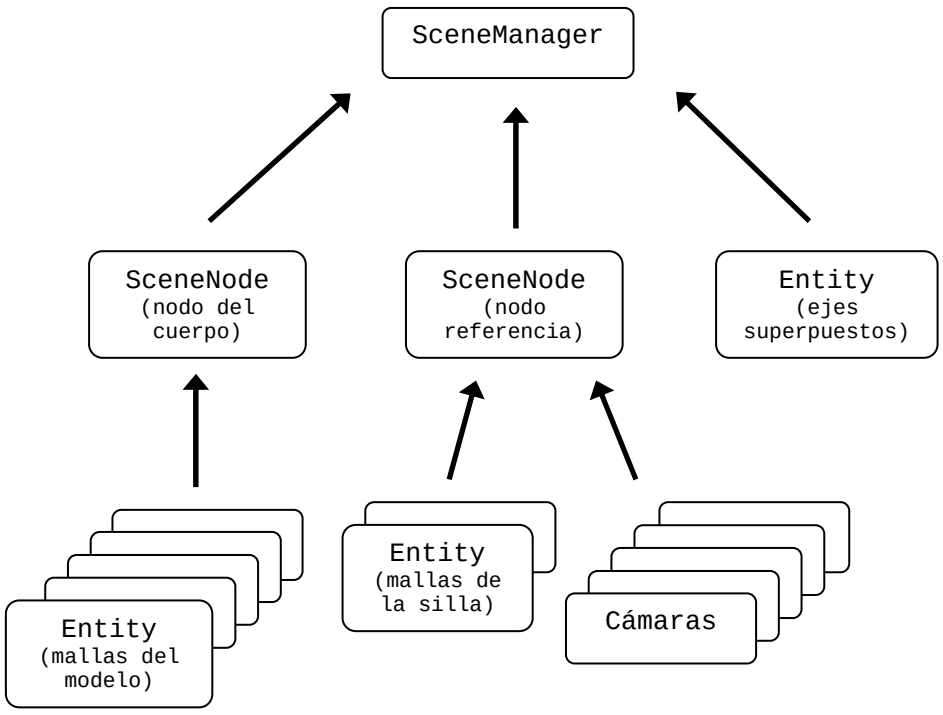


Figura 47: Jerarquía de nodos del proyecto actual

2.3.2.2.- Añadir materiales al modelo texturizado

Para añadir materiales a un conjunto de mallas, Ogre se basa en el enlace a unos scripts que son cargados una vez se cargan los recursos de la ventana de Ogre. Estos scripts son conocidos como scripts de materiales.

Scripts de materiales

Los scripts de materiales ofrecen la posibilidad de definir complejos materiales que pueden ser reutilizados de forma sencilla. Los scripts se definen en ficheros ajenos al código de la aplicación permitiendo, además de reducir el tamaño del código, su utilización mediante el empleo de las clases Material y TextureLayer.

La carga de estos scripts se realiza cuando los archivos de recursos son inicializados: Ogre localiza en todos los directorios de recursos los archivos *.material* y permite su utilización. Sin embargo cabe comentar que no se cargan completamente con este proceso; únicamente se cargan las definiciones, dejando la carga de texturas y otros recursos para su uso posterior. Esto es debido a que normalmente las librerías de materiales son de gran tamaño pero sólo se suele utiliza una pequeña cantidad de estos en cada aplicación.

Para incluir scripts de materiales propios sin necesidad de cargarlos de forma manual se pueden introducir el archivo correspondiente en el siguiente directorio de Ogre: "Directorio_de_librería_Ogre\Samples\Media\materials\scripts".

El lenguaje utilizado para definir los scripts de materiales es una especie de pseudo-C++ en los que cada sección está acotada por llaves ('{', '}') y los comentarios son señalizados con '//'. Las secciones más importantes son las siguientes:

- *material* nombre: aquí se especifica el nombre que tendrá el material. Debe ser único puesto que será la forma de enlazarlo para su utilización en Ogre utilizando el método `setMaterialName(nombre)`.
- *technique*: identifica la zona en la que se define el material. Puede haber más de una zona technique dentro de un material: se identifica la primera como la preferencial y, si se da el caso de que no pueda renderizarla, acudiría a las otras definiciones.
- *pass*: cada zona technique puede estar formada por una o múltiples secciones pass. Estas secciones se encargan del renderizado completo del objeto: cada pass con unas propiedades definidas se genera en un tiempo determinado y la combinación de todas estas secciones producen la composición del material del objeto. En esta sección se especifican los valores de color ambiente, difuso, especular o emisivo.
- *texture_unit*: dentro de cada sección pass se pueden definir las texturas a utilizar. En su interior se pueden incluir propiedades referentes a efectos de las texturas. Esta sección no es obligatoria para la carga de materiales, pero se emplea en la mayoría de definiciones.

Además de las secciones comentadas también existe la posibilidad de definir referencias a vértices, fragmentos de programa y shaders para la creación de materiales más complejos.

Una vez definida la teoría sobre los scripts de materiales, es conveniente explicar su manipulación para el proyecto actual. Al exportar el modelo 3D desde Blender, se crea de forma automática un archivo de extensión *.material* que contiene el enlace de cada uno de los materiales empleados con la información de las texturas UV. De esta forma si las texturas UV se añaden a la ruta "Directorio_de_librería_Ogre\Samples\Media\materials\textures" para que Ogre

pueda acceder a las mismas y el archivo *.material* generado también se coloca en la ruta correcta, la visualización de materiales debería ser correcta y visualmente realista. Sin embargo esto no sucede así debido a una consideración: los materiales añadidos son ajenos a la posición de la cámara y la luz y, por tanto, se muestran como materiales planos sin sombras.

Para solucionar este aspecto es necesario combinar los archivos *.material* generados de forma automática con Blender con funciones de los scripts de materiales. Para ello, se ha definido un nuevo módulo *texture_unit* dentro de la sección *pass* que combina la textura UV definida con una textura a la que afectan los cambios de luminosidad en función de la posición y orientación de la cámara que permite visualizar el modelo.

Figura 48: Textura afectada por la luminosidad

Siguiendo este proceso para cada uno de los materiales de los que dispone el modelo, la visualización mejora considerablemente ya que es mucho más realista.

```
material Material.012/SOLID/TEX/camisa.tga/VertCol
{
	technique
	{
		pass
		{
			diffuse vertexcolour
			specular 0.500000 0.500000 0.500000 12.500000
			texture_unit
			{
				texture camisa.png
			}
			ambient 0.9 0.7 0.6
			emissive 0.5 0.5 0.5

			texture_unit
			{
				texture flare.png
				colour_op_ex add_signed src_texture src_current
				env_map planar
			}
		}
	}
}
```

Figura 49: Ejemplo de un archivo .material: material referido a la camiseta del modelo

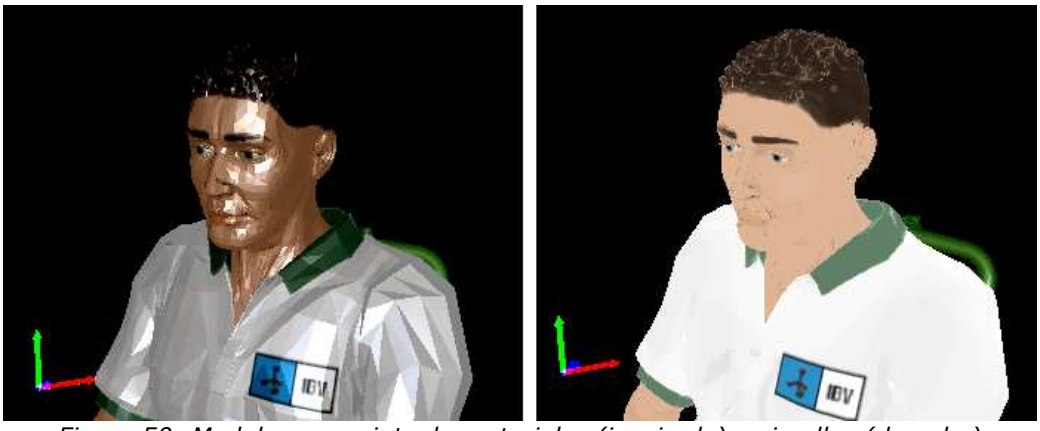


Figura 50: Modelo con scripts de materiales (izquierda) y sin ellos (derecha)

2.3.2.3.- Animación del modelo

Este paso se basa en la animación del modelo que ya se encuentra cargado según los requisitos del motor gráfico de Ogre. En primer lugar debemos leer los datos que permiten realizar la animación para, una vez almacenados adecuadamente, poder interpretarlos y conseguir un efecto de correlación entre los movimientos del modelo.

Los datos se obtienen de los ficheros de prueba (descritos el apartado 2.2.3) y simplemente hay que almacenarlos en una estructura creada con los siguientes campos:

```
typedef struct{
      float fTime;      //tiempo de la medición
      double ang_x;     //angulo X
      double ang_y;     //angulo Y
      double ang_z;     //angulo Z
}stAngleMarkTime;
```

Cabe comentar que los datos de los ángulos en los ficheros de prueba se encuentran en grados. Por tanto y con el fin de facilitar cálculos posteriores, se ha obtenido y almacenado su equivalencia en radianes mediante el método `DegreesToRadians` de la clase Math que proporciona Ogre.

Una vez se dispone de estos datos el siguiente paso para conseguir una animación fluida es interpretarlos. Se debe seguir una cálculo matemático que permita combinar los valores de los ángulos en los tres ejes posibles en uno solo movimiento.

De esta forma Ogre permite la rotación de objetos mediante tres órdenes sencillas: *pitch*, *yaw* y *roll*. Estos tres métodos se basan en proporcionar un determinado giro (respecto al eje X en caso de *pitch*, eje Y en caso de *yaw* y eje Z en *roll*) teniendo como datos de entrada un ángulo y el punto de referencia respecto del cual se produce la rotación (puede ser local, del mundo o referente al nodo padre). El problema que se encuentra para utilizar estas órdenes es que aunque se combinen, el movimiento se realiza de forma secuencial: primero se gira en el eje X, después en el eje Y y por último en el Z (el orden es arbitrario). Esto provoca problemas a la hora de conseguir un movimiento realista y fluido; es necesario combinar los tres ángulos.

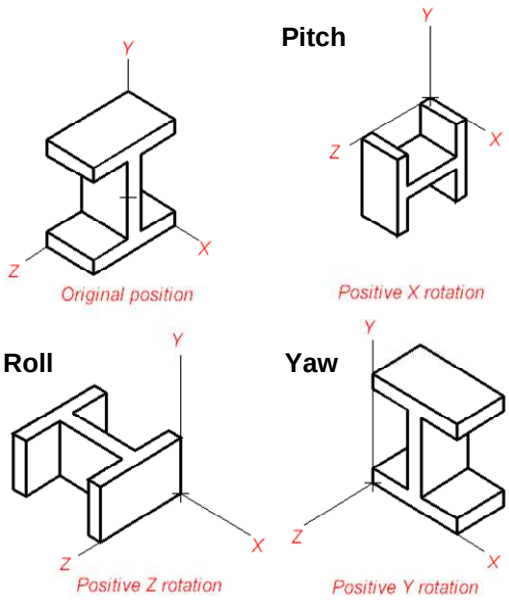


Figura 51: Ejemplo de rotación en los ejes X, Y, Z

Siguiendo con este razonamiento, es imprescindible describir dos conceptos previos relacionados con la rotación: la utilización de matrices de rotación y de *quaternions (*cuaterniones*)*.

Las matrices de rotación permiten, mediante la multiplicación de las mismas, combinar la rotación producida en ejes distintos. Así disponemos de una matriz de rotación por cada eje:

Rotación en X:

$$\begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix} * \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

Rotación en Y:

$$\begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} * \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

Rotación en Z:

$$\begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

Hay que tener en cuenta que la multiplicación de matrices no es conmutativa: no es lo mismo A * B que B * A. Por este motivo la multiplicación debe realizarse en el orden Rotación_X * Rotación_Y * Rotación_Z.

Por otra parte, los *quaternions* o cuaterniones proporcionan una notación matemática para representar las orientaciones y rotaciones de los objetos en tres dimensiones. El motivo principal de su utilización es que Ogre los emplea para representar rotaciones; sin embargo tienen las siguientes ventajas:

- Son más eficientes y más estables numéricamente que las matrices de rotación.
- Facilitan la modificación de rotaciones. Al igual que las matrices de rotación, mediante la multiplicación de cuaterniones se pueden combinar ángulos de giro.
- Evita la alineación costosa de la deriva de la matriz. Este hecho se produce cuando se tienen una gran cantidad de productos entre matrices debido al redondeo de números reales y pueden provocar anomalías en la rotación.

- Interpolación de rotaciones. Los cuaterniones incorporan dos métodos de interpolación: lineal y cúbica. La interpolación lineal realiza un cálculo más eficiente pero no consigue la fluidez de resultados que implementa la interpolación cúbica.

Sin embargo y con el fin de facilitar la creación de cuaterniones, Ogre permite su generación a partir de matrices de rotación mediante el método `FromRotationMatrix`. El hecho de que también incorpore una implementación de matrices 3x3 (mediante la clase `Matrix3`) y tenga sobrecargado el operador '*' para realizar el producto de matrices favorece en gran parte la utilización de esta técnica.

Una vez aclarados los aspectos matemáticos para el tratamiento de rotaciones en Ogre, el foco de atención se debe centrar en la implementación de la estructura o técnica que permita realizar la animación deseada. Los datos ya se encuentran organizados y se conoce la forma de interpretarlos, falta un método que permita encapsularlos en un movimiento.

La técnica utilizada es la implementación de una línea de tiempo en la que se definen una serie de *KeyFrames* o fotogramas clave. Los fotogramas clave son eventos o acciones que ocurren en un momento determinado dentro de los límites de una línea de tiempo definida. De esta forma en el caso del proyecto actual es simple determinar cómo se emplea esta técnica: la línea de tiempo de cada animación está vinculada al valor de tiempo almacenado en la estructura; mientras que los eventos o acciones que representan los *KeyFrames* se corresponden al ángulo de giro que debe tener la articulación del cuello del modelo. En consecuencia se dispone de un intervalo entre los fotogramas clave equivalente a la frecuencia con la que se toman las muestras (0.04 segundos) que Ogre se encarga de interpolar, según un patrón de interpolación marcado, con el objetivo de conseguir una animación fluida.

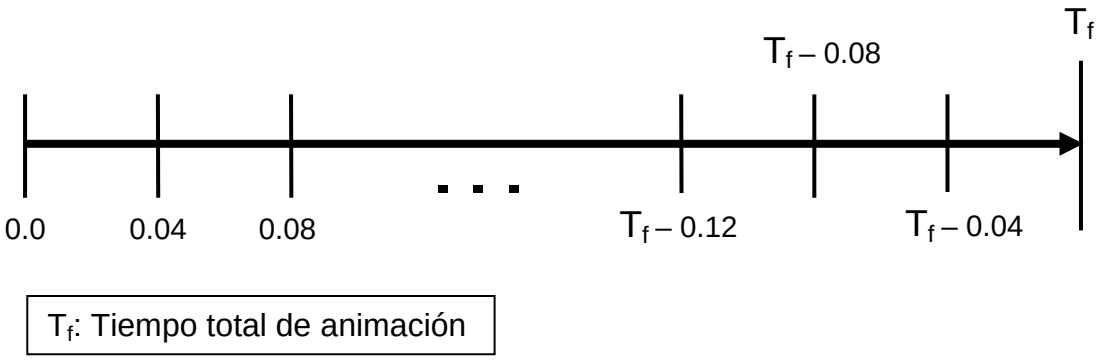


Figura 52: Línea de tiempo tipo de una animación

Teniendo toda la base teórica para realizar una animación, el último paso se centra en vincular la animación al modelo cargado. Para ello hay que emplear el esqueleto generado en Blender y exportado en el fichero .skeleton correspondiente.

Mediante la entidad que define la malla que debe realizar el movimiento es posible obtener el esqueleto mediante el método `getSkeleton()`. Este esqueleto contiene la definición de los huesos que forman las articulaciones del modelo, accesibles mediante el método `getBone("nombre_del_hueso")`. En este punto es importante recordar los nombres puestos en Blender a cada uno de los huesos puesto que son

necesarios para realizar su manipulación, hecho que se produce poniendo a `true` el método `setManuallyControlled(bool)` relativo al hueso obtenido.

En este momento ya se puede controlar de forma manual el hueso considerado, que en el proyecto actual se trata únicamente del hueso correspondiente a la cabeza.

El último paso para finalizar el proceso de creación de la animación es vincular el hueso obtenido a una línea de tiempo dividida en fotogramas clave. En primer lugar se crea la animación que posee como argumentos un nombre y un tiempo total de reproducción. En segundo, se permite que el manejador del hueso considerado y el hueso en sí sean controlados por la animación creada a través del método `createNodeTrack(manejador_del_hueso,hueso)`. En tercero y último, se declara un estado de animación con un nombre que representa a la misma en el bucle de renderizado.

A partir de este momento se rellenan los fotogramas clave: se sitúan en un tiempo igual a la frecuencia con la que se obtienen las muestras y tienen un valor obtenido por la multiplicación de las matrices de rotación, encapsulado en los cuaterniones correspondientes.

Con todos los datos necesarios para generar la animación, únicamente queda realizar su renderizado en escena para comprobar los resultados. Ogre utiliza un bucle o ciclo de renderizado llamado de forma continua que permite ir actualizando los momentos de animación (el método `startRendering` inicia este renderizado).
A partir de este momento la función a la que se accede y permite el control de parámetros de la animación (reproducción de un momento de animación concreto, modificación de cámaras y luces…) es `frameStarted(const FrameEvent &evt)`. Como se comprueba en la cabecera, este método responde a "eventos de frame", es decir, se ejecuta cada vez que un frame está en disposición de ser renderizado. Por tanto para dar la sensación de movimiento fluido, se hace variar el momento de la animación en función de estos eventos mediante el método `addTime(evt.timeSinceLastFrame)` asociado al estado de la animación declarado de forma previa.

2.3.2.4.- Interacción con el usuario

La interacción con el usuario en el caso de la animación en Ogre está limitada en cuanto a sus posibilidades debido a que se va a integrar como una ventana adicional en otra interfaz. Sin embargo, cabe comentar que Ogre permite la captura de todo tipo de eventos: eventos de ratón, de teclado e incluso la posibilidad de incorporar joysticks son contemplados por este motor gráfico.

En contraposición, en el proyecto actual únicamente se ha considerado la captura de eventos de teclado como función del motor gráfico; el resto de eventos están vinculados a las acciones que se lleven a cabo en la interfaz y se trasfieren mediante métodos vinculados a dicho motor gráfico. En consecuencia, junto con la carga de la ventana se debe iniciar un recurso que implemente un manejador de eventos de teclado, enlazado como es coherente a los eventos que sucedan en la misma ventana (método `setupInputSystem`).

NOTA: una breve descripción de los recursos inicializados por Ogre se encuentra en el apartado 2.2.2.1 como métodos de la clase `COgreApplication`. Posteriormente en esta memoria se explica de forma más detallada cada uno de ellos.

Con el fin de dividir la funcionalidad referida a la carga de recursos de la ventana Ogre y la captura de eventos sobre la misma se ha creado la clase `MyFrameListener`. En esta clase se incorporan todos los métodos vinculados a acciones de la interfaz, reproducción de la animación y, por supuesto, eventos de teclado. La función que representa el ciclo de animación es el lugar destinado a controlar la pulsación de teclas, puesto que al ser ejecutada de forma continua, permite una respuesta en tiempo real de los elementos modificables en la animación.

La pulsación de teclas contemplada se ha implementado con el fin de variar la posición de la cámara presente en la escena. Aunque esta funcionalidad también se incorpora a la interfaz, la facilidad que proporciona el teclado es lo suficientemente atractiva e intuitiva como para considerar su incorporación. Se ha vinculado funcionalidad a las siguientes teclas:

D, →: aumenta coordenada X
A, ←: disminuye coordenada X
Q, Re Pág: aumenta coordenada Y
E, Av Pág: disminuye coordenada Y
S, ↓: aumenta coordenada Z
W, ↑: disminuye coordenada Z

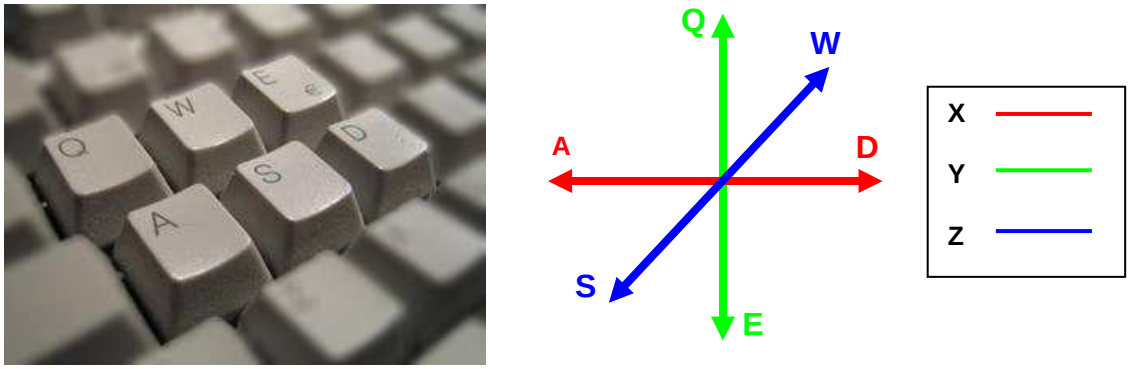


Figura 53: Teclas empleadas y sentido en el cambio de posición de la cámara

2.3.3.- Visual Studio 2005: MFC

El entorno de desarrollo elegido para implementar la interfaz de usuario y la estructura general del programa es Visual Studio 2005. La facilidad de uso, la incorporación de buenas herramientas de soporte para el lenguaje C++, la gran cantidad de documentación y comunidades de que dispone, así como la variedad de *plugins* desarrollados lo convierten en el entorno de programación ideal para el proyecto actual.

Además para el desarrollo de la interfaz se ha utilizado un conjunto de clases que facilitan de manera notable y evitan reticencias en el acceso al API de Windows: las clases MFC (Microsoft Foundation Classes).

2.3.3.1.- Desarrollo básico de la interfaz

Visual Studio 2005 permite la creación de un proyecto que ya incorpora las clases base de una interfaz que emplea MFC. Para ello basta con seleccionar como tipo de proyecto una aplicación MFC y establecer sus parámetros o características iniciales.

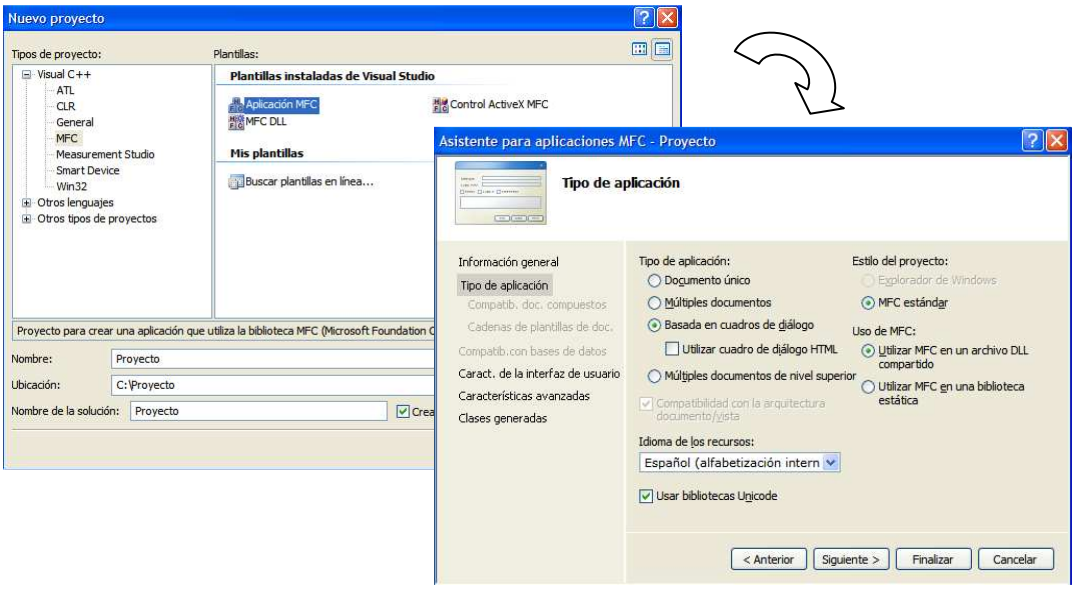


Figura 54: Aplicación MFC y propiedades iniciales

Tras realizar este paso, ya se dispone de la base de una interfaz con un diálogo principal, un diálogo secundario habitual del tipo "Acerca de…" y de algunos botones por defecto. Además se implementan de forma automática las clases y archivos de encabezamiento necesarios para tener una estructura sólida sobre la que comenzar a programar los elementos que componen la interfaz.

Un aspecto a comentar que proporcionan las clases MFC es la posibilidad de modificar el aspecto de la interfaz mediante un entorno de ventanas. En la vista de recursos de Visual Studio es posible seleccionar el diálogo que se desea modificar y mediante las opciones disponibles en el cuadro de herramientas, añadir algún elemento de forma muy intuitiva.

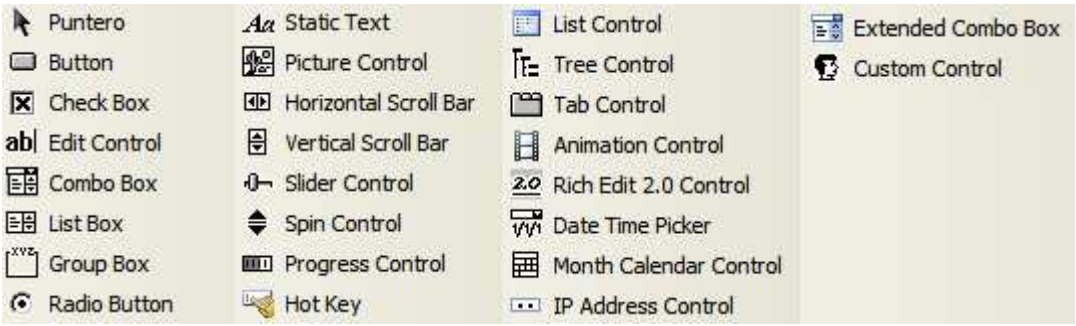
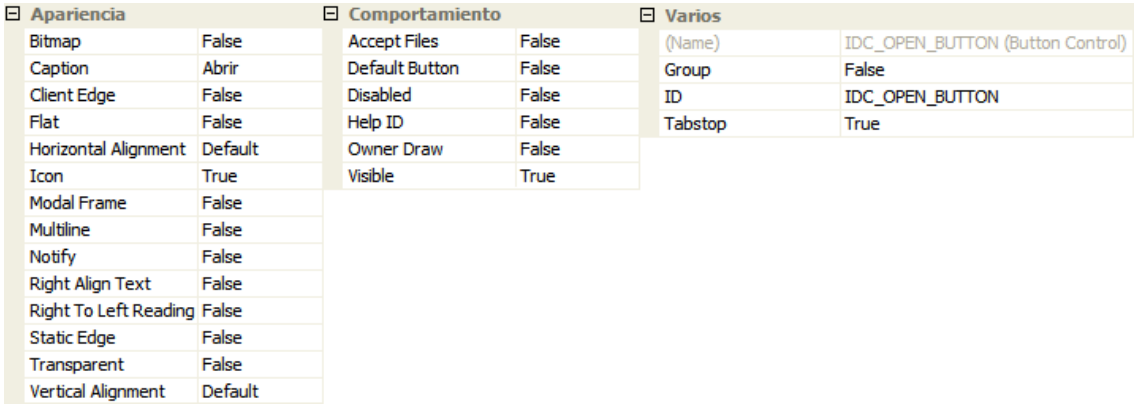


Figura 55: Elementos que proporcionan las MFC

Además tras añadir un elemento, en la ventana de propiedades asociada al mismo se puede consultar y modificar sus características iniciales que se encuentran agrupadas en tres módulos: apariencia, comportamiento y varios.



Apariencia	
Bitmap	False
Caption	Abrir
Client Edge	False
Flat	False
Horizontal Alignment	Default
Icon	True
Modal Frame	False
Multiline	False
Notify	False
Right Align Text	False
Right To Left Reading	False
Static Edge	False
Transparent	False
Vertical Alignment	Default

Comportamiento	
Accept Files	False
Default Button	False
Disabled	False
Help ID	False
Owner Draw	False
Visible	True

Varios	
(Name)	IDC_OPEN_BUTTON (Button Control)
Group	False
ID	IDC_OPEN_BUTTON
Tabstop	True

Figura 56: Características iniciales asociadas a un botón

La interfaz de la aplicación está compuesta por los siguientes elementos físicos:

- Botones (ya sean de pulsación o de marcado alternativo)
- Una lista de control
- Marcos de grupo
- Cuadros de texto
- Una barra de reproducción
- Controles mediante flechas
- Subventana de la gráfica
- Subventana de la animación

Esta enumeración tan general de los componentes que forman la aplicación se puede jerarquizar de la siguiente forma en función de su origen y funcionalidad:

Elementos propios de las MFC:

- Control de archivos cargados
  - Botones de pulsación
  - Lista de control

Marco de grupo de cámaras

- Elección de cámaras
  - Botones de marcado alternativo
- Variación de la posición de la cámara libre
  - Controles mediante flechas
  - Botón de pulsación

- Control del momento de animación
  - Barra de reproducción
  - Botones de pulsación
- Muestra de datos
  - Cuadros de texto

Elementos ajenos a las MFC:

Marco de grupo de animación

- Animación 3D
  - Ventana de Ogre

- Gráfica con los valores de los ángulos en función del tiempo
  - Elemento de Measurement Studio

En el apartado actual se va a realizar un análisis de los elementos propios de las clases MFC incluidos.

- Control de archivos cargados

  - Botones de pulsación: en la aplicación hay tres botones que se encargan de la gestión de los archivos cargados en el programa:
    - Botón salir: sale de la aplicación con una ventana de confirmación sólo si hay una animación cargada.
    - Botón abrir: su pulsación abre un diálogo de abrir archivo y, si el archivo es válido, se añade a la lista de control.
    - Botón borrar: elimina de la lista de control los archivos seleccionados, tras mostrar un mensaje de confirmación.
    - Botón cargar: carga el contenido del archivo y genera la animación. Para ello la ventana Ogre se inicializa de forma previa. No se permite cargar un archivo si el diálogo de apertura de pruebas está abierto.

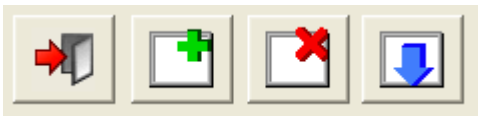


Figura 57: Botones de pulsación para control de archivos

  - Lista de control: contiene información de las pruebas que están en disposición de ser cargadas en el programa. La información a mostrar es la siguiente:
    - Prueba: nombre de la prueba realizando una correspondencia entre el nombre del archivo y la misma, como se indica en el apartado 2.2.3.1 de la memoria actual.
    - Ruta: ruta donde se encuentra el fichero a cargar.
    - Duración: tiempo en segundos que dura la prueba a cargar.
    - Fecha de modificación: fecha de modificación de los ficheros cargados. Este dato permite realizar un seguimiento del mismo paciente pudiendo comparar las fechas en las que se realizaron las pruebas.

    Cabe comentar que no se han descartado pruebas idénticas, es decir, no se ha omitido la carga de ficheros exactamente iguales siempre y cuando se traten de ficheros válidos.

Otro aspecto a comentar de la lista de control es que permite la carga de ficheros, además de mediante la pulsación del botón cargar, haciendo "dobleclick" sobre un elemento de la lista de control.

Prueba	Ruta	Duracion (s)	Fecha de Modificacion
Prueba de flexión lateral (1)	d:\Profiles\PABSILSO\Escritorio\modelos\Proyectocongrafica\Pacientes\angulos\v2_3.txt	29.96	22-9-2009 10:21
Prueba de flexión lateral (2)	d:\Profiles\PABSILSO\Escritorio\modelos\Proyectocongrafica\Pacientes\angulos\v2_4.txt	30.00	22-9-2009 10:23
Prueba de flexión-extensión (1)	d:\Profiles\PABSILSO\Escritorio\modelos\Proyectocongrafica\Pacientes\angulos\v2_1.txt	29.96	22-9-2009 10:18
Prueba de flexión-extensión (2)	d:\Profiles\PABSILSO\Escritorio\modelos\Proyectocongrafica\Pacientes\angulos\v2_2.txt	29.96	22-9-2009 10:21
Prueba de rotación (1)	d:\Profiles\PABSILSO\Escritorio\modelos\Proyectocongrafica\Pacientes\angulos\v2_5.txt	29.96	22-9-2009 10:23

Figura 58: Lista de control

- Elección de cámaras

  - Botones de marcado alternativo: estos botones o *Radio-button* se caracterizan porque sólo es posible mantener presionado uno, dejando sin presionar los demás. Esto permite cargar distintas posiciones de la cámara de visualización de la animación. Se han dispuesto 4 botones que representan cámaras fijas y uno que representa una cámara libre. La correspondencia entre las cámaras fijas y su orientación es la siguiente:
    - Cámara 1: vista frontal del modelo.
    - Cámara 2: vista trasera del modelo.
    - Cámara 3: vista de la parte derecha del modelo.
    - Cámara 4: vista de la parte izquierda del modelo.

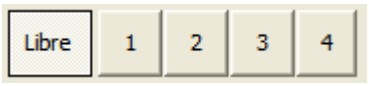


Figura 59: Botones de marcado alternativo

- Variación de la posición de la cámara libre

  - Controles mediante flechas: estos botones o *Spin Control* permiten variar la posición de la cámara de forma equivalente a la realizada con el teclado, comentada en el apartado 2.3.2.4. Se dispone de tres controles, uno por cada uno de los ejes (X, Y, Z). Estos controles sólo estarán activos si la cámara dispuesta es la cámara libre.

Figura 60: Controles mediante flechas

  - Botón de pulsación: este botón permite devolver a la cámara libre a su posición por defecto (vista central del modelo). Como es coherente, sólo estará disponible si está activa la cámara libre.

Figura 61: Botón de pulsación de cámara libre

- Control del momento de animación

  - Barra de reproducción: la barra de reproducción o *Slider Control* permite variar el momento de reproducción actual. Funciona de forma

equivalente a cualquier reproductor de vídeo: la variación de su punto de reproducción está sincronizado con el momento de la animación cargada. Además también está sincronizado con el cursor de reproducción de la gráfica y los cuadros de texto con información del tiempo y los ángulos. Cabe comentar también la posibilidad de interactuar con este control mediante la rueda del ratón: aplicar un giro a la rueda, modifica la posición del punto de reproducción de la barra.

Figura 62: Barra de reproducción

- Botones de pulsación: estos botones permiten interactuar con el momento de animación mediante los clásicos eventos de reproducción, rebobinado, pausado, reiniciado y finalización (*play*, *rewind*, *pause*, *reset* y *final*). Facilitan en gran medida obtener el momento de reproducción deseado.

Figura 63: Botones de reproducción

- Muestra de datos
  - Cuadros de texto: se utilizan para mostrar información relativa a la animación actual: tiempo de animación y valores de los ángulos en los tres ejes X, Y, Z.

Figura 64: Cuadros de texto

2.3.3.2.- Gestión de ficheros de pacientes

Apertura de pruebas

Los ficheros que representan las pruebas realizadas a los pacientes deben poseer una nomenclatura y estructura determinadas (descritas en el apartado 2.2.3.2), para así comprobar si se trata de archivos válidos y agregarlos a la lista de control. Se ha utilizado para ello un típico diálogo de abrir ficheros, mediante la utilización de la clase externa `COSFDialog` (clase comentada en el apartado 2.2.2.3), que permite seleccionar los archivos de texto que contienen las pruebas a cargar.

Una vez obtenidos los ficheros que se pretende cargar, se realizan las siguientes comprobaciones:

- Se comprueba que el nombre del fichero sea válido.
- Se realiza una lectura del fichero comprobando que los valores de tiempo siempre sean crecientes.

En caso de que no se obtenga un fichero válido, el fichero no se carga en la lista de control y se muestra el siguiente mensaje:

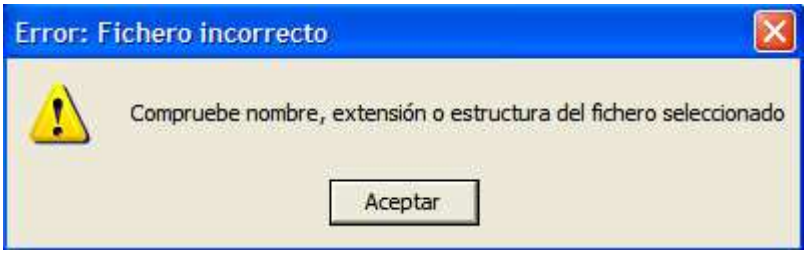


Figura 65: Mensaje de fichero de pruebas incorrecto

En caso contrario, se obtiene un mensaje con el directorio en el que se cargan los archivos y el número de archivos que se van a añadir a la lista de control.

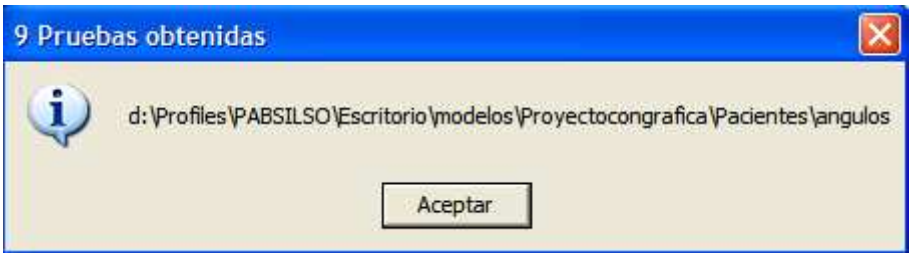


Figura 66: Mensaje obtenido al cargar pruebas correctas

Cabe tener en cuenta que si se realiza una selección múltiple, únicamente se muestra el número de archivos válidos que se añaden a la lista de control, junto al ya nombrado directorio de los mismos.

Borrado de pruebas

La aplicación permite seleccionar múltiples pruebas para eliminarlas todas a la vez. Para comprobar que no se trata de un error del usuario, siempre se muestra un mensaje con el número de pruebas a borrar pidiendo la confirmación del usuario.

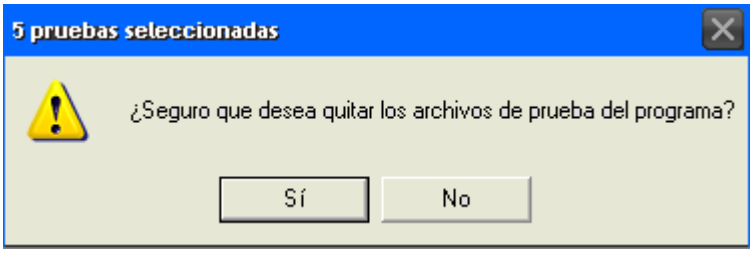


Figura 67: Mensaje obtenido al intentar borrar pruebas del list control

En el caso de que la prueba a borrar esté actualmente en carga, tanto la animación como la gráfica vuelven a un estado inactivo. Del mismo modo el resto de elementos de la interfaz asociados a la manipulación de la animación, también deshabilitan su estado.

2.3.3.3.- Cargar animación de Ogre

Hasta ahora, la animación de Ogre se ha implementado como una ventana independiente de la interfaz base de la aplicación. El objetivo principal de este apartado es enlazar dicha ventana para que se integre como un elemento más en dicha estructura de componentes.

Para ello se distinguen dos procesos: inicialización de la ventana de Ogre y carga de la animación con el modelo 3D.

Inicialización de la ventana de Ogre

Supone la carga de recursos iniciales para poder generar la ventana. Con tal fin, se realiza la siguiente secuencia de pasos:

- Cargar la configuración inicial obtenida tras instalar la librería de Ogre mediante la lectura del fichero *resources.cfg*.
- Definir las opciones de renderizado de que dispondrá la ventana. Se han contemplado las siguientes:
  - Utilizar `Direct3D9 Rendering Subsystem` como sistema de renderizado.
  - No reproducción a pantalla completa.
  - Resolución 800x600 con color de 32 bits.
  - Sin técnica de *antialiasing*.
  - Emplear la tarjeta gráfica mostrada en al archivo *resources.cfg*.
- Crear la ventana físicamente. Para ello es necesario el manejador de la ventana principal de la aplicación con el fin de establecer la ventana Ogre como ventana hija de la misma. Además se establece el tamaño y la posición respecto a su ventana padre, eligiendo un tamaño de 400x300 y situándola aproximadamente en la parte inferior izquierda.
- Inicializar recursos de grupo.
- Crear elementos básicos de la escena como el controlador de nodos, una cámara y un punto de vista.
- Inicializar eventos de entrada de teclado. Este proceso también precisa del manejador de la ventana principal con el fin de asociar dichos eventos de entrada de teclado a la misma.
- Carga de las entidades y nodos que componen el modelo. Aunque se carguen estos elementos, el modelo no es visible hasta que no se ejecute el bucle de renderizado de la animación. Esto permite que posteriormente sea menos costoso ejecutar una animación ya que el modelo se encuentra cargado de forma previa.

Este proceso de inicialización se realiza únicamente la primera vez que se carga el programa ya que será la base que posteriormente se emplea para soportar las distintas animaciones que se pueden cargar.

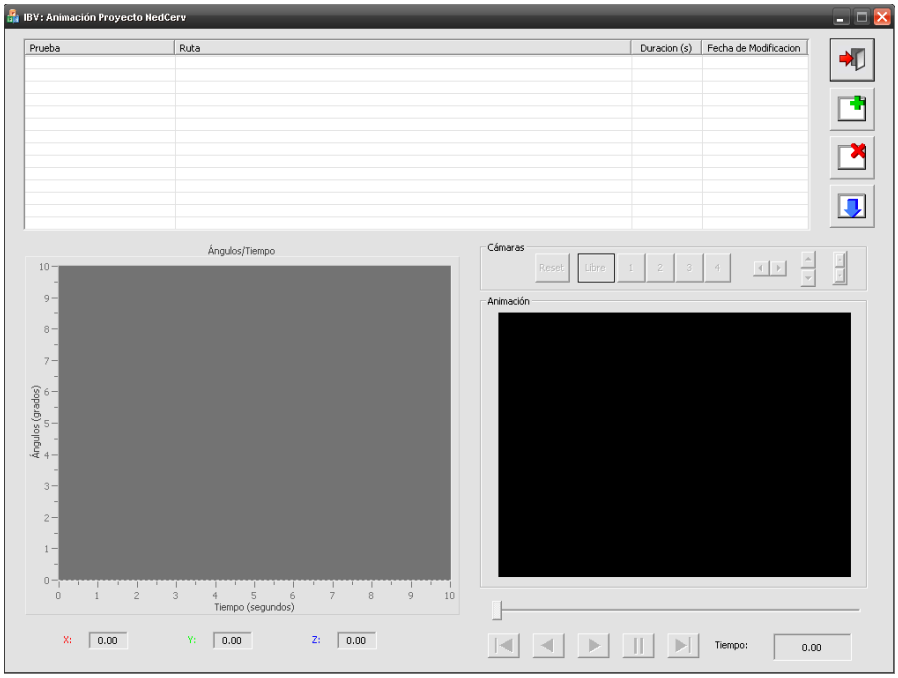


Figura 68: Aspecto de la interfaz tras la inicialización de la ventana Ogre

Carga de la animación con el modelo 3D

Para continuar con este proceso, se dispone de un archivo válido mostrado en la lista de control. Al realizar su carga para mostrar la animación, se producen de forma interna los siguientes procesos:

- Lectura del fichero seleccionado y guardado en la estructura correspondiente.
- Carga de la gráfica con los datos almacenados en la estructura.
- Se activan todos los elementos que pueden interactuar con la animación (botones, barra de reproducción, flechas…).
- Inicialización de valores de elementos de la interfaz (rango de la barra de reproducción, valores de los cuadros de texto…).
- Creación de la animación:
  - Cada animación posee un nombre distinto.
  - Cálculo de los *quaternions* y matrices de rotación que permitirán realizar el movimiento.
- Inicialización de eventos de entrada de la ventana.
- Actualización de los valores inicial y final de la animación de Ogre en función de los cursores límite de la gráfica.
- Activación de la cámara libre.
- Iniciación del bucle de renderizado que permite cargar el modelo y la animación.

Cabe comentar que si ya había una animación cargada de forma previa, se realiza de anteriormente al proceso comentado una liberación de recursos necesaria para no sobrecargar la aplicación. Esta liberación elimina todos los datos de la animación antigua, los posibles eventos que pudiera tener en cola y toda inicialización de eventos de entrada asociados a dicha animación.

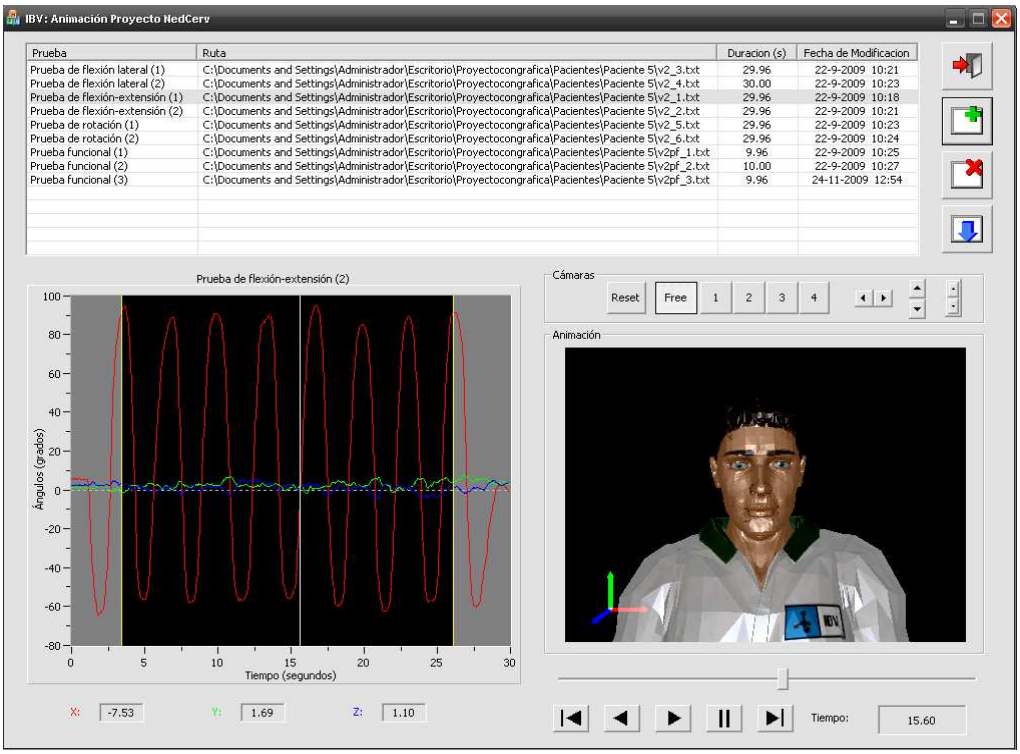


Figura 69: Aspecto de la interfaz tras la carga de la animación con el modelo 3D

Otro aspecto a tener en cuenta es que únicamente se debe seleccionar un archivo de la lista para cargarlo. En caso contrario, se avisa al usuario mediante un mensaje informativo.

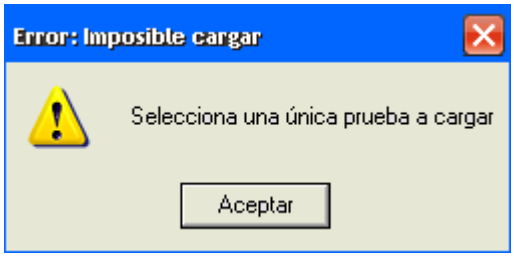


Figura 70: Mensaje obtenido al seleccionar más de un archivo a cargar

2.3.3.4.- Capturar eventos de la interfaz

La interfaz dispone de una gran cantidad de elementos con los que el usuario puede interactuar de una forma u otra. Esto conjunto de "interactuadores" es lo que se denomina eventos de la interfaz, los cuales es preciso tratar para controlar cualquier acción que el usuario pueda realizar sobre los mismos.

Las clases MFC incorporan varios métodos estándar asociados al comportamiento de cada elemento añadido a la interfaz. Los métodos varían en función del tipo de elemento que se añade y se adecúan a unos patrones básicos de funcionamiento. En consecuencia es lógico pensar que los eventos asociados a la pulsación de un botón no serán los mismos que los incorporados para tratar el comportamiento de un cuadro de texto.

BCN_HOTITEMCHANGE
BN_CLICKED
BN_DOUBLECLICKED
BN_KILLFOCUS
BN_SETFOCUS
NM_THEMECHANGED

NM_THEMECHANGED
STN_CLICKED
STN_DBLCLK
STN_DISABLE
STN_ENABLE

Figura 71: Métodos estándar de botones (izquierda) y cuadros de texto (derecha)

Además, mediante la selección de los mismos se proporciona una descripción de su funcionamiento, facilitando esto en gran medida la elección de métodos a utilizar para la aplicación:

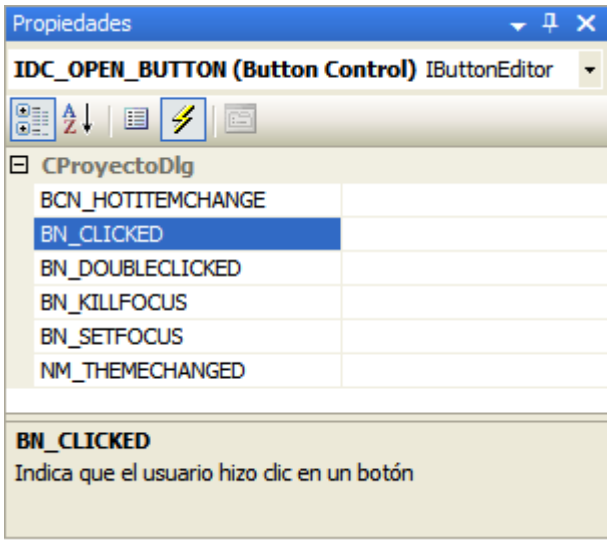


Figura 72: Descripción de funcionamiento de un método asociado a un botón

La clasificación de los eventos tratados atendiendo a su funcionalidad es la siguiente:

- Carga de pantalla: corresponden a eventos relacionados con el aspecto y el funcionamiento de la ventana principal. En el proyecto actual se ha incluido un evento que permita la carga de colores sobre algún elemento de la pantalla (en este caso los cuadros de texto con los valores X, Y, Z) y se ha tratado el cierre de la ventana principal para asegurarse de cerrar también la subventana de Ogre.

- Control de botones: este tipo de eventos son los más habituales. Se ha tratado la pulsación en todos los botones de la pantalla y se ha añadido la funcionalidad correspondiente a cada uno.

- Control de barra de reproducción: para la barra de reproducción se han tenido en cuenta las pulsaciones en la misma y las acciones de "pinchar y arrastrar" el momento de reproducción, actualizando consecuentemente toda información que se refiera a ese momento de reproducción.

- Temporizador: se ha añadido un temporizador al proyecto que tiene tres funciones principales:

    - Actualizar de forma coherente la barra de reproducción y el cursor de animación de la gráfica en función del momento de reproducción.
    - Cambiar el modo en el que se encuentra la gráfica para capturar eventos.
    - Capturar movimientos rápidos del usuario para entrar en modo de cambiar orientación de la cámara. Esto se debe a que en ocasiones la función `OnMouseMove` que se encarga de detectar el movimiento del ratón no se actualiza lo suficientemente rápido como para detectar la entrada a la ventana de Ogre.

- Control del ratón: teniendo en cuenta lo comentado en el temporizador, se ha añadido una funcionalidad para interpretar el pinchado y arrastrado en la ventana de Ogre a la variación de la orientación de la cámara de la animación y, en consecuencia, de los ejes superpuestos vinculados con la orientación de ésta. Esta funcionalidad se ha incorporado al método `OnMouseMove`, que es ejecutado cada vez que se desplaza el ratón. La equivalencia entre dicho movimiento y el cambio de orientación de la cámara se realiza de la siguiente forma:
    - Si el ratón se encuentra en la posición de la ventana de Ogre, se capturan eventos de ratón y se centra el foco en dicha ventana.
    - Si no es la primera vez que se llama al método, se calcula la diferencia de posición del ratón entre la última vez que se ejecutó y la actual.
    - En caso de pulsarse el botón izquierdo del ratón se envía dicha diferencia de posición a una función creada en Ogre para modificar la orientación de la cámara.

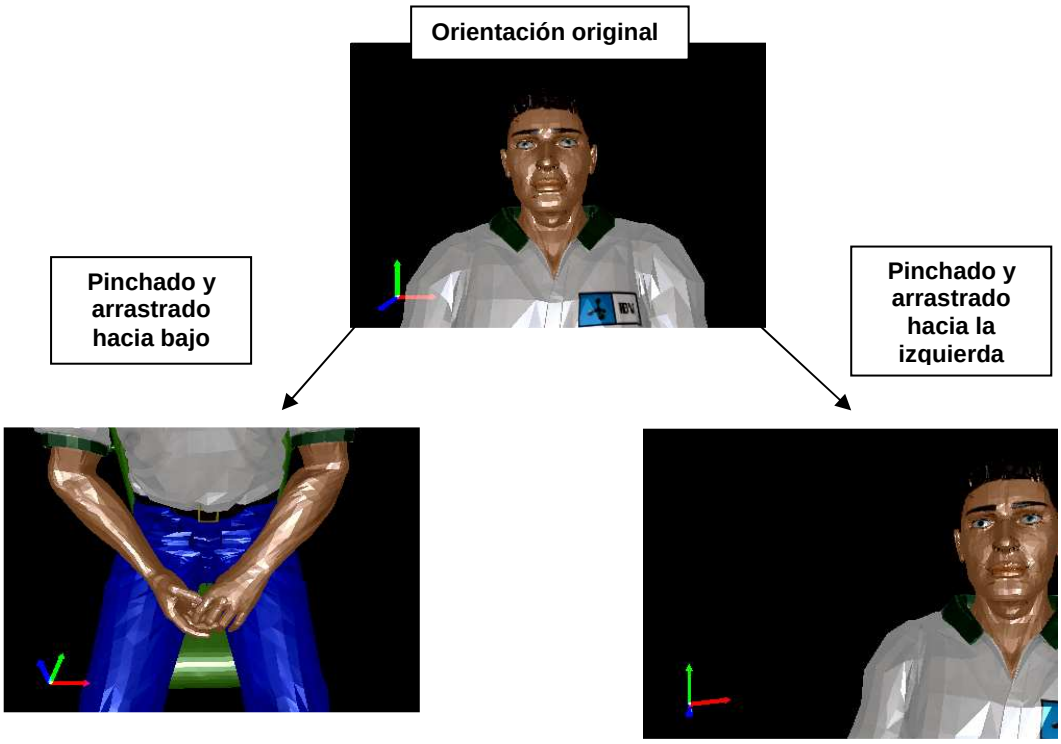


Figura 73: Ejemplo de funcionamiento de cambio de orientación de la cámara

Otro evento vinculado al control del ratón es el manejo de la rueda del mismo. Se ha incorporado una doble funcionalidad atendiendo a la posición en la que se encuentra el ratón en la ventana:

- Si se encuentra sobre la subventana de Ogre, se modifica la posición de profundidad de la cámara.
- En caso contrario, se modifica el momento de animación de la reproducción actual.

- Interacción con la gráfica: la gráfica incorporada permite interactuar con la misma variando tanto el momento de animación como el rango de reproducción de la misma.
  NOTA: Cualquier evento relacionado con la gráfica se explica de forma más detallada en el punto 2.3.3.5.

Esta clasificación atiende a todos los eventos que se capturan en la interfaz; sin embargo, para los eventos que interaccionan con la animación es preciso realizar un enlace entre los módulos de la interfaz y los de Ogre (estructurados en el punto 2.2.1 de la memoria actual).

Este proceso se realiza a través de vincular los métodos adheridos a la interfaz descritos previamente con un objeto de la clase `COgreApplication`, que a su vez incorpora un objeto de la clase `MyFrameListener` en el que se incluyen los métodos para controlar dichos eventos (en la Figura 17 asociada al apartado 2.2.2 se puede comprobar de forma gráfica este funcionamiento).

NOTA: en el apartado 2.2.2.1 se realiza una descripción general de los los métodos asociados a las clases `MyFrameListener` y `COgreApplication`.

2.3.3.5.- Measurement Studio

La animación en 3D del modelo proporciona una visión general y relativamente subjetiva de las pruebas obtenidas a los pacientes; es necesaria una representación de los datos que pueda ser estudiada de forma más objetiva. Este es el motivo principal por el que se ha incorporado una gráfica a la aplicación.

Para ello se ha elegido el conjunto de clases incorporadas en el paquete Mesurement Studio debido a su fácil integración con el entorno de programación en el que se ha realizado el proyecto y las posibilidades que dichas clases proporcionan.

Tras realizar la instalación del paquete, se incorporan de forma automática los elementos referentes a Measurement Studio al cuadro de herramientas.

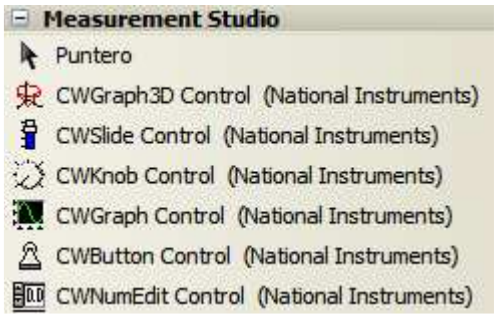


Figura 74: Elementos añadidos por Measurement Studio

A tener en cuenta es que los elementos de Measurement Studio no soportan la codificación *Unicode* con la que se crean los proyectos en Visual Studio por defecto. Cuando se agrega las clases para poder incorporar un elemento de este tipo se muestra un cuadro de diálogo para permitir la modificación de la codificación del proyecto.

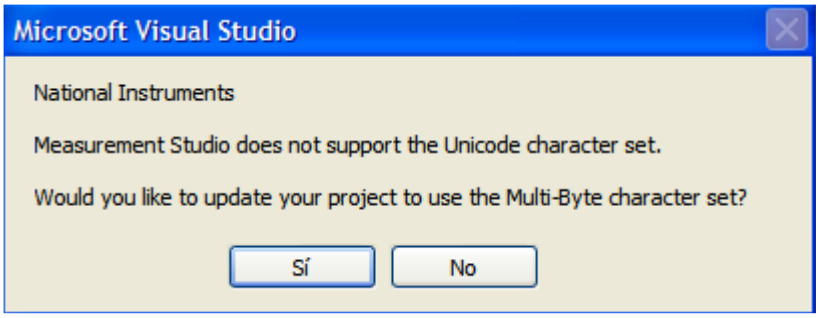


Figura 75: Cuadro de diálogo para cambio de codificación del proyecto

El cambio principal que supone este cambio de codificación en el proyecto actual es la variación para realizar la conversión entre algunos de cadenas y el ligero cambio de aspecto que muestran algunos elementos de la interfaz.

Representación de valores

Realizada la inicialización comentada del paquete Measurement Studio, ya es posible agregar el elemento empleado para realizar la gráfica que represente el cambio de valor de los ángulos en los tres ejes en función del tiempo: *CWGraph Control*.

Los elementos de tipo *CWGraph Control* poseen una amplia gama de propiedades y eventos que permiten bastante interactividad. Es posible controlar sus propiedades iniciales, al estilo de cómo se comenta en el apartado 2.3.3.1, y sus eventos, de la misma forma que en el punto 2.3.3.4, pero incorpora una novedad respecto del resto de elementos de la interfaz: una ventana de propiedades independiente y personalizada que nos muestra toda la información de configuración de la gráfica.

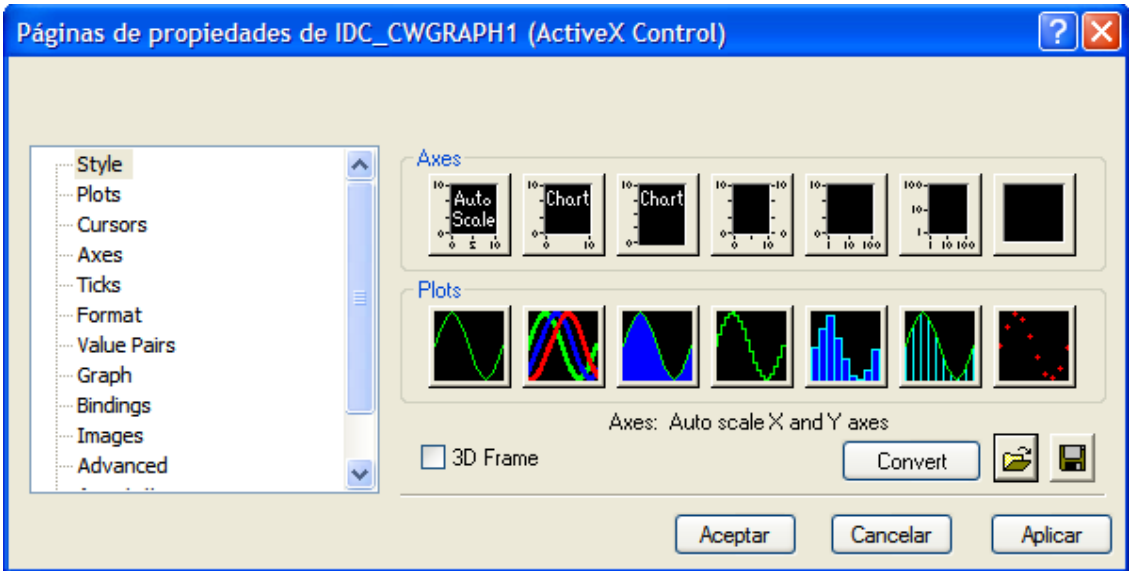


Figura 76: Ventana de propiedades de un elemento de tipo CWGraph Control

A través de esta ventana es posible cambiar el estilo de la gráfica, los ejes, los cursores, añadir anotaciones…entre otras muchas características.

En el proyecto actual se ha considerado agregar los siguientes componentes a la gráfica:

- 3 trazos que representen al valor de los ángulos X, Y, Z.
- Dos cursores entre los cuales se indica el rango de reproducción de la animación.
- Un cursor acorde al momento de reproducción de la animación.
- Una región delimitada entre ambos cursores indicando la zona de reproducción activa.
- Leyendas reflejando los datos mostrados en ambos ejes, así como las unidades en las que se representan.

Tras establecer esta configuración inicial en la gráfica, se debe programar dichos componentes para que representen tanto los datos obtenidos de las pruebas como para que se coordinen con el momento de animación y el resto de elementos agregados a la interfaz.

El proceso para representar los datos de los ángulos en los tres trazos tiene como base los datos guardados en la estructura de marcadores. Se accede a estos datos y se separan en 4 vectores independientes: un vector para tiempos y los otros tres para los valores de los ángulos en X, Y, Z. Posteriormente y con el fin de que el elemento *CWGraph Control* pueda interpretar los mismos, se realiza una conversión entre estos vectores de tipo `double` y una clase propia del elemento de Measurement Studio: `CNiReal64Vector`.

El último paso para representar los trazos es considerar como eje de abscisas el vector de tiempos y como eje de ordenadas los tres vectores con los valores de los ángulos.

Con esta descripción ya se dispone de la base que forma la gráfica; sin embargo hay que considerar la programación de otros componentes como los cursores y la región activa.

Programación de cursores

En la gráfica se distingue dos tipos de cursores: cursores para limitar el rango de reproducción y un cursor que representa el momento de animación (de colores amarillo y blanco respectivamente).

La representación de los cursores que limitan el rango de reproducción se puede considerar trivial: el cursor izquierdo se sitúa en el instante inicial de reproducción (por defecto 0.0 segundos) mientras que el cursor derecho se coloca en el último dato de tiempo que presenta la prueba almacenada (varía en función de la duración de la prueba). Un aspecto a considerar que facilita la representación de estos cursores es que la gráfica es de tipo autoescalable, es decir, que ante unos datos de entrada ajusta de forma automática los ejes de representación para mostrar la información de la forma más clara posible.

Por otra parte, el cursor que representa el momento de animación en la gráfica requiere un seguimiento mayor de la ejecución del programa: debe actualizar su posición ante cualquier cambio de reproducción del momento de animación actual. Por tanto cualquier evento lanzado por la barra de reproducción tiene su repercusión en este cursor de animación, al que se obliga a actualizar su posición.

Programación de la región activa

La región de reproducción activa se señala en la gráfica mediante un fondo negro, mientras que la zona de reproducción inactiva está señalada con un fondo gris.

Este efecto se programa en la gráfica a través de anotaciones. Las anotaciones permiten añadir a la gráfica multitud de elementos tales como texto, dibujos, fondos… y tienen una gran cantidad de características modificables. Por tanto se trata de una anotación en la que se establece el tamaño de representación a través de dos valores en X y dos valores en Y. De esta forma es trivial idealizar de forma teórica su representación: los valores del eje Y corresponden al tamaño de representación de la gráfica, mientras que los valores en el eje X corresponden a los cursores que limitan el rango de reproducción.

Capturar eventos de la gráfica

Uno de los motivos por los que se han elegido las clases incorporadas por el paquete de Measurement Studio para la representación de la gráfica es la interactividad que permiten sus elementos. El elemento *CWGraph Control* incorpora una gran cantidad de eventos que es posible capturar y manejar para obtener un comportamiento deseado ante una acción del usuario.

AnnotationChange	KeyUp
AnnotationMouseDown	MouseDown
AnnotationMouseMove	MouseMove
AnnotationMouseUp	MouseUp
Click	Pan
CursorChange	PlotAreaBoundsChange
CursorMouseDown	PlotAreaMouseDown
CursorMouseMove	PlotAreaMouseMove
CursorMouseUp	PlotAreaMouseUp
CWBindingDataUpdated	PlotMouseDown
CWBindingStatusUpdated	PlotMouseMove
DblClick	PlotMouseUp
KeyDown	ReadyStateChange
KeyPress	Zoom

Figura 77: Eventos del elemento CWGraph Control

Las posibles interacciones con CWGraph Control que incorpora la aplicación se resumen en:

- Pinchar y arrastrar los cursores que delimitan el rango de reproducción de la gráfica.
- Pinchar y arrastrar el cursor de animación.
- Pinchar en alguna zona de la gráfica ajena a los cursores.

La principal dificultad para implementar las tres funcionalidades de manera simultánea es el modo en el que se configura el elemento *CWGraph Control*: dispone de una opción que permite seleccionar el tipo de eventos a los que es sensible.

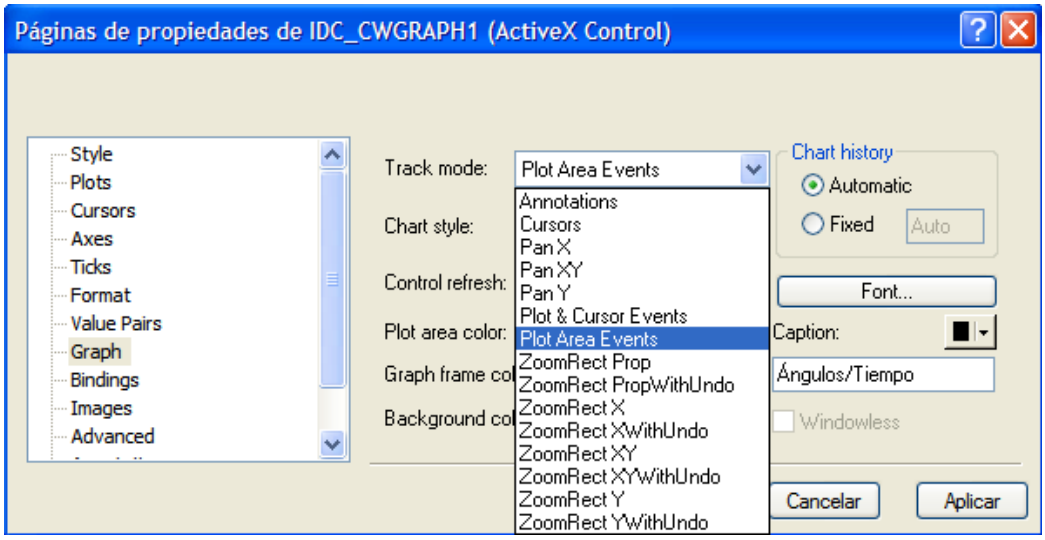


Figura 78: Modos de selección de eventos disponibles en CWGraph Control

Se distinguen cuatro modos fundamentales de selección de eventos:

- *Cursors*: únicamente es sensible a eventos de pinchado y arrastrado de cursores.
- *Pan & Zoom Events*: eventos relacionados con el cambio de profundidad y tamaño de las gráficas.
- *Plot Area Events*: sensible a eventos de ineracción del ratón en la gráfica (no considera modificación de cursores).

- *Plot & Cursor Events*: considera los eventos controlados por Plot Area Events y los eventos de pulsación sobre cursores (no incluye eventos para controlar la pulsación y arrastre de cursores).

Por tanto, el modo de actuar para conseguir una respuesta satisfactoria ante las tres funcionalidades de manera simultánea es variar entre los modos *Cursors* y *Plot Area Events* en función de la situación del ratón en la gráfica. Cuando el ratón se sitúa sobre un cursor el modo de selección de eventos es *Cursors*, mientras que en cualquier otra posición el modo de selección es *Plot Area Events*.

Cabe comentar que con el fin de facilitar la manipulación de los cursores, se ha considerado un rango de acción vinculado a la posición de los mismos con el fin de no tener que pulsar exactamente en el centro del cursor deseado para seleccionarlo.

Teniendo en cuenta estos aspectos, se va a realizar un análisis de cada una de las funcionalidades implementadas distinguiendo la relación con el comportamiento de otros elementos de la interaz, así como comportamientos no deseados considerados en los mismos.

Pinchar y arrastrar sobre cursores que delimitan el rango de reproducción

Cambiar el rango de reproducción de la animación afecta a los momentos inicial y final de reproducción de la animación. Estos cambios tienen una repercusión en los siguientes elementos:

- En la animación en sí ya que mediante los botones de reproducción únicamente se puede acceder a este rango delimitado.
- La barra de reproducción debe modificar su dispoción para actualizar su tamaño total al nuevo rango definido por estos cursores de la gráfica.
- La región activa modifica su tamaño en función de la posición que adopten ambos cursores.
- En caso de que alguno de los cursores supere al cursor de animación dejándolo en una posición de región inactiva, el cursor de animación varía su posición de forma equivalente al cursor que se está modificando.

Un comportamiento no deseado al modificar este tipo de cursores es que se intenten intercambiar de posición, es decir, que el cursor izquierdo intente colocarse en una posición temporal mayor que el cursor derecho o viceversa. Este comportamiento muestra un mensaje de error advirtiendo al usuario y coloca la disposición de los cursores por defecto: el cursor izquierdo en el momento inicial de animación, el cursor derecho en el final y el cursor de animación al comienzo de la misma.

Figura 79: Error mostrado al intercambiar cursores

Pinchar y arrastrar el cursor de animación

Esta acción cambia el momento de reproducción de la animación y, en consecuencia, afecta a los siguientes elementos de la interfaz:

- A la animación en sí puesto que varía la posición temporal de la animación, actualizándose consecuentemente los movimientos del modelo.
- La barra de reproducción ya que está correlacionada con este cursor: un cambio en uno de estos dos elementos y el otro refleja la misma modificación.
- La información mostrada por los cuadros de texto con los datos del tiempo y los valores de los ángulos.

Un comportamiento no deseado tratado es la modificación de la posición del cursor de animación a una posición perteneciente a la región inactiva. En caso de superar este rango de reproducción, el cursor de animación permanece fijo en la misma posición en la que se encuentra el cursor límite superado.

Pinchar en alguna zona de la gráfica ajena a los cursores

Este evento desplaza el momento de animación representado a la zona de tiempo de la gráfica en la que se haya pulsado el botón izquierdo del ratón. Esta acción esta relacionada con los siguientes elementos de la interfaz:

- La animación se actualiza dependiendo del momento de animación en el que se pulse sobre la gráfica.
- El cursor de animación se sitúa en el tiempo correspondiente a la posición en la que se haya pulsado sobre la gráfica.
- La barra de reproducción se actualiza de forma equivalente al cursor de animación.
- Los valores de los cuadros de texto se actualizan también en función del momento de animación mostrado.

Un comportamiento no deseado se produce al pulsar sobre una zona correspondiente a una región inactiva. La respuesta de la aplicación es situar el cursor de animación en la misma posición del cursor límite más cercano, evitando esta situación anómala.

3.- Resultados y discusión

El software generado posee todos los fundamentos para tener gran aplicación y difusión en los ámbitos de valoración y diagnóstico de cervicalgias. Combinado con el proyecto NedCervical, que le otorga toda la base estadística de diagnóstico y seguimiento de pacientes, facilita en gran medida la labor del médico encargado de tratar este tipo de dolencias.

La facilidad de uso, el *feedback* o respuesta continua ante interacciones del usuario, la solidez ante posibles datos de entrada incorrectos y lo intuitivo que resulta utilizar el programa son algunas de las normas en las que se ha basado su desarrollo. Cualquier persona cualificada sin una gran experiencia en la utilización de componentes informáticos puede otorgar un sentido útil y práctico al manejo del programa.

Son estos aspectos los considerados fundamentales para el éxito de este programa: la usabilidad del mismo y la posibilidad de obtener de una forma ágil un diagnóstico etiológico correcto. Estas dos características adquieren una mayor importancia ante el siguiente dato: se estima que más de la mitad de la población padece algún tipo de cervicalgia en algún momento de su vida. Por tanto el número de pacientes a consultar está sobredimensionado en comparación a los recursos que presenta la sanidad pública actual; es necesario el impulso y desarrollo de herramientas que permitan agilizar todo este proceso.

Existen otros datos quizá más impactantes que justifican el empleo de herramientas como la presentada en esta memoria: según un estudio realizado por la Universidad Politécnica de Valencia, que obtuvo sus datos de dos mutuas de accidentes de trabajo de la ciudad de Murcia, la mitad de los pacientes con diagnóstico y baja laboral por cervicalgia tienen una estructura de personalidad desadaptativa y con un claro perfil neurótico. Este tipo de personalidad está vinculado a personas que presentan una inestabilidad emocional, escasos recursos para afrontar las situaciones de estrés y dificultades para adaptarse e implicarse activamente en el trabajo. Ciñéndose a las cifras, el 61.9% de los sujetos participantes en el estudio muestran una personalidad inapropiada para tener éxito laboral. Además el 49.1% posee un malestar psicológico que se focaliza en la presencia de síntomas somáticos.

Este tipo de datos obliga a resolver una cuestión: la necesidad de establecer en la medida de lo posible si la etiología presentada por el paciente tiene su origen en una cervicalgia o se debe a otro tipo de dolencias, ya sean físicas o psicológicas. Cabe comentar que el programa NedCervical está preparado para hacer frente a algunos casos problemáticos: es capaz de detectar casos de no colaboración del paciente con un acierto del 95%.

Todo este compendio de datos conlleva una consecuencia lógica: el programa presentado, debido a su estructura y siempre acompañado del proyecto NedCervical desarrollado por el Instituto de Biomecánica de Valencia, tiene posibilidades en el mercado de la valoración médica actual.

3.1.- Análisis de resultados obtenidos

A continuación se va a proceder a realizar un análisis para detectar movimientos anómalos entre algunos de los pacientes a los que se les han realizado las pruebas. Se va a realizar una comparación con los resultados obtenidos en pacientes sanos con el fin de observar de una forma clara las diferencias entre ellos.

Paciente sano

Las pruebas registradas a este paciente son las que se tomarán como referencia, ya que los parámetros de todos los movimientos se encuentran dentro de los rangos de normalidad establecidos.

Las gráficas a mostrar corresponden a las de las 3 pruebas de movimientos con variación principal en cada uno de los tres ejes (flexo-extensión, eje X; rotación, eje Y; y flexión lateral, eje Z) y las de las 3 pruebas funcionales obtenidas con la aplicación.

Flexo-extensión: la flexo-extensión mostrada varía entre unos valores de los ángulos que se pueden catalogar como normales. Se caracteriza por no presentar apenas variación respecto de los valores de los ángulos Y y Z, mientras que la variación en el valor del ángulo respecto del eje X tiene una gran amplitud. Por esta razón se considera que el paciente no presenta limitaciones funcionales en este movimiento.

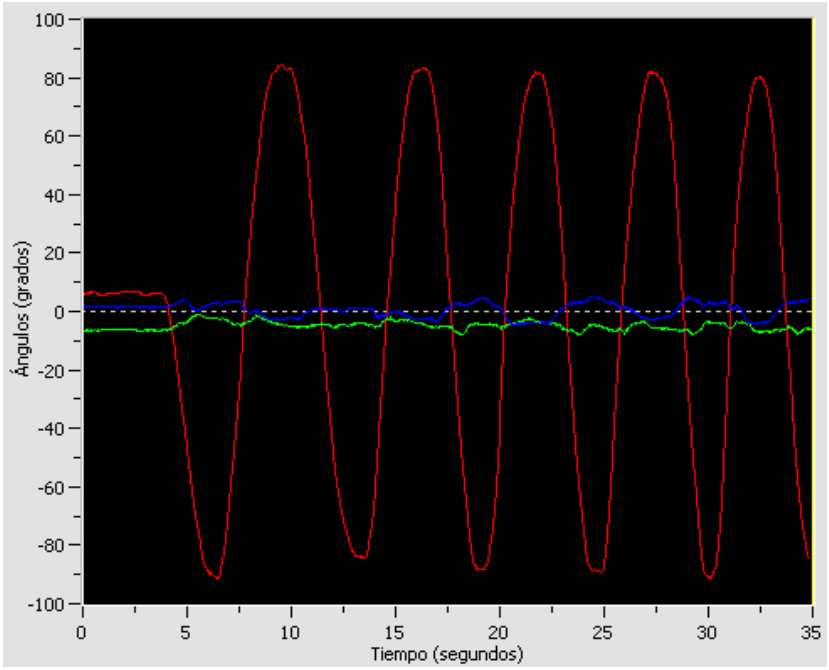


Figura 80: Prueba de flexo-extensión en un paciente sano

Rotación: la prueba de rotación mostrada presenta un comportamiento funcional normal. Se caracteriza por presentar una gran variación en el valor del ángulo referido al eje Y, además de por unas pequeñas ondulaciones continuas en los valores de los ángulos referidos a los ejes X y Z.

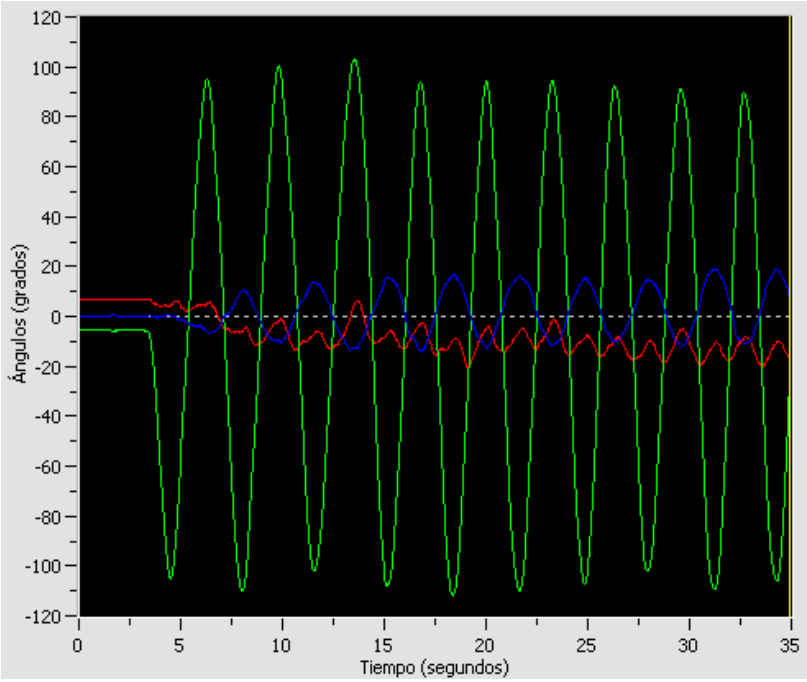


Figura 81: Prueba de rotación en un paciente sano

Flexión lateral: la flexión lateral o lateralización mostrada se produce dentro de los rangos catalogados como normales. Además se acompaña de un sistemático ángulo menor de rotación típico al pretender realizar el gesto de la forma más amplia posible, hecho que proporciona detalles de que no posee ninguna afección visible en este movimiento.

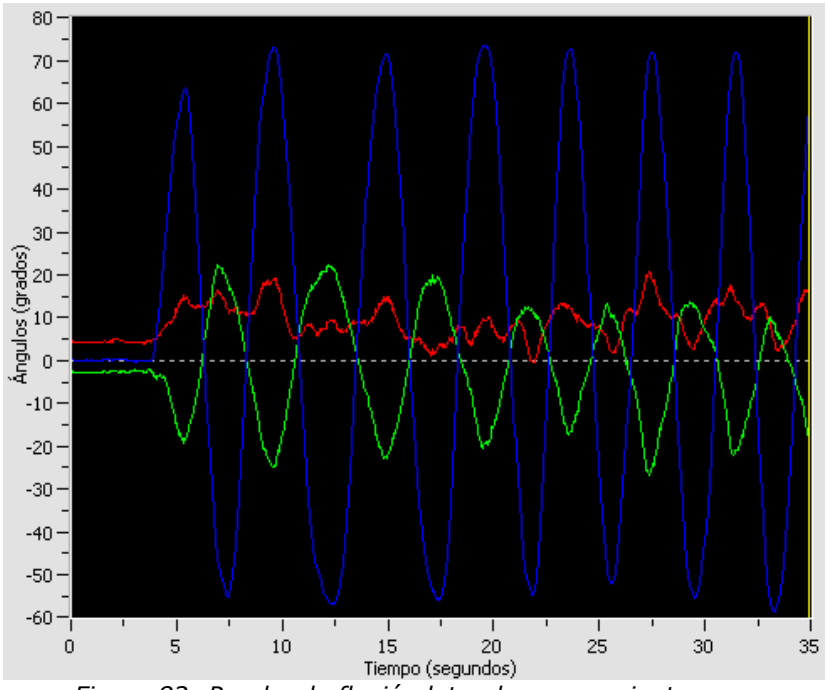


Figura 82: Prueba de flexión lateral en un paciente sano

Prueba funcional 1: la gráfica mostrada para esta prueba funcional no presenta ningún índice que permita identificar una afección. Esta prueba se caracteriza por una variación hacia valores positivos de los ángulos en los tres ejes en un primer momento, seguida de una variación fundamental en el eje X y de un retorno a la posición original.

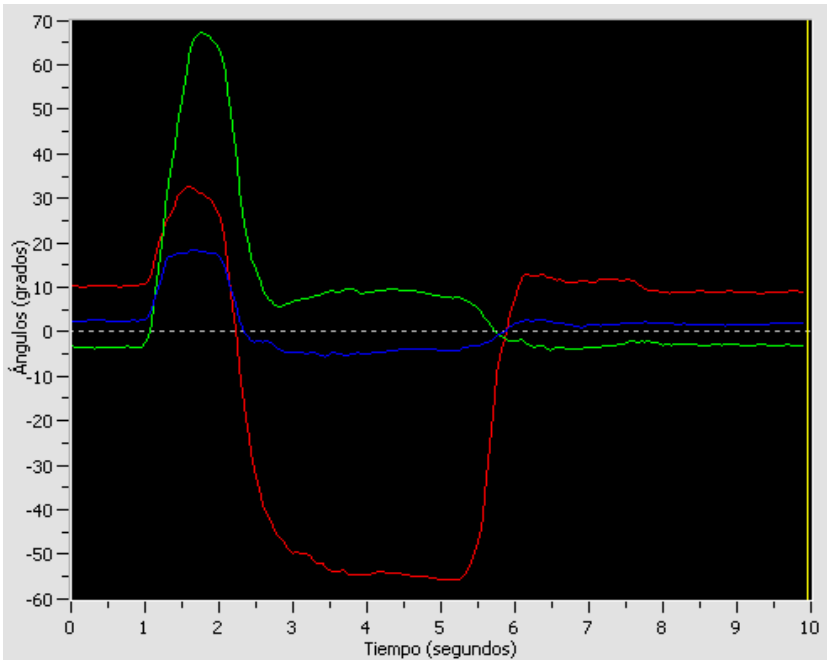


Figura 83: Prueba funcional 1 realizada a un paciente sano

Prueba funcional 2: la gráfica dispuesta para este tipo de prueba funcional posee un comportamiento funcional normal. Esta prueba se caracteriza por presentar una variación positiva del valor del ángulo referido al eje X, seguida de una variación negativa y mantenida del mismo y de su retorno a la posición original. Los valores de los ángulos de los otros ejes permanecen constantes durante toda la prueba.

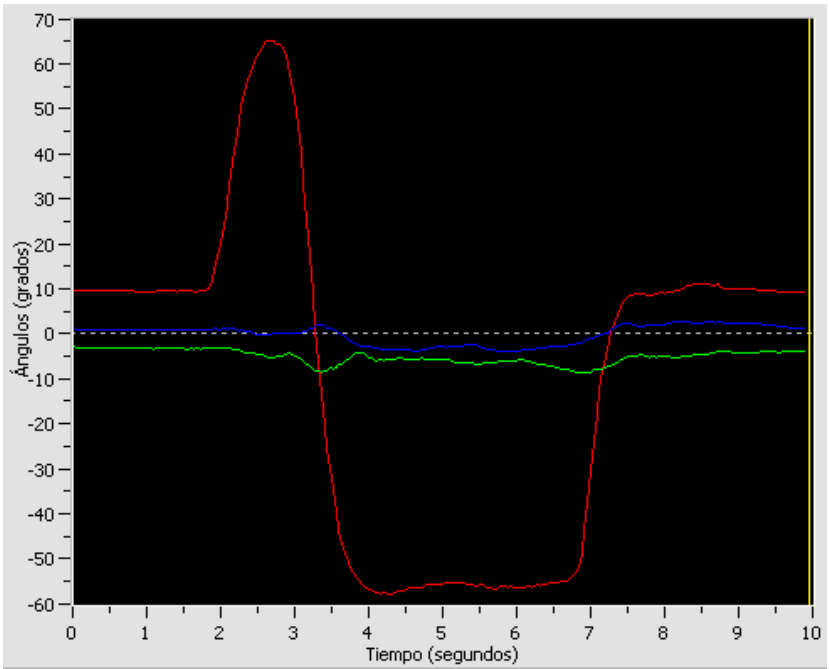


Figura 84: Prueba funcional 2 realizada a un paciente sano

Prueba fucional 3: la gráfica mostrada a continuación que refleja la información obtenida en este tipo de prueba funcional presenta unos parámetros calificados como normales. Sus características básicas son, en un primer momento, unos valores de los ángulos elevados y con signo negativo para el eje Y, al igual que para el eje Z aunque con un valor mucho menor, acompañado de una ligera extensión. A continuación se produce un movimiento de flexión pronunciado acompañado de una rotación hacia valores positivos, seguido de un retorno a la posición origen.

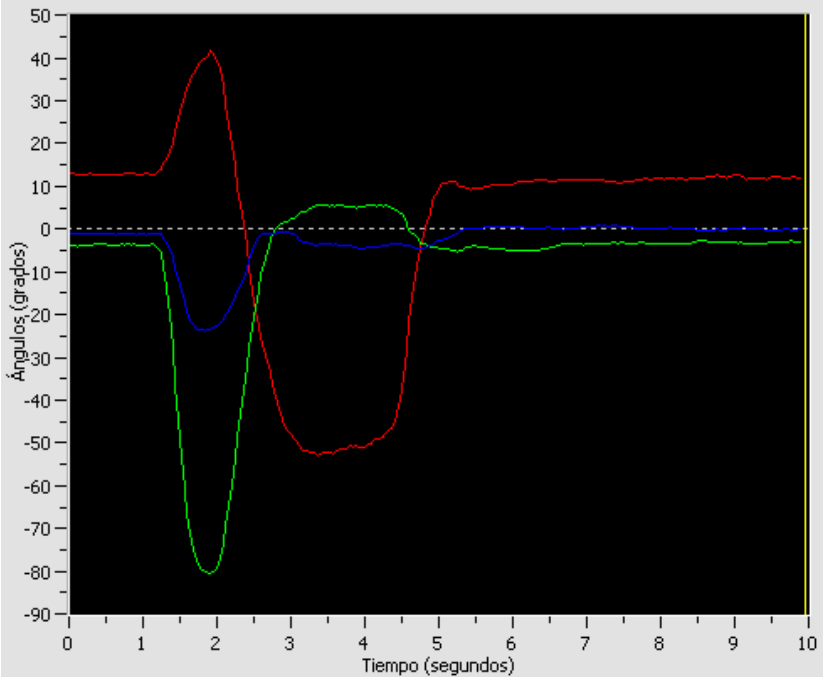


Figura 85: Prueba funcional 3 realizada a un paciente sano

Si se desea comprobar las animaciones para cada uno de estos movimientos, se pueden localizar en los archivos incluidos en la carpeta llamada "Paciente 1":

- Flexo-extensión: `angulos_2.txt`
- Rotación: `angulos_6.txt`
- Flexión lateral: `angulos_3.txt`
- Prueba funcional 1: `angulos_9.txt`
- Prueba funcional 2: `angulos_8.txt`
- Prueba funcional 3: `angulos_7.txt`

A continuación se va a realizar una comparación entre los datos mostrados y los obtenidos en otros pacientes que presentan algún tipo de anomalía en las pruebas.

NOTA: Al visualizar las gráficas mostradas hay que recordar que disponen de un sistema autoescalable para apreciar mejor pequeños cambios en los valores de los ángulos.

Pacientes con movimientos anómalos

- *Paciente 3*: este paciente no puede realizar de forma correcta ninguna de las pruebas mostradas. Los movimientos mostrados están muy limitados y, comparándolo con las gráficas del paciente sano, se puede comprobar que existe un déficit completo de movilidad cervical.

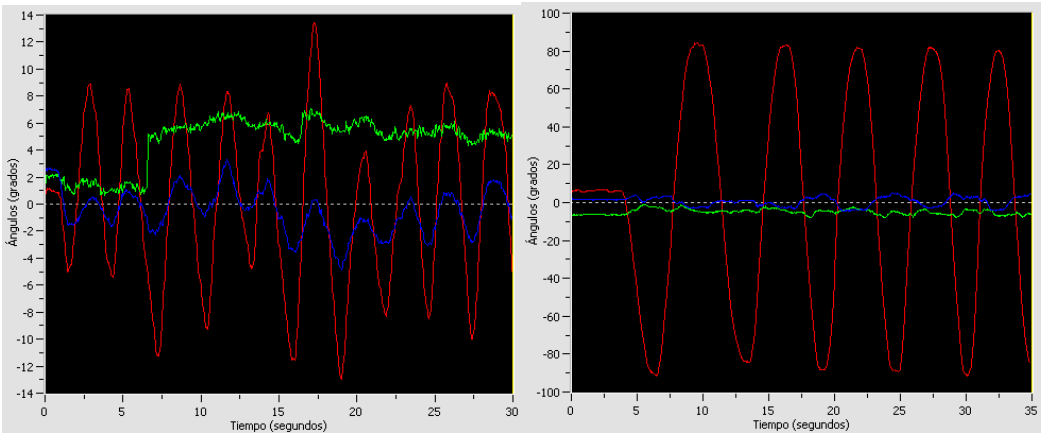


Figura 86: Comparación de flexo-extensión entre paciente 3 (izqda.) y sano (dcha.)

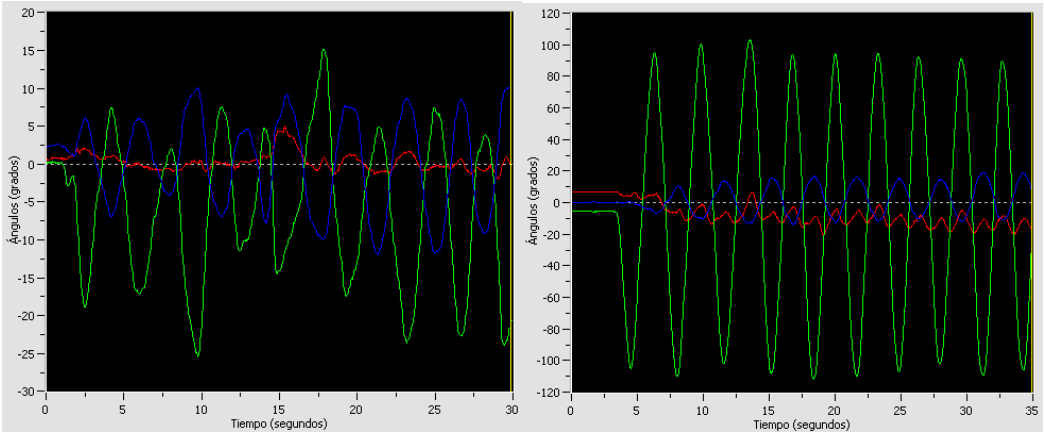


Figura 87: Comparación de rotación entre paciente 3 (izqda.) y sano (dcha.)

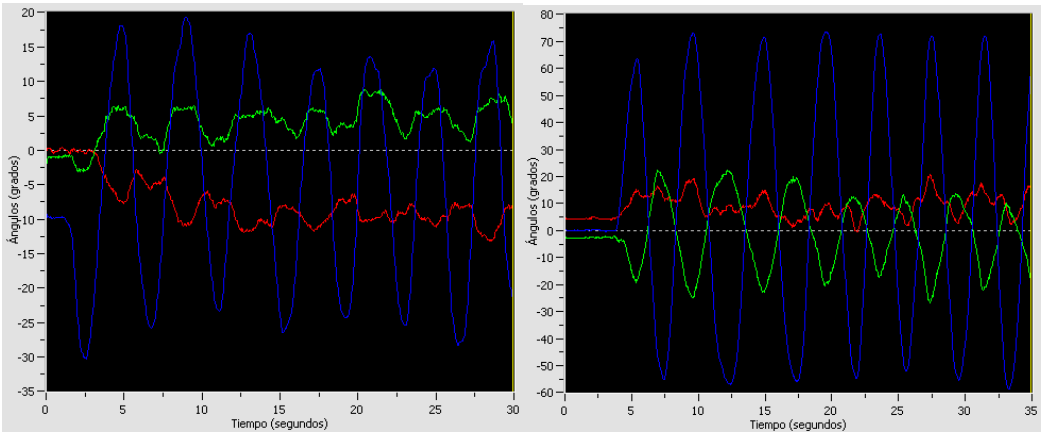


Figura 88: Comparación de flexión lateral entre paciente 3 (izqda.) y sano (dcha.)

- *Paciente 4*: este paciente no realiza un movimiento de laterización válido. Esto se puede deber a que en el momento de la grabación de la prueba no entendió el gesto que debía realizar para hacer la flexión lateral, ya que el resto de pruebas presentan un comportamiento perfectamente correcto.

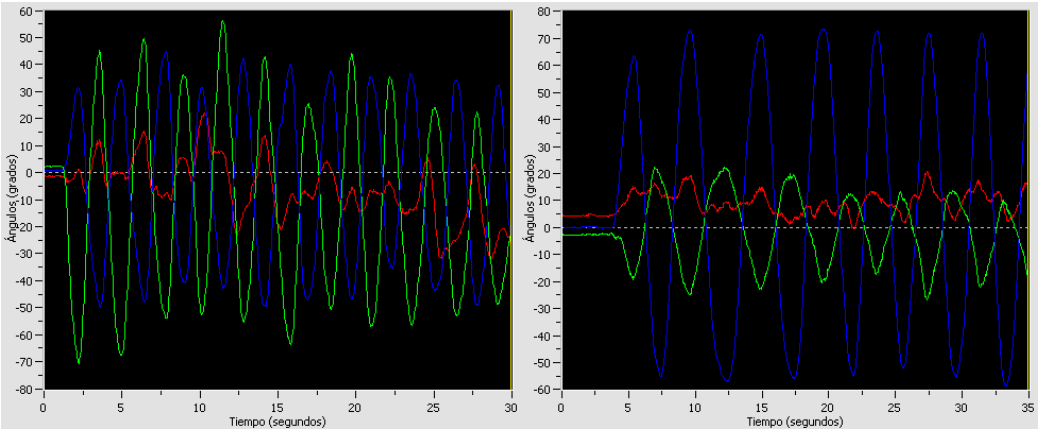


Figura 89: Comparación de flexión lateral entre paciente 4 (izqda.) y sano (dcha.)

- *Paciente 6*: este paciente tiene una gran flexibilidad: todos los movimientos son de una gran amplitud y los realiza de manera muy rápida, fijando muy bien cada uno de los tiempos. Además el hecho de que en las pruebas de rotación realice un rebote final es compatible con que disponga de una potente musculatura cervical.

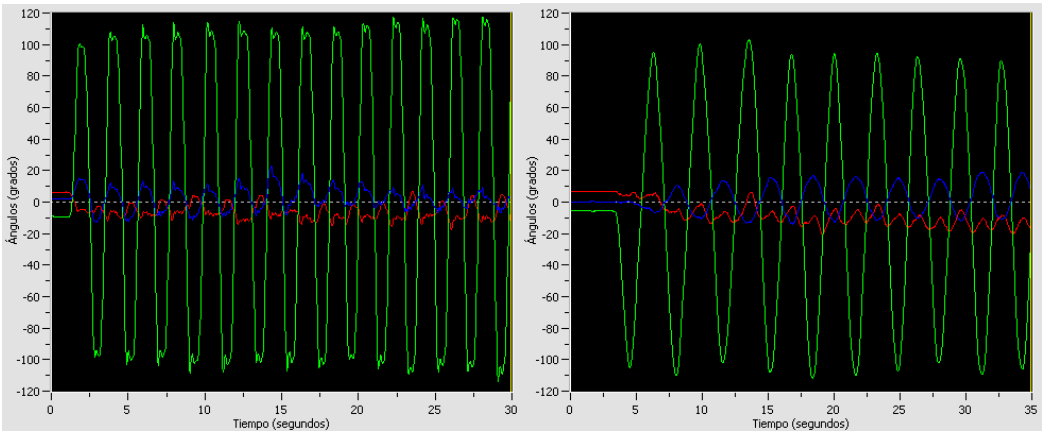


Figura 90: Comparación de rotación entre paciente 6 (izqda.) y sano (dcha.)

- *Paciente 7*: viendo las gráficas y la animación de este paciente se identifican dos problemas de distinta índole: el primero, es que tiene limitada la lateralización hacia su parte izquierda; y el segundo, es que posee una laxitud excesiva.

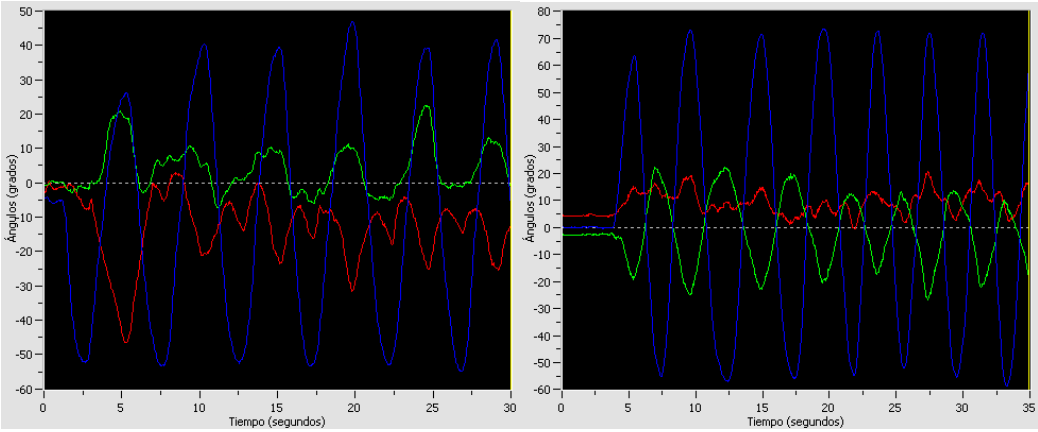


Figura 91: Comparación de flexión lateral entre paciente 7 (izqda.) y sano (dcha.)

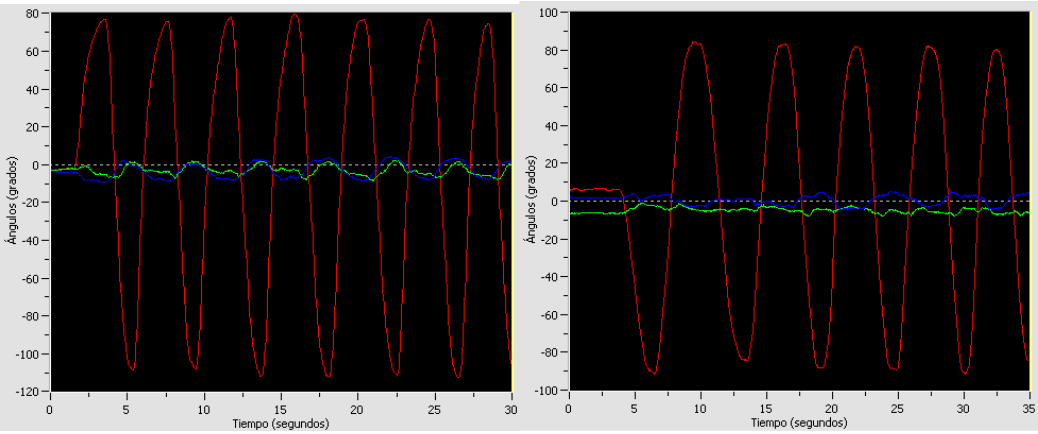


Figura 92: Comparación de flexo-extensión entre paciente 7 (izqda.) y sano (dcha.)

- *Paciente 9*: este paciente tiene limitaciones en los movimientos visibles en gran parte de las pruebas. La flexión, la rotación, el gesto realizado en la segunda prueba funcional son algunos de los movimientos en los que se detecta esta deficiencia. Este conjunto de síntomas, es compatible con un paciente cervicoartrósico.

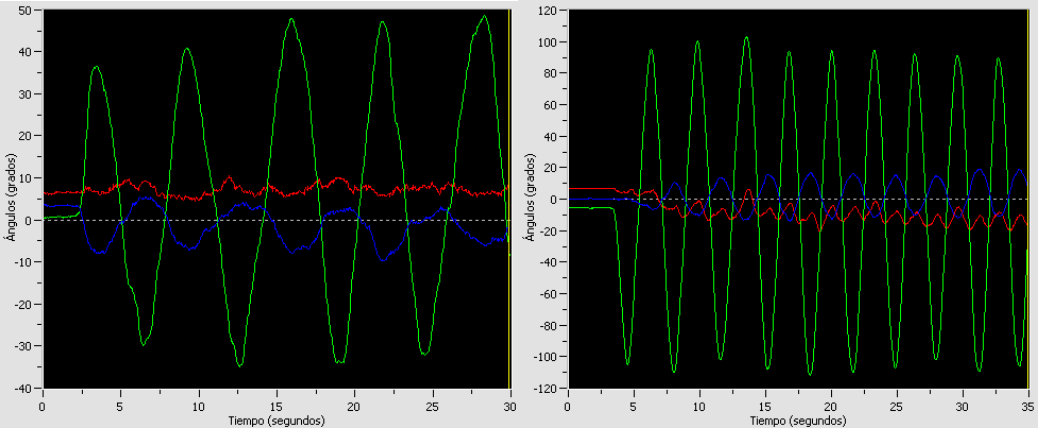


Figura 93: Comparación de rotación entre paciente 9 (izqda.) y sano (dcha.)

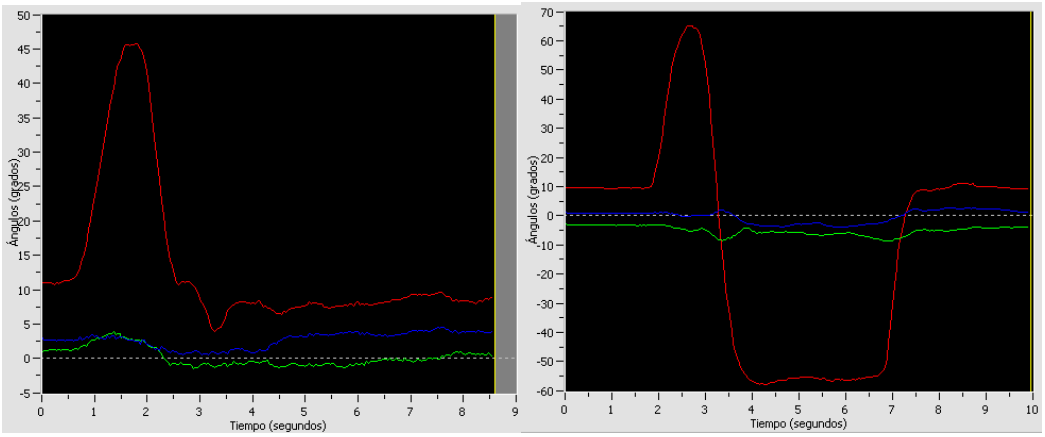


Figura 94: Comparación de la p.funcional 2 entre paciente 9 (izqda.) y sano (dcha.)

- *Paciente 11*: las pruebas realizadas a este paciente tienen, por lo general, un movimiento funcional bastante correcto. Sin embargo el hecho de que alguna de ellas no tenga un patrón constante posibilita encasillar a este paciente como un posible simulador de una dolencia.

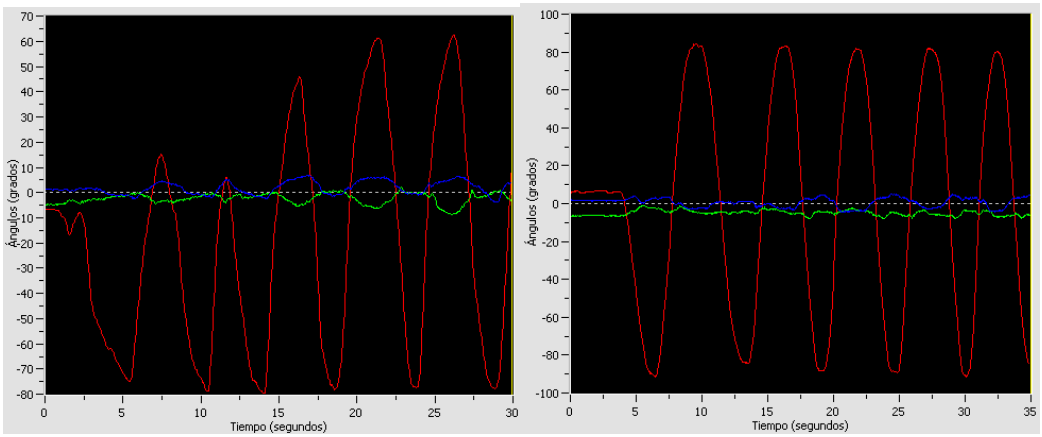


Figura 95: Comparación de flexo-extensión entre paciente 11 (izqda.) y sano (dcha.)

Las animaciones correspondientes a todos los pacientes estudiados se encuentran en la carpeta "Pacientes" que contiene subcarpetas con la información de cada uno de los pacientes, y son accesibles a través de la ejecución del proyecto presentado.

Las consideraciones realizadas en este apartado se pueden confirmar de forma objetiva introduciendo dichos pacientes en la aplicación NedCervical/IBV, ya que es el programa encargado de establecer los parámetros de normalidad y compararlos cada una de las pruebas.

4.- Conclusiones

La sociedad de hoy en día necesita, cada vez más, un apoyo de la tecnología acorde a las nuevas ideas y pretensiones que la misma propone e impone. Los estudios científicos ya no se conciben sin la presencia de un ordenador y cada vez se realizan programas y funciones más automatizadas, que se adaptan a unas condiciones o datos de entrada y generan unos resultados válidos sin la presencia de un ser humano al cargo de los mismos.

Este desarrollo conceptual se está extendiendo a la mayoría de ámbitos posibles: sanidad, educación, deporte… con el fin último de facilitar las tareas y cálculos, así como otorgar la posibilidad de realizar estas gestiones de manera más eficiente y, por tanto, con un grado de progreso mayor.

Dentro de este gran compendio de ámbitos, la aplicación descrita en esta memoria se sitúa en una temática cercana a la sanidad y medicina de forma muy general, y realizando una clasificación más minuciosa, dentro de la estructura que compone la valoración funcional.

Uno de los grandes problemas a los que hace frente la valoración funcional es la necesidad de intentar objetivar un diagnóstico ante unos resultados obtenidos mediante medidas precisas. Para ello el proceso seguido realiza, entre otros cálculos, una comparación entre las medidas obtenidas en pacientes sanos y los pacientes que son objeto de estudio. Esta comparación requiere la disposición de una enorme base de datos con pacientes de todo tipo y muy diversificados, ya que sólo es posible establecer unos parámetros como pertenecientes a un rango de normalidad en el caso de que la población estudiada sea lo suficientemente heterogénea como para definir unos comportamientos medios o de referencia.

Este cálculo estadístico y matemático se muestra, en su mayoría, en forma de gráficas y datos numéricos. La ventaja fundamental para mostrar de esta manera la información es el hecho de disponer unos resultados discretos ajenos a cualquier tipo de subjetividad asociada a una apreciación de un diagnóstico. Sin embargo esto que al principio se puede clasificar como una gran ventaja, en algunas ocasiones tiene un efecto contraproducente: aleja al valorador o médico encargado de realizar un diagnóstico de una apreciación subjetiva de la afección física del paciente, además de obligarle a comprender todos los datos que se le muestran para que realmente sirvan de apoyo para obtener una etiología correcta.

La aplicación desarrollada ataca directamente este posible problema: intenta otorgar al médico valorador de una perspectiva mucho más amigable, alejada de los datos numéricos, que toma cuerpo en una animación en tres dimensiones obtenida a través de las pruebas realizadas al paciente tratado. Así en cualquier momento se puede consultar dicha animación para comprobar in situ cualquier aspecto relacionado con el movimiento realizado en la prueba.

Sin embargo, esto puede llevar a pensar que una animación 3D no aporta muchas más ventajas que las que puede aportar una grabación en vídeo de la prueba: nada más lejos de la realidad. El problema de una grabación en vídeo es que la única interactuación que se puede realizar en ella es la variación del momento de reproducción de la misma. Es posible rebobinar, reproducir fotograma a fotograma, pausar la reproducción para fijarse en algún detalle determinado entre otras acciones similares. En contraposición, una animación en tres dimensiones, además de incorporar todas las interactuaciones de las grabaciones de vídeo, añade una variable más: la posibilidad de modificar la posición y orientación espacial de la

cámara encargada de visualizar la animación. Este hecho, sin lugar a dudas, es de gran utilidad ya que permite obtener el punto de vista idóneo de cada uno de los movimientos facilitando en gran medida la apreciación de cierta deficiencia en algún gesto.

Además el hecho de incorporar esta animación 3D, aparte de resultar tremendamente útil en el proceso de valoración, proporciona cierta dosis de modernidad y actualidad al programa actual. La mayoría de programas y aplicaciones de éxito en la actualidad incorporan una interfaz con componentes en tres dimensiones que la hacen muy atractiva para el público. Aunque suene algo trivial e incluso simple, el 3D está de moda.

Otro aspecto en el que actualmente también se está aplicando especial incidencia en la programación es el concepto de usabilidad. Los programas y aplicaciones desarrollados en la actualidad no deben exigir un aprendizaje previo para darles una utilidad práctica a nivel de usuario. Por ello las interfaces pretenden ser muy intuitivas y de fácil manejo y comprensión, hecho que se ha tenido en cuenta para la realización del proyecto actual ya que el destinatario del mismo no tiene porqué poseer conocimientos informáticos avanzados.

Por tanto y teniendo en cuenta todos los aspectos comentados, el proyecto realizado, considerado como una estructura global, responde a los objetivos marcados en el mismo y puede tener una gran aplicación en el ámbito de la valoración funcional de cervicalgias. Se trata de un producto que responde a una demanda o carencia en los sistemas de valoración estudiados a los que proporciona una solución.

Trabajo futuro

A pesar de considerar la aplicación como un producto cerrado e independiente, tiene varios frentes de trabajo abiertos para su perfeccionamiento en un futuro cercano.

Como se ha comentado en el origen del trabajo, la aplicación surge ante la carencia que presenta un programa de valoración funcional de cervicalgias (NedCervical/IBV). En consecuencia, la integración con dicho programa es una de las vías que puede mantener vivo el desarrollo del proyecto explicado en la memoria actual ya que, de esta forma, se formará una aplicación para la valoración funcional con las ventajas de ambos proyectos y disminuyendo las limitaciones que ambos presentan.

Otra fuente de trabajo en un futuro está vinculada a la mejora visual, tanto de la animación 3D como de la interfaz. La funcionalidad sería la misma que en el programa mostrado, pero incorporar un modelo mucho más realista (quizá con unos polígonos más suavizados) puede mejorar sensiblemente el aspecto de la aplicación final. En el *lookandfeel* o aspecto de la interfaz general también se puede incidir para lograr una aplicación final más innovadora, aunque en el diseño de la interfaz se ha intentado seguir la línea de diseño de las aplicaciones típicas de Windows ya que es el sistema operativo más extendido en la actualidad para ordenadores personales.

BIBLIOGRAFÍA

❖ CERVICALGIAS Y VALORACIÓN FUNCIONAL:

- INSS (Instituto Nacional de la Seguridad Social), Ministerio de Trabajo e Inmigración. *Tiempos estándar de incapacidad Temporal*. 2009. pág.:1-7, 66.

- José Paz Jiménez, Miguel Ángel Belmonte Serrano, Formación continuada de la SECOT y de la SER. *Monografías médico-quirúrgicas del aparato locomotor. Cervicobraquialgia*. 2000. pág: 1-12.

- Enrique Viosca Herrero, Jaime Prat Pastor, Carlos Soler Gracia, Mª Francisca Peydro de Moya, Mª José Vivas Broseta, Carme Gimeno Mollà, Javier Sánchez Lacuesta, IBV (Instituto de Biomecánica de Valencia). *Valoración Funcional*. 2004.

- Revista Consumer, *Cervicalgias: sobrecargas musculares, la causa más habitual*. Octubre 2003 [Visitado: 28/01/10]
  http://revista.consumer.es/web/es/20031001/salud/

- DMedicina, *Advierten del cambio en causas de baja laboral*. 19 de Junio de 2002 [Visitado: 28/01/10]
  http://www.dmedicina.com/enfermedades/musculos-y-huesos/actualidad/advierten-del-cambio-en-causas-de-baja-laboral

- Compendio de Artículos publicados en el THMJ, *Monográfico sobre salud cervical de la task forcé al efecto de la THMA*. 1998 [Visitado: 29/01/10]
  http://www.txoriherri.com/cervical.htm

- Dolores Jiménez-Peña Mellado, Joaquina Ruiz del Pino, Silvia Hazañas Ruiz, Melchor Conde Melgar, Elena Enríquez Álvarez. *Traumatología del raquis: cervicalgias y lumbalgias*. [Visitado: 29/01/10]
  http://www.medynet.com/usuarios/jraguilar/Manual%20de%20urgencias%20y%20Emergencias/cervilum.pdf

- Carlos Albadalejo CAP Llefià, Unidad Docente de Medicina Familiar y Comunitaria (Badalona). *Cervicalgia*. Noviembre 2007 [Visitado: 29/01/10]
  http://www.medicinageriatrica.com.ar/viewnews.php?id=EEAukEVFllUlajWafj

- Bartolomé Llor Esteban, Mariano García Izquierdo, Aurelio Luna Maldonado. *Variables psicosociales y de personalidad asociadas a la cervicalgia recurrente*. 2006 [Visitado: 29/01/10]
  http://dialnet.unirioja.es/servlet/articulo?codigo=2594927

❖ ANTECEDENTES TÉCNICOS:

- Grupo de informática gráfica, Universitat Jaume I. *Motores de juegos. Modelado multirresolución en juegos por ordenador* [Visitado: 29/01/10]
  http://ima.udg.edu/iiia/GGG/TIC2001-2416-C03-01/docs/EnginesUJI.pdf

- Página oficial de Autodesk. [Visitado: 29/01/10]
  http://www.autodesk.es/adsk/servlet/home?siteID=455755&id=458320

❖ **OGRE 3D:**

- Gregory Junker, Founder/Lead Developer of OGRE 3D. *Pro OGRE 3D Programming*. 2006.

- Comunidad oficial Ogre 3D. [Visitado: 27/01/10]
http://www.ogre3d.org

- Alejandro Woywood. [Visitado: 27/01/10]
http://www2.ing.puc.cl/~iic3686/Tutorial_Ogre1/ogre1.htm

- Blog de gazzell. [Visitado: 27/01/10]
http://gazzell.wordpress.com/2008/03/19/uso-bsico-de-compositores-en-ogre/

❖ **BLENDER:**

- Página oficial de Blender 3d [Visitado: 28/01/10]
http://www.blender.org/

- Niel 3d-Comunidad de Blender 3d en español [Visitado: 27/01/10]
http://www.niel3d.com

- Blog de fisicolomolon. *Curso de creación de videojuegos* [Visitado: 28/01/10]
http://fisicomolon.blogspot.com/2009/06/curso-creacion-de-videojuegos-en.html

- Wikilibros. *Blender 3D* [Visitado: 28/01/10]
http://es.wikibooks.org/wiki/Blender_3D:_novato_a_profesional/blender_interfaz

- Foro soliman. *Apuntes de Blender* [Visitado: 28/01/10]
http://apuntesdeblender3.iespana.es/

➢ TEXTURAS UV**:**

o Foro 3DPoder [Visitado: 28/01/10]
http://www.foro3d.com/f240/mapas-uv-79726.html
http://www.foro3d.com/f240/mapeado-texturas-79727.html

o Claudio Andaur, Maleficio 3D [Visitado: 28/01/10]
http://www.malefico3d.org/viejos-tutoriales/UVmap.html

o Usuario Garman, Youtube [Visitado: 28/01/10]
http://www.youtube.com/watch?v=6O8HC3n5wWE&feature=related

o Derek Marsh, UV Mapping a Human Head Mesh Utilizing LSCM [Visitado: 28/01/10]
http://bgdm.katorlegaz.com/lscm_tute/lscm_tute.htm

- EXPORTACIÓN A OGRE:
  - Blog de Antonio Serrano [Visitado: 28/01/10] http://www.aserrano.com/2007/12/19/blender-exportar-modelos-a-ogre/
  - Source forge, Ogre Meshes Exporter[Visitado: 28/01/10] http://ogre.svn.sourceforge.net/viewvc/ogre/trunk/Tools/BlenderExport/ogrehelp/ogremeshesexporter.html#Armature%20Animations
  - ToM's Blog [Visitado: 28/01/10] http://thomas.trocha.com/wp/?page_id=248
  - Blog de Juan Diego Uran, Valtovar [Visitado: 28/01/10] http://valtovar.blogspot.com/2009/06/formatos-mesh-y-skeleton-de-diversas.html

- **INTERFAZ GRÁFICA:**

- The Code Project, Development Resource [Visitado: 28/01/10] http://www.codeproject.com/
- Function X [Visitado: 28/01/10] http://www.functionx.com/visualc/
- Microsoft msdn [Visitado: 28/01/10] http://social.msdn.microsoft.com/
- Jodrell Bank Centre for Astrophysics [Visitado: 28/01/10] http://www.jb.man.ac.uk/~slowe/cpp/lastmod.html
- Foro Wreckedgames [Visitado: 28/01/10] http://www.wreckedgames.com/forum/index.php?topic=1046.0
- Kelvin Sung, Microsoft Foundation Classes Tutorial [Visitado: 28/01/10] http://depts.washington.edu/cmmr/biga/chapter_tutorials/mfc_tutorial/

- MEASUREMENT STUDIO
  - Página oficial de National Instruments [Visitado: 28/01/10] http://www.ni.com/es/

Listado de ilustraciones

ANEXO:

a) Manual del instalador

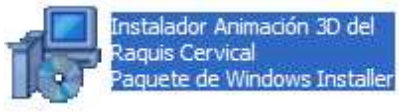
Instalación de la aplicación

La aplicación está diseñada para ser instalada en un sistema operativo *Windows* y, de forma recomendable, *Windows XP*. El proceso de instalación se realiza de forma prácticamente automática e incluye todos los archivos necesarios para ejecutar la aplicación. De esta forma en el directorio de instalación se encuentran estructurados los siguientes elementos:

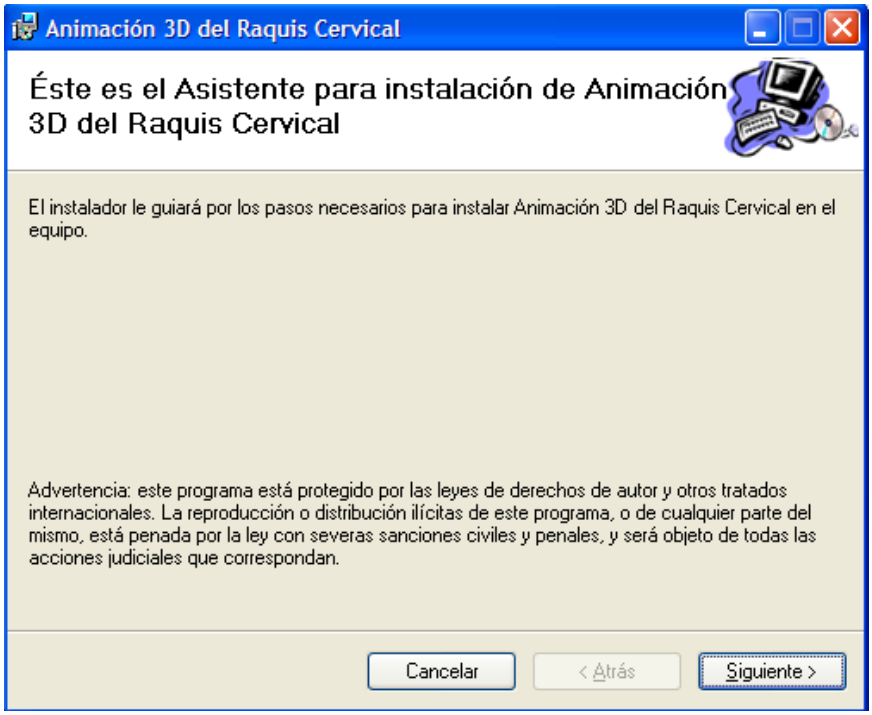
- El ejecutable de la aplicación (*Proyecto.exe*).
- Una carpeta llamada *Media* que contiene los recursos de Ogre.
- Una carpeta llamada *Fuentes* con el código fuente de la aplicación.
- Una carpeta llamada *Pacientes* que contiene subcarpetas con las pruebas de unos sujetos determinados, que se emplearan para cargar las animaciones.
- El resto son archivos con extensión *.dll* y otros necesarios para la aplicación.

A continuación se va a realizar una traza de instalación típica, mostrando cada una de las ventanas del proceso de instalación hasta llegar a la ejecución del programa.

En primer lugar hay que pulsar doble clic sobre el instalador de la aplicación (archivo *Instalador Animación 3D del Raquis Cervical.msi*).

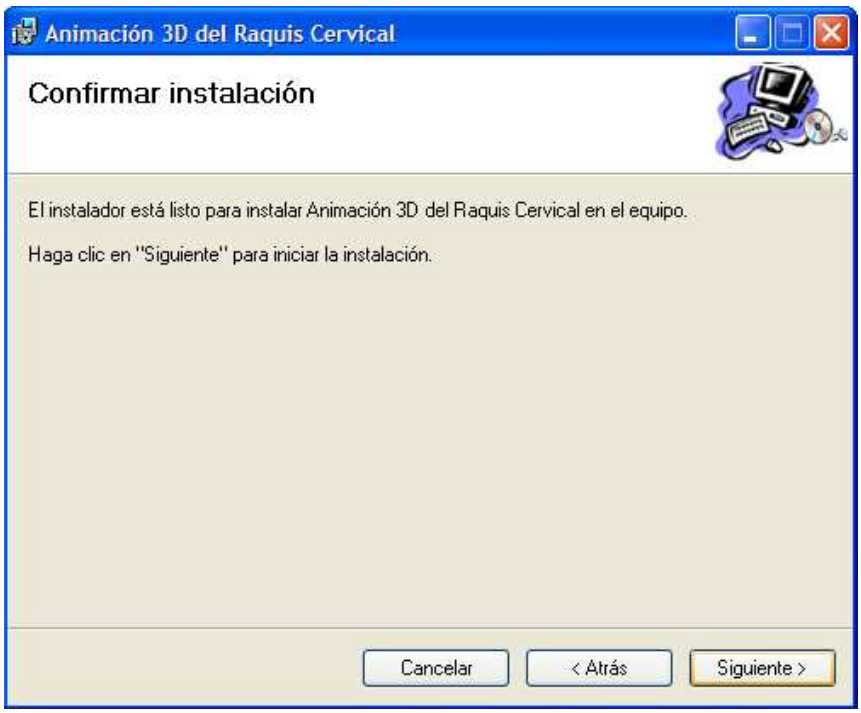


Una vez se ha ejecutado el instalador, se inicia el asistente para instalar el programa. Tras leer la información sobre los derechos de la aplicación, se pulsa *Siguiente* >.

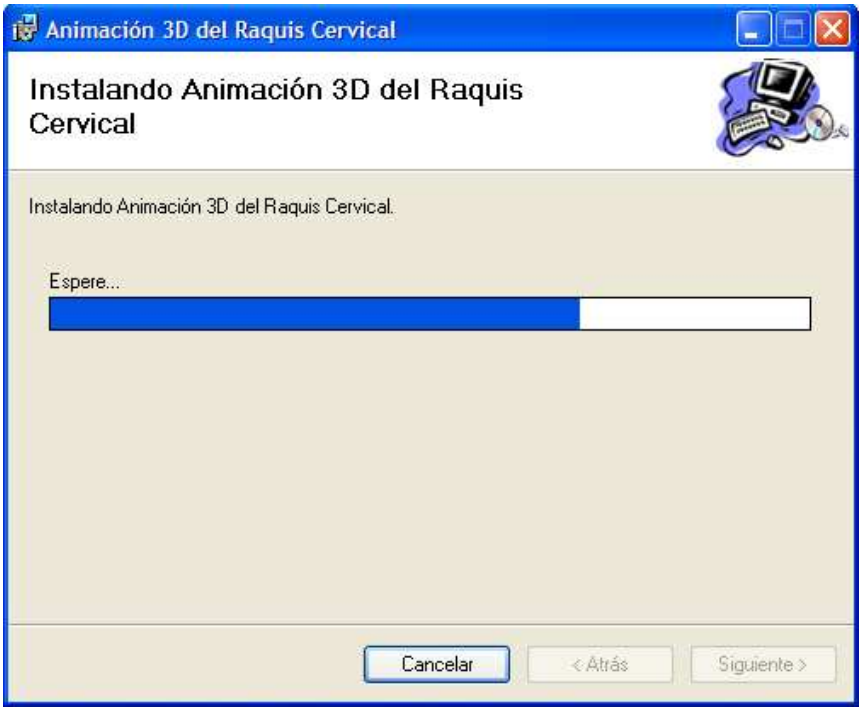


Seguidamente se muestra una ventana que contiene la ruta donde se va a realizar la instalación, así como la posibilidad de seleccionar si la instalación se realiza para un usuario o para todos los usuarios del equipo. Seleccionadas las opciones deseadas, se pulsa en *Siguiente* >.

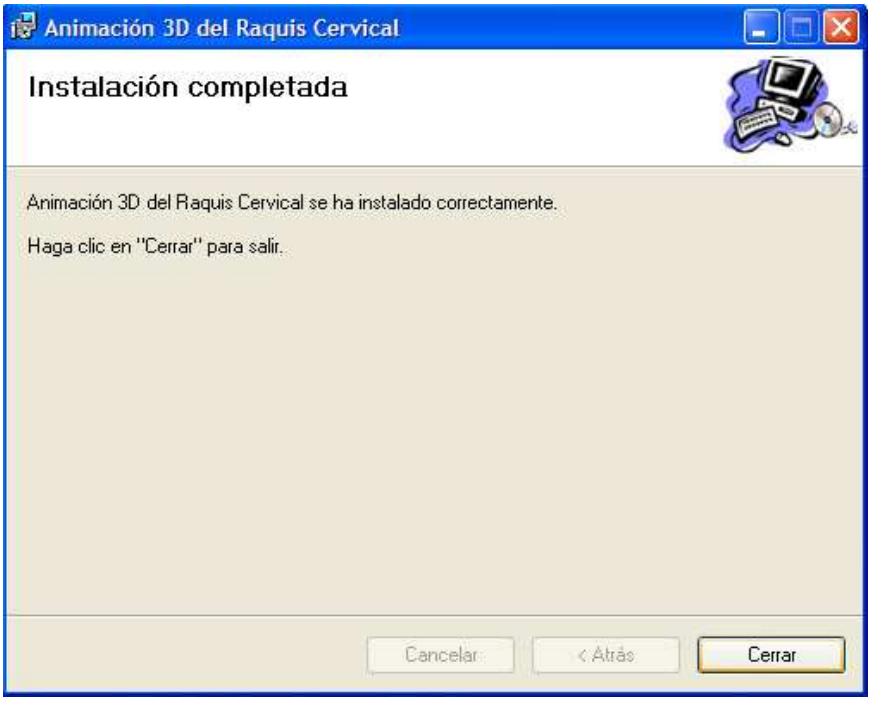
En este momento ya está todo preparado para instalar la aplicación, se pide una última confirmación y se pulsa en *Siguiente* >.



A partir de este instante, una barra de progreso indica el porcentaje de instalación completado.



En el momento acaba la instalación, se muestra una ventana indicando la finalización de la misma.



Ahora para ejecutar la aplicación, hay que acceder a la ruta indicada y ejecutar el archivo *Proyecto.exe*.

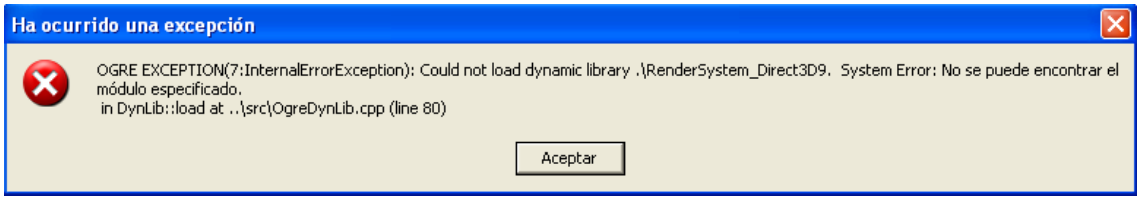
Si es la primera vez que la aplicación se ejecuta, se muestra una ventana de Ogre como la indicada a continuación permitiendo elegir el sistema de renderizado para la aplicación. Se debe elegir *Direct3D9 Rendering Subsystem*.

En el momento se ha seleccionado un sistema de renderización, se muestra la información de configuración del mismo, así como el modelo de tarjeta gráfica instalada en el equipo.

Además, en el directorio de la aplicación se crea un archivo llamado *ogre.cfg* que guarda esta información para posteriores utilizaciones de la aplicación. A medida que se van cargando recursos en la aplicación, también se genera un archivo con nombre *Ogre.txt* con la información de los recursos cargados con éxito.

En caso de que no se haya elegido ninguno, el programa se cierra de forma inmediata.

NOTA: es posible que al ejecutar la aplicación se muestre el siguiente error.



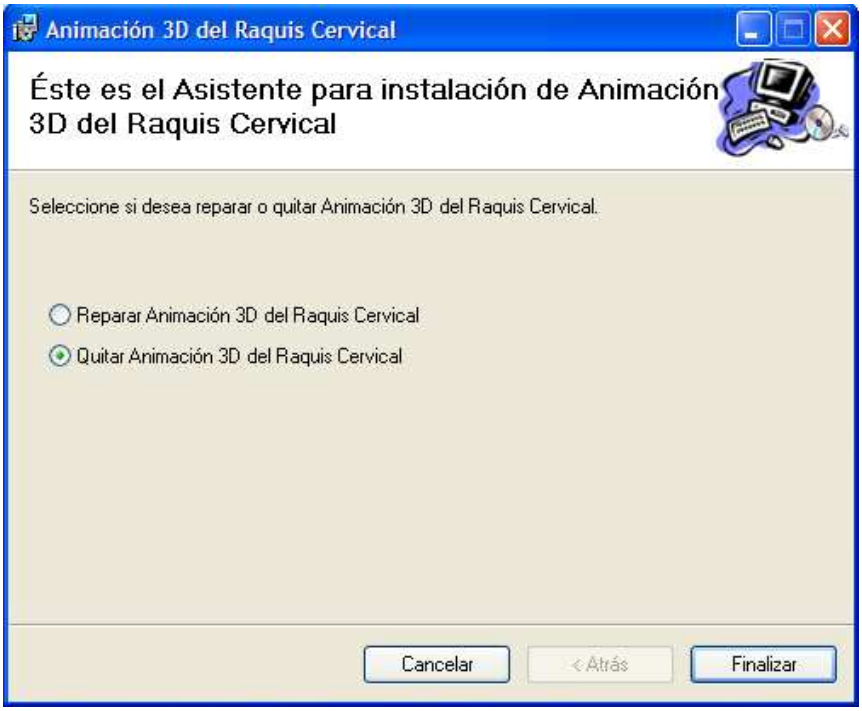
Esto se debe a que el equipo no dispone de la instalación de una versión de *DirectX* adecuada. Habrá que instalar una versión la última versión de *DirectX 9* para ejecutar la aplicación (en el cd adjunto se añade un instalador con una versión actualizada de *DirectX 9* con nombre *dxwebsetup.exe*).

Desinstalación de la aplicación

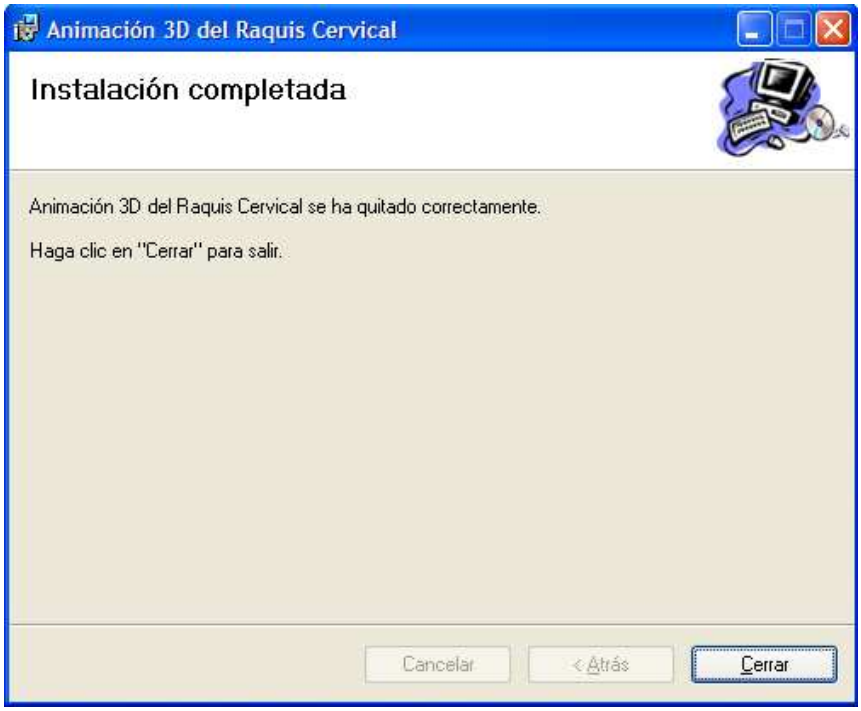
Para realizar la desinstalación de la aplicación, se puede realizar de dos formas:

- A través del instalador *Instalador Animación 3D del Raquis Cervical.msi*
- Aaccediendo al *Panel de Control* a *Agregar o quitar programas*.

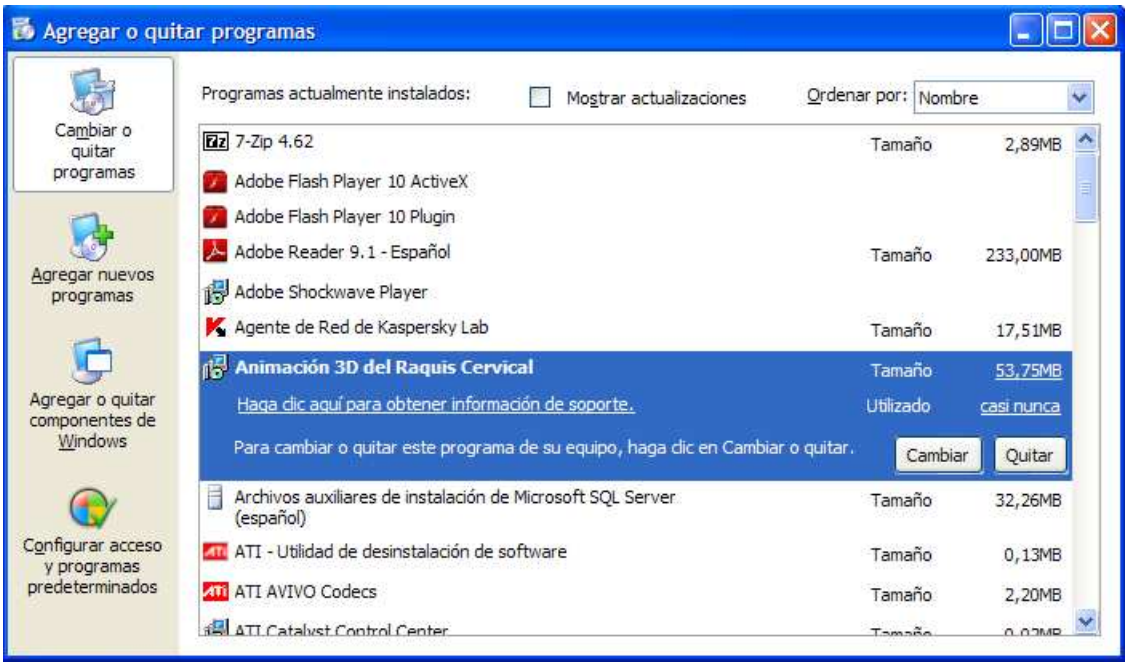
En caso de seleccionarse la primera opción, se muestra un asistente con dos opciones sobre el programa: *Reparar* o *Quitar*. Se selecciona *Quitar* y se pulsa en *Finalizar*, hecho tras el cual comienza de forma automática la desinstalación.



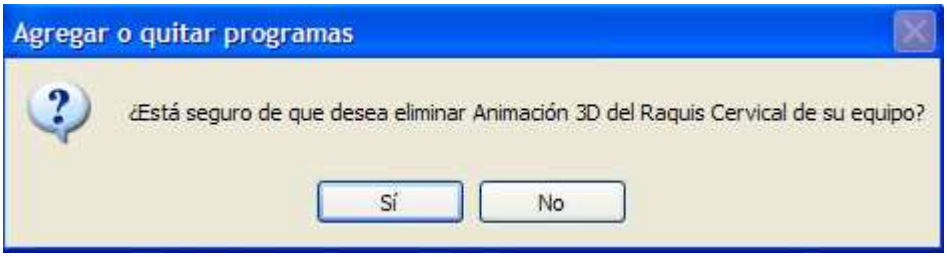
Cuando acaba la instalación, se muestra una ventana informando de que el programa ha sido desinstalado con éxito.



En caso de seleccionar la opción de desinstalar el programa a través del panel de control, se busca el nombre del programa en la lista y se pulsa en *Quitar* para desinstalar de forma automática, o *Cambiar* para que aparezca la ventana mostrada anteriormente con las opciones de reparar y quitar.



A continuación se pide una confirmación para desinstalar el programa, a la que se debe pulsar en *Sí*, y la desinstalación se realiza de forma automática.



NOTA: aunque el proceso de desinstalación parezca congelado o parado durante bastante tiempo, no hay que pausarlo o detenerlo. Además tras acabar el proceso de desinstalación, habrá que eliminar a mano los archivos *ogre.cfg*, *Ogre.txt* y la carpeta donde se instaló la aplicación.

b) Manual de usuario

El programa actual no requiere un periodo de aprendizaje previo y es muy intuitivo; sin embargo y se establece la siguiente estructura dividida en función de su comportamiento:

- Traza de utilización estándar
- Manipulación de archivos cargados
- Manipulación de cámaras de la animación
- Manipulación del momento de reproducción
- Cierre de la aplicación

- Traza de utilización estándar

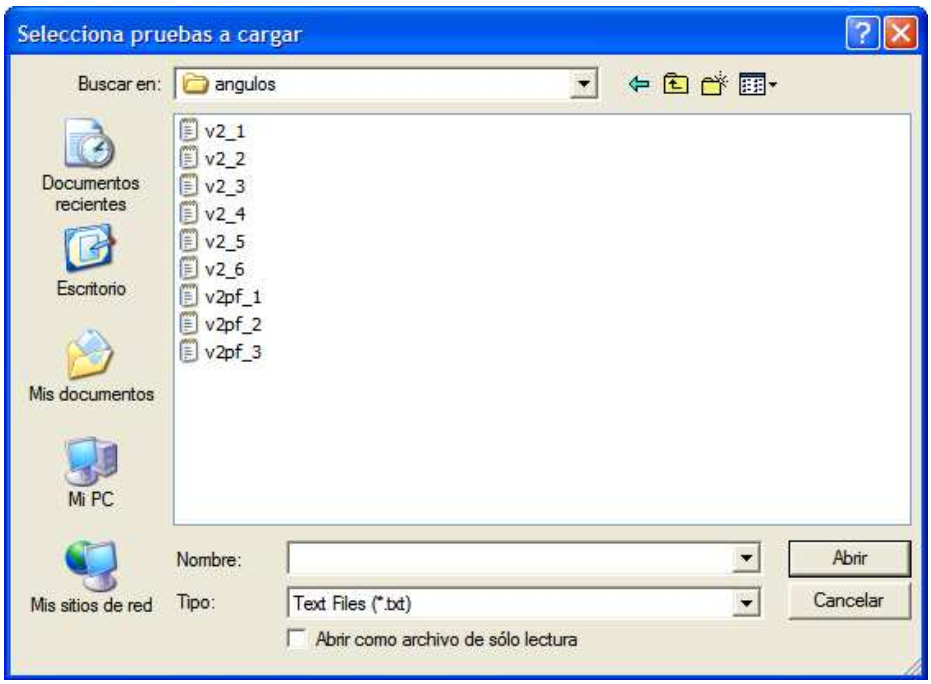
Se va a realizar una descripción de los pasos a seguir para obtener la visualización de la animación de las pruebas.

En primer lugar se ejecuta el programa y se carga la ventana principal de la aplicación (observar aspecto de la *Figura 68* de la memoria actual). A partir de este momento, el proceso para utilizar el programa descrito se puede dividir en tres partes:

- Búsqueda de los archivos de prueba a emplear
- Carga de la animación obtenida de archivos validados
- Visualización de resultados

❖ Búsqueda de los archivos de prueba a emplear

Para obtener obtener ficheros de prueba, hay que pulsar en el botón  de la interfaz, el cual abre un cuado de diálogo sobre el que se puede buscar la ruta en el que se encuentran los archivos de pruebas de pacientes a cargar.



Cabe recordar que el tipo de los ficheros a abrir es *.txt* y la nomenclatura y contenido de los mismos se describe en el apartado 2.2.3.1 y 2.2.3.2 respectivamente.

Una vez seleccionados los ficheros a cargar se pulsa en el botón *Abrir* y el programa informa del número de ficheros de prueba válidos para ser cargados mediante una ventana saliente (ver figura 50).

En este momento los ficheros son cargados por la lista de control de la interfaz, mostrando la información de los mismos: nombre de prueba, ruta de la que se han obtenido, duración de la prueba y fecha de modificación (ver figura 42).

❖ Carga de la animación obtenida de archivos validados

Los archivos mostrados en la lista de control contienen las pruebas que pueden ser cargadas como una animación. Para cargar la animación que contienen, basta con seleccionar el archivo y pulsar en el botón  o simplemente hacer doble click sobre alguno de los elementos de la lista.

Tras este proceso de cargado, la animación se muestra en la subventana de Ogre y se carga la gráfica asociada a dicha prueba, al estilo de lo mostrado en la figura 52.

❖ Visualización de resultados

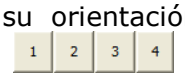
A partir de este momento ya es posible interactuar con todos los elementos que modifican la visualización de la animación actual. Modificar el rango de reproducción de la animación mediante la variación de los cursores límite de la gráfica, pulsar en la gráfica para ver un momento de reproducción deseado, cambiar entre las distintas cámaras fijas, modificar la posición de la cámara libre, pulsar los botones de reproducción de animación… son algunas de las funcionalidades que permiten visualizar de la mejor manera posible la prueba cargada.

Manipulación de archivos cargados

Los archivos cargados en la lista de control pueden ser cambiados dinámicamente. Para añadir nuevos archivos, basta con pulsar en el botón  como se ha comentado en el apartado anterior. Para eliminar archivos de la lista de control, el funcionamiento es muy simple: seleccionar el archivo y pulsar el botón  .

Cabe tener en cuenta que el programa permite cargar dos pruebas idénticas, añadiéndolos como archivos diferentes a la lista de control, se permite el borrado de múltiples pruebas tras mostrar un diálogo de confirmación al usuario.

Manipulación de cámaras de la animación

Para visualizar la animación 3D se dispone de 4 cámaras fijas y una cámara libre sobre la que es posible modificar tanto su posición como su orientación. Las cámaras fijas son accesibles a través de los botones 

y se corresponden a las siguientes posiciones:

- Cámara 1: vista frontal
- Cámara 2: vista trasera
- Cámara 3: vista lateral derecha del modelo
- Cámara 4: vista lateral izquierda del modelo

Por otro lado, cuando se activa la cámara libre mediante el botón 

se habilitan los spins() para poder modificar su posición. Mediante el primer spin la cámara se desplaza por el eje X, con el segundo por el eje Y y con el tercero a través del eje Z. Si en algún momento se desea colocar la cámara libre en su posición por defecto (vista frontal), únicamente hay que pulsar en el botón 

. Cabe comentar que al cargar una animación, la cámara que se coloca por defecto es la cámara libre.

En cuanto a variar la orientación de la misma, se realiza mediante la pulsación del botón izquierdo del ratón y arrastre en la ventana Ogre. En el momento se realiza una pulsación, se toma ese punto como referencia y al realizar el arrastre con el botón pulsado cambia la orientación de la cámara.

- Manipulación del momento de reproducción

Las acciones para cambiar la reproducción de la animación se pueden realizar desde tres puntos: los botones de reproducción, la barra de reproducción y la gráfica con la representación de los datos.

Mediante los botones de reproducción () se pueden realizar acciones típicas para modificar la reproducción de la animación: rebobinado, pausado, reinicio de la animación…

Por otro lado la interacción con la barra de reproducción () también es la habitual: mediante acciones de pulsado en la barra o arrastre de la marca que marca el momento de reproducción es posible modificar la animación.

En cuanto a la interacción con la gráfica es posible realizar dos tipos de acciones: modificación del momento de reproducción, ya sea con acciones de pulsación en la misma o con pulsación y arrastre del cursor de animación (el de color blanco); y acotado de la región activa de la gráfica a través de la pulsación y arrastre de los cursores límite de la animación (de color amarillo). Cabe comentar que la región activa se muestra con un fondo negro, mientras que la inactiva tiene un fondo grisáceo.

- Cierre de la aplicación

Se puede realizar a través del botón y a través de la cruceta de cierre de la ventana. Se cierra también cualquier ventana saliente asociado al programa. Sin embargo, en el caso de que se encuentre una animación cargada, se requiere una confirmación para cerrar el programa.