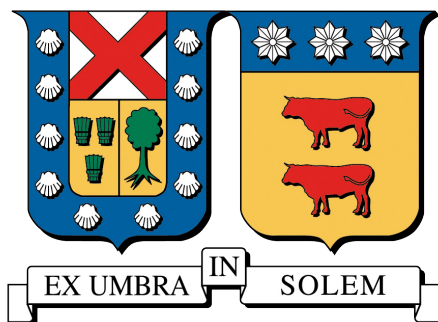


UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE INFORMÁTICA
VALPARAÍSO - CHILE



SINTETIZACIÓN DE IMÁGENES DE ULTRASONIDO PARA SIMULADOR DE EXÁMENES ECOGRÁFICOS

FRANCISCO JOSÉ SEPÚLVEDA VENEGAS
francisco.sepulvedav@alumnos.usm.cl

MEMORIA DE TITULACIÓN PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL INFORMÁTICO

PROFESOR GUÍA : HUBERT HOFFMANN NAGEL
PROFESOR CORREFERENTE : PABLO REYES CABRERA

Noviembre - 2016

A mis padres, Orba y Patricio, y mi hermana, Camila

Quienes fueron mis primeros profesores, mis primeros amigos y mis primeros compañeros en la vida.

A toda mi gran familia

Que aunque vivieran en las faldas de un cerro enclavados y faltaran solo dos minutos para la medianoche, siempre prestaron ayuda y cariño solo esperando un “gracias” como pago.

A todos los amigos encontrados en el camino

Que desde todos los países y con todas las edades fueron apoyo y compañía en los estudios, la política y la fe.

A todos quienes me han tenido infinita paciencia en estos años.

Y a la Luna Llena

Que apareció inesperada una noche de noviembre y no ha dejado de mostrar su luz desde entonces.

Resumen

La formación de estudiantes de medicina contempla una etapa de entrenamiento práctico que incluye la observación y examen de pacientes “reales”. Esto implica problemas desde el punto de vista del mismo paciente, quien normalmente requiere de un diagnóstico e inicio de su tratamiento lo antes posible, y del estudiante, quien se ve limitado en los casos que puede estudiar en la práctica a los que presenten los pacientes que estén dispuestos a ayudar en estas labores formativas.

Una forma de abordar este problema es el desarrollo de simuladores de exámenes, que permitan emular las condiciones de realización y los resultados de los mismos. Un simulador de estas características tendría dos aspectos centrales: la interfaz usuaria y el “motor lógico” que genere los datos que se esperaría observar en un examen real, a partir de la configuración del sistema y las acciones del usuario. Este trabajo presenta un generador de imágenes simuladas de ultrasonido, basado en la herramienta PLUS, desarrollado con estas consideraciones.

Este informe comienza por revisar el estado del arte en el desarrollo de simuladores de ultrasonido, presentando en el análisis de ese levantamiento de información los criterios que llevaron a seleccionar a PLUS como base para la fase de desarrollo. Luego, a medida que se revisan las características particulares de esta herramienta, se explica cómo estas se relacionan con los requerimientos técnicos que se establecen como necesarios para cumplir el objetivo del sintetizador de imágenes y las modificaciones que se debieron realizar a PLUS para satisfacer completamente los requerimientos antes mencionados.

El sintetizador de imágenes finalmente implementado cumple con todos los requerimientos de usabilidad establecidos y, aunque las pruebas de rendimiento no lograron alcanzar los tiempos establecidos como definición de una simulación “en tiempo real”, sí se lograron determinar cambios que debieran permitir cumplir con ese objetivo - la mayoría relacionadas con las condiciones de uso de la herramienta y no con su implementación en sí.

Abstract

The training of a medical student requires a practical stage that covers patient observation and examination, among others. This usually means a student performs an examination on a real patient, under supervision. Such examination might be problematic from the point of view of both the patient - who would probably rather be seen by a “real” doctor so they may get a quicker treatment for they ailment - and the students themselves, who are limited in the scope of their training to the range of different cases offered by those patients who are willing to be part of said training.

Development of clinical simulators might be a solution to the aforementioned problems. Such simulators should replicate both the examination conditions and results and would have to consider two key aspects: user interface and “logic engine”, the part that receives both the initial configuration data and user input to generate the expected output for the simulated examination. This document presents an ultrasound image generator, based on the toolkit PLUS, meant to be the logic engine for an ecographic examination simulator.

The first part of this document presents and analyses the state of the art regarding ultrasound simulation. This analysis ends up defining the requirements that the planned image generator should fulfil, same which tipped the balance towards choosing PLUS as the basis for this development work. Then a detailed study of the technical characteristics of PLUS is shown, along with an explanation about how said characteristics relate to the fulfilment of the requirements previously defined and the modifications that had to be done on PLUS so that said requirements were met completely.

The image generator developed in this work meets all usability and functional requirements but fails to fulfil the non functional requirement of execution time, although achieving this goal seems to be possible making changes to the execution environment and conditions without requiring modification to the software itself. Such possible changes are identified and presented in the conclusions of this report.

Índice general

1. Introducción	1
1.1. Objetivo general	2
1.2. Objetivos específicos	3
2. Estado del arte	4
2.1. Definiciones	4
2.1.1. Sistema tutor inteligente	4
2.1.2. Examen ecográfico	5
2.2. Técnicas de sintetización de imágenes	8
2.2.1. Modelo físico	9
2.2.2. Texturas	10
2.2.3. <i>Ray tracing</i>	11
2.3. Resumen	14
3. Definición del trabajo de desarrollo	16
3.1. Requerimientos funcionales	17
3.2. Requerimientos no funcionales	18
3.3. Materiales de trabajo	18
3.3.1. Magnetómetro trakSTAR™	19
3.3.2. Modelo tridimensional	20
3.4. Herramientas de desarrollo	22
3.4.1. PLUS	22
3.5. Conclusiones	27

ÍNDICE GENERAL

4. Desarrollo de la aplicación	29
4.1. Modificaciones a PLUS	30
4.1.1. Modelos 3D animados	30
4.1.2. Variación de parámetros de configuración en tiempo de ejecución	33
4.1.3. Múltiples fuentes de datos de posicionamiento simultáneas	37
4.2. Interfaz gráfica	39
4.3. Conclusiones	41
5. Validación de la aplicación	42
5.1. Cumplimiento de requerimientos funcionales	43
5.2. Cumplimiento de requerimientos no funcionales	51
5.3. Conclusiones	54
6. Observaciones y comentarios finales	56
A. Descripción técnica de bibliotecas	62
A.1. PLUS	62
A.1.1. Captura de datos de posicionamiento	64
A.1.2. Lectura de datos de modelo 3D	65
A.1.3. Recepción de datos mediante OpenIGTLink	66
A.2. OpenIGTLink	68
A.2.1. Mensajes TRANSFORM	69
A.2.2. Mensajes NDARRAY	70
B. Descripción técnica de implementaciones	71
B.1. Aplicación OBJtoVTP	71
B.2. Clase IGTLServer	72
B.3. Clase IGTLClient	72
B.4. Clase UsSimulatorWidget	73
B.5. Aplicación UsSimulatorGUI	74
B.6. Clase UsSimulatorGUI	75

Índice de figuras

2.1. Ejemplo de examen ecocardiográfico, modos B (superior) y M (inferior).	8
2.2. Ejemplo de examen ecocardiográfico, modo B.	8
3.1. Dispositivo controlador del magnetómetro trakSTAR™.	20
3.2. Sensor (<i>probe</i>) del magnetómetro trakSTAR™.	20
3.3. Malla del modelo tridimensional de corazón.	21
3.4. Diagrama general de funcionamiento de PLUS.	24
3.5. Fuentes de datos para el algoritmo de simulación de ultrasonido.	27
4.1. Fuentes de datos para el algoritmo de simulación de ultrasonido, con PLUS modificado.	30
4.2. Flujo de datos de configuración dentro del servidor PLUS.	31
4.3. Flujo de datos propuesto inicialmente para dar soporte a variación de parámetros en tiempo de ejecución.	36
4.4. Diagrama de comunicación entre servidor PLUS e interfaz gráfica.	39
4.5. Captura de pantalla de la interfaz gráfica implementada.	41
5.1. Imagen sintetizada con intensidad de señal de 300 [mW/cm ²] (izquierda) y 1500 [mW/cm ²] (derecha).	44
5.2. Imagen sintetizada con frecuencia de señal de 3.0 [MHz] (izquierda) y 1.5 [MHz] (derecha).	45
5.3. Imagen sintetizada con factor de corrección <i>gamma</i> 0.2 (izquierda) y 0.4 (derecha).	46
5.4. Imagen sintetizada con escala de brillo 30 (izquierda) y 60 (derecha).	46
5.5. Imagen sintetizada con offset de brillo 30 (izquierda) y 60 (derecha).	47
5.6. Imagen sintetizada con 128 (izquierda) y 64 (derecha) líneas de muestreo.	48
5.7. Imagen sintetizada con 1000 (izquierda) y 125 (derecha) muestras por línea.	48
5.8. Imagen sintetizada con amplitud de ruido 5.0 (izquierda) y 10 (derecha).	49

ÍNDICE DE FIGURAS

5.9. Imagen sintetizada con frecuencias de ruido (2.5, 3.5, 1.0) (izquierda) y (5.0, 7.0, 2.0) (derecha).	50
5.10. Imagen sintetizada con fases de ruido (50, 20, 0) (izquierda) y (100, 40, 0) (derecha). . .	50
A.1. Diagrama general de ejecución del servidor PLUS.	63
A.2. Diagrama de ejecución - ejemplo “ <i>Vascular needle insertion simulation</i> ”.	64
A.3. Diagrama de clases de entrada de datos de rastreo.	65
A.4. Diagrama de relaciones para algoritmo de simulación de ultrasonido.	66
A.5. Flujo de datos recibidos y enviados mediante OpenIGTLink.	67
B.1. Diagrama de flujo de ejecución de una instancia de <code>IGTLClient</code>	73

Índice de cuadros

5.1. Promedios de tiempo de ejecución, en [ms], para casos “ <i>Static heart simulation</i> ” y “ <i>Ani- mated heart simulation</i> ”.	51
5.2. Promedios de tiempo de ejecución, en [ms], utilizando el dispositivo trakSTAR™.	52
5.3. Promedios de tiempo de ejecución, en [ms], sin simulación de ruido.	53
5.4. Promedios de tiempo de ejecución, en [ms], sin simulación de ruido.	53
5.5. Promedios de tiempo de ejecución, en [ms], en un equipo distinto al usado en el desarrollo.	54

Capítulo 1

Introducción

En el proceso de instrucción y formación de toda profesión u oficio existe una fase de entrenamiento práctico. No se trata solo de la adquisición de contenidos, teorías y principios, sino que de la aplicación de los mismos: ya sea un aprendiz de carpintero, un ingeniero o un físico teórico, todos deben aprender a utilizar las herramientas que les serán propias (la escuadra y el serrucho, un modelo matemático o la razón pura, por nombrar algunas para los ejemplos antes mencionados) y a desenvolverse en los ambientes en que se desempeñarán laboralmente.

Lo anteriormente dicho es especialmente relevante para los estudiantes de medicina. En ellos se confiarán, en el futuro, decenas de vidas de forma directa, por lo que se espera que sean peritos en la utilización de sus instrumentos de trabajo y conocedores eximios de su ámbito de desempeño, el cuerpo humano. Sin embargo, el entrenamiento de estos estudiantes posee una dificultad intrínseca no menor; en efecto, la naturaleza misma de su disciplina implica que para los futuros médicos el entrenamiento ideal sea con pacientes reales, lo cual dificulta mucho este proceso: el tiempo dedicado al estudio por parte de los médicos en formación es tiempo que no se destina al cuidado del paciente, algunas patologías raras son difíciles de vivenciar directamente (precisamente porque ocurren con baja frecuencia), la disponibilidad de los estudiantes y la “disponibilidad” de los pacientes no necesariamente serán coincidentes para una sesión de aprendizaje, por mencionar algunas de las condiciones que dificultan el entrenamiento directo de estos estudiantes - de hecho, este tópico es de por sí un tema de estudio actual, como se puede ver, por ejemplo, en [1].

“La formación de profesionales que trabajan en el cuidado y tratamiento de la salud debe tener una sólida formación en distintos ámbitos. Una de estas áreas es el entrenamiento y capacitación que necesitan los futuros profesionales de la salud en el uso de tecnología que entrega un soporte al momento

CAPÍTULO 1. INTRODUCCIÓN

de realizar un diagnóstico médico, como la adquisición de imágenes médicas.

El aprendizaje en esta área presenta el gran inconveniente de no poseer las herramientas para practicar a voluntad del estudiante la cantidad de veces y horas que él estime conveniente según sus necesidades.”[2].

Como se ve en la cita anterior, esta problemática no es de interés reciente ni mucho menos ha sido ignorada. De hecho, el trabajo de titulación mencionado está relacionado a un proyecto Fondecyt (Fondecyt 11070226), el cual tenía por objetivo la elaboración de un sistema tutor inteligente que, además de proveer de un simulador de un equipo de ultrasonido (tanto en su comportamiento como en su interfaz y experiencia)[3], permitiera el entrenamiento de los estudiantes, planteando ejercicios y evaluando directamente el desempeño de los alumnos frente a los mismos[4]. Este enfoque permite reducir o eliminar las dificultades mencionadas párrafos atrás, además de reportar otros beneficios ([2], citando a [4]). Estos tres trabajos ([2], [3] y [4]) son el antecedente inmediato de esta memoria de titulación.

En vista de la magnitud del esfuerzo requerido para implementar un sistema tutor inteligente completo, este trabajo de desarrollo se enfoca en la generación de una herramienta que permita sintetizar simulaciones de exámenes de ultrasonido. En una primera fase, se revisa la literatura existente para establecer los distintos enfoques ideados y probados en el mundo para crear herramientas de simulación. Con los resultados de esta primera fase, se establecen los requerimientos particulares a cumplir en este desarrollo y las métricas a utilizar para medir su cumplimiento, continuando con la descripción del trabajo técnico realizado y los resultados de las evaluaciones practicadas a la herramienta finalmente implementada y finalizando con una valoración cualitativa del producto final y el establecimiento de potenciales objetivos para futuros trabajos de desarrollo que se realicen a partir de la labor presentada en este informe.

1.1. Objetivo general

El objetivo general de este trabajo de titulación es la implementación de un sistema de sintetización de imágenes de ultrasonido, que responda a las exigencias de rendimiento y configuración que se requieren para su eventual integración en un simulador de exámenes ecocardiográficos¹.

¹Estudio del corazón mediante ultrasonido.

1.2. Objetivos específicos

- Determinar los requerimientos técnicos específicos implicados en el desarrollo del sistema de sintetización de imágenes de ultrasonido.
- Seleccionar las herramientas de desarrollo - en particular, bibliotecas que permitan abordar la generación de las imágenes de ultrasonido - que mejor se adapten a los requerimientos técnicos.
- Implementar un sistema de sintetización de imágenes de ultrasonido que cumpla con los requerimientos técnicos.
- Realizar pruebas y evaluaciones para comprobar que el sistema implementado cumpla con los requerimientos técnicos.

Capítulo 2

Estado del arte

2.1. Definiciones

2.1.1. Sistema tutor inteligente

Un **sistema tutor inteligente** (ITS, por sus siglas en inglés) puede definirse como un “sistema de aprendizaje, basado en computadores, que intenta adaptarse a las necesidades de los estudiantes y que, por tanto, es el único tipo de sistema que intenta ‘preocuparse’ de los estudiantes” (traducido de [5]). La misma fuente anterior da a conocer el avance de la investigación en el campo de los ITS hace casi 15 años. Para los efectos de este trabajo de titulación no es particularmente interesante el profundizar sobre el avance en este punto, aunque sí puede mencionarse que, en el contexto del trabajo inmediatamente relacionado con el proyecto [4] se han realizado avances no solo en la concreción de un sistema de este tipo, sino en el desarrollo de la teoría asociada a los mismos.

A partir de la definición anterior, se pueden identificar tres grandes componentes que serían parte de un ITS enfocado a estudiantes de medicina y dedicado a exámenes de ultrasonido :

1. La *interfaz hombre-máquina*, que consiste no solo de una interfaz gráfica que emule la experiencia de uso de un ecógrafo, sino en la simulación de la experiencia de la toma del examen mediante el uso de un equipo que permite replicar la manipulación física del ecógrafo y el uso de maniqués anatómicos, similares a los que se utilizan en el entrenamiento de primeros auxilios.
2. El *sintetizador de imágenes de ultrasonido*.
3. El *motor de manejo del conocimiento*, el que convierte al simulador en un ITS propiamente

CAPÍTULO 2. ESTADO DEL ARTE

tal. El estudio acabado del mismo, así como sus posibilidades de mejora, son lo suficientemente complejos como para ameritar por sí mismo un trabajo de igual o mayor envergadura que el desarrollado en esta memoria de titulación.

Este trabajo de desarrollo se enfocará principalmente en el segundo punto de los anteriormente enumerados, con atención también a los aspectos que puedan afectar al aspecto de la interfaz entre el usuario final y el sistema. Este último elemento va delimitando los requerimientos funcionales de la herramienta de simulación a implementar: por ejemplo, el sintetizador debe soportar el uso de mecanismos de captura de datos que utilicen de alguna forma un maniquí anatómico.

2.1.2. Examen ecográfico

Un examen ecográfico es aquel que utiliza ultrasonido para la observación de una parte del cuerpo humano. En el caso de ciertas áreas cuyo estudio sea particularmente común, se pueden utilizar nombres específicos para este tipo de examen. Por ejemplo, un examen ecocardiográfico puede definirse como “un conjunto de exámenes, que se obtienen a partir de una técnica de imágenes diagnóstica no invasiva, que radica en la aplicación y recepción de ondas de ultrasonido sobre el área cardíaca. Estas ondas son producidas mediante un transductor compuesto por cristales de material piezoeléctrico que generan ondas de presión con frecuencias de rango que varían desde los 2 a 7 MHz. La información obtenida de los ecos es procesada y desplegada en una pantalla para su visualización y análisis.” [3].

Para efectos de este trabajo de titulación, deben considerarse las características procedimentales propias de este tipo de examen y las propiedades físicas de un examen realizado mediante ultrasonido. Los trabajos utilizados como punto de partida de este desarrollo ([2], [3] y [4]) se concentran en exámenes ecocardiográficos. Por ello, aunque un sistema de sintetización “ideal” debiera ser independiente del objetivo del examen simulado, desde este punto en adelante se hablará en particular de exámenes ecocardiográficos, y será objeto de medición el ver si el producto final de este desarrollo puede generalizarse a otra forma específica de ecografía.

Características físicas

Un examen de ultrasonido utiliza señales sonoras cuya frecuencia se encuentra por sobre el rango audible por el ser humano. Por ello, el estudio del fenómeno físico debe realizarse desde la perspectiva de un evento ondulatorio, toda vez que el sonido corresponde a una onda mecánica. Las principales características propias de un examen de este tipo son [2]:

CAPÍTULO 2. ESTADO DEL ARTE

- **Atenuación:** a medida que la onda de ultrasonido viaja por el medio de transmisión (en este caso, los tejidos del área examinada) y regresa hacia los sensores respectivos, sufre una pérdida de energía (atenuación) debido a efectos como la reflexión, la dispersión y la convección, entre otros - este efecto, además, también depende de características de la onda como su frecuencia. Esta combinación de eventos es altamente dependiente del tipo de tejido por el que atraviesa la onda sonora, por lo que el estudio de la atenuación es importante para extraer información acerca de los mismos. Además, la atenuación también influye directamente en la imagen que se genera a partir de las ondas de sonido: si en un punto en particular el eco es atenuado muy levemente, en la imagen se verá como un punto “brillante” pero, al igual que ocurre directamente con fenómenos lumínicos, la presencia del mismo “opacará” la señal de los puntos adyacentes, los que serán más difíciles de percibir.
- **Scattering:** o dispersión de la señal, ocurre por la naturaleza poco homogénea de los tejidos del cuerpo humano, lo que causa reflexiones cuyo efecto neto es el de una reducción de la tonalidad y el contraste en la imagen final generada. Este fenómeno está relacionado con el antes mencionado y con el ruido *speckle*, que se ve a continuación.
- **Ruido *speckle*:** es un resultado directo del fenómeno de *scattering*, y corresponde al típico patrón de interferencia que se asocia a una imagen de ultrasonido (el efecto “granular” que poseen estas imágenes). Si bien se puede pensar que son de naturaleza aleatoria, en realidad responden a un evento físico determinístico, medible y modelable. Resulta crucial tener en mente que, como corresponde a una propiedad intrínseca a un examen de ultrasonido, su eliminación en la sintetización de imágenes de este tipo no constituiría una mejora, sino todo lo contrario, puesto que alejaría el resultado final de la sintetización de lo que se observaría en un examen real.
- **Anisotropía:** en tejidos muy fibrosos, como los músculos o los nervios, la intensidad de la onda reflejada depende del ángulo de incidencia, pues la reflexión y refracción de la señal de sonido varía dependiendo de si incide en un tejido cuyas fibras estén perpendiculares o paralelas a la dirección de emisión de la onda.

Características de procedimiento

Es importante tener en cuenta que un “pulso” de ultrasonido no está compuesto de una única emisión. Por el contrario, cada haz está compuesto por la señal emitida desde cada cristal o canal que contenga el instrumento de medición utilizado[3]. Con esto en mente, se puede explicar la primera

CAPÍTULO 2. ESTADO DEL ARTE

parte de las características procedimentales del examen, la existencia de ciertos parámetros que pueden adecuarse para realizar un mejor estudio - según sea la necesidad en cada caso:

- Foco: distancia entre los haces que componen el pulso de ultrasonido y/o entre los frentes de onda emitidos.
- Contraste: corresponde al mismo concepto de “contraste” en el caso de una imagen regular aunque, en este caso, se relaciona con la intensidad de la onda emitida y con los efectos físicos que sufren las ondas, por lo que no se puede simplificar este punto con la idea de “a mayor intensidad, mayor contraste”, pues también cambian los resultados de los efectos antes vistos.
- Frecuencia: como se mencionó al explicar el efecto de atenuación, la frecuencia de la onda es uno de los factores determinantes de las características de los ecos generados y medidos por el ecocardiógrafo. En particular, distintas frecuencias reaccionan de manera distinta con los tejidos, por lo que variar la frecuencia del ultrasonido emitido permite estudiar diferentes estructuras anatómicas.

Por otro lado, para el caso específico de las ecocardiografías existen varios tipos de medición, conocidos como “modos”, de los cuales en primera instancia interesan dos[3]:

- Modo M (*Motion*): consiste en el estudio en una dimensión de un haz de ultrasonido en particular. En este modo se genera una gráfica (ejemplificada en 2.1), en función del tiempo, donde se observan las características del tejido medido y su variación (la presentación de este modo posee una “semántica temporal”, toda vez que permite observar y comparar el comportamiento del tejido en el eje determinado por el haz de sonido durante uno o varios ciclos cardíacos).
- Modo B (*Brightness*): el modo más conocido del examen ecocardiográfico, la figura 2.2 muestra este modo. Aunque no posee “memoria” como el modo M, permite visualizar de mejor manera las estructuras cardíacas al ofrecer una imagen en dos dimensiones de las mismas. Se genera a partir de la información proveniente de varios haces de ultrasonido, a diferencia del modo M que solo utiliza uno.

Existe también un tercer modo de examen que es conveniente mencionar, llamado **modo Doppler**, que permite observar el flujo sanguíneo mediante el uso del efecto ondulatorio del que toma su nombre. Este trabajo de desarrollo se enfocará solo en el modo B de un procedimiento ecográfico.

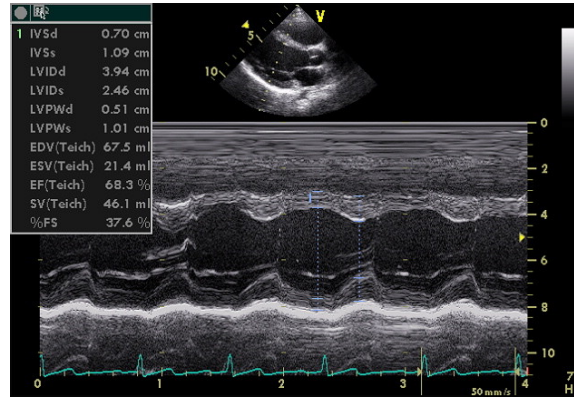


Figura 2.1: Ejemplo de examen ecocardiográfico, modos B (superior) y M (inferior).
Fuente: Dominio público (Wikimedia Commons).

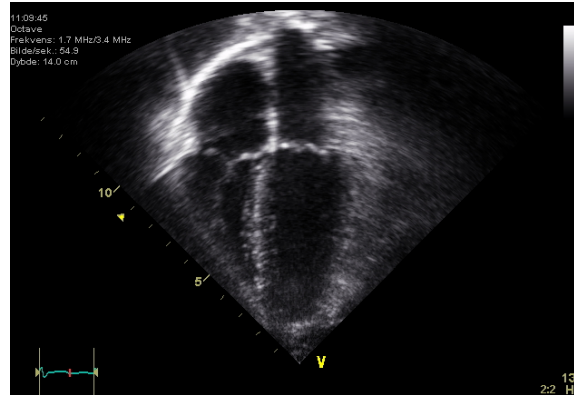


Figura 2.2: Ejemplo de examen ecocardiográfico, modo B.
Fuente: Dominio público (Wikimedia Commons).

2.2. Técnicas de sintetización de imágenes

Para este trabajo, cuando se habla de “sintetización” se hace referencia al proceso de generar, mediante técnicas computacionales, imágenes similares a las de un examen ecocardiográfico, sin contar con un sujeto real ni una señal verdadera de sonido. Se trata, en ese sentido, de emular el comportamiento de un examen real mediante algún mecanismo o enfoque que permita obtener los resultados esperados. Se analizaron tres tipos de técnicas: la implementación directa del modelo físico que describe el comportamiento del ultrasonido, la generación de texturas en un procesador gráfico y la generación de imágenes mediante *ray tracing*.

2.2.1. Modelo físico

Una primera aproximación sería implementar computacionalmente el modelo físico que explica los fenómenos ondulatorios que tienen lugar cuando se realiza un examen de ultrasonido, de manera tal de construir un “simulador” de dichos sistemas y obtener así los resultados que luego serán interpretados al generar la representación visual requerida - la imagen de ultrasonido. En ese sentido, el trabajo realizado en [2] en el marco del proyecto Fondecyt [4] es no solo un completo compendio de las fórmulas requeridas para explicar estos fenómenos, sino que muestra los resultados de una implementación realizada en MatLab de las mismas para la generación de las imágenes. Este punto es especialmente importante, pues le permitió al autor realizar pruebas de rendimiento del modelo y optimizaciones del mismo. Se realizaron mejoras importantes en cuanto a los tiempos de ejecución requeridos, con respecto a los sistemas que se utilizaron de referencia: mientras en una de las citas contenidas en ese trabajo se menciona que una imagen tomaba alrededor de 5,8 días en generarse, incluyendo la creación del fantoma¹ inicial, al final del trabajo se obtuvieron resultados satisfactorios en menos de tres horas: 40 minutos para generar el fantoma y 2 horas para, a partir de éste, sintetizar un cuadro de imagen en ultrasonido.

La investigación y el desarrollo en esta línea ha continuado a nivel mundial hasta la fecha. En efecto, en [6] se encuentra la última publicación encontrada con respecto a este tema. Sin embargo, a pesar de avances notables en cuanto a los rendimientos obtenidos, los tiempos de ejecución siguen siendo prohibitivos a la hora de pensar en este enfoque como una alternativa de implementación para el sintetizador de imágenes: dependiendo de las configuraciones y el modelo usados, los tiempos requeridos para generar una imagen varían entre 3 horas con 30 minutos y 2 minutos - valores mucho mayores a las fracciones de segundo que requiere un simulador en tiempo real (en el orden de 10^{-2} , esto es, de decenas de cuadros por segundo).

Dentro de esta familia de técnicas, una variación posible consiste en realizar aproximaciones matemáticas del modelo físico teórico, las cuales permitan generar una simulación de un examen de ultrasonido con un menor esfuerzo computacional (sin sufrir pérdidas inaceptables de calidad, desde el punto de vista de la utilidad de las imágenes sintetizadas como instrumento de entrenamiento médico). Por ejemplo, [7] ofrece una implementación de la simulación de los efectos de ultrasonido mediante la ecuación de Westervelt, obteniendo resultados para sus dos casos de prueba en 55 y 78 minutos, respectivamente. Cabe destacar que el foco de este trabajo es la obtención de una imagen lo más cercana posible a la realidad, sin tener como objetivo la sintetización de la misma en tiempo real.

¹Imagen base a partir de la cual se sintetiza una imagen de ultrasonido.

CAPÍTULO 2. ESTADO DEL ARTE

En el ámbito de las simulaciones de ultrasonido, existe una aplicación utilizada ampliamente llamada Field II - de hecho, este programa se utiliza en [2] como referencia durante las pruebas de rendimiento. En [8] y [9], además de explicar el funcionamiento de Field II, se presentan implementaciones basadas en este sistema. Con respecto a lo primero, Field II trabaja en base al cálculo de respuestas de impulso espacial, un concepto que considera la naturaleza acústica del problema que se quiere simular, pero que no viene al caso explicar en detalle. Sobre los rendimientos, [8] logra obtener una imagen completa en 15 minutos y 44 segundos, utilizando computación paralela en un clúster de 16 computadores con CPUs de un núcleo; por otro lado, [9] entrega los resultados para la simulación de un examen completo (una secuencia de imágenes) y se concentra más en la reducción de los tiempos de ejecución que se pueden lograr paralelizando el cómputo. De todos modos, el mejor resultado obtenido es de 391 horas para simular un examen completo.

Terminando la revisión de los métodos de simulación matemática, en [10] se presenta una implementación basada en convoluciones, cuyo objetivo es el lograr obtener imágenes en tiempo real o, al menos, avanzar en esa dirección. En las conclusiones de dicha publicación se indica un techo para el tiempo de sintetización de una imagen de 2 segundos, aunque se reconoce también una serie de simplificaciones que reducen la calidad de la simulación en comparación a alternativas como el mismo Field II. De todos modos, se prevee en el mismo trabajo que aún hay espacio para reducir el tiempo de ejecución a niveles que se puedan considerar de tiempo real.

2.2.2. Texturas

Abandonando la perspectiva de la computación científica, una segunda alternativa de solución se encuentra en la computación gráfica: utilizar técnicas que, si bien puede que no se relacionen con la teoría detrás del fenómeno del ultrasonido, permitan sintetizar imágenes que tengan las características que se esperaría encontrar en un examen de este tipo.

En este sentido, [11] presenta un sistema basado en la generación y refinamiento de *texturas*, imágenes generadas por un procesador gráfico (GPU), cuyo flujo de ejecución se puede resumir como:

- Se reciben los valores iniciales de configuración del procesamiento de imágenes y un fantoma “base” - igual que en el caso del modelo físico.
- Se realizan 3 procesos de refinamiento del fantoma, cada uno encargándose de agregar un subconjunto de las características seleccionadas como esenciales a una imagen de ultrasonido. El resultado de una fase constituye la entrada para la siguiente; el primer “barrido” recibe la ima-

gen base generada al comenzar la ejecución.

- Se realizan los tratamientos finales a la imagen sintetizada (ej.: acotar el área mostrada al “cono” característico del modo B) y se entrega el resultado final para su uso por parte del usuario.

En el marco del proyecto Fondecyt [4], se realizó una implementación del sistema presentado en [11], a la cual se tiene acceso para efectos de este trabajo de titulación, aunque no cuenta con publicaciones a las cuales hacer referencia en este texto. Los resultados de la observación de la misma se darán a conocer en las conclusiones de este informe. Sí puede mencionarse, en esta parte, que los fantasmas son generados a partir de un modelo tridimensional: no existe una “biblioteca” de cortes, sino que una implementación que genera estas imágenes base a partir de dicho modelo y de un aparato que simula el acto de manipular el cabezal del ecocardiógrafo. Los detalles de este sistema se verán más adelante, durante el desarrollo de este trabajo de titulación.

2.2.3. *Ray tracing*

La familia de técnicas de generación de imágenes (*rendering*) conocidas como *ray tracing* (rastreo de rayos) fue presentada por primera vez en 1968, en [12], y parten de la idea básica de emitir un “rayo” de luz desde un punto determinado hacia otro y calcular su comportamiento (reflexión, refracción, sombra generada, etc.). La abstracción original es emitir un rayo de luz desde un punto que representa al observador hacia la escena observada, para detectar el punto en el espacio en el que colisiona con un objeto opaco. En [12] se llega hasta este punto, presentando la versión más básica de la técnica; de hecho, este algoritmo en particular hoy es conocido como *ray casting* (**emisión** de rayos, en contraste con **seguimiento**). En lo que hoy se conoce propiamente como *ray tracing*, el procedimiento no termina con la emisión del rayo y la detección de su interacción con alguna parte del ambiente observado, sino que continúa evaluando el rayo emitido con más efectos - como su reflexión hacia el observador, su refracción hacia otros puntos, su transmisión a otro medio si el objeto impactado no es opaco, entre otros. Cada punto evaluado en el escenario observado, entonces, recibe el rayo que le corresponde desde el origen - el observador - y los refractados desde puntos adyacentes, a la vez que genera un rayo dirigido hacia el observador y los que refractan desde este punto hacia puntos vecinos. Por la misma mecánica del procedimiento, éste es iterativo y altamente paralelizable (se evalúa en cada punto con las entradas y salidas antes mencionadas; además, las salidas de una iteración constituyen las entradas de la siguiente, si se considera la simulación completa). Por otro lado, la demanda computacional de estas técnicas puede llegar a ser bastante alta, dependiendo de la cantidad de rayos emitidos y las

CAPÍTULO 2. ESTADO DEL ARTE

propiedades que se evalúen de los mismos.

En [13] se presenta brevemente una implementación de simulación de ultrasonido que utiliza *ray tracing*. Sin embargo, como el énfasis estuvo en obtener imágenes de buena calidad para utilizarlas de referencia en evaluaciones de otro modelo, no se entregan datos acerca del rendimiento computacional de la implementación. En [14] también utiliza una técnica de la familia del *ray casting/tracing*, obteniendo resultados de entre 10 a 15 cuadros por segundo (*frames per second*, o fps), pero sobre un modelo estático generado en más de 30 horas. Por otro lado, en [15] se mencionan otros trabajos que presentan soluciones basadas en *ray tracing* -, ninguno de los cuales cumple totalmente con el requisito de generación de la simulación en tiempo real - además de presentar una implementación parcialmente desarrollada como *ray tracing* y que, además, aprovecha la tecnología CUDA (*Compute Unified Device Architecture*, de NVIDIA) para paralelizar el cómputo y obtener, así, resultados en tiempo real: 13,28 ms por *frame*, lo que permite, aproximadamente, un máximo de 75 fps. La lógica de generación de los fantasmas, sin embargo, requiere del uso de registros de exámenes de tomografías computarizadas (CT) realizados previamente, lo cual no es compatible con las exigencias de diseño de este trabajo. En [16] se presenta un trabajo muy parecido a [15], pero implementado en OpenGL (una herramienta de desarrollo gráfico, no pensada para cómputo general, como CUDA).

Revisando los *papers* publicados en los últimos años, se aprecia un incremento importante, tanto en el volumen de publicaciones y sistemas desarrollados sobre esta temática y técnica en particular, como en la cercanía con la solución ideal para el caso planteado en este trabajo (cantidad de cuadros por segundo generados, parámetros considerados, etc.). Se pueden distinguir 3 “frentes” de desarrollo en las publicaciones revisadas:

- Mejoras *formales* a la simulación, que incluyen análisis más exhaustivos a las funciones matemáticas involucradas[17], implementación y evaluación de métodos distintos a los que ya han alcanzado un cuerpo de desarrollo que los convierte en los estándares para crear soluciones de simulación de ultrasonido[18] y la inclusión de nuevas propiedades, como la simulación del flujo sanguíneo[19].
- Mejoras *técnicas* a la simulación, como la transformación e implementación de los algoritmos clásicamente utilizados a una lógica paralela, para aprovechar de mejor manera la computación general en GPU (GPGPU)[20].
- Ampliación de las *fuentes de datos* disponibles para la sintetización de las imágenes de ultrasonido. Si bien [4] no utiliza información CT para generar la simulación de ultrasonido, sino un modelo

CAPÍTULO 2. ESTADO DEL ARTE

3D generado directamente en un editor *ad hoc*, la mayoría de los trabajos enfocados en técnicas de tipo *ray tracing*, publicados hasta antes la década de 2010, utilizaban la primera alternativa (junto con información de resonancia magnética, MRI) como fuente primaria de las imágenes 2D que serían posteriormente tratadas para simular un cuadro de imagen en ultrasonido. En [21] y [22], empero, se presentan simuladores desarrollados bajo la lógica de *ray tracing*, utilizando computación paralela (GPGPU) y que generan las imágenes a partir de modelos poligonales 3D.

Todo lo anterior se ve presente en el kit de herramientas *toolkit* llamado *Public Library for Ultrasound* (PLUS), presentado en [23] y desarrollado en [24] y [25]: “El propósito de PLUS es proveer un *toolkit* disponible gratuitamente que facilite el desarrollo rápido de sistemas de intervención guiados por ultrasonido para la investigación traslacional clínica. El *toolkit* PLUS otorga acceso a varias herramientas de rastreo de posición y de imágenes de ultrasonido usando una interfaz única e independiente del hardware, e incluye software, hardware, documentación y conocimientos know-how para calibraciones, procesamientos de datos, visualizaciones y operaciones de transferencia comúnmente requeridas.” (traducido desde [23]). Uno de los problemas cubiertos en el desarrollo de esta herramienta de software es la dificultad para acceder a los datos de imágenes y posicionamiento entregados por dispositivos ecográficos comerciales; para solucionar este punto, además de utilizar protocolos de comunicación abiertos como OpenIGTLink (este último será analizado más adelante, en la sección de desarrollo y los apéndices de este informe), otorga la posibilidad de generar simulaciones de imágenes de ultrasonido sin requerir información de exámenes previos o de equipos físicos de ecografía.

Esta biblioteca, de código abierto (licencia tipo BSD), presenta, entre otras características de diseño y técnicas:

- Trabajo en base a modelos poligonales 3D (como [21] y [22]).
- Resultados en tiempo real: 50fps para un formato de imagen de 564x597 píxeles en [25]; 30fps para una imagen de 1280x1024 píxeles en [23].
- Implementación en C++ y basada en el *toolkit* Qt, para las partes del software que implementan una interfaz gráfica.
- Implementación serial (en CPU).

Al ser de código abierto y libre, PLUS ofrece la posibilidad de adaptación a las necesidades particulares de un proyecto y la oportunidad de estudiar y, eventualmente, implementar mejoras (como la paralelización).

2.3. Resumen

El problema de la generación de imágenes de ultrasonido sintetizadas es de larga data y, por tanto, posee una amplia literatura con respecto a su solución. La variedad de planteamientos y desarrollos sobre este tema se diferencian en el aspecto en particular que desean abordar, existiendo dos grandes “facetas”: la fidelidad de los resultados obtenidos con respecto a imágenes reales y el rendimiento computacional, medido en el tiempo empleado para generar una imagen sintética.

En el caso de los estudios revisados en este trabajo, la similitud de las imágenes generadas a las obtenidas en un examen real siempre alcanzó niveles aceptables para los autores, aunque las soluciones implementadas como expresión directa del modelo físico subyacente a un examen de ultrasonido (o una versión simplificada del mismo) obtienen, evidentemente, los mejores resultados bajo este criterio. Sin embargo, dado que la “calidad” de la imagen siempre alcanzó un nivel suficiente, el foco de comparación entre las distintas publicaciones fueron los tiempos requeridos para la obtención de una imagen sintética y, ya en aspectos más propios de la ingeniería de software, la facilidad de uso de la herramienta y su flexibilidad para dar soporte a los requerimientos funcionales del sistema a implementar - los que fueron insinuándose en este capítulo y serán formalizados en el siguiente.

Los desarrollos revisados que se pueden categorizar como técnicas de generación de texturas gráficas ofrecen rendimientos computacionales muy buenos, cercanos a lo que hasta este punto se ha denominado como “tiempo real”. No obstante aquello, su dificultad de uso como herramienta de desarrollo es relativamente alta, tanto por la complejidad propia de hacer programación cercana al hardware (en este caso, a la GPU) como por la lejanía entre este paradigma y el dominio teórico del problema a solucionar, la simulación de un examen médico. Un desarrollo basado en este tipo de técnica requeriría de varios niveles de abstracción.

Finalmente, investigando sobre herramientas basadas en la técnica de *ray tracing* (un paradigma de computación gráfica que se acerca más, por su definición, al fenómeno físico del ultrasonido) se descubrió la biblioteca PLUS, que entre su amplia gama de herramientas y opciones ofrece la posibilidad de generar imágenes de ultrasonido sintéticas, utilizando una implementación basada en *ray tracing*. Su diseño y arquitectura aparentan ofrecer un buen punto de partida para el desarrollo (tanto por alguien más cercano al dominio médico que al de desarrollo de software o viceversa) y soporte para dar cumplimiento a los requerimientos funcionales que en este punto del trabajo se avizoran, en cuanto a capacidad de configuración de los parámetros de una simulación. Por último, aunque los resultados experimentales mostrados en [23] no son entregados con todo el detalle técnico que permitiría una evaluación concreta de su capacidad para cumplir el requerimiento de rendimiento computacional

CAPÍTULO 2. ESTADO DEL ARTE

(tiempo de ejecución requerido para obtener una imagen) - por ejemplo, no indican las características de los equipos en los que se ejecutaron las simulaciones -, los valores que sí se indican muestran que el rendimiento se encuentra en el rango de valores donde podría encontrarse la meta a definir con precisión en el capítulo que viene a continuación en este informe.

Capítulo 3

Definición del trabajo de desarrollo

En el capítulo anterior se fueron preconfigurando los requerimientos específicos del sistema que se busca implementar. Estos vienen determinados por el objetivo general de este trabajo, las características de los materiales disponibles a partir de los trabajos que sirven de antecedente directo a este desarrollo y de las herramientas disponibles para la implementación - en particular, la biblioteca PLUS, que es la que mejor se acomoda al objetivo general de las estudiadas en el estado del arte de este trabajo.

Es conveniente recapitular el objetivo principal de este trabajo de desarrollo: la implementación de un sistema de sintetización de imágenes de ultrasonido que pueda utilizarse en un simulador de exámenes ecográficos. En la siguiente sección se irán extrayendo los requerimientos funcionales y las métricas de medición asociadas a ellos, a partir de este objetivo. La primera parte de esta declaración, “implementar un sistema de sintetización de imágenes de ultrasonido”, por sí sola, resulta sumamente amplio. En efecto, como se vio en el estado del arte, existen al menos dos grandes aspectos en los cuales enfocarse para evaluar una aplicación en este sentido (calidad de la imagen generada y tiempo de ejecución requerido para sintetizar una imagen). Los requerimientos específicos para este trabajo de desarrollo vienen, por tanto, dados por la segunda parte del objetivo general: “que pueda utilizarse en un simulador de exámenes ecográficos” - o, en el caso inicial de esta memoria, específicamente de exámenes ecocardiográficos.

3.1. Requerimientos funcionales

La condición de “poder utilizarse en un simulador de exámenes” constituye el gran requerimiento funcional de este trabajo. Es conveniente desagregarlo en requerimientos más específicos, para poder determinar el cumplimiento de los mismos de manera más sencilla.

El primer requerimiento, esbozado en el estado del arte, es la posibilidad de generar imágenes sintéticas solo a partir de los datos de entrada de cada uso; esto es, sin recurrir a “grabaciones” o, en general, a los resultados de simulaciones anteriores. Por la naturaleza de este requerimiento, este se resuelve en la fase de diseño de la aplicación final y, de hecho, ya se sabe que PLUS ofrece esta opción. Por tanto, la elección de esta herramienta para la implementación del sintetizador resuelve este requisito.

Los siguientes requerimientos vienen dados por la experiencia de realizar un examen de ultrasonido. Esto implica, para efectos de este trabajo de desarrollo y desde el punto de vista funcional:

- Responder a las acciones del usuario.
- Permitir la variación de los mismos parámetros de configuración que un examen real, en el momento respectivo de la realización del examen.

El primer punto viene del aspecto mecánico de un examen ecográfico: el examinador coloca un componente de hardware (conocido como sensor, sonda o *probe*) sobre el área del paciente que desea auscultar, desde el cual se emite la señal de ultrasonido y se capta el eco de la misma, el que luego es procesado por la máquina de ultrasonido para generar la imagen presentada. Luego, el sistema debe ser capaz de recibir datos de algún componente físico que emule esta mecánica de examinación y asociarlos al caso específico que se esté simulando, de manera que la imagen finalmente generada refleje los movimientos del usuario.

La segunda condición nace de las características de los equipos utilizados para realizar ecografías, los cuales permiten ajustar algunos parámetros de la señal de ultrasonido durante la realización de un examen para variar los elementos del área examinada que se resaltan en la imagen generada. En abstracto, las características que pueden modificarse dependen del diseño de cada equipo de auscultación, pero en general se requiere control sobre las tres características fundamentales mencionadas en el estado del arte, al definir un examen ecográfico: *foco*, *contraste* y *frecuencia* de la señal de ultrasonido.

Se tienen, entonces, tres requerimientos funcionales:

1. Generar simulaciones solo a partir de datos de entrada, sin recurrir a simulaciones anteriores.

CAPÍTULO 3. DEFINICIÓN DEL TRABAJO DE DESARROLLO

2. Recibir un dato de entrada, llamado “de posicionamiento”, desde un dispositivo físico que emule la mecánica de la realización de un examen ecográfico.
3. Permitir la variación de los parámetros de configuración de la simulación que se relacionen con las características de foco, contraste y frecuencia de un examen de ultrasonido.

Finalmente, y para poder observar los resultados del sintetizador de imágenes, se debe ofrecer una interfaz gráfica que permita observar la imagen generada y realizar las variaciones de los parámetros cubiertos por el tercer requerimiento funcional antes mencionado.

3.2. Requerimientos no funcionales

El mismo objetivo de generar una herramienta que pueda utilizarse en un simulador de exámenes ecográficos plantea una limitación de rendimiento al sintetizador de imágenes: las imágenes generadas deben presentarse en “tiempo real”. Esto significa que no debe existir retraso entre una acción del usuario y la sintetización de la imagen respectiva - en estricto rigor, que el retardo que pueda haber entre la acción y el resultado no sea mayor al que se observa en un examen real.

Esta restricción se debe considerar en dos puntos: las latencias que pueda presentar el hardware que se utilice para emular el aspecto mecánico de un examen de ultrasonido y el tiempo que toma generar cada imagen sintetizada. El primer aspecto escapa del ámbito de control de este trabajo de desarrollo, toda vez que el equipo que se utilizará para generar los datos de posicionamiento son un insumo entregado y no existe la posibilidad de cambiarlo (por temas de costos). Por tanto, el foco estará en el rendimiento computacional del sistema de simulación.

La restricción de rendimiento puede expresarse, numéricamente, en la necesidad de generar al menos 25 imágenes por segundo[11] o, lo que es equivalente, de utilizar hasta 40 milisegundos de tiempo de ejecución para sintetizar una imagen de ultrasonido. Estos valores se convierten, así, en las metas a alcanzar en las mediciones del rendimiento del sistema.

3.3. Materiales de trabajo

Para la implementación del sintetizador de imágenes se contó con dos insumos:

- Un dispositivo de hardware que cumple la función de emular el aspecto operacional de un examen de ultrasonido (en particular, un magnetómetro¹).

¹Dispositivo que mide la potencia de campos magnéticos.

- Un modelo tridimensional de un corazón latiendo, para utilizarse como dato de entrada para las simulaciones.

3.3.1. Magnetómetro trakSTAR™

El dispositivo trakSTAR™ es un magnetómetro que se utiliza para realizar seguimientos espaciales (o sea, para determinar la posición de un punto en el espacio), desarrollado para su uso en sistemas de simulación y entrenamiento médico. Fabricado por la compañía Ascension Technology Corporation², permite realizar seguimiento con 6 grados de libertad - esto es, detecta movimientos de traslación y rotación en cada uno de los tres ejes de un espacio tridimensional euclideo.

De acuerdo al fabricante, el dispositivo provee “tecnología de seguimiento de posición y orientación económica y de alta precisión. Cada movimiento dentro del volumen de seguimiento, sin importar cuán sutil o preciso, es rastreado en tiempo real con precisión excepcional. Esta exactitud, combinada con una baja latencia y altas tasas de actualización, permite que los datos de posición y orientación del sensor se integren fluidamente con la simulación desplegada, para producir una experiencia de entrenamiento interactiva y muy cercana a la vida real” (traducido desde el original³, en inglés).

El magnetómetro consta de un dispositivo controlador (figura 3.1), un transmisor que genera un campo electromagnético y al menos un sensor - el disponible para este trabajo corresponde al modelo de 8 [mm] (figura 3.2). La precisión de las mediciones realizadas es de 1,4 [mm] y 0,5 grados (ambos valores, medias cuadráticas). La tasa de actualización de mediciones es variable por el usuario, en un rango entre 20 y 255 [Hz], con un valor por defecto de 80 Hz. La conexión a un computador se puede realizar mediante un puerto serial (RS-232) o USB (este último fue el usado en este trabajo de desarrollo).

²<http://www.ascension-tech.com/>

³<http://www.ascension-tech.com/products/trakstar-drivebay/>



Figura 3.1: Dispositivo controlador del magnetómetro trakSTAR™.

Fuente: Sitio Web del fabricante.



Figura 3.2: Sensor (*probe*) del magnetómetro trakSTAR™.

Fuente: Sitio Web del fabricante.

3.3.2. Modelo tridimensional

Para representar el corazón se adquirió el modelo tridimensional conocido como “The Zygote 3D Human Heart Model”, el que fue desarrollado a partir de datos obtenidos mediante exámenes de resonancia magnética (MRI) y tomografías computarizadas (CT).

De acuerdo a la descripción del publicador⁴, este modelo 3D “mantiene fielmente la forma del corazón humano, con válvulas y vasos sanguíneos en correcta proporción y posición”, mostrando un gran detalle en sus estructuras internas (válvulas, aurículas, músculos papilares, etc.) a la vez que mantiene una geometría eficiente. Las superficies interiores y exteriores están casi completamente

⁴http://www.3dscience.com/3D_Models/Human_Anatomy/Heart/Heart.3.php

CAPÍTULO 3. DEFINICIÓN DEL TRABAJO DE DESARROLLO

compuestas por cuadros (*quads*).

El modelo completo incluye, además de la información geométrica inicial, texturas (que no fueron utilizadas en este trabajo) y la animación de un ciclo cardíaco humano completo, contenida en 25 modelos estáticos independientes que representan el estado del corazón en un punto determinado de dicho ciclo. “Con los cortes lateral, anterior y superior del modelo 3D, cada válvula y cámara es revelada completamente en el contexto del ciclo cardíaco, en tiempo real”.

En la figura 3.3 se muestra, en vista de malla, uno de los modelos tridimensionales que componen la animación.

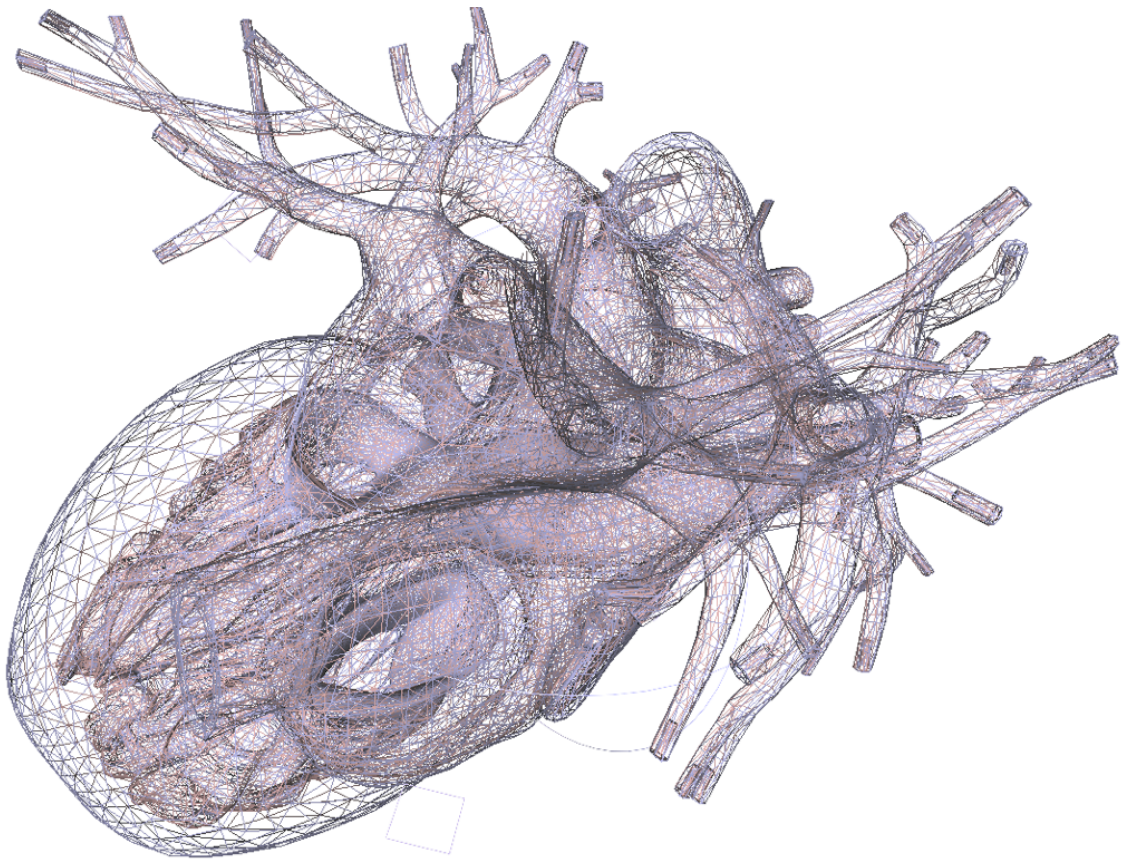


Figura 3.3: Malla del modelo tridimensional de corazón.

Fuente: Sitio Web del fabricante.

3.4. Herramientas de desarrollo

Este trabajo de desarrollo se efectuó en un computador con las siguientes características:

- **Procesador:** Intel[®] Core[™] i7 M640 (doble núcleo, frecuencia de reloj 2,80 GHz).
- **Tarjeta gráfica:** NVIDIA[®] NVS 3100M.
- **Memoria RAM:** 8GB, DDR3.

El sistema operativo en el que la aplicación final funciona es Microsoft Windows. El entorno de desarrollo utilizado fue Microsoft Visual Studio 2013.

Las dos grandes bibliotecas utilizadas durante este trabajo son el *toolkit* Qt, para la interfaz gráfica, y PLUS. Qt es una herramienta de software bastante conocida y con una amplísima documentación disponible libremente en la Internet; además, la interfaz gráfica que se incluye en este desarrollo solo fue implementada por la necesidad de mostrar y evaluar las capacidades del sintetizador de imágenes. Por ello, no se explicará en detalle esta biblioteca.

3.4.1. PLUS

La biblioteca PLUS es distribuida, tanto en forma de código fuente como de binarios compilados – estos últimos, a su vez, tanto ejecutables como bibliotecas dinámicas –, bajo una licencia tipo BSD, la que otorga un amplio margen de acción para su descarga, uso, modificación y distribución. En su presentación original, PLUS entrega una aplicación que cumple la función de “servidor”, recibiendo requerimientos y entregando las imágenes generadas a partir de estos. La configuración inicial del servidor se realiza mediante un archivo XML, cuya estructura se explicará en informe.

PLUS es una pieza de software compleja, dependiente de otras bibliotecas. Es importante mencionar una en particular, porque será nombrada recurrentemente durante este informe: OpenIGTLink, que corresponde a un protocolo de comunicación entre aplicaciones y dispositivos desarrollado especialmente para aplicaciones médicas - de hecho, la sigla IGT viene del inglés “*image-guided therapy*” (terapia guiada por imágenes). En los apéndices de este informe se presenta un detalle del funcionamiento de OpenIGTLink.

Un primer análisis de la documentación[26], de los archivos de configuración asociados a la batería de ejemplos que se distribuyen junto a PLUS y la ejecución de estos últimos, permite comenzar a especificar el modo de funcionamiento de la biblioteca en, al menos, el caso de uso de interés en este desarrollo – entiéndase, la generación de una simulación de imagen de ultrasonido, a partir de

CAPÍTULO 3. DEFINICIÓN DEL TRABAJO DE DESARROLLO

una “escena” (compuesta de al menos un objeto 3D) e información de posición y orientación de un sustituto de la sonda de ecografía. Entre los casos de ejemplos incluidos en la biblioteca, existe uno que se asemeja bastante al escenario final de la aplicación a implementar, llamado “*PlusServer: Vascular needle insertion simulation*”, cuyo flujo de ejecución se puede resumir en los siguientes pasos:

- Se ejecuta una instancia del servidor PLUS, configurada a partir de los parámetros contenidos en el archivo XML seleccionado (el asociado al ejemplo elegido).
- Se reciben los primeros datos de posicionamiento - que, en el caso de este ejemplo, no solo corresponden al sensor del equipo de ultrasonido, sino que también a los distintos objetos que componen la escena (existe una entrada de datos por cada uno de estos elementos). Estos datos pueden provenir desde la interfaz usuaria o desde un dispositivo físico soportado por PLUS.
- El servidor genera una primera versión de la imagen sintetizada, a partir de las condiciones iniciales contenidas en el archivo XML y las posiciones de los objetos de la escena, y transmite el resultado de la misma a través del protocolo OpenIGTLink.
- Si existe alguna modificación en los datos de posicionamiento de los objetos involucrados en la simulación, el servidor PLUS genera una imagen actualizada y emite la misma a través de OpenIGTLink.

En el caso de que los datos de posicionamiento provengan de la interfaz usuaria, estos son recibidos por PLUS a través de OpenIGTLink. La documentación de PLUS revela que también existe soporte para la administración remota del servidor mediante “comandos”, formados como comandos XML y transmitidos a través de OpenIGTLink. La figura 3.4 muestra un esquema general de la ejecución de una instancia de PLUS.

La arquitectura de PLUS, en cuanto aplicación, permite su uso relativamente directo, toda vez que cualquier otro programa que quiera utilizarlo debe implementarse bajo la lógica de comunicación con un servidor, obviando la necesidad de gestionar los detalles técnicos internos de PLUS. Además, entrega flexibilidad a la hora de diseñar e implementar el sistema final: por ejemplo, permite ejecutar PLUS en una máquina y la interfaz usuaria en otra, o utilizar una única instancia de PLUS para entregar imágenes a varios clientes-terminales.

Los datos de configuración del algoritmo de simulación de ultrasonido se pueden agrupar según dos criterios: el tiempo en que pueden ser entregados y la información que representan. Con respecto al primer criterio, los datos pueden ser entregados al ejecutar una instancia del servidor PLUS mediante

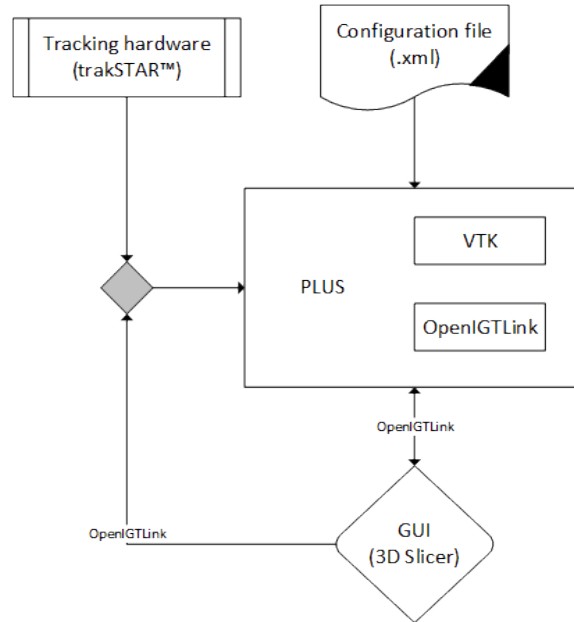


Figura 3.4: Diagrama general de funcionamiento de PLUS.

Fuente: Elaboración propia.

el archivo de configuración XML o pueden provenir, durante la ejecución de la instancia de PLUS, de alguna fuente externa (interfaz usuaria o dispositivo de hardware). Con respecto al segundo criterio, los datos de entrada pueden corresponder a:

- Información de configuración de la fuente de datos espaciales: parámetros para el establecimiento de la conexión mediante OpenIGTLink o para la comunicación con algún dispositivo físico.
- Configuraciones generales de PLUS: datos para la difusión de las imágenes generadas, sistemas de coordenadas para los objetos involucrados en la simulación.
- Parámetros físicos y de configuración general para el algoritmo de simulación de ultrasonido.
- Parámetros geométricos y físicos de los objetos 3D que forman parte de la escena sobre la que trabaja el algoritmo de ultrasonido (estos se discutirán en detalle en la sección “Descripción de modificaciones de PLUS e implementación de aplicaciones”).
- Datos de localización espacial de los objetos 3D que forman parte de la escena.

Los parámetros físicos y de configuración general, soportados por PLUS, son:

CAPÍTULO 3. DEFINICIÓN DEL TRABAJO DE DESARROLLO

- Frecuencia de la señal de ultrasonido simulada, en [MHz].
- Intensidad de la señal de ultrasonido simulada, en [mW/cm²].
- Factor de corrección *gamma*, escala y *offset* para el cálculo de intensidad de brillo, donde el brillo de cada píxel es una función de la señal de ultrasonido reflejada.
- Número de líneas de muestreo – a mayor cantidad, mejor definición tendrá la imagen sintetizada.
- Número de muestras por línea – a mayor cantidad, mejor definición tendrá la imagen sintetizada.
- Parámetros de configuración de la simulación de ruido en la imagen sintetizada.
- Datos geométricos (archivo modelo 3D) y físicos de cada uno de los objetos 3D que componen la escena sobre la cual se genera la imagen de ultrasonido simulada:
 - Densidad (en [kg/m³]).
 - Velocidad del sonido en el medio, en [m/s].
 - Coeficiente de atenuación, en [dB/cm*MHz].
 - Coeficiente de reflexión de dispersión de retorno o retrodispersión (backscatter) (adimensional).
 - Decaimiento de intensidad de reflexión de superficie, en [dB/mm], en el caso de objetos que interrumpen la reflexión de otros (ej.: aire, que actúa de “medio” contenedor de todos los demás objetos).
 - Coeficiente de reflexión especular de superficie (adimensional).
 - Coeficiente de reflexión difusa de superficie (adimensional).

En este punto, es conveniente dar una breve explicación del cálculo del brillo: las imágenes de ultrasonido, en general - y, en particular, las generadas por PLUS -, están en escala de grises, por lo que el brillo de cada píxel corresponde a su “color”. Esto suele expresarse, computacionalmente, como un número entero almacenado en un *byte* sin signo, donde el valor mínimo (0) corresponde al negro absoluto y el máximo (255), al blanco total. Expresado matemáticamente, PLUS determina el brillo de cada píxel de una imagen sintetizada con la siguiente expresión:

$$b_{pixel} = \max(\min(B + SI^\gamma + N, 255), 0)$$

CAPÍTULO 3. DEFINICIÓN DEL TRABAJO DE DESARROLLO

Con:

- b_{pixel} : el brillo finalmente asignado a un píxel determinado.
- B : brillo mínimo asignado a todos los píxeles de la imagen sintetizada. Actúa como *offset* del valor final de b_{pixel} .
- S : factor escalar que amplifica o disminuye el brillo determinado para cada píxel por el algoritmo de simulación.
- I : intensidad de la señal reflejada de ultrasonido correspondiente al píxel. Esto es, simplificando, el resultado numérico directo del algoritmo de simulación de ultrasonido.
- γ : factor de corrección *gamma* del brillo. Este es un concepto utilizado en el tratamiento de imágenes.
- N : valor entregado por la función de simulación del ruido asociado a un examen ecográfico. Este parámetro se discutirá más adelante, en el capítulo de validación de resultados.

Como se puede ver, los parámetros asociados al brillo (factor *gamma*, escala y *offset*) corresponden a características computacionales del algoritmo, no físicas. En un examen real, serían propias de la configuración del equipo de ultrasonido, determinando las características del procesamiento que convierte la señal reflejada de ultrasonido en la imagen presentada al examinador.

La relación entre estas fuentes de datos y los tiempos en que son entregados a PLUS se reflejan en la figura 3.5.

Finalmente, cabe destacar que los datos de posicionamiento pueden ser obtenidos solo desde una fuente indicada al momento de ejecutar el servidor PLUS: la interfaz usuaria (mediante el protocolo OpenIGTLink) o algún dispositivo físico soportado por PLUS (entre los que se encuentra el tipo de magnetómetro disponible para este trabajo de desarrollo).

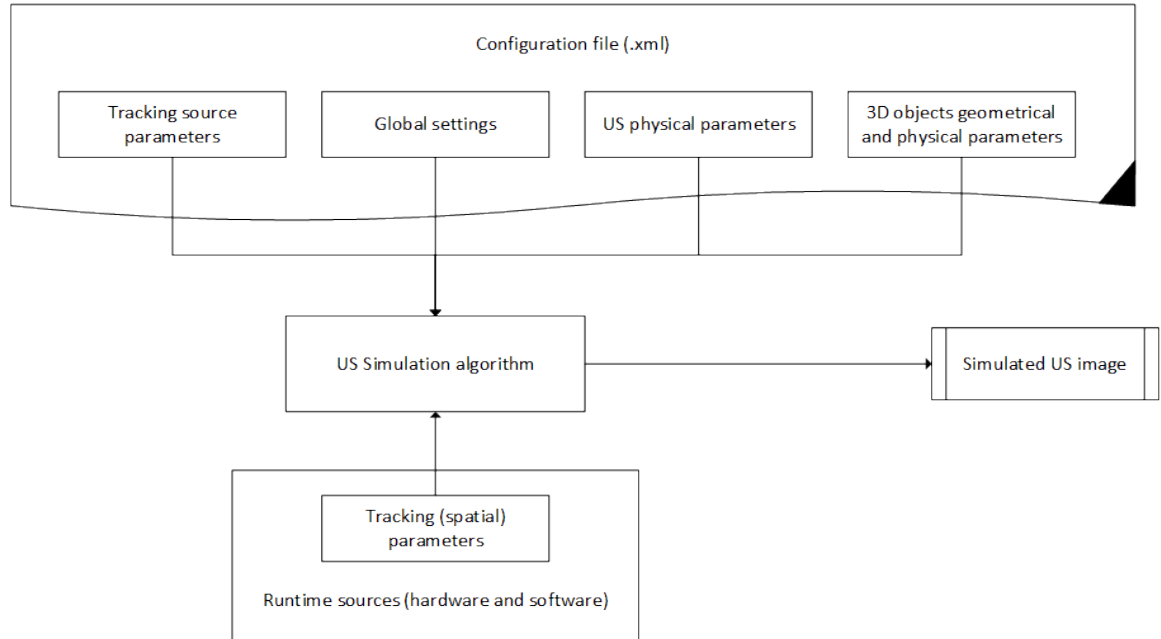


Figura 3.5: Fuentes de datos para el algoritmo de simulación de ultrasonido.

Fuente: Elaboración propia.

3.5. Conclusiones

En este punto del trabajo, lo central es determinar las capacidades que posee PLUS en su implementación original, con respecto a las necesidades que derivan de los requerimientos funcionales determinados para este desarrollo. Así, las características importantes encontradas en PLUS son las siguientes dos:

- Una instancia del servidor PLUS se ejecuta, utilizando como entrada una batería de parámetros entregados a través de un archivo de configuración XML asociado al caso que se desea simular.
- El servidor PLUS recibe datos de posicionamiento de las entidades relacionadas a la simulación, entre ellas la posición del sensor de ultrasonido con respecto a los demás objetos (lo que se conoce como el “observador”), desde una única fuente de datos.

De la primera característica se desprenden, a su vez, dos elementos a considerar:

- Un tipo de elementos cuya configuración inicial se puede considerar “inmutable” durante una simulación corresponde a los objetos tridimensionales que la componen - por ejemplo, los órganos a observar. Sin embargo, para uno de los parámetros a considerar, el modelo 3D propiamente

CAPÍTULO 3. DEFINICIÓN DEL TRABAJO DE DESARROLLO

tal, PLUS solo soporta formatos de archivo que no permiten definir modelos “animados”, cuya geometría varíe en el tiempo.

- Los parámetros de configuración general de la simulación contienen los valores a modificar para dar cumplimiento al tercer requerimiento funcional, pero para cumplir plenamente este requisito los parámetros deben poder modificarse en cualquier momento de la simulación - así como se pueden variar durante la realización de un examen real. Por tanto, no es suficiente que puedan configurarse al ejecutar el servidor PLUS.

Sobre la segunda característica indicada al comienzo de esta sección se puede realizar otra observación: dadas las condiciones materiales en que se encuentra el magnetómetro trakSTAR™ disponible, este solo puede proveer los datos de posicionamiento del observador, no de las demás entidades pertenecientes a la escena simulada. Además, el mismo dispositivo posee un límite superior a los canales de datos que puede entregar - en este caso particular, el máximo es cuatro -, lo que incluso en una situación ideal limitaría los elementos que pueden participar de una simulación. Por último, varios de los objetos tridimensionales en una escena debieran variar de posición poco o, incluso, nunca - hay que considerar que lo normal es que un paciente no se mueva durante un examen ecográfico. Esto volvería innecesario obtener los datos de posicionamiento de estos objetos de forma permanente desde el magnetómetro, pero la arquitectura actual de PLUS no permite entregar esa información desde una fuente distinta (por ejemplo, la interfaz gráfica a través de OpenIGTLink) al magnetómetro, que es de donde se obtendrán siempre los datos de posicionamiento del observador.

Por todo lo anterior, se puede determinar la necesidad de realizar las siguientes adiciones y modificaciones a PLUS, para dar cumplimiento a los requerimientos de este trabajo de desarrollo:

- Agregar soporte para que objetos 3D “animados” formen parte de la escena sobre la que se ejecuta la simulación de ultrasonido.
- Variación de los puntos en que puede asignarse un valor a los parámetros físicos del algoritmo de simulación, para poder cambiarlos en tiempo de ejecución.
- Añadir soporte para la recepción de datos de posicionamiento desde múltiples fuentes, en forma simultánea.

El análisis que permitió caracterizar PLUS y determinar las modificaciones a realizar se documenta de manera más extensa y técnica en los apéndices de este informe.

Capítulo 4

Desarrollo de la aplicación

El capítulo anterior de este informe terminó con la determinación de los cambios que habría que realizar a la biblioteca PLUS para poder dar cumplimiento a los requerimientos funcionales de la aplicación a implementar. El trabajo de desarrollo consistió, en líneas generales, en realizar estas modificaciones y en programar la interfaz gráfica a utilizar para poder observar los resultados de la ejecución de PLUS y para que el usuario pudiera variar los parámetros que se le permitieran al dar cumplimiento al tercer requerimiento funcional de este trabajo.

Las modificaciones a realizar a PLUS se relacionan principalmente con las fuentes de datos utilizadas desde donde una instancia del servidor PLUS obtiene la información utilizada en una simulación. Considerando el diagrama mostrado en la figura 3.5, se pueden graficar los cambios que deben realizarse; este nuevo diagrama está en la figura 4.1, en la cual se destacan las adiciones realizadas a las fuentes de datos soportadas en color rojo, mientras que las fuentes ya existentes que son alteradas en su implementación son destacadas en azul.

Las alteraciones realizadas a PLUS, así como la implementación de la interfaz gráfica, fueron realizadas mediante un proceso de desarrollo incremental e iterativo, donde la consecución de un hito de desarrollo fue seguida de una fase de pruebas que permitían determinar mejoras a lo ya realizado y establecer la base para la siguiente etapa de implementación.

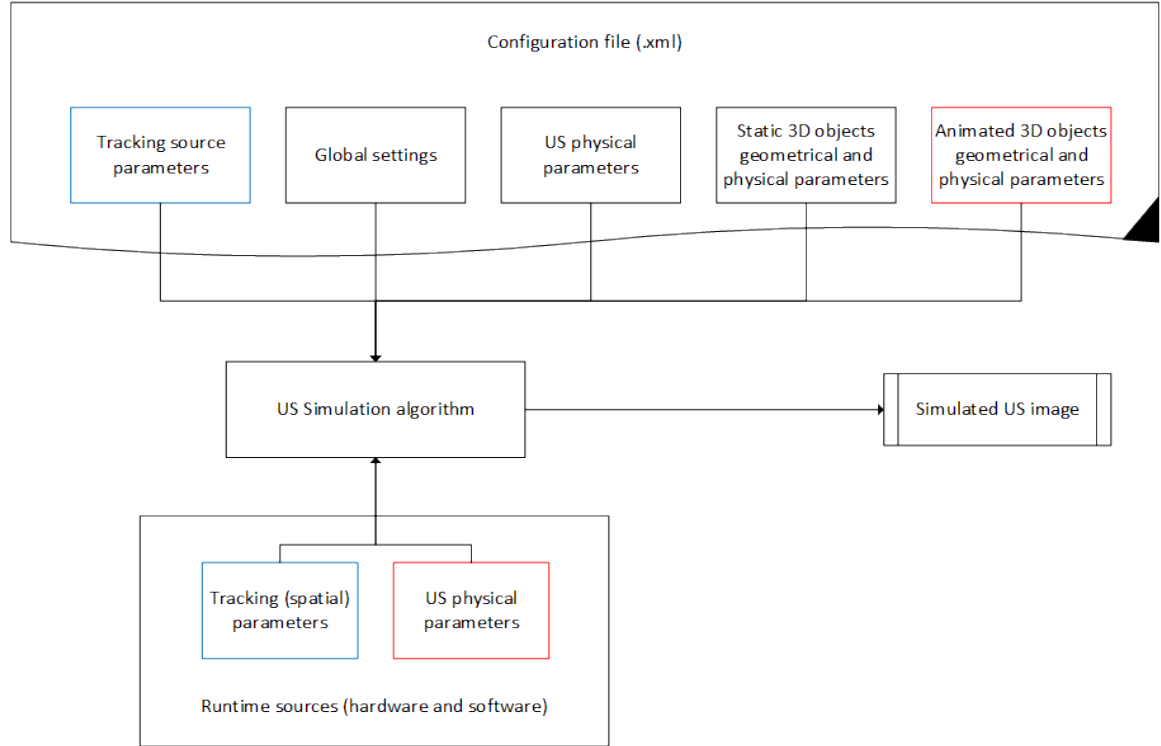


Figura 4.1: Fuentes de datos para el algoritmo de simulación de ultrasonido, con PLUS modificado.
Fuente: Elaboración propia.

4.1. Modificaciones a PLUS

Las modificaciones realizadas a la biblioteca PLUS se presentan a continuación, en el orden en que fueron implementadas.

4.1.1. Modelos 3D animados

La implementación original de PLUS permite realizar simulaciones de ultrasonido a partir de escenas compuestas de múltiples objetos tridimensionales. Sin embargo, estos objetos pueden estar definidos geométricamente solo por modelos 3D sin semántica temporal; esto es, por modelos “inanimados” que, por definición, no contemplan variaciones en su geometría en un espacio determinado de tiempo. Como el caso de uso que inspira este trabajo de desarrollo es la simulación de un examen ecocardiográfico, esta limitación debe eliminarse: el órgano de interés es el corazón, el que por su esencia tiene variaciones temporales, pues late - este efecto, de hecho, está considerado en el modelo tridimensional recibido como material de trabajo, al inicio de este desarrollo.

CAPÍTULO 4. DESARROLLO DE LA APLICACIÓN

El objetivo central al realizar esta adaptación en PLUS es hacerla de manera tal que se aproveche la funcionalidad ya existente y minimizar los cambios internos a la biblioteca que deben realizarse para dar soporte a la nueva funcionalidad. Para ello, primero debe entenderse la secuencia que inicia cuando se indica a PLUS qué archivo de modelo 3D utilizar hasta que los datos contenidos en este son utilizados en la sintetización de una imagen de ultrasonido.

El punto inicial de la secuencia antes mencionada es el archivo de configuración XML que contiene la información de la simulación a ejecutar. El diagrama en la figura 4.2 muestra la dirección del flujo de datos, desde este archivo hasta las instancias de los modelos espaciales que componen una escena dada. La estructura del archivo XML refleja esta misma lógica: el nodo de un nivel contiene, además de los datos propios de su configuración, los nodos de los elementos subsiguientes - por ejemplo, la configuración de la fuente de imágenes generadas por el algoritmo de simulación de ultrasonido contiene los datos de configuración de este último.

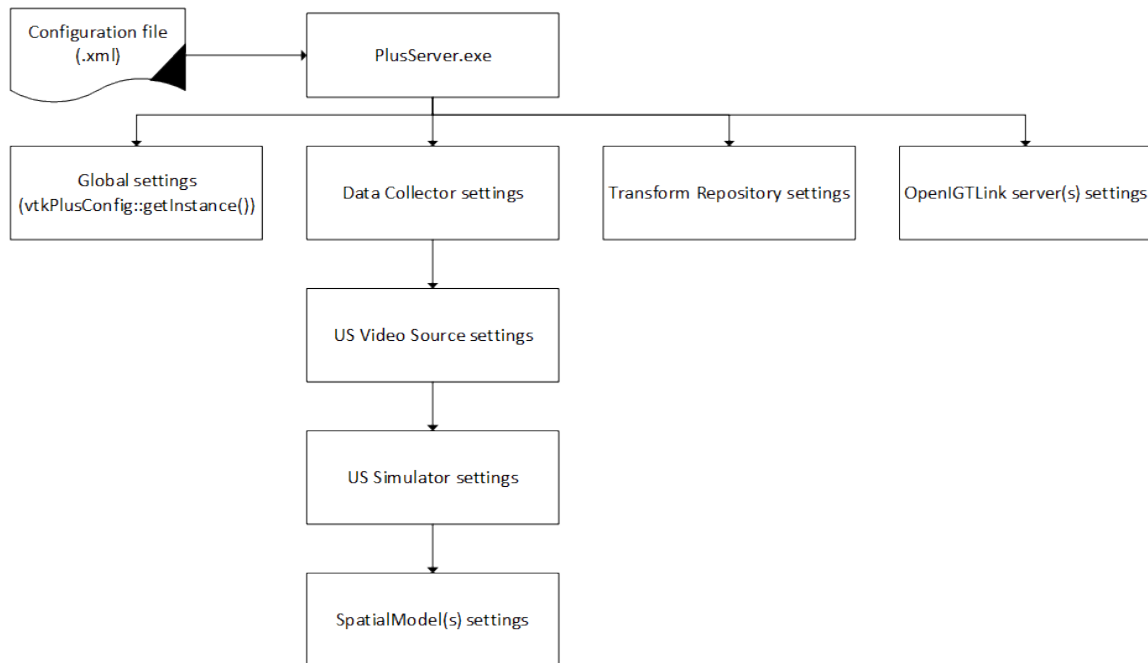


Figura 4.2: Flujo de datos de configuración dentro del servidor PLUS.

Fuente: Elaboración propia.

En el caso particular de la información geométrica de los objetos de una escena, estos llegan hasta un objeto de la clase `SpatialModel`, dentro de PLUS. Esta clase contiene un arreglo con la información geométrica propiamente tal y otro con información derivada de los datos geométricos.

CAPÍTULO 4. DESARROLLO DE LA APLICACIÓN

Como se mencionó en la sección 3.3.2 de este informe, el modelo “animado” disponible para este desarrollo es, en realidad, un conjunto de 25 modelos estáticos. Por tanto, se decidió agregar el soporte para modelos animados a través de una clase que herede de `SpatialModel` y que, internamente, contenga tantos arreglos con datos geométricos y derivados como archivos compongan la “animación”. Así, para el algoritmo de simulación, un objeto tridimensional estático y uno animado debieran ser indistinguibles, pues la generación de una imagen se realiza con el modelo que corresponda al instante de tiempo en que se encuentre la simulación. Dado que la memoria utilizada por la aplicación no es limitante, esta solución, además de minimizar los cambios que deben realizarse a las partes del código de PLUS que utilizan los objetos de una escena simulada, reduce los costos computacionales al guardar los datos derivados de la información geométrica de cada uno de los modelos 3D que componen un objeto animado.

La segunda alteración que debió realizarse en PLUS es en la lectura del archivo de configuración XML, para poder incorporar ahí la información relativa al objeto animado. En particular, se agregó el soporte a un nuevo tipo de nodo XML que contenga:

- Una expresión regular que describa el nombre de los archivos con los modelos 3D que componen el objeto animado.
- El número de modelos 3D que componen el objeto.
- Un nodo con la información que describe el objeto, usando el mismo formato que los nodos de los objetos estáticos.

La tercera parte de la lógica de PLUS que fue cambiada es el manejo de la semántica de tiempo. Inicialmente, se pensó en agregar una hebra de ejecución, administrada por el objeto de la clase implementada para manejar objetos animados, que cambiara el modelo 3D expuesto al resto del sistema en un momento dado. Sin embargo, además de aumentar la carga computacional, esta solución habría impuesto exigencias de sincronización - por lo menos, habría que evitar que se cambiaran los datos durante la sintetización de una imagen nueva. Por ello, se consideró una segunda posibilidad de diseño: existe, entre los parámetros de configuración de la simulación contenidos en el archivo XML, uno que corresponde a la tasa de actualización de la imagen sintetizada (llamada “tasa de adquisición”), expresada en [imagen/s]. Se tomó como supuesto que esta tasa es igual a la velocidad de animación (en igual unidad de medida) de los objetos tridimensionales animados. Esta decisión obedece a dos criterios:

CAPÍTULO 4. DESARROLLO DE LA APLICACIÓN

- La equiparación entre la tasa de refresco del algoritmo de simulación y la cantidad de frames por segundo que contiene un objeto animado permite utilizar la misma hebra de ejecución para administrar el avance del tiempo que ya existe para la instancia del algoritmo. En otras palabras, no debe instanciarse una nueva hebra de ejecución que sirva de “temporizador” para cada objeto tridimensional.
- Como consecuencia del criterio anterior, la animación no presenta discontinuidades entre las distintas imágenes sintetizadas (“saltos” en la animación desplegada al usuario).

En atención a ambos criterios, se agregó una porción de código, dentro del *loop* de ejecución del algoritmo de simulación, que indica a todos los objetos animados que deben avanzar a los datos del siguiente modelo de la animación.

Como última modificación dentro de PLUS, se alteró el tratamiento de la generación del ruido de la imagen de ultrasonido. Esta parte de la simulación se establece una única vez para todo el algoritmo, a partir de datos contenidos en el archivo de configuración XML. Para crear la sensación de animación para el ruido calculado, los valores de inicialización de la función de ruido (en particular, la fase) se multiplican por un valor aleatorio entre 0 y 1 calculado al momento de cambiar de modelo utilizado por el objeto tridimensional animado, solo en caso de que el algoritmo de simulación contenga al menos un objeto de este tipo. Esta variación permite generar la sensación de animación en todos los casos; en PLUS, el ruido es generado como una función de los datos de posicionamiento del observador, por lo que variar esta información resulta en cambios en el ruido simulado. Sin embargo, en un examen real el ruido obedece en mucho mayor medida a las características del medio por el que se propaga la señal (en este caso, el cuerpo humano), las que no son constantes. Por tanto, se requiere que el ruido no varíe solo en función del observador.

4.1.2. Variación de parámetros de configuración en tiempo de ejecución

Para dar cumplimiento al tercer requerimiento funcional determinado en 3.1, se debió añadir la posibilidad de que ciertos parámetros, configurados originalmente solo al iniciar una simulación, puedan modificarse durante la ejecución de la misma. En el capítulo anterior, en la sección 3.4.1, se detallan los parámetros que pueden pasarse a PLUS a través del archivo de configuración XML asociado al caso a simular. En particular, son de interés los que pueden llamarse “parámetros globales”; esto es, que determinan características propias del caso simulado y no de los objetos particulares que lo componen. Estos son:

CAPÍTULO 4. DESARROLLO DE LA APLICACIÓN

- Frecuencia de la señal de ultrasonido simulada.
- Intensidad de la señal de ultrasonido simulada.
- Factor de corrección *gamma*, escala y *offset* para el cálculo de intensidad de brillo.
- Número de líneas de muestreo.
- Número de muestras por línea.
- Parámetros de configuración de la simulación de ruido en la imagen sintetizada.

A su vez, cuando se definieron las características de un examen ecográfico, se determinaron tres aspectos fundamentales que describen dichos procedimientos: el *foco* de la señal de ultrasonido - que determina la resolución de la imagen generada -, el *contraste* de la imagen obtenida y la *frecuencia* de la señal acústica, que determina qué tejidos destacan por sobre otros en la imagen producida por el equipo de ultrasonido. Para proceder con las modificaciones que deben realizarse a PLUS, se debió identificar la relación entre los parámetros de configuración y la o las propiedades a las que afectan:

- La *frecuencia* de la señal se encuentra directamente expresada en un parámetro de configuración.
- La *intensidad* de la señal afecta el contraste de la imagen generada. El efecto es ejemplificado más en la validación de la aplicación implementada, pero ahora puede decirse que la intensidad de la señal de ultrasonido puede asimilarse a la potencia de una ampolla. Además, sin ser parámetros directamente relacionados a un concepto físico, los valores de configuración del *brillo* también afectan, entre otros, al contraste de la imagen sintetizada.
- El número de líneas de muestreo es inversamente proporcional a la distancia entre los haces de la señal de ultrasonido (a mayor número, menor distancia). A su vez, el número de muestras por línea corresponde al número de frentes de onda emitidos. Ambos conceptos determinan el *foco* (o resolución) de la imagen de ultrasonido.

Considerando el análisis hecho hasta este punto, se determinó añadir a PLUS el soporte para variar, en tiempo de ejecución, los llamados parámetros globales de configuración que, originalmente, se determinan a través del archivo XML asociado a la simulación que se ejecuta en un uso determinado de la aplicación.

El medio que ofrece PLUS para comunicarse con una instancia del servidor es el protocolo Open-IGTLink. La aplicación mantiene dos hebras de ejecución dedicadas específicamente a dicha labor:

CAPÍTULO 4. DESARROLLO DE LA APLICACIÓN

- Una hebra para la recepción de mensajes de estado y solicitudes desde los clientes conectados al servidor PLUS.
- Una hebra para la recepción de mensajes de transformaciones aplicadas a la escena 3D sobre la que se realiza la simulación de ultrasonido.

Al momento de decidir en qué punto de PLUS realizar las modificaciones, se optó por trabajar a partir de la recepción de datos de transformaciones por los siguientes motivos:

- La instanciación de los objetos relacionados a la recepción de transformaciones y el algoritmo de ultrasonido se instancian en el mismo punto del código y, por tanto, es relativamente sencillo “conectarlos” mediante una referencia para que se puedan transmitir datos. En contraposición, la lógica de procesamiento de comandos de PLUS está ubicada en un punto completamente distinto dentro de la implementación del servidor, por lo que enviar los datos recibidos en este lugar al objeto de la clase que realiza el algoritmo de simulación de ultrasonido requeriría una modificación mucho mayor a la implementación de PLUS.
- La implementación del siguiente punto de este trabajo, la recepción de datos espaciales desde distintas fuentes simultáneamente, requiere trabajar en el mismo punto del código donde se reciben los datos de transformaciones mediante OpenIGTLink – de hecho, el objetivo final de la tercera modificación de PLUS fue la de recibir datos de posicionamiento a través de OpenIGTLink y desde el dispositivo trakSTAR™ de manera simultánea.

Inicialmente, se procedió a separar la recepción de los mensajes del procesamiento de los mismos. Para estos efectos, se implementaron dos clases: una para manejar los mensajes con información de rastreo y transformaciones y otra para que reciba y clasifique los mensajes recibidos mediante OpenIGTLink. Una instancia de esta última clase entregaría la información recibida a un objeto de la primera, cuando el mensaje recibido correspondiera a una transformación, o al algoritmo de simulación de ultrasonido, cuando el mensaje recibido fuese el nuevo valor de algún parámetro de configuración. Además, bajo este esquema, la clase encargada de procesar las transformaciones también soportaría recibirlas desde el magnetómetro, en la implementación final de la aplicación. El flujo de datos ideado originalmente se grafica en la figura 4.3. Aquí, la clase que procesaría los mensajes de posicionamiento y transformaciones se denomina “*Multisource Tracker*” para reflejar el cumplimiento del segundo requerimiento funcional, cuya implementación se discute en la sección siguiente.

Sin embargo, la implementación de PLUS entreteje de manera muy compacta la recepción y el procesamiento de datos. En efecto, fue posible implementar, a partir del código original, las clases

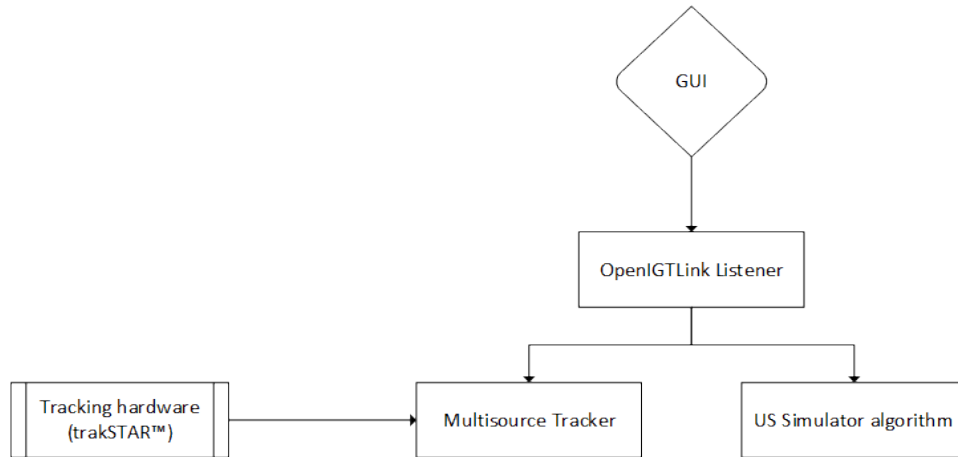


Figura 4.3: Flujo de datos propuesto inicialmente para dar soporte a variación de parámetros en tiempo de ejecución.

Fuente: Elaboración propia.

indicadas en el párrafo anterior y recibir exitosamente los mensajes que permitían cambiar los valores de los parámetros indicados en tiempo de ejecución, pero los datos de rastreo – si bien eran recibidos sin problemas en PLUS – no pudieron ser correctamente procesados por problemas de coordinación temporal entre los distintos componentes que esperaban dichos datos. Por ello, finalmente se optó por crear una clase que, a partir de una copia de la lógica original de la clase que recibe los mensajes de transformaciones, filtrase y enviase los mensajes que no fuesen de transformación al algoritmo de simulación de ultrasonido, para que este último los procesara y modificara los parámetros de manera acorde.

Además de la creación de la nueva clase antes mencionada, se debieron realizar tres modificaciones adicionales en PLUS:

- La lógica de instanciación de objetos dentro de PLUS debió ser modificada para incorporar esta nueva clase dentro de los posibles tipos de objetos a instanciar. Además, cuando se estuviese en presencia de un objeto de esta nueva clase, se debió incorporar una instrucción para que este objeto tuviera una referencia al objeto que encapsula el algoritmo de simulación de ultrasonido, para la transmisión de los mensajes respectivos.
- Existe en PLUS una clase que se encarga específicamente de la conversión de los mensajes de OpenIGTLink desde el formato propio de este protocolo a uno inteligible para PLUS. Esta clase fue modificada para añadir soporte a los mensajes utilizados para procesar los valores de

los parámetros de configuración. El tipo elegido para estos mensajes, llamado `NDARRAY`¹, no es soportado por la implementación original de PLUS.

- Se debió modificar la clase que encapsula el algoritmo de simulación para que soportara la recepción y procesamiento de los datos de modificación del valor de un parámetro que se recibieran mediante OpenIGTLink.

4.1.3. Múltiples fuentes de datos de posicionamiento simultáneas

Con respecto a los datos de posicionamiento, la arquitectura de PLUS no permite el uso de más de una entrada de datos de manera simultánea. En particular, las condiciones materiales en que se desarrolló este trabajo indicaron que se requería congeniar la recepción de datos de posicionamiento, expresados a través de transformaciones homogéneas, desde una fuente de software comunicada con PLUS a través del protocolo OpenIGTLink y desde el dispositivo de hardware trakSTAR™.

La biblioteca PLUS incluye, en su implementación original, clases que gestionan la recepción de datos desde ambas fuentes. Aunque ambas son clases “hermanas”, con una clase padre común, están pensadas para utilizarse de forma excluyente entre sí - de hecho, las partes de PLUS que utilizan instancias de estas clases asumen que solo existe un objeto de cualquiera de las clases hermanas disponibles: intentar instanciar otro objeto de esta familia (por ejemplo, indicando la configuración inicial de dos receptores de datos en un archivo de configuración XML) implica la destrucción del ya existente (en este caso, la instanciación del segundo objeto especificado en el archivo de configuración implicaría la destrucción del primero).

La decisión de diseño de utilizar un único objeto para recibir los datos de posicionamiento no se refleja en un único punto de la implementación de la biblioteca. Por el contrario, permea toda la arquitectura de PLUS y, a nivel de código, se observa en múltiples clases y puntos del desarrollo de las lógicas de ejecución: además de los datos de configuración propios de cada tipo soportado, la clase que implementa el manejo de una fuente de datos de posicionamiento contiene la definición de un flujo de datos y, dentro de este, de varios “canales” que representan la información de una entidad en particular dentro de la escena de la simulación - un objeto tridimensional o el “observador”, por ejemplo. Dada la arquitectura interna de PLUS en este punto, las alternativas para dar soporte al uso simultáneo de varias fuentes de datos eran dos:

¹Ver anexo A.2.

CAPÍTULO 4. DESARROLLO DE LA APLICACIÓN

1. Introducir la posibilidad de instanciar varias fuentes de datos y que estas envíen información a un mismo flujo de datos.
2. Dentro del marco definido por la arquitectura de PLUS, generar una clase que encapsulara la lógica de recepción de datos desde, específicamente en este caso, una conexión OpenIGTLink y un dispositivo trakSTAR™.

La primera opción requeriría la modificación de varios niveles en el diseño interno de PLUS, pues correspondería más a una modificación de la arquitectura que al cambio del funcionamiento interno de una clase (o familia de clases). La segunda alternativa, si bien correspondería a una implementación *ad hoc* a las condiciones materiales de este desarrollo, no tan fácilmente generalizable a otros casos de uso, no necesitaría de modificaciones fuera de la implementación de la nueva clase. Por lo anterior, se optó por crear una nueva clase que “aglutinara” las lógicas de recepción y tratamiento de datos provenientes de una fuente de tipo OpenIGTLink y del dispositivo trakSTAR™; en líneas generales, se copió la lógica de las clases respectivas dentro de la nueva y se realizaron las adecuaciones necesarias para que todo quedara consistente.

La implementación de esta nueva clase significó, en particular, la fusión de las lógicas de configuración interna de las dos clases originales, de manera que, a partir de un mismo elemento dentro del archivo de configuración XML de una simulación, se instanciaran los distintos canales del flujo de datos utilizando la lógica que corresponde a la fuente de donde provienen. Como, independiente de los datos y códigos de configuración inicial utilizados, todos los canales de datos corresponden a objetos de una misma clase, los objetos ya instanciados pueden agregarse a un único flujo de datos; dicho de otro modo, la diferenciación de configuración se encapsula dentro de la clase y, para el resto de PLUS, solo existe el flujo de datos y no se hace distinción entre las fuentes externas de datos (la interfaz usuaria, a través de OpenIGTLink, y el magnetómetro). Asimismo, se consolidaron las lógicas de actualización de los datos de la fuente de datos, de forma que una misma llamada a la función respectiva actualice todos los canales de acuerdo a la lógica correspondiente con su fuente de datos. En este punto, la implementación se concentró en la eliminación de redundancias e inconsistencias en la consolidación de las lógicas de las clases originales.

Como esta nueva clase contiene la lógica de recepción de datos a través de OpenIGTLink, también corresponde a la clase cuya creación se menciona en la sección anterior de este capítulo; en otras palabras, la implementación realizada para cumplir el requerimiento de variación de parámetros en tiempo de ejecución y el de recepción de datos de posicionamiento desde múltiples fuentes simultáneas se realizó en una única clase. Las modificaciones hechas dentro de PLUS para que la aplicación utilice

esta nueva clase fueron explicadas en la sección anterior.

4.2. Interfaz gráfica

La interfaz gráfica implementada como parte de este trabajo debió cumplir con los siguientes requisitos:

- Permitir el despliegue de la imagen de ultrasonido simulado sintetizada.
- Permitir la variación de los parámetros del algoritmo de ultrasonido que pueden modificarse en tiempo de ejecución.

Ambos requisitos necesitaron del uso del protocolo OpenIGTLink. Desde la perspectiva de la interfaz gráfica, el primer punto necesitó la configuración de la aplicación como cliente OpenIGTLink (para recibir las imágenes desde PLUS), mientras que el segundo requirió implementar el programa como un servidor del mismo protocolo (para enviar los valores de los parámetros hacia PLUS). Esta dualidad de comportamientos se grafica en la figura 4.4.

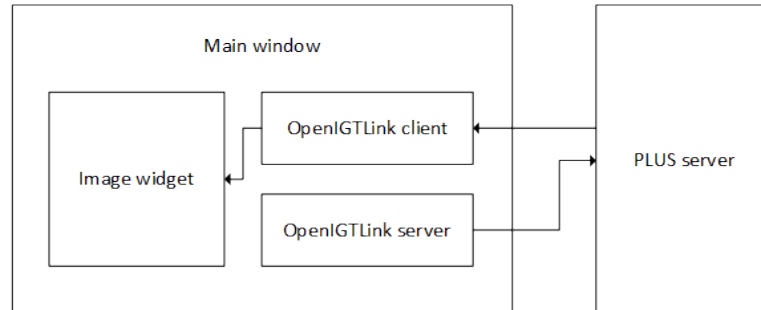


Figura 4.4: Diagrama de comunicación entre servidor PLUS e interfaz gráfica.

Fuente: Elaboración propia.

Una vez que PLUS genera una imagen y la deja disponible para su transmisión, crea e instancia un mensaje de OpenIGTLink con la información de la imagen sintetizada. A partir de uno de los ejemplos incluidos en el código fuente de OpenIGTLink se implementó la lógica de cliente de la aplicación gráfica. Para determinar que la recepción fuese exitosa y, además, permanente en el tiempo – esto es, que no fuese solo de una primera imagen y que las posteriores se perdieran en el canal de comunicación –, se montó un sistema de “3 frentes”:

CAPÍTULO 4. DESARROLLO DE LA APLICACIÓN

- Primero, se ejecutó una instancia del servidor PLUS configurada para utilizar uno de los ejemplos incluidos con la biblioteca.
- Segundo, se ejecutó la aplicación que PLUS utiliza como interfaz usuaria por defecto, con la configuración respectiva al mismo ejemplo seleccionado para PLUS. Esta aplicación permite activar y desactivar, de manera independiente, los dos flujos de comunicación de OpenIGTLink que existen en este ejemplo (como cliente y servidor, con respecto a PLUS). Para este caso, se desactiva la señal de cliente; esto es, se deshabilita la recepción de la imagen.
- Finalmente, se ejecutó la interfaz gráfica usuaria implementada.

En este esquema, la conexión entre PLUS y la interfaz usuaria por defecto permitió introducir cambios en la imagen – por ejemplo, la ubicación de los objetos 3D dentro de la escena simulada –, los cuales fueron reflejados en la imagen mostrada mediante la aplicación gráfica implementada en este trabajo de desarrollo.

Con la funcionalidad de recepción de imágenes probada, se continuó con la implementación de la funcionalidad de servidor OpenIGTLink de la aplicación gráfica. Esto también se realizó a partir de un ejemplo incluido con el código fuente de OpenIGTLink. La interfaz gráfica está diseñada para funcionar solo con la simulación que se creó para hacer las pruebas de validación de la aplicación final; por ello, solo ofrece controles para variar los datos de posicionamiento del único objeto 3D contemplado en dicho caso de uso. La funcionalidad de la interfaz quedó ordenada bajo la siguiente lógica:

- La modificación en el valor de alguno de los controles asociados a los parámetros de configuración desencadena el envío de un mensaje de OpenIGTLink con el nombre del parámetro modificado y el nuevo valor a asignarle.
- La modificación en los datos de posicionamiento desencadena el cálculo de una nueva matriz de transformación homogénea (explicada en los apéndices de este informe; en particular, en la sección A.2.1) y el envío de esta matriz a través de un mensaje de OpenIGTLink.

La figura 4.5 incluye una captura de pantalla de una ejecución de esta aplicación. Resulta interesante, desde lo técnico, hacer notar que el desarrollo de la aplicación gráfica se hizo en paralelo a las modificaciones a PLUS: en la medida que en este último se realizaban modificaciones y adiciones de funcionalidad, la interfaz gráfica fue cambiando para permitir probar y utilizar dichos cambios en PLUS.

CAPÍTULO 4. DESARROLLO DE LA APLICACIÓN

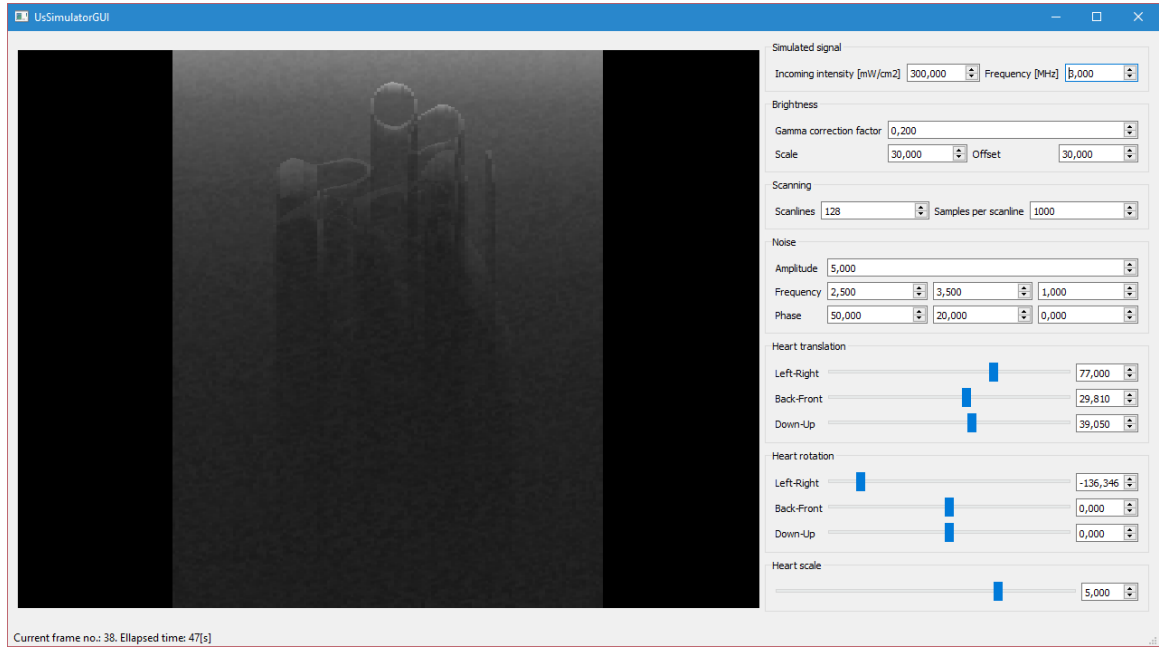


Figura 4.5: Captura de pantalla de la interfaz gráfica implementada.

Fuente: Elaboración propia.

4.3. Conclusiones

La fase de desarrollo de este trabajo culminó con la realización de las modificaciones a PLUS y la implementación de la interfaz gráfica que permite utilizar dichas modificaciones. Ambas aplicaciones quedaron en condiciones de ser ejecutadas de manera estable, sin errores críticos que terminen la ejecución; por tanto, fue posible pasar a la fase de validación del trabajo realizado.

Las modificaciones a PLUS fueron explicadas en este capítulo. La implementación de la interfaz gráfica fue tratada de manera más general, por ser esta un medio para evaluar el cumplimiento de los requerimientos funcionales. De todos modos, el desarrollo de la interfaz usuaria se detalla en los apéndices de este informe, en particular en el apéndice B, desde la sección B.2 en adelante.

Capítulo 5

Validación de la aplicación

Una vez terminada la fase de implementación de este trabajo, se procedió a realizar pruebas para verificar el cumplimiento de los requerimientos funcionales y no funcionales que se determinaron al inicio del desarrollo. Para ello, a partir del caso de uso de ejemplo llamado “*PlusServer: Vascular needle insertion simulation*”, incluido con la distribución de PLUS, se configuraron dos casos de uso:

- Uno, llamado “*Static heart simulation*”, que utiliza uno de los archivos que componen el modelo tridimensional del corazón para configurar una simulación estática - esto es, sin modelos animados. Este caso utiliza la misma tasa de adquisición que el ejemplo utilizado de base, 15 imágenes por segundo.
- Otro, llamado “*Animated heart simulation*”, que utiliza todos los archivos del modelo 3D para configurar una simulación animada. Este caso utiliza una tasa de adquisición de 25 imágenes por segundo (igualando la duración de la animación del modelo tridimensional a un segundo).

Además de las diferencias indicadas, ambos casos están configurados para considerar, al menos de manera inicial, los siguientes valores para los parámetros de configuración global:

- Frecuencia de señal: 3,0[MHz].
- Intensidad de señal: 300[MW/cm²].
- Factor de corrección *gamma*: 0,2.
- Escala de brillo: 30.

- *Offset* de brillo: 30.
- Número de líneas de muestreo: 128.
- Número de muestras por línea: 1000.

5.1. Cumplimiento de requerimientos funcionales

El primer requerimiento funcional quedó parcialmente satisfecho con la sola elección de PLUS como herramienta de desarrollo, pues esta biblioteca considera, dentro de sus modos de uso, la generación de imágenes de ultrasonido sintetizadas a partir de simulaciones “puras” - esto es, sin utilizar datos de exámenes reales o de simulaciones previas -, configurados con los modelos tridimensionales que conforman una escena, los parámetros iniciales de la simulación y de cada objeto de la escena y los datos de posicionamiento de estos últimos y el observador. Para completar el cumplimiento de este requerimiento se debió agregar el soporte para objetos tridimensionales animados; la correcta implementación de esta característica se verificó con la ejecución exitosa del caso “*Animated heart simulation*”.

El cumplimiento del segundo requerimiento se verificó durante las pruebas prácticas realizadas al sistema completo. Se observó que la aplicación funcionaba correctamente al utilizar datos de posicionamiento provenientes tanto desde la interfaz usuaria, para los objetos tridimensionales (en este caso particular fue solo uno, para el corazón), como desde el magnetómetro trakSTAR™. Durante estas pruebas se encontró una limitación: el rango de movimientos del sensor del magnetómetro que mantenía el corazón tridimensional dentro del campo de visión fue relativamente limitado, comparado con el rango de movimientos de un examen real - en otras palabras, el sensor simulado resultó ser más “sensible” que uno real, al menos con la simulación configurada tal como estaba durante la realización de las pruebas.

La realización de los primeros dos requerimientos funcionales no puede documentarse de manera más detallada en este informe por limitaciones propias de su naturaleza: ambos requieren la observación continua del sistema en un lapso de tiempo determinado. Además, la verificación del segundo requerimiento requiere el uso de hardware especializado - el dispositivo trakSTAR™. La verificación del tercer requerimiento, la variación de parámetros de configuración en tiempo de ejecución, también requiere la manipulación directa del sistema, para notar la diferencia con la asignación de valores a dichos parámetros a través del archivo de configuración XML asociado a cada caso. Sin embargo, las pruebas de este requerimiento sí arrojan un resultado presentable en este informe: las variaciones en la

CAPÍTULO 5. VALIDACIÓN DE LA APLICACIÓN

imagen sintetizada. A continuación, se presenta un análisis resumido del efecto de la variación de cada uno de los parámetros cuya modificación en tiempo de ejecución es soportada por PLUS posterior a este trabajo de desarrollo.

El primer parámetro estudiado fue la intensidad de la señal de ultrasonido simulada. En líneas generales, en la generación de una imagen de ecografía el brillo de cada punto es directamente proporcional a la señal de ultrasonido reflejada desde dicho punto: aumentar la intensidad de la señal implica que cada punto refleja más de dicha señal y, por tanto, se ve más brillante en la imagen final. Así, esto es conceptualmente equivalente a aumentar el brillo de una ampolleta. El efecto de este cambio en la intensidad de la señal en la imagen sintetizada se observa en la figura 5.1.

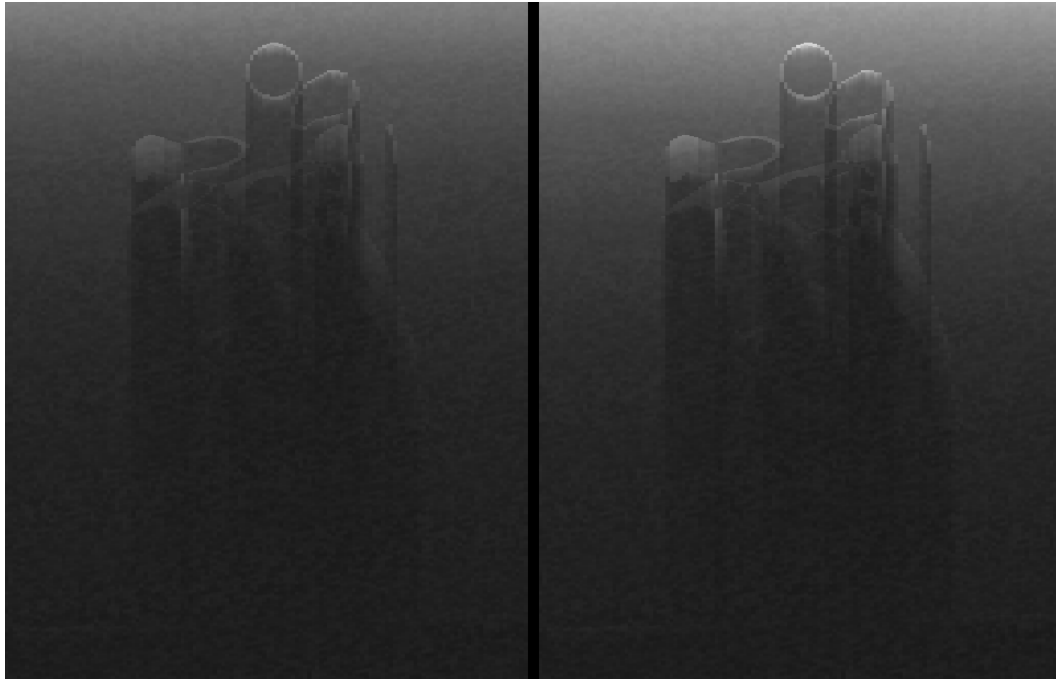


Figura 5.1: Imagen sintetizada con intensidad de señal de 300 [mW/cm²] (izquierda) y 1500 [mW/cm²] (derecha).

Fuente: Elaboración propia.

La frecuencia de la señal de ultrasonido afecta la interacción de esta con el medio por el que viaja; por ejemplo, en un examen de ecografía, determina qué tejidos (y hasta qué punto) atraviesa. En la figura 5.2 se observa este fenómeno; en particular, el valor de la frecuencia de la señal simulada impacta en hasta qué punto la señal se refleja de manera útil para la generación de la imagen, antes de ser sobrepasada por efecto de la difusión y el ruido. El efecto de la variación de este parámetro en la

CAPÍTULO 5. VALIDACIÓN DE LA APLICACIÓN

imagen sintetizada coincide con el observado en un examen real.

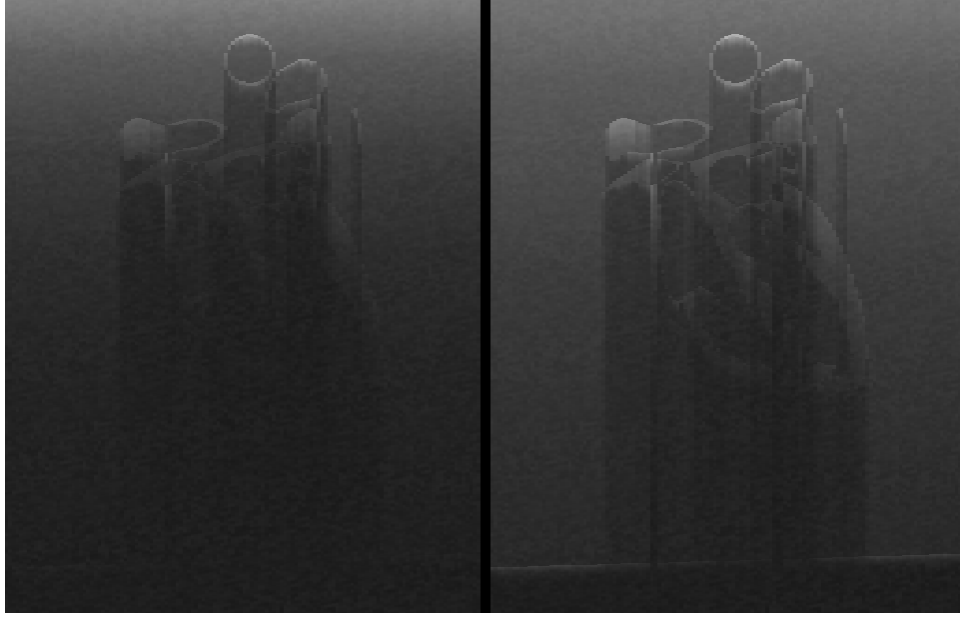


Figura 5.2: Imagen sintetizada con frecuencia de señal de 3.0 [MHz] (izquierda) y 1.5 [MHz] (derecha).
Fuente: Elaboración propia.

El brillo de cada píxel en la imagen sintetizada, como se mencionó anteriormente, se calcula como una función de la señal de ultrasonido reflejada por el punto respectivo en la escena 3D sobre la que se realiza la simulación. Sobre este valor se realiza un posprocesamiento para convertir la información acústica (la señal reflejada) en información visual. La ecuación matemática que describe este procesamiento, explicada en detalle en la sección 3.4.1 de este informe, es

$$b_{pixel} = \max(\min(B + SI^\gamma + N, 255), 0)$$

Con γ siendo el factor de corrección *gamma* del brillo, S la escala y B , el *offset*. La variación de cada uno de estos parámetros se muestran en las figuras 5.3, 5.4 y 5.5, respectivamente. En los casos del factor gamma y de la escala, el cambio en el brillo es proporcional (exponencial y directamente, respectivamente) a la intensidad de la señal reflejada en cada punto, mientras que el offset aplica de manera uniforme a todos los píxeles de la imagen.

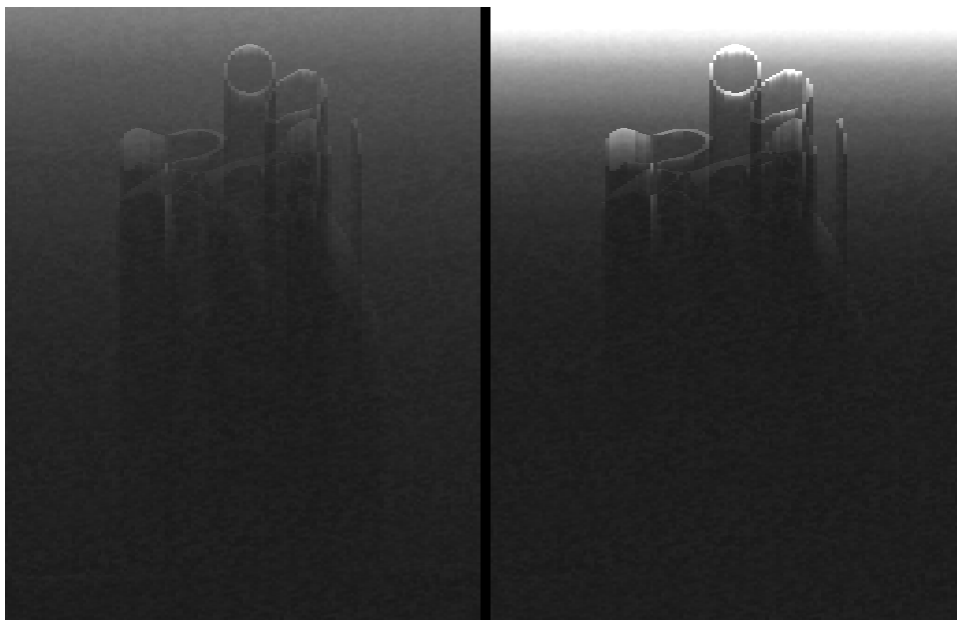


Figura 5.3: Imagen sintetizada con factor de corrección *gamma* 0.2 (izquierda) y 0.4 (derecha).

Fuente: Elaboración propia.

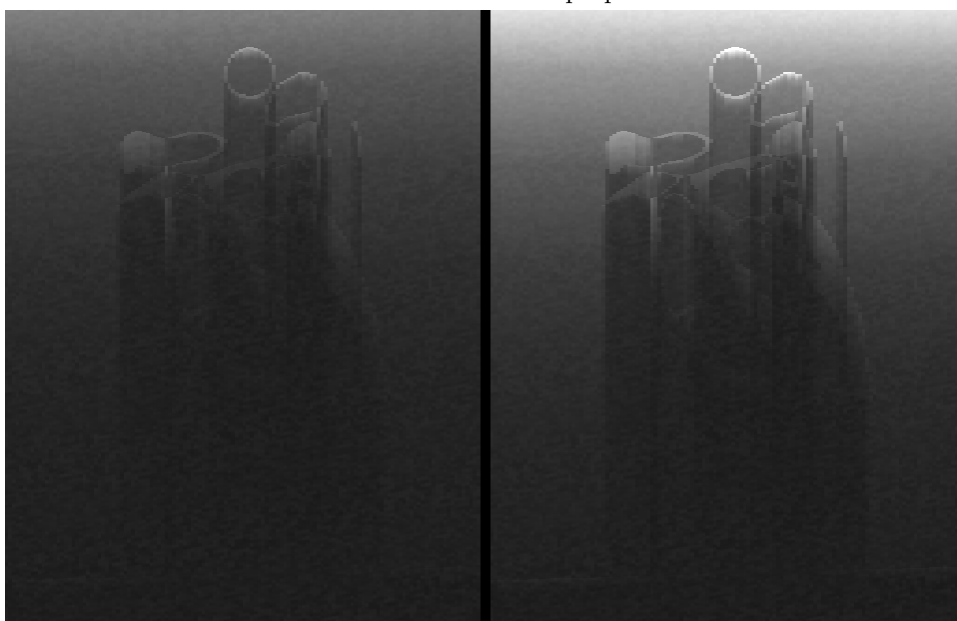


Figura 5.4: Imagen sintetizada con escala de brillo 30 (izquierda) y 60 (derecha).

Fuente: Elaboración propia.

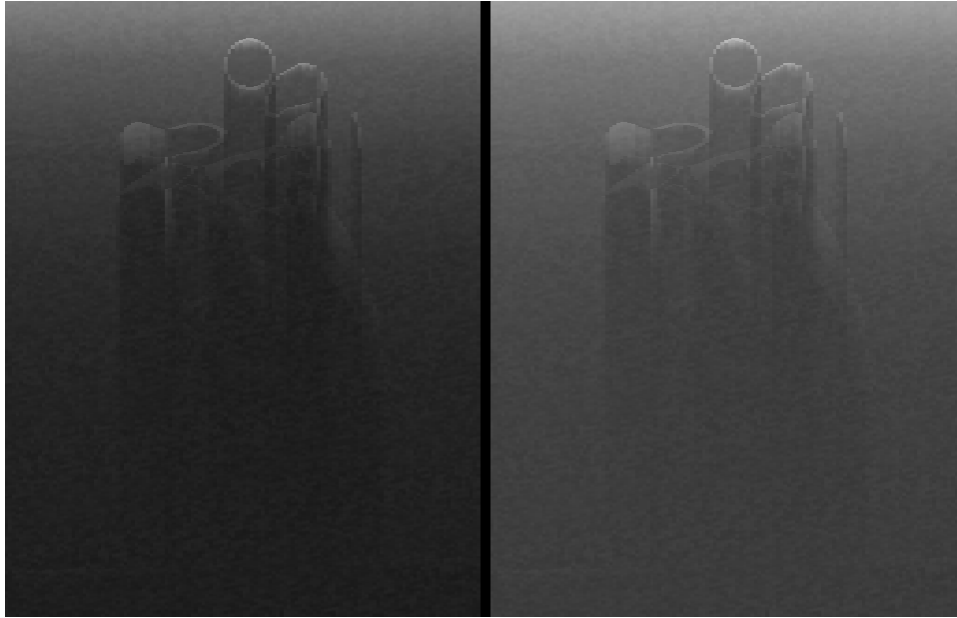


Figura 5.5: Imagen sintetizada con offset de brillo 30 (izquierda) y 60 (derecha).

Fuente: Elaboración propia.

El número de líneas de muestreo y el número de muestras por líneas afectan la definición de la imagen final, como se observa en las figuras 5.6 y 5.7, respectivamente. Las líneas de escaneo indican el número de “columnas” en que se divide la escena 3D para efectos computacionales, mientras que el número de muestras por línea determina las “filas” que conforman la escena. Ambas divisiones se vuelven perceptibles con números suficientemente bajos de los parámetros respectivos.

El cálculo final del brillo de cada píxel de la imagen incorpora ruido (la variable N en la ecuación mencionada anteriormente). PLUS simula este efecto utilizando un algoritmo conocido como ruido de Perlin, cuya configuración inicial requiere de tres parámetros llamados amplitud, frecuencia y fase, respectivamente. En la medida que estos valores no cambien, y sin considerar los datos de posicionamiento del observador de la escena simulada, el ruido calculado para un punto dado no cambia; por esto, la implementación del soporte de una escena 3D animada requirió incluir la modificación automática de algunos de estos valores para dar la apariencia de animación al ruido calculado. Además, dado que estos parámetros se encuentran dentro de los parámetros globales configurables en el archivo XML de cada caso simulado, se agregó la posibilidad de variarlos desde la interfaz usuaria. El efecto de cada parámetro sobre el ruido es el siguiente:

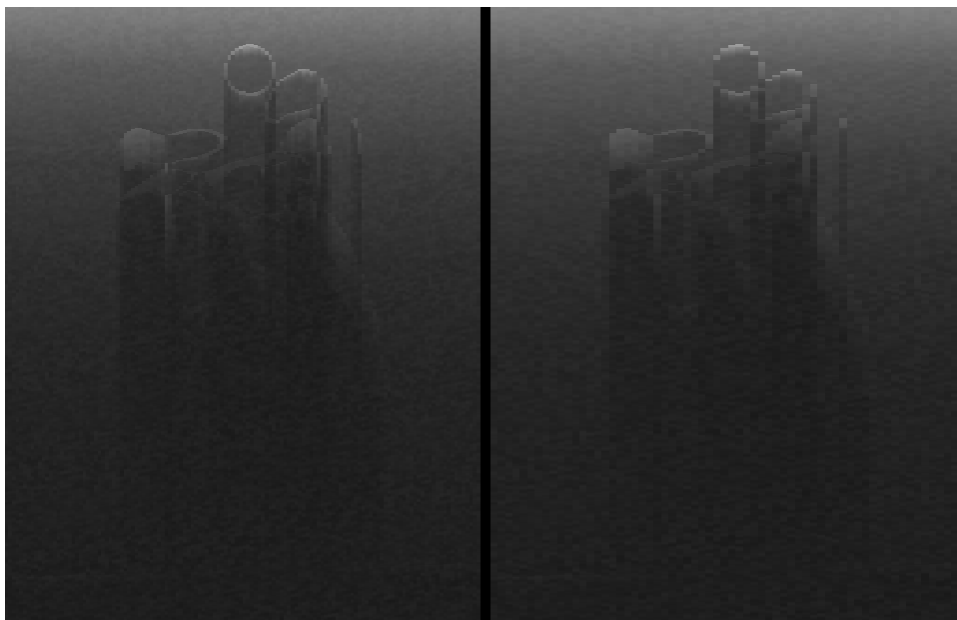


Figura 5.6: Imagen sintetizada con 128 (izquierda) y 64 (derecha) líneas de muestreo.

Fuente: Elaboración propia.

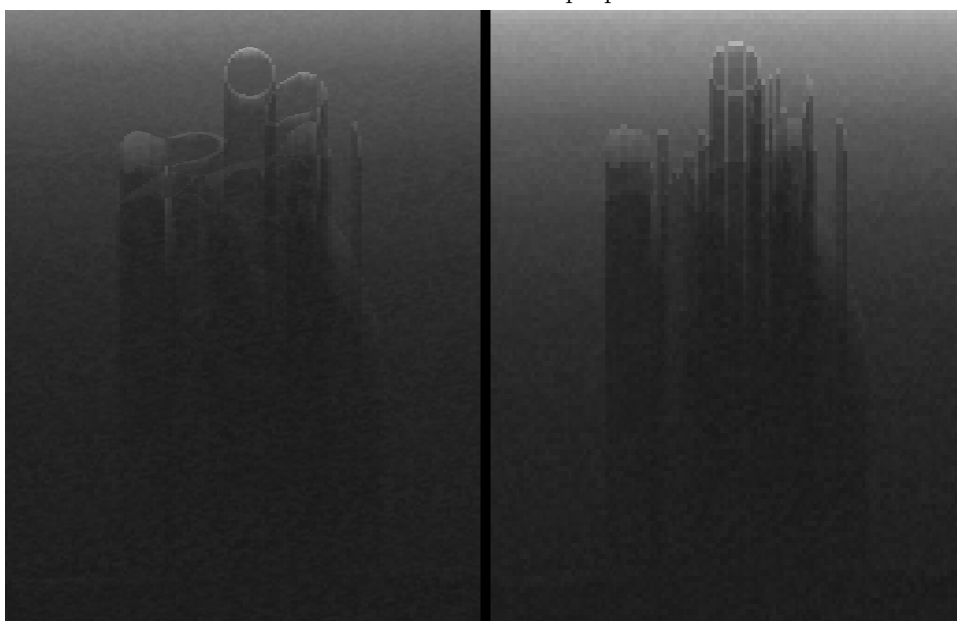


Figura 5.7: Imagen sintetizada con 1000 (izquierda) y 125 (derecha) muestras por línea.

Fuente: Elaboración propia.

CAPÍTULO 5. VALIDACIÓN DE LA APLICACIÓN

- La amplitud del ruido determina el tamaño del rango dentro del cual se mueven los valores entregados por el generador de ruido; esto es, entre mayor este valor de este parámetro, más “intenso” el ruido calculado, como se ve en la figura 5.8.
- La frecuencia determina la “fineza” del ruido calculado - sin rigurosidad técnica, se puede describir como el “tamaño de las manchas” en la imagen final. Este valor se determina para cada eje coordenado del espacio 3D, y el efecto de su variación se observa en la figura 5.9.
- La fase del ruido actúa como *offset*: se podría considerar que el ruido se genera como una “capa” infinita que luego es superpuesta sobre la imagen sintetizada, y la fase indica en qué medida, para cada dirección espacial, esta capa se desplaza antes de computarse la imagen final; por esto, se utiliza este parámetro para simular la animación del ruido. La figura 5.10 muestra dos imágenes sintetizadas con distintas fases. En todo caso, las diferencias se observan mejor en el caso del modelo animado que entre dos imágenes fijas en particular.

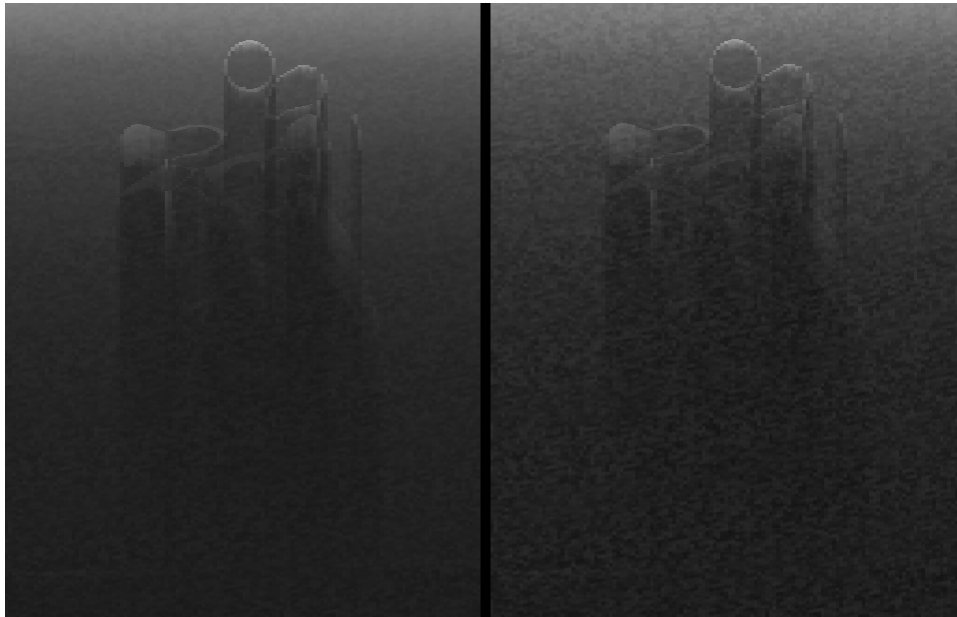


Figura 5.8: Imagen sintetizada con amplitud de ruido 5.0 (izquierda) y 10 (derecha).

Fuente: Elaboración propia.

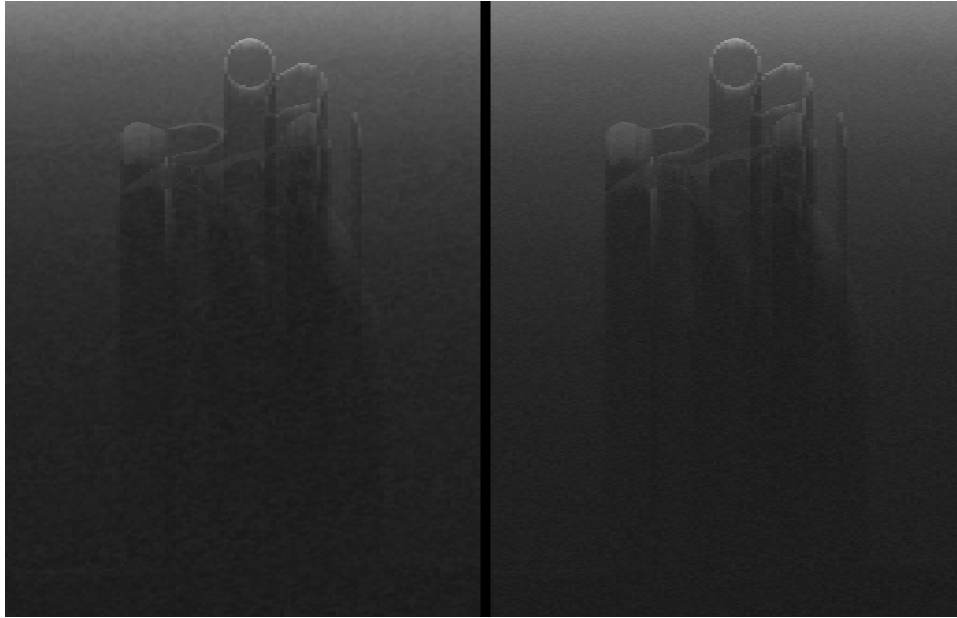


Figura 5.9: Imagen sintetizada con frecuencias de ruido (2.5, 3.5, 1.0) (izquierda) y (5.0, 7.0, 2.0) (derecha).

Fuente: Elaboración propia.

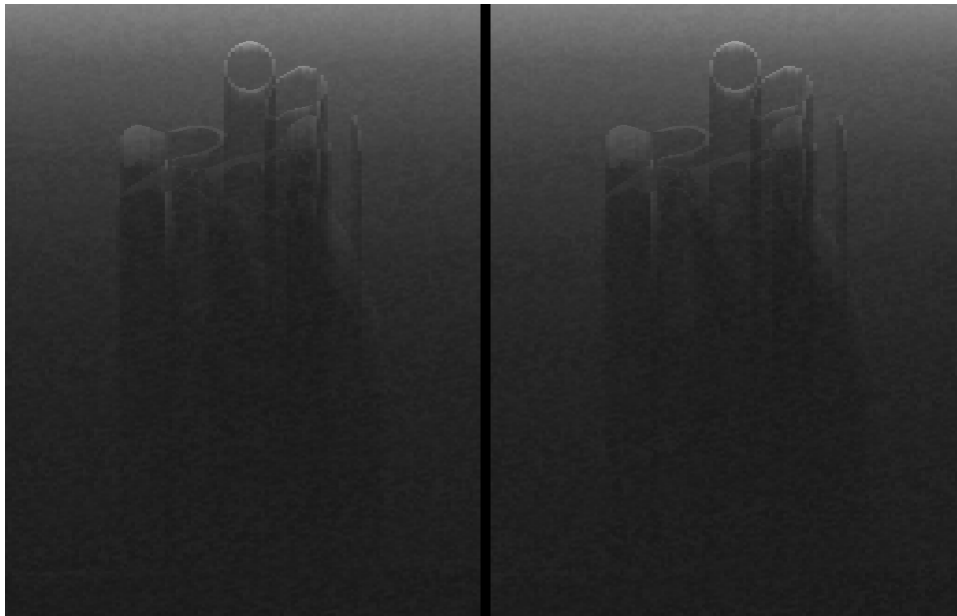


Figura 5.10: Imagen sintetizada con fases de ruido (50, 20, 0) (izquierda) y (100, 40, 0) (derecha).

Fuente: Elaboración propia.

5.2. Cumplimiento de requerimientos no funcionales

Como se indicó en el capítulo 3, en la sección 3.2, la aplicación final tiene una restricción de rendimiento: debe generar una imagen sintetizada en 40 milisegundos o menos. Para medir el tiempo de ejecución utilizado, en PLUS, en la generación de una imagen, se utilizó la API de Windows; en particular, la función `GetSystemTimeAsFileTime`¹, que retorna, mediante un valor cuyo tipo es la estructura `FILETIME`² el tiempo en que ocurre la llamada a la función, con una resolución de 100 nanosegundos. Para obtener el tiempo transcurrido, se obtiene el tiempo al inicio y al término de la ejecución del algoritmo de simulación para una imagen, y luego se restan los valores obtenidos. Esta forma de medición no consideró el tiempo utilizado en la interfaz gráfica para recibir y dibujar la imagen sintetizada, pues esta parte de la aplicación es, en teoría, completamente paralelo a la simulación - de hecho, por la arquitectura de la aplicación, podría hasta ejecutarse en un equipo distinto al que corre la simulación.

Se presentan los tiempos de ejecución promedio calculados sobre dos rangos, 600 y 3000 imágenes generadas. El tamaño del primer rango se estableció a partir del resultado de la primera medición, pues corresponde a la aproximación, a la centena, de la cantidad de imágenes generadas en 3 minutos de ejecución (rango temporal seleccionado arbitrariamente); el tamaño del segundo rango corresponde a 5 veces el tamaño del primero. Los tiempos medidos no consideran la transmisión de la imagen a la interfaz gráfica, ni su despliegue en la misma.

La primera medición se realizó ejecutando, tal cual como fueron configurados, los casos “*Animated heart simulation*” y “*Static heart simulation*”, sin conectar el magnetómetro trakSTAR™ al computador. Esta medición, cuyos resultados se muestran en la tabla 5.1 sirvió para establecer una referencia para las demás evaluaciones y para estudiar el efecto del uso de objetos tridimensionales animados en el rendimiento de PLUS.

	Animated heart simulation	Static heart simulation
600 imágenes	280	275
3000 imágenes	233	266

Cuadro 5.1: Promedios de tiempo de ejecución, en [ms], para casos “*Static heart simulation*” y “*Animated heart simulation*”.

Fuente: Elaboración propia.

¹[https://msdn.microsoft.com/en-us/library/windows/desktop/ms724397\(v=vs.85\).aspx](https://msdn.microsoft.com/en-us/library/windows/desktop/ms724397(v=vs.85).aspx)

²[https://msdn.microsoft.com/en-us/library/windows/desktop/ms724284\(v=vs.85\).aspx](https://msdn.microsoft.com/en-us/library/windows/desktop/ms724284(v=vs.85).aspx)

CAPÍTULO 5. VALIDACIÓN DE LA APLICACIÓN

Las siguientes mediciones se realizan solo para el caso “*Animated heart simulation*”, el cual es modificado en un parámetro a la vez para observar el efecto de la variable en cuestión sobre el tiempo de ejecución. Se establecieron cuatro variaciones:

- Utilizando el dispositivo trakSTAR™.
- Desactivando la simulación de ruido.
- Utilizando 256 líneas de escaneo.
- En un computador distinto al utilizado para el desarrollo.

Un factor que no fue medido es el efecto de la complejidad del modelo 3D en el tiempo de ejecución, debido a la inexistencia de un segundo modelo 3D significativo (los que viene incluido con PLUS son excesivamente simples, con un número de vértices y planos en el orden de las decenas, contra los 187.278 y 62.426 del modelo del corazón, respectivamente) y que permita simular la animación – en particular, con 25 archivos que representen un frame – que sirviera de comparación. Otro factor que pudiese haber sido medido es el efecto del número de muestras por línea de muestreo, el otro parámetro que determina la resolución de la imagen. Entre esta variable y el número de líneas de muestreo se optó arbitrariamente por evaluar solo este último, debido a que, revisando el código de PLUS en la parte donde estas variables se utilizan, el tipo de efecto sobre el rendimiento es similar y, por tanto, se estimó que medir ambos sería redundante. Sobre este punto en particular se elaborará más adelante.

La tabla 5.2 muestra el resultado de la medición realizada conectando al computador, y utilizando para la simulación, el magnetómetro. Además, se observó que, a diferencia de las demás mediciones, en este caso hubo “pérdida” de imágenes: por una parte, la simulación parecía dejar de generarse en algunos momentos (se “congelaba”), mientras que, por otro lado, la cantidad de imágenes que PLUS reportaba como generadas – esto fue indirectamente observado en el número de mediciones obtenidas – correspondía casi exactamente al doble de las imágenes recibidas en la interfaz gráfica, a la que fue agregado un contador de imágenes dibujadas.

600 imágenes	286
3000 imágenes	278

Cuadro 5.2: Promedios de tiempo de ejecución, en [ms], utilizando el dispositivo trakSTAR™.

Fuente: Elaboración propia.

La tabla 5.3 muestra los resultados medidos al desactivar la aplicación de ruido en las imágenes sintetizadas. El código de PLUS donde se efectúa este cálculo tiene una optimización que implica no

CAPÍTULO 5. VALIDACIÓN DE LA APLICACIÓN

invocar las funciones relacionadas a dicho cómputo cuando la amplitud del ruido (uno de los parámetros de inicialización del algoritmo de Perlin) es igual a cero; esta optimización se utilizó para “desactivar” la generación de ruido para esta medición sin modificar el código fuente.

600 imágenes	186
3000 imágenes	152

Cuadro 5.3: Promedios de tiempo de ejecución, en [ms], sin simulación de ruido.

Fuente: Elaboración propia.

La tabla 5.4 muestra los resultados obtenidos al duplicar la cantidad de líneas de muestreo con respecto al caso base. Un estudio del código fuente revela la estructura general de la generación de la imagen y características sobre la lógica de este algoritmo; en particular, que la ejecución de la simulación se distribuye, por cada imagen, en una iteración doblemente anidada: por cada línea de muestreo, y por cada muestra en una línea. Por tanto, el impacto de ambos parámetros en el tiempo de ejecución viene dado, en líneas generales, de la forma

$$N^{\circ} \text{ de líneas de escaneo} * N^{\circ} \text{ de muestras por línea}$$

Por tanto, el efecto de estos parámetros, en el rendimiento del sistema completo, es comparable entre sí, y es por esto que su medición por separado se consideró redundante.

600 imágenes	394
3000 imágenes	470

Cuadro 5.4: Promedios de tiempo de ejecución, en [ms], sin simulación de ruido.

Fuente: Elaboración propia.

Finalmente, la tabla 5.5 muestra los efectos de ejecutar el caso base en un equipo de escritorio, distinto al utilizado para el desarrollo y la medición en los demás casos evaluados. Las características de este equipo son las siguientes:

- **Procesador:** Intel[®] Core[™] i5-3300 (doble núcleo, frecuencia de reloj 3,00 GHz).
- **Tarjeta gráfica:** NVIDIA[®] GeForce[™] GT 640.
- **Memoria RAM:** 8GB, DDR3.

CAPÍTULO 5. VALIDACIÓN DE LA APLICACIÓN

600 imágenes	162
3000 imágenes	136

Cuadro 5.5: Promedios de tiempo de ejecución, en [ms], en un equipo distinto al usado en el desarrollo.
Fuente: Elaboración propia.

5.3. Conclusiones

Al comienzo de este trabajo de desarrollo, se establecieron los siguientes requerimientos a cumplir:

1. Funcionales:

- a) Generar simulaciones solo a partir de datos de entrada, sin recurrir a simulaciones anteriores.
- b) Recibir un dato de entrada, llamado “de posicionamiento”, desde un dispositivo físico que emule la mecánica de la realización de un examen ecográfico.
- c) Permitir la variación de los parámetros de configuración de la simulación que se relacionen con las características de foco, contraste y frecuencia de un examen de ultrasonido.

2. No funcionales:

- a) Generar una imagen sintética en hasta 40 milisegundos.

Con respecto a los requerimientos funcionales, el primero y el tercero fueron satisfechos completamente, sin restricciones ni consideraciones sobre el modo de uso del sistema. El segundo, relativo a la recepción de datos de posicionamiento, fue satisfecho con dos observaciones: la “sensibilidad” del magnetómetro a los movimientos del usuario y la degradación en lo que puede llamarse la “estabilidad” del sistema - en particular, la aparente pérdida de imágenes generadas. Ambas consideraciones, que limitan el rango de operación del sistema en general al limitar los movimientos que puede realizar el usuario, parecen relacionarse a una discordancia entre los sistemas de referencia (ejes coordenados y escalas) de los datos espaciales de los objetos tridimensionales de la escena simulada - entregados desde la interfaz usuaria mediante el protocolo OpenIGTLink - y los del observador, obtenidos desde el magnetómetro. En particular, con respecto a la pérdida de imágenes, lo que pareciera ocurrir es que, dentro de PLUS, se reciben datos que resultan inutilizables, sea porque son numéricamente sin sentido (por ejemplo, un valor indefinido (*not a number*, NaN)) o porque están fuera de los rangos de diferencia tolerables con respecto a los datos de los otros objetos; esto causaría que, en una iteración de la simulación de ultrasonido, se genere una imagen vacía que, por lo mismo, no es transmitida a

CAPÍTULO 5. VALIDACIÓN DE LA APLICACIÓN

la interfaz - esto explicaría la diferencia entre la cantidad de valores medidos en las pruebas de rendimiento y el número de imágenes recibidas. En cualquier caso, esta posible disconformidad entre las fuentes de datos espaciales solo coloca restricciones al modo de uso, sin comprometer el cumplimiento del requerimiento funcional.

El requerimiento no funcional, que cada imagen sintetizada sea computada en 40 milisegundos o menos, no fue cumplido en este desarrollo. Sin embargo, en comparación a los rendimientos observados durante la fase investigativa de esta memoria y mostrados en el estado del arte de este informe, los tiempos obtenidos se encuentran muy cerca del objetivo - a un orden de magnitud, expresados en milisegundos, contra varios minutos por imagen - y, así, prácticamente en el rango que se puede considerar como “tiempo real”.

Las mediciones realizadas permiten dilucidar caminos para cumplir efectivamente con el requisito de rendimiento. Primero, la resolución de la imagen es ajustable, por lo que la alternativa más directa es reducir el número de líneas de muestreo y de muestras por línea simuladas por PLUS. Esto requeriría determinar, junto a profesionales del área médica, el mínimo valor para estos parámetros que siga entregando imágenes de la calidad suficiente para utilizarse en un simulador de exámenes ecográficos.

En segundo lugar, la implementación del algoritmo de simulación utilizado por PLUS es completamente secuencial. Además, un análisis general y preliminar del código da a entender que al menos partes del algoritmo (por ejemplo, la simulación del ruido) está compuesto de cálculos independientes entre sí. Por tanto, la posibilidad de agregar paralelización al algoritmo está abierta; esto podría sumarse a la utilización de cómputo en GPU, cuando haya una disponible, para mejorar aún más el rendimiento del sistema.

Por último, viendo la magnitud de la mejora de rendimiento necesaria para cumplir el requerimiento de tiempo, la solución más “fácil”, mejorar el sistema subyacente - el computador y su configuración -, es factible. La diferencia en los tiempos medidos entre los dos equipos comparados es de alrededor de un 100 % (con respecto al menor valor registrado); coincidentemente, la diferencia de “potencia” entre los dos procesadores, según varias comparaciones disponibles públicamente³ es de igual proporción: el procesador menos potente rinde alrededor de la mitad del otro. Esto también vuelve interesante considerar otro aspecto, la configuración del *software* del equipo. En particular, ambos equipos de prueba estaban configurados como computadores de uso general, con una serie de programas y herramientas compitiendo por el uso de la CPU con el sintetizador de imágenes. Un sistema configurado para un uso dedicado a PLUS debiera permitir obtener mejores rendimientos solo en virtud de dicha característica.

³<http://cpuboss.com/cpus/Intel-Core-i7-640M-vs-Intel-Core-i5-3330>

Capítulo 6

Observaciones y comentarios finales

La situación inicial desde la cual se elaboró esta memoria, alcanzada en el contexto del trabajo introducido en [4], consistía en un sistema de sintetización de imágenes basado en la generación de texturas, que alcanzaba rendimientos que cumplen con la definición de “tiempo real” utilizada en este informe y permitía utilizar el dispositivo trakSTAR™ como fuente de datos de posicionamiento del “observador” de la simulación. Sin embargo, no contemplaba la variación de los parámetros del examen simulado durante la ejecución de la aplicación.

Con respecto a este punto de partida, este desarrollo implica un avance en dos frentes:

1. Desde el punto de vista de un usuario final, se trata de una herramienta más general, no asociada a un único caso a nivel de código o diseño, por lo que su uso es más flexible. Esto se refleja en el soporte para variar los parámetros que usualmente se pueden modificar durante la realización de un examen (sea en la configuración inicial de una simulación o en tiempo de ejecución) y en la posibilidad de configurar nuevas simulaciones creando archivos XML para cada caso - un proceso por definición automatizable y realizable a través de, por ejemplo, una interfaz gráfica hecha para tal efecto - sin que sea necesario modificar el sistema completo, por mencionar dos ventajas.
2. Desde el punto de vista de un desarrollador, se trata de un componente de *software* más modular y, además, mucho más sencillo de utilizar en la implementación de otros sistemas, debido al uso de protocolos de comunicación abiertos y conocidos - en particular, OpenIGTLink.

Las ventajas antes mencionadas derivan del uso de la biblioteca PLUS como base para este desarrollo. En efecto, si hubiera que resumir este trabajo en una línea, debiera decirse que la tarea fue

CAPÍTULO 6. OBSERVACIONES Y COMENTARIOS FINALES

modificar PLUS para dar soporte a nuevas características: soporte para modelos animados, modificación de algunos parámetros en tiempo de ejecución y captura de datos de posicionamiento desde múltiples fuentes simultáneas. Esta herramienta de *software* resultó ser sumamente robusta en cuanto a las funciones ofrecidas y de poseer una arquitectura interna que permita agregar nuevas características - de hecho, la mayor dificultad a la hora de implementar las modificaciones fue familiarizarse con el código fuente (una tarea siempre compleja cuando se trabaja con una herramienta de este tamaño), no la programación en sí. Este desarrollo se suma así a una creciente batería de casos que validan a PLUS como una herramienta de trabajo en el campo de la creación de herramientas relacionadas al uso de ultrasonido, sea real o simulado.

Una vez terminado este trabajo, se puede decir que esta memoria tuvo un alto grado de exploración. Si bien constituye un avance hacia la implementación de una herramienta que pueda considerarse “final”, en el sentido de que sea utilizable en simuladores de exámenes ecográficos, como producto terminado, por usuarios finales, la aplicación resultante de este desarrollo no se encuentra completamente refinada. La razón más importante es el no haber alcanzado, en las pruebas realizadas, el nivel de rendimiento que se definió como “en tiempo real” - y, aunque esto podría obtenerse solo con cambios en las condiciones de uso de la aplicación (en particular, en la configuración de la imagen sintetizada y del equipo en que se ejecute el sistema), hay espacio para optimizaciones a nivel de código que no pueden descartarse de antemano. También está el asunto de la sensibilidad del magnetómetro a los movimientos del usuario, con respecto a los demás objetos que componen la simulación; esto, aunque no impide el uso del sistema, impone una restricción que debiera eliminarse en un producto que se considere “terminado”.

Con respecto al rendimiento de la aplicación, y como ya se mencionó en el capítulo anterior, se advierte que la implementación linear e iterativa de PLUS ofrece, a primera vista, espacio para optimizaciones de cómputo; en particular, para posibles paralelizaciones. Esto no implicaría, necesariamente, trasladar cálculos a una GPU ni requeriría el uso de nuevas dependencias, sino que podría aprovechar de mejor manera la CPU utilizando las facilidades de programación de procesos múltiples ofrecidas por VTK (que ya es dependencia de PLUS) o las que existen en la última revisión del estándar de C++ que, por ejemplo, ya están soportadas por la versión de Microsoft de la biblioteca estándar, que es la utilizada durante este trabajo. Se requieren, eso sí, revisiones más profundas y mediciones más precisas para determinar exactamente qué puntos del código sería más interesante optimizar.

Sobre el uso de múltiples fuentes de datos de posicionamiento, la implementación realizada en este trabajo va a contrapelo de lo que podría llamarse la “filosofía de diseño” de PLUS, mezclando

CAPÍTULO 6. OBSERVACIONES Y COMENTARIOS FINALES

de manera algo forzada funciones que originalmente no solo estaban separadas, sino que trabajan de manera excluyente. Esto se refleja tanto directamente en el código como indirectamente en el archivo de configuración de la simulación. Si bien es posible realizar modificaciones adicionales a PLUS para que tenga pleno soporte a esta característica, una solución más sencilla podría ser entregar todos los datos de posicionamiento desde la interfaz gráfica, capturando en ese punto la información provista por el dispositivo trakSTAR[™] y enviándola a PLUS, junto con los datos de los demás objetos, mediante OpenIGTLink. Esto, además de respetar la arquitectura original de la biblioteca, permitiría realizar preprocesamientos (con respecto a PLUS) a los datos para robustecer su consistencia y hacer congeniar de mejor manera las distintas fuentes de datos – por ejemplo, normalizando los sistemas coordinados en que trabaja cada una. También facilitaría el uso de una cantidad arbitraria de fuentes de datos, pues para PLUS siempre aparecería solo una, que es el supuesto con el que está implementado originalmente.

Finalmente, cabe realizar una observación sobre la modificación de parámetros de la simulación en tiempo real. En este momento la implementación de esta funcionalidad se encuentra acoplada a la de recepción de datos de posicionamiento. Existe otro canal de recepción de datos, que es utilizado para comunicarse con PLUS como aplicación, para entregar comandos como - por ejemplo - indicar que las imágenes sintetizadas se guarden como archivos en el disco duro. Desde una perspectiva de diseño “purista”, los parámetros que se modifican constituyen un canal de datos intermedio entre los ya existentes, pues se aplican a la simulación en general, no al servidor PLUS como aplicación ni a los objetos 3D particulares que componen la simulación. La implementación debiera reflejar esta separación, por “corrección técnica”, aunque funcionalmente el diseño actual no implique limitaciones funcionales ni genere problemas detectables en la ejecución de la aplicación.

Bibliografía

- [1] D. P. Mandavia, J. Aragona, L. Chan, D. Chan y S. O. Henderson, «Ultrasound training for emergency physicians—a prospective study», *Academic Emergency Medicine*, vol. 7, n.º 9, págs. 1008-1014, 2000.
- [2] P. Z. González, «Implementación de un modelo de sintetización de imágenes de ultrasonido», Tesis de pregrado, Departamento de Ingeniería Civil Biomédica, Universidad de Valparaíso, Chile, 2007.
- [3] H. B. Erices, «Desarrollo de una interfaz de usuario y de algoritmos para el manejo y análisis en imágenes para un sistema ecocardiográfico simulado», Tesis de pregrado, Departamento de Ingeniería Civil Biomédica, Universidad de Valparaíso, Chile, 2007.
- [4] P. Reyes, «Integrating deliberate practice in a simulation-based cardiac sonographer training by implementing an Intelligent Tutoring Systems based on Conversation Theory (Proyecto Fondecyt 11070226)», 2007.
- [5] J. Self, «The defining characteristics of intelligent tutoring systems research: ITSs care, precisely», *International Journal of Artificial Intelligence in Education*, vol. 10, págs. 350-364, 1998.
- [6] R. Tronco Gasparini y V. L. da Silveira Nantes Button, «Development of a MATLAB Environment Software for Simulation of Ultrasonic Field», *Revista Ingeniería Biomédica*, vol. 7, n.º 13, págs. 57-67, 2013.
- [7] A. Karamalis, W. Wein y N. Navab, «Fast ultrasound image simulation using the westervelt equation», en *Medical Image Computing and Computer-Assisted Intervention—MICCAI 2010*, Springer, 2010, págs. 243-250.
- [8] J. A. Jensen e I. Nikolov, «Fast simulation of ultrasound images», en *Ultrasonics Symposium, 2000 IEEE*, IEEE, vol. 2, 2000, págs. 1721-1724.

BIBLIOGRAFÍA

- [9] J. A. Jensen, «Simulation of advanced ultrasound systems using Field II», en *Biomedical Imaging: Nano to Macro, 2004. IEEE International Symposium on*, IEEE, 2004, págs. 636-639.
- [10] A. Marion y D. Vray, «Toward a real-time simulation of ultrasound image sequences based on a 3-D set of moving scatterers», *Ultrasonics, Ferroelectrics and Frequency Control, IEEE Transactions on*, vol. 56, n.º 10, págs. 2167-2179, 2009.
- [11] K. Schwenk, G. Reis y R. H. van Lengen, «Realtime artificial ultrasound images», *Submitted to 2005/06 Elsevier Journal Simulation Modelling Practice and Theory*, 2005.
- [12] A. Appel, «Some techniques for shading machine renderings of solids», en *Proceedings of the April 30–May 2, 1968, spring joint computer conference*, ACM, 1968, págs. 37-45.
- [13] M. Song, R. M. Haralick y F. H. Sheehan, «Ultrasound imaging simulation and echocardiographic image synthesis», en *Image Processing, 2000. Proceedings. 2000 International Conference on*, IEEE, vol. 3, 2000, págs. 420-423.
- [14] R. Shams, R. Hartley y N. Navab, «Real-time simulation of medical ultrasound from CT images», en *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2008*, Springer, 2008, págs. 734-741.
- [15] T. Reichl, J. Passenger, O. Acosta y O. Salvado, «Ultrasound goes GPU: Real-time simulation using CUDA», en *SPIE Medical Imaging*, International Society for Optics y Photonics, 2009, págs. 726 116-726 116.
- [16] O. Kutter, R. Shams y N. Navab, «Visualization and GPU-accelerated simulation of medical ultrasound from CT images», *Computer methods and programs in biomedicine*, vol. 94, n.º 3, págs. 250-266, 2009.
- [17] S. U. Gjerald, R. Brekken, L. E. Bø, T. Hergum y T. A. Nagelhus Hernes, «Interactive development of a CT-based tissue model for ultrasound simulation», *Computers in biology and medicine*, vol. 42, n.º 5, págs. 607-613, 2012.
- [18] H. Gao, T. Hergum, H. Torp y J. D’hooge, «Comparison of the performance of different tools for fast simulation of ultrasound data», *Ultrasonics*, vol. 52, n.º 5, págs. 573-577, 2012.
- [19] W. Cong, J. Yang, Y. Liu e Y. Wang, «Fast and Automatic Ultrasound Simulation from CT images», *Computational and mathematical methods in medicine*, vol. 2013, 2013.

BIBLIOGRAFÍA

- [20] R. Shams, F. Luna y R. I. Hartley, «An algorithm for efficient computation of spatial impulse response on the GPU with application in ultrasound simulation», en *Biomedical Imaging: From Nano to Macro, 2011 IEEE International Symposium on*, IEEE, 2011, págs. 45-51.
- [21] Y. Law, S. Ullrich, T. Knott, T. Kuhlen y S. Weg, «Ultrasound Image Simulation with GPU-based Ray Tracing», *Virtuelle und Erweiterte Realität*, págs. 183-194, 2011.
- [22] B. Burger, S. Bettinghausen, M. Radle y J. Hesser, «Real-time GPU-based ultrasound simulation using deformable mesh models», *Medical Imaging, IEEE Transactions on*, vol. 32, n.º 3, págs. 609-618, 2013.
- [23] A. Lasso, T. Heffter, A. Rankin, C. Pinter, T. Ungi y G. Fichtinger, «PLUS: Open-source toolkit for ultrasound-guided intervention systems», *IEEE Transactions on Biomedical Engineering*, n.º 10, págs. 2527-2537, oct. de 2014. DOI: 10.1109/TBME.2014.2322864. dirección: <http://perk.cs.queensu.ca/contents/plus-open-source-toolkit-ultrasound-guided-intervention-systems>.
- [24] A. Lasso, T. Heffter, C. Pinter, T. Ungi y G. Fichtinger, «Implementation of the PLUS open-source toolkit for translational research of ultrasound-guided intervention systems», *MICCAI-Systems and Architectures for Computer Assisted Interventions*, págs. 1-12, 2012.
- [25] L. Bartha, A. Lasso, C. Pinter, T. Ungi, Z. Keri y G. Fichtinger, «Open-source surface mesh-based ultrasound-guided spinal intervention simulator», *International journal of computer assisted radiology and surgery*, vol. 8, n.º 6, págs. 1043-1051, 2013.
- [26] *Plus applications user manual*, <http://perk-software.cs.queensu.ca/plus/doc/nightly/user/>, Visto en 2016-09-01.
- [27] *PlusLib: Main Page*, <http://perk-software.cs.queensu.ca/plus/doc/nightly/dev/>, Visto en 2016-09-01.
- [28] *Specification*, <http://openigtlink.org/spec.html>, Visto en 2016-09-01.
- [29] *The OpenIGTLink library*, <http://openigtlink.org/api/index.html>, Visto en 2016-09-01.

Apéndice A

Descripción técnica de bibliotecas

A.1. PLUS

La compilación de PLUS, a partir del código fuente, requiere de la presencia de cuatro dependencias obligatorias:

- *The Visualization Toolkit* (VTK)¹ : biblioteca para computación gráfica, procesamiento y visualización de imágenes.
- *Insight Segmentation and Registration Toolkit* (ITK)² : biblioteca para análisis gráfico.
- *Open Image-Guided Therapy Link* (OpenIGTLink)³ : protocolo de comunicación de procesos y dispositivos, enfocado en usos médicos.
- Qt⁴ : *framework* para la implementación de aplicaciones gráficas.

Otras funcionalidades (ej.: fuentes de datos de entrada asociadas a dispositivos de hardware específicos) pueden requerir la configuración y presencia de más dependencias al momento de compilar PLUS.

La documentación técnica de PLUS (lo que comúnmente se llamaría la “API” de la biblioteca) se encuentra en [27].

¹<http://www.vtk.org/>

²<http://www.itk.org/>

³<http://openigtlink.org/>

⁴<http://www.qt.io/>

APÉNDICE A. DESCRIPCIÓN TÉCNICA DE BIBLIOTECAS

La figura A.1 muestra un esquema general de la ejecución del servidor PLUS. Se distinguen los principales flujos de datos y las relaciones entre las clases que componen la biblioteca. En este caso general, se distinguen dos posibles flujos de imágenes: una entrada de datos a partir de un dispositivo físico o de software (Video Input, en el diagrama), el cual entrega directamente imágenes de ultrasonido para su procesamiento y/o transmisión a través de PLUS, y una entrada de datos para el algoritmo de simulación de ultrasonido (Spatial Input), el cual genera imágenes a partir de esta información sin requerir de imágenes base; el mismo algoritmo genera las imágenes “base” que después serán posprocesadas para agregarle las características de una imagen obtenida mediante ultrasonido (por ejemplo, ruido). Para efectos de este proyecto no se requirió considerar el flujo de datos Video Input; solo se poseen los modelos 3D a entregar al algoritmo de simulación y, por tanto, se usarán las imágenes generadas por el mismo. De hecho, el ejemplo “*Vascular needle insertion simulation*” permite, en su archivo XML, elegir entre ambos flujos de imágenes, pero se utilizará la opción por defecto (el algoritmo de simulación de ultrasonido). La lógica de ejecución de este caso en particular se grafica en la figura A.2.

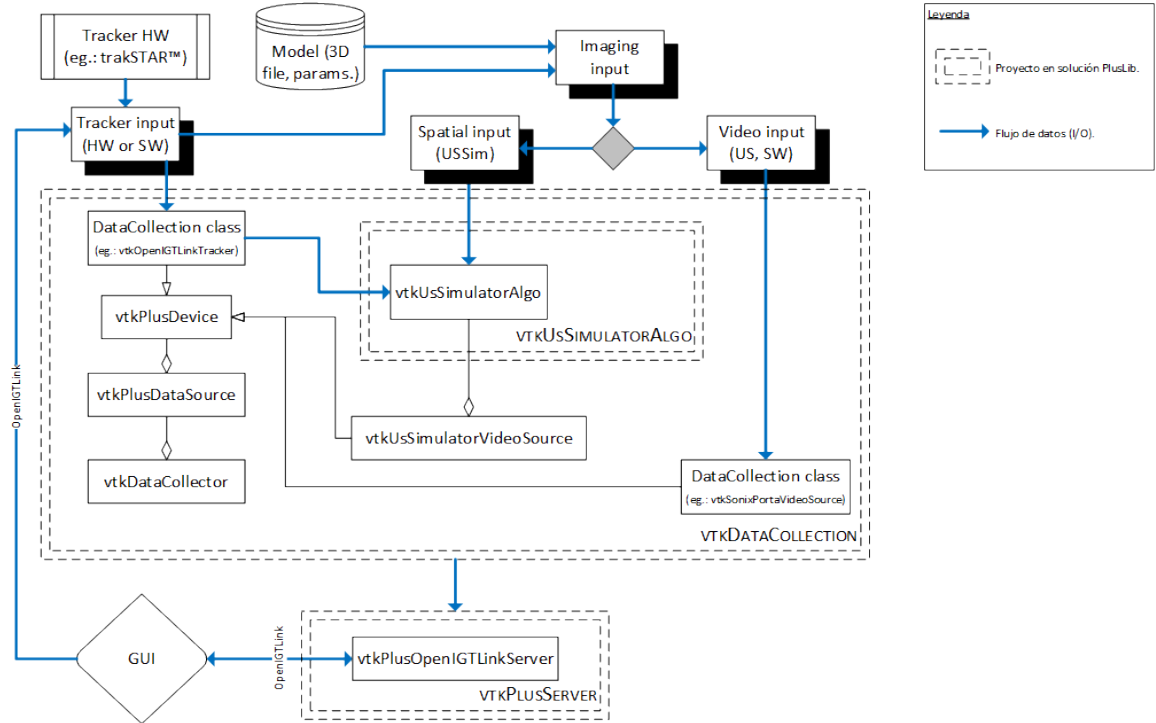


Figura A.1: Diagrama general de ejecución del servidor PLUS.

Fuente: Elaboración propia.

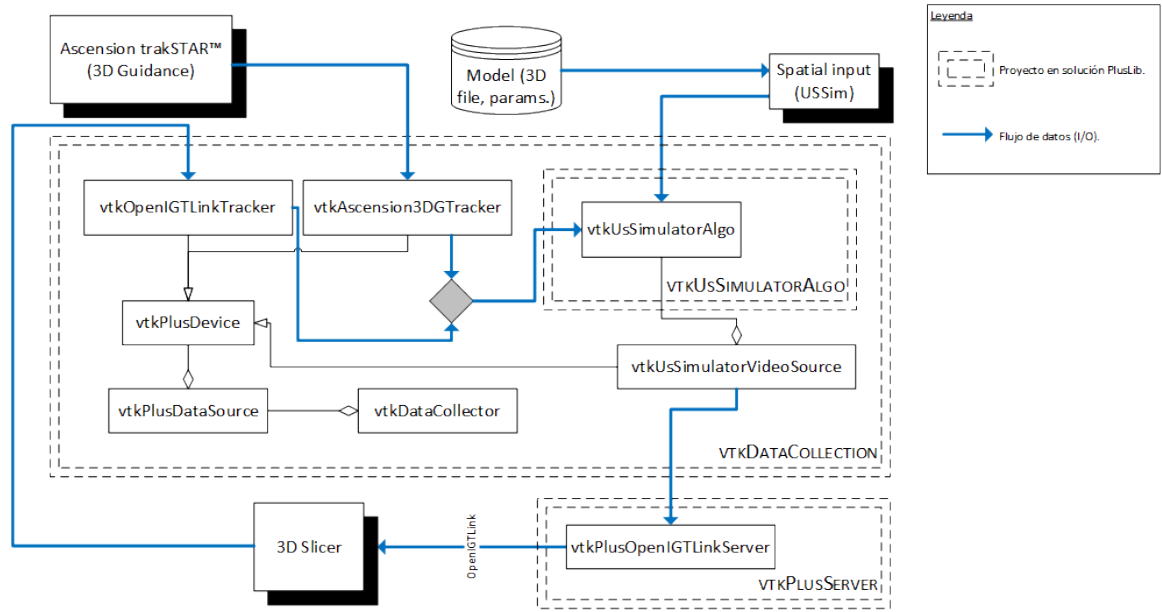


Figura A.2: Diagrama de ejecución - ejemplo “*Vascular needle insertion simulation*”.
Fuente: Elaboración propia.

A.1.1. Captura de datos de posicionamiento

En general, la estructura del archivo XML que configura un ejemplo incluye:

- Definiciones para un conjunto de “dispositivos”, los que pueden ser dispositivos físicos o unidades de software que simulen/implementen la funcionalidad de algún hardware en particular; por ejemplo, las fuentes de las imágenes y los datos de rastreo (tracking) que determinan la posición del sensor de ultrasonido en el modelo simulado.
- Definiciones de sistemas de coordenadas, tantas como sean necesarias para una simulación.
- Definiciones particulares de cada caso – información de calibración o de configuración de una conexión OpenIGTLink, por ejemplo.

En el caso del ejemplo estudiado, el archivo XML ofrece dos alternativas como fuente de datos de rastreo: un dispositivo virtual que recibe los datos de posicionamiento desde la aplicación de interfaz usuaria utilizada para ejecutar los ejemplos (llamada 3D Slicer⁵), mediante OpenIGTLink, y un dispositivo de simulación de rastreo basado en campos electromagnéticos llamado trakSTAR™.

⁵<http://www.slicer.org/>

APÉNDICE A. DESCRIPCIÓN TÉCNICA DE BIBLIOTECAS

El ejemplo “*Vascular needle insertion simulation*” define un flujo de datos de posicionamiento con 3 “canales”, toda vez que contiene tres objetos en la escena de simulación: la sonda de ultrasonido (el observador), un objeto genérico que actúa de referencia y un segundo que representa una aguja vascular. La implementación del servidor PLUS considera el uso de una sola fuente de datos de rastreo; esto es, en el caso de este ejemplo, se puede seleccionar la entrada desde 3D Slicer (OpenIGTLink) o desde el dispositivo trakSTAR™. Esta decisión de diseño de la biblioteca se refleja en varios niveles de profundidad – desde las clases `vtkOpenIGTLinkTracker` y `vtkAscension3DGTracker` hasta, al menos, la primera clase padre común que comparten (ver figura A.3), sin considerar algunas que se relacionan con las primeras, como `vtkUsSimulatorAlgo`.

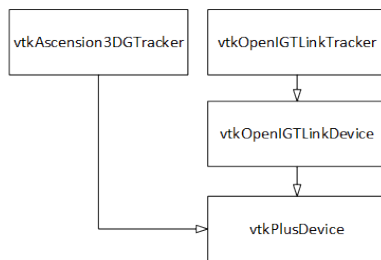


Figura A.3: Diagrama de clases de entrada de datos de rastreo.

Fuente: Elaboración propia.

El flujo de datos de posicionamiento generado por un dispositivo de rastreo tiene un nombre que PLUS espera que sea único; además, los elementos que requieren dichos datos esperan que exista un único flujo de datos – por ejemplo, si en la configuración de una simulación se definen dos flujos de datos de igual nombre, la instanciación del segundo genera el descarte de la primera instancia. Ambas características impiden configurar en el archivo XML respectivo más de un dispositivo de rastreo.

A.1.2. Lectura de datos de modelo 3D

La clase que recibe como parámetro un archivo de modelo 3D y lo ofrece al algoritmo para su utilización es `SpatialModel`. La figura A.4 muestra una versión simplificada de las relaciones entre las distintas clases que intervienen en la generación de una imagen sintetizada por el algoritmo de simulación. La documentación de `SpatialModel` indica que se soportan los formatos de archivo STL y VTP:

- El formato STL (*STereoLithography*)⁶, especificado para estructuras internas en ASCII (texto

⁶[https://en.wikipedia.org/wiki/STL_\(file_format\)](https://en.wikipedia.org/wiki/STL_(file_format))

APÉNDICE A. DESCRIPCIÓN TÉCNICA DE BIBLIOTECAS

plano) y binario, permite expresar únicamente la geometría de un modelo 3D. No permite almacenar información de texturas, color o animación.

- El formato VTP (*VTK PolyData*)⁷ es un archivo XML que representa un objeto de la clase *vtkPolyData*⁸, parte de VTK. Tampoco permite almacenar información de texturas, color o animación.

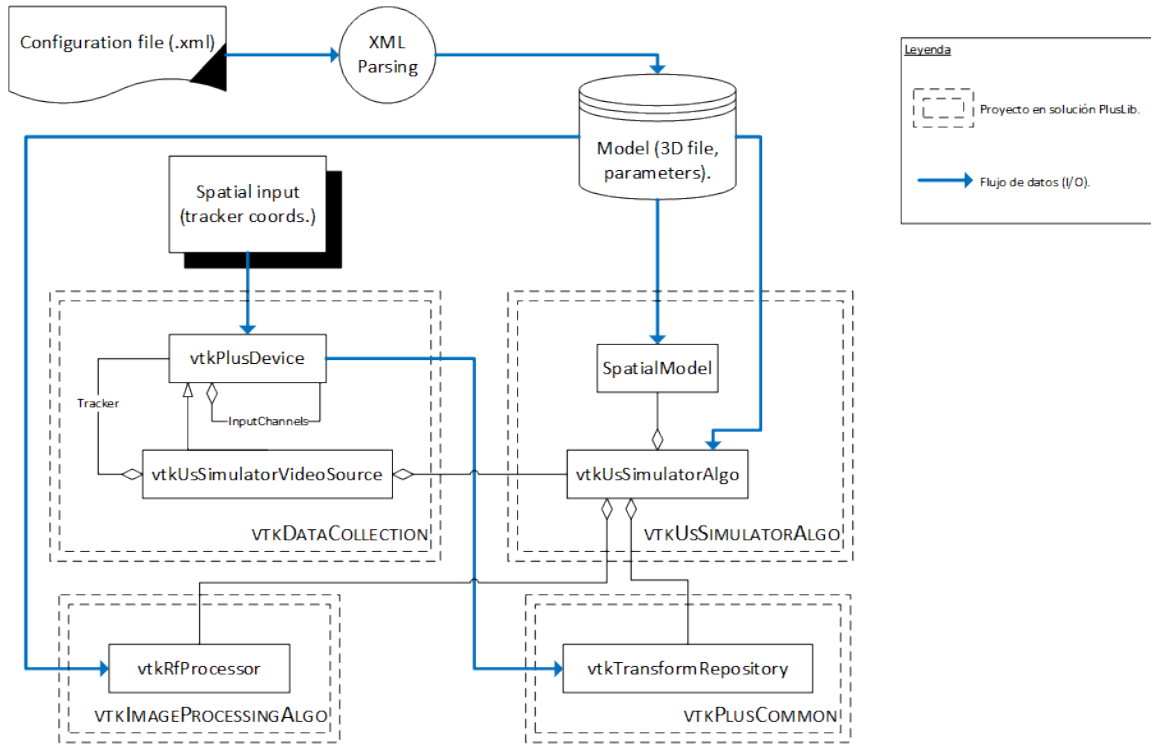


Figura A.4: Diagrama de relaciones para algoritmo de simulación de ultrasonido.

Fuente: Elaboración propia.

A.1.3. Recepción de datos mediante OpenIGTLink

Como se observa en la figura A.1, PLUS recibe información transmitida por OpenIGTLink por medio de las clases *vtkOpenIGTLinkServer* y *vtkOpenIGTLinkTracker*. En la figura A.5 se diagrama de forma más detallada la relación entre estas clases dentro de PLUS.

⁷<http://www.vtk.org/wp-content/uploads/2015/04/file-formats.pdf>

⁸<http://www.vtk.org/doc/nightly/html/classvtkPolyData.html>

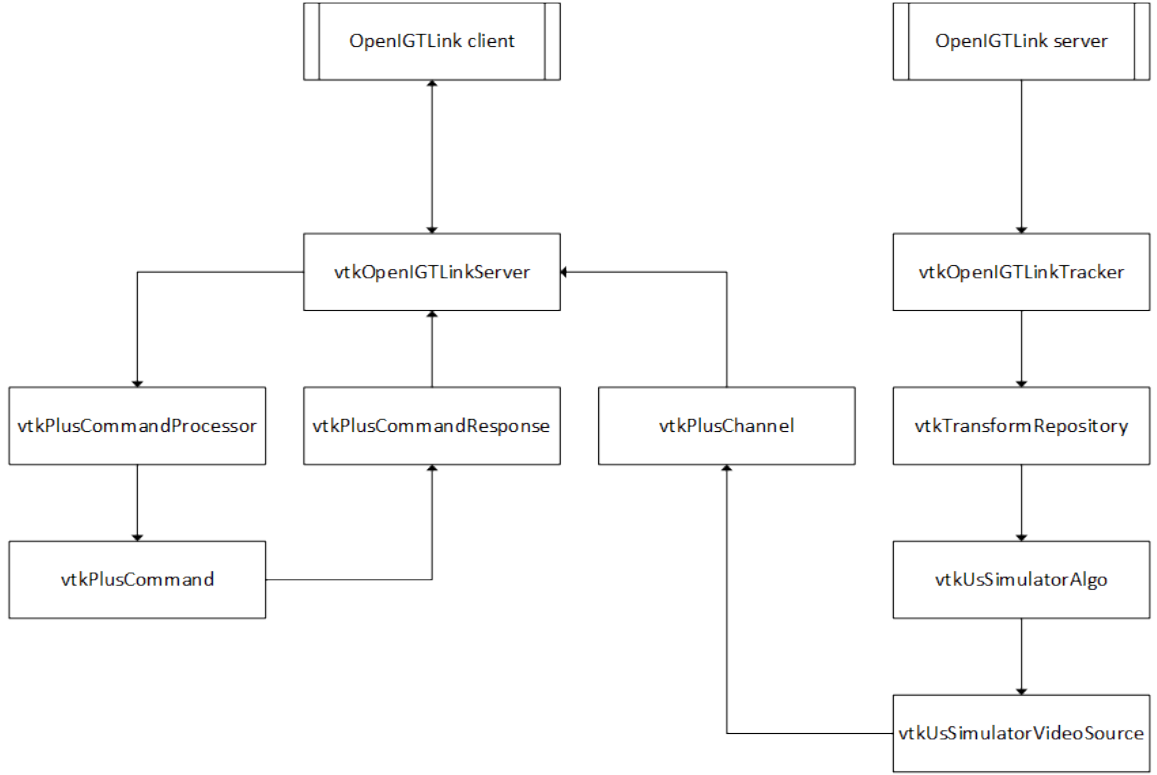


Figura A.5: Flujo de datos recibidos y enviados mediante OpenIGTLink.

Fuente: Elaboración propia.

Cuando PLUS actúa como cliente, recibe datos sobre transformaciones aplicadas a la escena 3D simulada a través de una hebra contenida en un objeto de clase `vtkOpenIGTLinkTracker`. En esta hebra se invocan las funciones `InternalUpdateTData()` y `InternalUpdateGeneral()`, en las cuales se maneja la recepción y procesamiento de mensajes según su tipo: en la primera función se aceptan mensajes de tipo `TDATA` – el cual, según la documentación de OpenIGTLink, indica la posición de una herramienta o instrumento de medición (por ejemplo, una sonda de ultrasonido) –, mientras que la segunda acepta mensajes de tipo `TRANSFORM` y/o `POSITION`: una matriz cuadrada de orden 4 y un vector de 3 dimensiones, respectivamente, que representan la posición en el espacio coordenado de la simulación de un modelo 3D y, en el caso de la matriz, datos sobre las transformaciones geométricas a las que ha sido sometido dicho objeto (rotación, traslación, escalamiento). Luego de procesados, estos datos son dejados a disposición del algoritmo de ultrasonido, el que procede a sintetizar una imagen de ultrasonido simulada a partir de los mismos. Esta imagen queda, luego, disponible para su transmisión.

Actuando como servidor, PLUS envía y recibe mensajes. Los mensajes enviados pueden corres-

APÉNDICE A. DESCRIPCIÓN TÉCNICA DE BIBLIOTECAS

ponder a las imágenes generadas por el algoritmo de ultrasonido o a las respuestas a comandos. Estos últimos son los mensajes que PLUS puede recibir en este rol, y corresponden a mensajes de tipo `STRING` que contienen una entidad XML cuya estructura está definida en la documentación de PLUS. Luego de recibido un comando, este es entregado a una clase procesadora que se encarga de determinar qué tipo de comando se ha recibido y de instanciar un objeto de la clase respectiva (heredera de la clase `vtkPlusCommand`) para que se ejecute la acción indicada y se genere la respuesta que corresponde para su posterior transmisión al cliente.

A.2. OpenIGTLink

OpenIGTLink es un protocolo de comunicación diseñado para utilizarse entre procesos y/o dispositivos involucrados en terapias médicas de imagen (como, por ejemplo, ecografías). De hecho, las dos aplicaciones más destacadas en el sitio Web de la biblioteca son, precisamente, 3D Slicer – como interfaz de visualización y manipulación de imágenes – y PLUS – como fuente de imágenes.

El protocolo trabaja con un modelo cliente-servidor, donde ambos son capaces de generar y recibir mensajes – de hecho, las lógicas relacionadas con los mensajes se implementan en una clase padre a ambos tipos de entidad en el modelo. A nivel lógico, la conexión entre distintas entidades se realiza mediante puertos de comunicación (en el caso de uso “*Vascular needle insertion simulation*”, por ejemplo, PLUS escucha como cliente el puerto 18946 y transmite como servidor hacia el puerto 18944).

Un análisis de la documentación de OpenIGTLink[28][29] permite determinar que la comunicación entre procesos se basa en la transferencia de “mensajes”, de los cuales existen una determinada cantidad de tipos que representan distintos contenidos y, por tanto, tienen asociados diferentes tratamientos. La identificación de un mensaje en particular se realiza mediante su tipo y, para distinguir entre mensajes de igual tipo, su nombre (*nombre de dispositivo*, en la terminología utilizada por OpenIGTLink). Existe la posibilidad de realizar transmisiones “a bajo nivel” que prescinden de esta diferenciación, para las cuales OpenIGTLink ofrece ciertas facilidades de implementación, pero tanto PLUS como 3D Slicer utilizan la comunicación de “alto nivel” basada en mensajes, por lo que resulta más práctico seguir el trabajo de desarrollo bajo este mismo paradigma, tanto para la fase actual de pruebas y estudio como para la de implementación.

El código fuente de la biblioteca que implementa este protocolo, de acceso público en el sitio Web de este último, incluye una batería de ejemplos que permiten revisar distintos casos de uso.

Para efectos de este trabajo, es conveniente analizar en detalle dos de los tipos de mensajes existentes

en OpenIGTLink.

A.2.1. Mensajes TRANSFORM

Los mensajes de tipo TRANSFORM, manejados mediante la clase `igt1TransformMessage` dentro de OpenIGTLink, contienen una matriz de transformación utilizada en computación gráfica tridimensional.

Para una explicación más detallada sobre qué es una matriz de transformación, es conveniente referirse a la documentación más específica al tema como, por ejemplo, la entregada por la API OpenGL⁹. En cualquier caso, para efectos de explicar las operaciones documentadas en este informe, vale considerar lo siguiente:

- Una coordenada en un espacio 3D corresponde a un arreglo de tamaño 3 (valores para X, Y y Z). Una dirección (vector) se representa numéricamente de igual forma. Para poder distinguir entre ambos usos, en computación gráfica se utilizan vectores de 4 dimensiones (X, Y, Z, W), donde $W = 1$ si el vector se utiliza como posición y $W = 0$ si se utiliza como dirección.
- En este esquema, las distintas operaciones que pueden aplicarse sobre un objeto en el espacio tridimensional se representan mediante matrices cuadradas de orden 4:

- Traslación.

$$\begin{bmatrix} 1 & 0 & 0 & T_x \\ 0 & 1 & 0 & T_y \\ 0 & 0 & 1 & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Escalamiento.

$$\begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

⁹<http://www.opengl-tutorial.org/beginners-tutorials/tutorial-3-matrices/>

- Rotación en el eje X.

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos \theta & -\sin \theta & 0 \\ 0 & \sin \theta & \cos \theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Rotación en el eje Y.

$$\begin{bmatrix} \cos \theta & 0 & \sin \theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin \theta & 0 & \cos \theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Rotación en el eje Z.

$$\begin{bmatrix} \cos \theta & -\sin \theta & 0 & 0 \\ \sin \theta & \cos \theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- La aplicación de sucesivas operaciones se representa, computacionalmente, como una multiplicación de matrices.

Con respecto al uso de la clase `igt1TransformMessage`, un objeto de esta clase se inicializa entregándole ya sea la matriz de transformación o la posición y 3 vectores normales que describan la transformación deseada.

A.2.2. Mensajes NDARRAY

Los mensajes de tipo NDARRAY, manejados mediante la clase `igt1NDArrayMessage` dentro de OpenIGTLink, contienen un arreglo multidimensional de datos numéricos.

Este tipo de mensajes puede contener arreglos de 1 o más dimensiones, con un máximo de 65.536 valores por dimensión (216, capacidad correspondiente a `unsigned short`). Los valores contenidos en el arreglo pueden ser enteros de 1, 2 o 4 bytes (con o sin signo), números de punto flotante de 4 u 8 bytes o 16 bytes que son interpretados como números complejos (donde los primeros 8 bytes corresponden a la parte real y los siguientes 8, a la parte imaginaria). La inicialización de un mensaje de este tipo requiere indicar el tipo de datos a enviar, las dimensiones y el tamaño por dimensión del arreglo, y los datos contenidos dentro de este.

Apéndice B

Descripción técnica de implementaciones

B.1. Aplicación OBJtoVTP

Esta aplicación de línea de comandos permite convertir un archivo de modelo 3D en formato OBJ, definido por Wavefront Technologies¹, a otro de formato VTP. Ambos formatos almacenan el mismo tipo de información (geometría y texturas, sin animaciones), por lo que no hay pérdida de datos en la conversión.

VTK ofrece dos ejemplos en particular, “ReadOBJ”² y “WriteVTP”³, liberados bajo licencia Creative Commons Attribution 2.5 Generic, los que sirvieron de base para la implementación de esta aplicación.

La ejecución de `OBJtoVTP` requiere el paso de un único argumento de línea de comandos, indicando la ruta del archivo a convertir. Este formato permite la utilización de la aplicación en un *script* para el procesamiento de varios archivos de entrada.

¹https://en.wikipedia.org/wiki/Wavefront_.obj_file

²<http://www.vtk.org/Wiki/VTK/Examples/Cxx/IO/ReadOBJ>

³<http://www.vtk.org/Wiki/VTK/Examples/Cxx/IO/WriteVTP>

B.2. Clase IGTLServer

Esta clase gestiona la conexión con el cliente de OpenIGTLink – en este caso, el servidor PLUS en su rol de recepción de datos desde la interfaz gráfica – y el envío de los mensajes con información generada por el usuario. La implementación de esta clase supone que solo existirá un cliente (la instancia respectiva del servidor PLUS) y que solo se enviarán mensajes de OpenIGTLink de imágenes y de datos numéricos.

La función que espera la conexión desde el cliente es “bloqueante”: la ejecución de la aplicación se detiene hasta que un cliente establece comunicación con el servidor. Por ello, esta parte de la inicialización se realiza en una hebra de ejecución aparte, de manera de no interrumpir la ejecución de los otros componentes de la aplicación (por ejemplo, los componentes gráficos de Qt). El inicio y término de esta hebra son completamente privados a la clase, en función de un tiempo de espera (*timeout*) especificado al momento de ordenar la espera de conexiones desde clientes por parte de quien utilice un objeto de esta clase (por defecto, el tiempo de espera es infinito). Una vez que se establece la conexión, se emite una señal mediante el mecanismo *signals and slots* de Qt⁴. La implementación de la hebra de ejecución fue basada en la biblioteca estándar de C++ (clase `std::thread`, disponible desde el estándar C++11⁵).

B.3. Clase IGTLClient

Esta clase gestiona la conexión con el servidor PLUS, en su rol de servidor de OpenIGTLink, y la recepción de los mensajes con la información de las imágenes generados por el algoritmo de simulación de ultrasonido. La implementación asume que la aplicación solo recibirá mensajes con la información de imágenes.

La función que realiza la conexión con el servidor OpenIGTLink también es bloqueante; sin embargo, como esta conexión es precondition para la correcta ejecución de la aplicación (sin ella, no hay imágenes para mostrar), la llamada a dicha función se realiza desde la hebra de ejecución principal – esto es, el flujo de la aplicación completa queda detenida hasta que se establece la comunicación. Además, este hecho sirve para sincronizar la ejecución del programa que utiliza esta clase con el servidor PLUS.

La recepción de mensajes desde el servidor también es de naturaleza bloqueante: la función res-

⁴<http://doc.qt.io/qt-5/signalsandslots.html>

⁵<http://en.cppreference.com/w/cpp/thread/thread>

APÉNDICE B. DESCRIPCIÓN TÉCNICA DE IMPLEMENTACIONES

pectiva detiene la ejecución de la aplicación hasta que exista un mensaje. Por ello, esta parte de la implementación se aloja en una hebra de ejecución propia, también basada en `std::thread`. El ciclo de ejecución, como se muestra en la figura B.1, consiste en iterar infinitamente (hasta que la aplicación termine), preguntando en cada ciclo si hay un mensaje disponible en el puerto que se está escuchando.

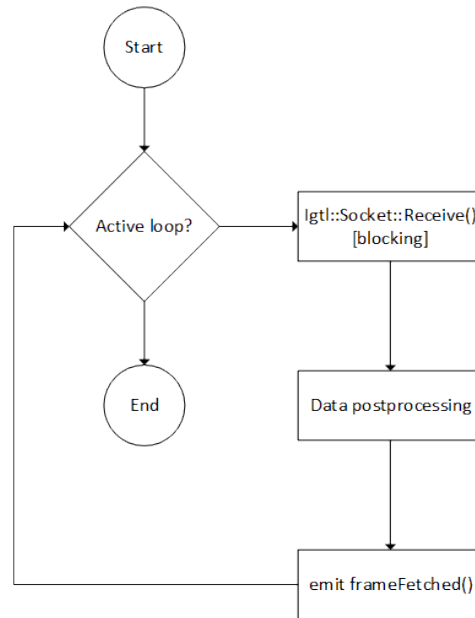


Figura B.1: Diagrama de flujo de ejecución de una instancia de `IGTLCClient`.

Fuente: Elaboración propia.

Una vez recibido un mensaje, el contenido del mismo es copiado a una nueva ubicación en memoria, para su posterior uso en el despliegue de la imagen sintetizada al usuario. Finalmente, una señal es emitida para avisar a la instancia de `UsSimulatorWidget` correspondiente de la disponibilidad de una nueva imagen para mostrar.

B.4. Clase `UsSimulatorWidget`

Esta clase gestiona la generación, a partir de datos entregados por una instancia de `IGTLCClient`, de la imagen mostrada al usuario. Esta clase es implementada como un *widget* dentro del framework Qt – esto es, como una clase heredera de `QWidget`⁶–, de manera que sea utilizable de manera directa en una aplicación gráfica implementada utilizando esta misma biblioteca.

⁶<http://doc.qt.io/qt-5/qwidget.html>

APÉNDICE B. DESCRIPCIÓN TÉCNICA DE IMPLEMENTACIONES

Los datos recibidos mediante OpenIGTLink corresponden a un arreglo de *bytes*, de tamaño $W \times H$ para una imagen de W píxeles de ancho y H píxeles de alto. Cada *byte* representa la posición de un píxel en la escala de grises (donde 0 es completamente negro y 255, completamente blanco). Este arreglo de datos es utilizado para instanciar un objeto de clase `QImage`, perteneciente a Qt⁷, la que soporta la generación de una imagen a partir de un arreglo de *bytes* interpretados en una gama de formatos (ej.: 4 *bytes* para cada píxel en formato RGBA – escala de colores RGB y transparencia, o canal alfa), entre ellos la misma escala de grises utilizada por PLUS. Esto permite el paso de los datos a Qt sin preprocesamiento, más allá de realizar una copia de los mismos para evitar la destrucción del arreglo antes de que sea completamente desocupado por las clases encargadas del despliegue de la imagen en pantalla. El objeto `QImage` instanciado es utilizado de base para el despliegue del objeto `UsSimulatorWidget` en pantalla⁸. Este objeto es generado cada vez que se recibe un nuevo mensaje de imagen mediante OpenIGTLink. Además, se almacena en memoria mientras no se reciba un nuevo mensaje, para reducir el costo computacional de la actualización del despliegue del *widget* en pantalla (lo que ocurre cada vez que, por ejemplo, la aplicación gráfica contenedora es movida, minimizada o maximizada).

Además de los requisitos propios para la instanciación de un objeto de clase `QWidget`, los objetos de esta clase requieren la dirección y el puerto que han de usarse para establecer la conexión, como cliente de OpenIGTLink, con el servidor PLUS.

B.5. Aplicación UsSimulatorGUI

Esta aplicación corresponde a la interfaz gráfica que el usuario utiliza para observar las imágenes sintetizadas por PLUS y enviar datos a este último en tiempo de ejecución, correspondientes a los parámetros de configuración del algoritmo de simulación de ultrasonido.

La ejecución de esta aplicación implica la realización de tres pasos:

1. La ejecución del servidor PLUS, entregándole a este la ruta del archivo de configuración XML correspondiente.
2. La lectura de este mismo archivo de configuración para la obtención de los datos para la instanciación del cliente y servidor de OpenIGTLink.
3. El despliegue de la ventana principal de la aplicación.

⁷<http://doc.qt.io/qt-5/qimage.html>

⁸<http://doc.qt.io/qt-5/qpainter.html>

B.6. Clase UsSimulatorGUI

Esta clase corresponde a la ventana principal de la aplicación gráfica del mismo nombre. Implementada en base al *framework* Qt, esta clase es heredera de `QMainWindow`⁹.

Además de una instancia de `UsSimulatorWidget`, un objeto de esta clase contiene una serie de *widgets* que permite la entrada y modificación de valores numéricos. Mediante el mecanismo de *signals and slots* de Qt, la entrada de un nuevo valor desencadena la generación y envío de un mensaje de OpenIGTLink con el contenido respectivo, el cual es enviado mediante una instancia de `IGTLServer` hacia el servidor PLUS.

⁹<http://doc.qt.io/qt-5/qmainwindow.html>