HERRAMIENTA SOFTWARE PARA LA PRÁCTICA Y EXPERIMENTACION DE LA ROBÓTICA QUIRÚRGICA

Diego Enrique Guzmán Villamarin

Universidad del Cauca

Facultad de Ingeniería Electrónica y Telecomunicaciones

Departamento de Electrónica, Instrumentación y Control

Ingeniería en Automática Industrial

Popayán, Agosto 2013

HERRAMIENTA SOFTWARE PARA LA PRÁCTICA Y EXPERIMENTACIÓN DE LA ROBÓTICA QUIRÚGICA

Monografía presentada como requisito parcial para optar por el título de Ingeniero en Automática Industrial

Diego Enrique Guzmán Villamarin

Director: Dr. Oscar Andrés Vivas Albán

Universidad del Cauca

Facultad de Ingeniería Electrónica y Telecomunicaciones

Departamento de Electrónica, Instrumentación y Control

Ingeniería en Automática Industrial

Popayán, Agosto de 2013

Nota de aceptación: _____

_____

_____

_____

_____

_____

Director: _____

Oscar Andrés Vivas Albán

_____

Firma del Jurado

_____

Firma del Jurado

Popayán, Agosto de 2013

Agradecimientos

Primero que todo quiero agradecer a Dios, quien me ilumino a través de este camino. Además puso en mi vida muchas personas quienes me ayudaron a formarme profesionalmente y como persona. Estos logros los describiría como dice una frase que vi a un jugador de futbol “no soy yo, es la gracia de Dios a través de mi”.

Aprovecho este espacio para una dedicatoria especial a mi abuelo Enrique Lemos, quien falleció el 10 de junio de 2013. Siempre fue una persona importante en la familia, nos enseño la importancia de ser un ser humano integro, aunque cascarrabias siempre pensó en lo mejor para sus nietos para que fuesen personas útiles en la sociedad, siempre lo querremos y lo llevaremos en nuestros corazones.

Agradezco también a mis padres Diego Guzmán y Victoria Villamarin, quienes apoyaron y me formaron en todo este proceso de hacerme un ciudadano responsable, me enseñaron, me guiaron por el buen camino y son una fuente de inspiración. A mis hermanos Juan David y Santiago quienes estuvieron ahí animándome en muchos momentos. A mi abuela Victoria Iragorri le agradezco por su constante perseverancia al mostrarme la importancia de hacer las cosas bien y pensar siempre en grande. A mi abuela Olivia, mis tíos, tías y primos quienes siempre me apoyaron.

Agradezco a Daniel Rosero, amigo y colega, quien me ayudo y enseño a usar Blender para dimensionar los objetos 3D en la herramienta. Juntos aprendimos a codificar y usar las herramientas software necesarias para el desarrollo de nuestros respectivos trabajos de grado y fue un valioso apoyo en el transcurso del proyecto.

Agradezco a mis amigos y personas más allegadas, con quienes pase los mejores momentos cuando el estrés se acumulaba, e hice muchos buenos recuerdos con ellos. Agradezco a aquellos compañeros y amigos en la u, con quienes desarrollamos muchos proyectos, aprendimos y adquirimos las habilidades necesarias para ser buenos ingenieros.

Agradezco a mi director Oscar Vivas, por su guía en el desarrollo de este proyecto de grado.

Agradezco a la Universidad del Cauca, especialmente a la Facultad de Ingeniería Electrónica la cual con sus docentes y administrativos siempre han estado a nuestro servicio.

Agradezco a los evaluadores de este proyecto que con su visión incrementarán el aporte científico de este proyecto.

Resumen

Indiscutiblemente, la labor de un ingeniero debe estar orientada a resolver las necesidades de potenciales usuarios, a permitirles hacer fácil lo que es difícil. Experimentar el manejo de instrumental sobre un cuerpo humano, buscando además técnicas mínimas invasivas, es un riesgo sobre los resultados que puedan surtir, por ello el desarrollo de ambientes virtuales, herramientas informáticas que trasladen este riesgo, y permitan aprender del error con el mínimo de recursos disponibles, es indiscutiblemente la mejor manera en la que un ingeniero puede llegarse a lucir.

En este proyecto, y pensando en la posibilidad de generar un aprendizaje autónomo o independiente (al menos para un mayor grado posible), se desarrolló una herramienta software académica que permite a estudiantes practicar y experimentar con robots quirúrgicos en un ambiente virtual 3D, permitiéndoles no solo conocer las posibles repuestas de comportamiento, sin contar con el hardware para ello (robot), sino también de su comportamiento dentro del espacio de un cuerpo humano (sin poner en riesgo a uno de verdad). Para ello se estudiaron trabajos de grado anteriores, revisando sus potencialidades, adaptándolos a la necesidad hacia la cual estaban enfocados e integrándolos bajo una misma plataforma.

Es importante demostrar que las necesidades de usuarios que aún persisten en este mundo, pueden ser atacadas desde la visión del ingeniero, proponer soluciones, facilitar la labor. Para el caso en concreto se trabajó en el tema de las cirugías mini-invasivas a nivel abdominal por medio de una técnica quirúrgica conocida como la laparoscopia, proyectando el uso de robots previamente modelados y llevados a simulación como el robot porta-endoscopio Hibou y el robot quirúrgico LapBot. Para la herramienta se aprovechó el modelo geométrico inverso de cada robot, integrándolos a un espacio único, espacializando la ubicación de sus piezas. El movimiento de sus articulaciones considera un punto de incisión por tratarse de una cirugía mínima invasiva, el cual es representado en coordenadas cartesianas y a partir del mismo se simula el movimiento en el espacio de trabajo (para el caso el abdomen del paciente).

La codificación de esta herramienta software se llevó a cabo en un entorno de desarrollo integrado conocido como “Microsoft Visual C++ 2008” para la codificación como tal, y se apoyó en el uso de otras herramientas informáticas de desarrollo como “Blender” para el modelado en 3D, “Qt creator” para el desarrollo de la interfaz y “Make Human” para los pacientes. Una herramienta clave en el proyecto fue el uso de una biblioteca en C++

compuesta por un conjunto de clases para la visualización de gráficos 3D por computador llamada VTK.

Y finalmente se obtuvo el software RoboSurgery, llamado así por su desarrollador, ya que este permite realizar procedimientos quirúrgicos en un espacio simulado utilizando robots.

Abstract

No doubt, the work of an engineer must be guided to solve the needs of potential users, he has to allowed to make easy doing what is difficult, experience the handling of instruments on a human body, seeking in addition minimal invasive techniques, is a risk outcomes that may be valid, therefore the development of virtual environments, computer tools that move this risk, earning from mistakes with the minimum of available resources, It is unquestionably the best way in which an engineer can shine.

In this project, and having the possibility to generate an autonomous or independent learning, (at least to a greater extent possible), is developing an academic software tool that allows students to practice and experiment with surgical robots in a 3D virtual environment, allowing them to not only know the possible behavior, without having the hardware to do so (robot), but also their behavior within the space of a body (without putting at risk of a human). To deploy this work we seek on previous degree works, reviewing their potential, adapting them to the needs to which they were focused, integrating them to a single geometric model.

It is important to me, demonstrate that the user needs that still in this world, can be attacked from the point of view of an engineer, propose solutions, facilitate the work. For this specific case, it allowed me to work on the issue of mini-invasive abdominal surgery by a surgical technique known as laparoscopy, projecting the use of previously modeled and led to a simulation robot, like the endoscope carrier Hibou and the surgical robot LapBot. The tool took advantage of each robot inverse geometric model, integrating them into a single space, spacializing the location of their parts. The movement of their joints considered a point of incision because it is a minimal invasive surgery, which is represented in Cartesian coordinates and from it simulates movement in the workspace (in case the patient's abdomen).

The coding of this software tool was carried out in an integrated development environment known as "Microsoft Visual C++ 2008" for it self, and relied on the use of other tools of development as "Blender" for modeling in 3D, "Qt creator" for the development of the interface, "Make Human" for patients. A key tool in the project was the use of a C++ library composed of a set of classes called VTK computer 3D graphics display.

And finally get the software called RoboSurgery, by its developer, since this allows surgical procedures in a simulated space using robots.

CONTENIDO

LISTA DE FIGURAS

LISTA DE TABLAS

INTRODUCCIÓN

No todas las personas aprenden de la misma manera. A unos el estímulo visual les llega primero, a otros el auditivo, el táctil; o la combinación de todos, técnica conocida como la nemotécnica. Igualmente el uso de técnicas de aprendizaje como el aprendizaje autónomo, le permite al interesado en un ambiente muy distinto al escolar, identificar la necesidad de aprendizaje y decir como satisfacerla. Ésta se realiza sin la presencia física de un profesor, aunque preferiblemente bajo la orientación de quien conozca del tema, aunque no sea bajo el seguimiento de un programa de estudio, dándole importancia al combinar el aprendizaje casual con el aprendizaje programado.

De una forma u otra permanentemente llevamos a cabo este tipo de estudios, cuando vamos a dar uso a un aparato nuevo y acudimos al manual, o cuando observamos a alguien para luego intentar realizar su labor. Es decir que no estamos tratando con un término nuevo.

En él cada persona aprende y se desarrolla de manera distinta y a ritmo diferente, se aplica o se experimenta el aprendizaje con la realidad, para el caso virtual, es muy importante desarrollar un aprendizaje autónomo pues la vida siempre está cambiando y siempre habrá algo nuevo que aprender. El estudiante desarrolla la habilidad o la capacidad de relacionar los problemas por resolver, buscar la información necesaria, analizar, generar ideas, sacar conclusiones y establecer el nivel de logro de sus objetivos.

El estudiante autónomo es emocionalmente independiente, tiene su auto-aprobación y a medida que acuda el profesor estará empleando un menor grado de autonomía.

Indiscutiblemente que el proceso de aprendizaje autónomo es para adultos, es decir, personas capaces de auto dirigirse.

Los cambios permanentes en el mundo actual obligan a aprender de forma autónoma, o en su defecto resignarse a permanecer desactualizado, sin embargo este cambio permanente en la tecnología permite el aprendizaje desde casa, sin grandes inversiones. Avances en el desarrollo de nuevas herramientas informáticas, permiten que se preste un soporte lógico suficiente para simular distintos ambientes y evaluar posibles resultados sin

necesidad de ir directamente al campo de trabajo, con los riesgos y costos que esto implicaría, apoyando así un proceso de aprendizaje autónomo, económico, sin riesgo.

Vale la pena mencionar que así como existen diferentes filosofías pedagógicas también existen diferentes enfoques para el desarrollo de herramientas de software educativo, por lo que este proyecto de grado se enfocó, hacia la instrucción asistida por computadora, que consiste en una secuencia de lecciones o módulos de aprendizaje.

El presente proyecto pretende realizar una aproximación inicial en el desarrollo de una herramienta software que permita observar en un entorno 3D el comportamiento de algunos robots modelados en la Universidad del Cauca, para permitir a los estudiantes de Ingeniería en Automática Industrial desarrollar prácticas en las que puedan simular cirugías asistidas por robots con el objetivo de darles a conocer cómo trabajan estos sistemas de robótica médica.

Para empezar hay que entender que la robótica como tecnología es multidisciplinaria, combina disciplinas como la mecánica, la electrónica, la informática, la inteligencia artificial, la ingeniería de control y la física, buscando diseñar y construir robots para diferentes aplicaciones, como la robótica industrial, o la robótica medica que viene siendo el campo de interés en este proyecto.

Hace unas pocas décadas, la cirugía era considerada un acto supremo donde el cirujano, mitad mito, mitad hombre, trataba de sanar a un enfermo. A mayor complejidad en una operación más importante era el cirujano, fundando la creencia de: «A grandes cirujanos, grandes incisiones». Era impredecible el cambio que esta especialidad requería. La tecnología, que marcó la segunda mitad del siglo XX, revolucionaría la cirugía en el primer cuarto del siglo XXI. Actualmente se opera sin abrir a las personas, «cirugía laparoscópica». Mediante pequeñas incisiones de 5 milímetros difícilmente perceptibles, y lentes de aumento que magnifican la zona donde se realiza la operación, se hace la cirugía más precisa y menos agresiva, con lo que se obtiene un mejor resultado. En los últimos diez años el cirujano ha pasado de «bisturí, tijera e hilo», a modernos sistemas de coagulación, láser y máquinas que cortan y suturan en un mismo gesto. Se utilizan instrumentos que permiten vaporizar los vasos y las arterias sin que estas sangren. La renovación constante de la tecnología ha permitido realizar todo esto sin abrir al paciente, minimizando el dolor y las cicatrices. Lo que hace veinte años parecía ciencia ficción, hoy es una realidad de uso cotidiano en los hospitales y centros tecnológicamente avanzados.

Es de anotar que en el campo de la robótica médica la precisión y exactitud juega un importante papel durante las intervenciones quirúrgicas, por lo que hay que destacar 2 técnicas, el desarrollo en la telecirugía, permitiéndole al cirujano realizar el procedimiento sin necesidad de desplazar al paciente; y la cirugía mínimamente invasiva que permite sanar sin necesidad de generar daños innecesarios. El avance que se logre con los robots usados en intervenciones quirúrgicas mediante herramientas informáticas, permitirán corregir las imprecisiones y errores humanos.

Los sistemas para robótica quirúrgica están compuestos tal como se puede observar en la Figura 1 por:

- El puesto maestro, donde hay un dispositivo de mando tipo joystick para controlar el robot y pantallas para visualizar imágenes (video en tiempo real) y datos.
- El quirófano robotizado y equipado con los sensores que le ofrecen los datos al puesto maestro donde se convierten en información útil para el cirujano.
- La red de comunicación entre los dos componentes anteriores.

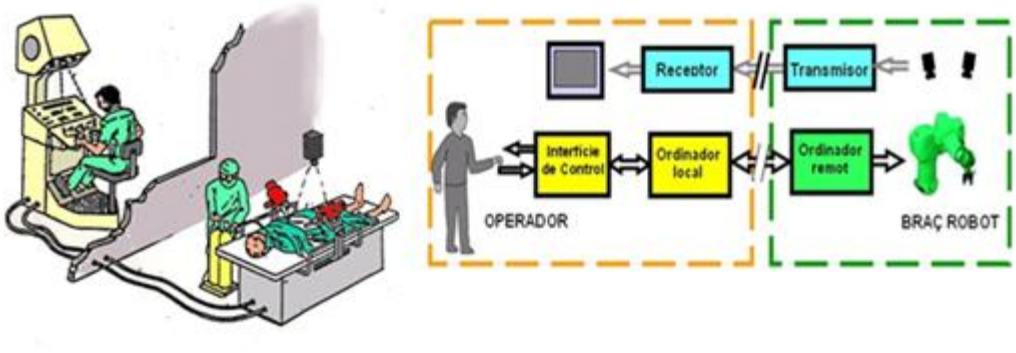

Figura 1: Componentes de un sistema de robótica quirúrgica [1].

Para el desarrollo de esta herramienta software, se definieron algunos objetivos y especificaciones funcionales pero durante el desarrollo como tal se fueron modificando y adaptando las especificaciones de la herramienta software a medida que se iba conociendo la capacidad de las herramientas de desarrollo usadas.

Este proyecto de grado involucra los componentes mostrados en la Figura 2. En esta figura se observa cómo por medio de una interfaz se pretende realizar una herramienta software que presente un entorno manipulado a través de un joystick y toda la información sea observada por el usuario a través de un computador.

Figura 2: Componentes de la herramienta software propuesta.

1. HERRAMIENTA SOFTWARE DE ROBOTICA Y SIMULACIÓN QUIRURGICA.

En este capítulo se presenta un estado del arte referente a la temática de este proyecto.

1.1. Herramienta software existente en el mercado.

Existen en el mercado algunas herramientas software para simular de manera virtual prácticas quirúrgicas (como la laparoscopia).

1.1.1. Herramientas software de simulación en medicina.

LAP MENTOR ™ [2]:

Este simulador proporciona el más alto nivel de formación quirúrgica, destaca entre los simuladores de laparoscopia pues proporciona una solución completa para la formación de los alumnos de todos los niveles y en todas las disciplinas. Cuenta con la opinión de expertos en cirugía, sociedades médicas y educadores con el fin de desarrollar un producto de mayor impacto.

Dispone de bibliotecas con módulos en constante desarrollo, proveen de una capacitación completa en procedimientos básicos y avanzados. El sistema incluye diferentes procedimientos de laparoscopia y pacientes en diferentes escenarios. Además incluye múltiples procedimientos quirúrgicos (incluyendo ginecología, urología y cirugía general).

Cabe anotar que continuamente se están desarrollando módulos para mantenerse a la par con los avances en las cirugías. Su tecnología única garantiza un alto grado de fidelidad en complejos procedimientos pero en un ambiente seguro para el aprendiz, permitiendo además la experiencia táctil cuando se usan las herramientas, sintiendo la resistencia durante la simulación de una cirugía.

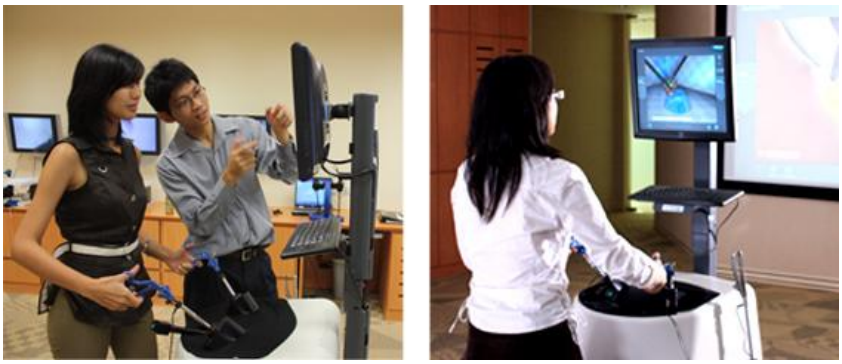

Figura 3: LAP MENTOR [2].

LapSim® [3]:

Este sistema digital de simulación laparoscópica ofrece una experiencia de aprendizaje eficaz, al permitir a los residentes practicar las habilidades básicas de la cirugía laparoscópica en un entorno de trabajo realista.

El entrenamiento básico incluye la navegación con cámara, la navegación con instrumental, la coordinación, grapado, disección, colocación de clips, sutura y medición de precisión (y velocidad).

Figura 4: LapSim® Laparoscopic trainer [3].

ProMIS [4]:

Este simulador quirúrgico para entrenamiento de cirujanos, posee 4 módulos: laparoscopia, manejo de instrumentos, disección y suturado. En la Figura 5 se muestra el simulador ProMIS para laparoscopia.

Figura 5: ProMIS simulator [4].

Existen otras herramientas software de simulación en el área de la robótica.

1.1.2. Herramientas software de simulación en robótica.

Cosimir [5]:

COSIMIR es una herramienta software de simulación 3D, se usa para planear las células de trabajo con los robots inmersos, para comprobar la accesibilidad de todas las posiciones, para desarrollar programas y controladores para robots, y para optimizar el diseño de la célula de trabajo. Funciona bajo sistemas operativos Windows 95/98 y Windows NT/2000.

Los movimientos de operación y manipulación pueden ser simulados, para evitar colisiones y optimizar tiempos de ciclo. Además permite la descarga del programa y el controlador con el robot real. Otra característica de COSIMIR es que permite importar otros sistemas CAD, modelados en herramientas como AutoCAD.

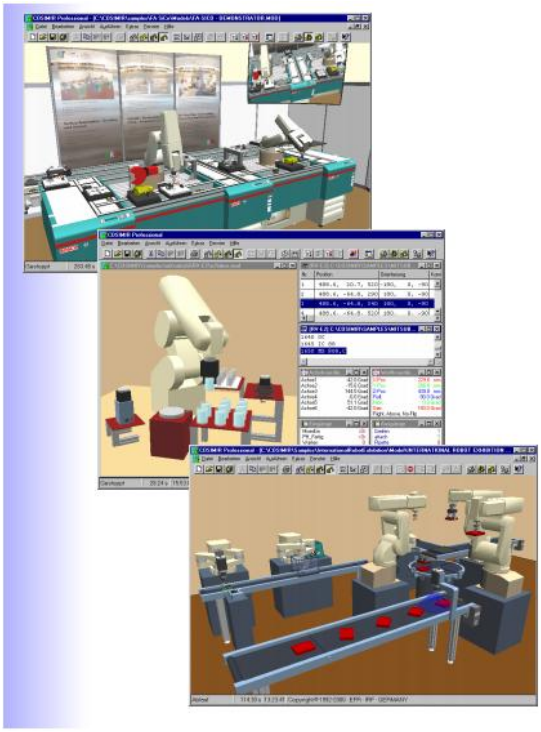

Figura 6: COSIMIR [5].

RoboCell [6]:

Esta herramienta software permite simular y observar un espacio 3D robotizado y con ambiente industrial, los movimientos y ejecución de programas para robots. RoboCell se integra plenamente con Scorbase, que es un paquete software de Windows de control y programación de robots.

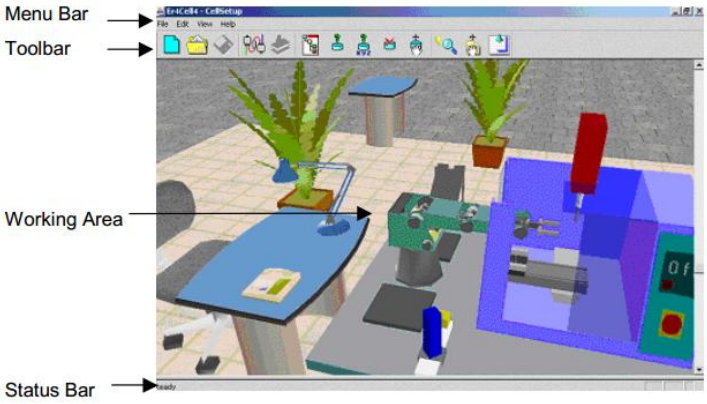

Figura 7: RoboCELL [6].

Easy-Rob [7]:

Permite desde un computador personal planificar y verificar las células de trabajo del robot. La planificación es altamente fiable ya que comprueba la accesibilidad, las colisiones según los rangos de viaje y estima los tiempos de ciclo.

Easy-Rob posee una potente API[1], desarrollada durante los últimos años, que permite personalizar la herramienta a la necesidad de casi cualquier cliente.

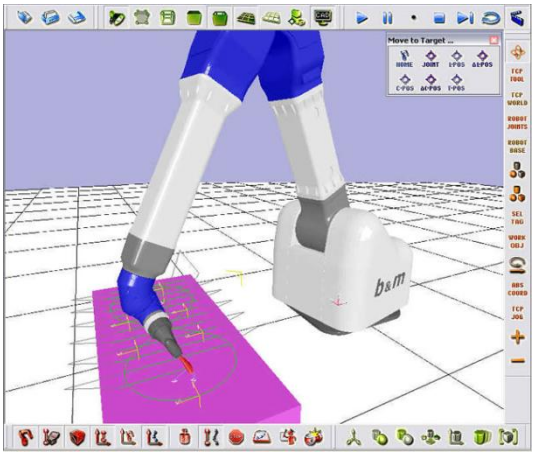

Figura 8: Easy-Rob [7].

[1]API: Interfaz de Programación de Aplicaciones.

RoboWorks [8]:

Esta herramienta software permite animar un modelo 3D por medio de un archivo '.dat' creado en herramientas como Matlab, LabView, etc. Es ideal para el modelado 3D y animación de robots y sistemas mecánicos, sirve de herramienta para enseñanza de ingeniería en pregrado y postgrado. Tiene una amplia colección de modelos de robots industriales.

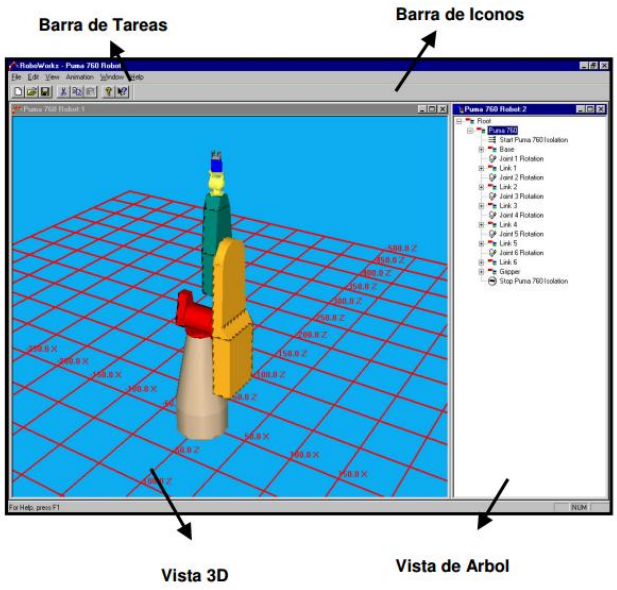

Figura 9: RoboWorks [8].

Se puede observar que hay una gran diferencia entre las herramientas software de medicina y las de robótica, pues sus enfoques son diferentes. Por esto este proyecto de grado propone el desarrollo de una primera versión de una herramienta software que muestre los robots y su comportamiento en el entorno quirúrgico.

2. CONCEPTOS GENERALES.

En este capítulo se presentan algunas definiciones de las temáticas tratadas a lo largo de este trabajo de grado. Teniendo en cuenta que este proyecto tiene fines académicos, y que en los proyectos de pregrado en la Universidad del Cauca se han desarrollado trabajos de grado sobre laparoscopia, lo que se pretende lograr en este proyecto es poner estos bajo una sola plataforma.

2.1. Cirugía Mini-invasiva (MIS[2]):

Esta técnica de intervención muy común hoy en día, evita abrir las cavidades del paciente para realizar una cirugía cerrada y local. Mediante unas pequeñas incisiones que pueden hacerse en el torax, en el abdomen (nuestro lugar de interés), en una articulación, etc, el médico introduce un equipo de visión (normalmente un endoscopio[3]) el cual envía imágenes de las estructuras internas a un monitor, como también otros instrumentos, los cuales por manipulación externa permiten realizar la exploración de los órganos contenidos en la cavidad y operar sobre ellos. Las principales ventajas y limitaciones de la MIS se pueden observar en la Tabla 1:

Ventajas	Limitaciones
No precisa de heridas importantes para acceder a la zona a operar.	No todos los órganos y tejidos del cuerpo humano pueden ser operados con estas modernas técnicas.
Post-operatorio más corto y menos doloroso.	A nivel técnico los instrumentos no pueden acceder a determinadas regiones del organismo.
Al paciente se le da de alta e incorpora a la vida normal más rápidamente.	Realimentación de fuerzas
Riesgo de infecciones y hernia es menor.	Visión 3D
Todas las limitaciones van desapareciendo día a día.	Ergonomía

Tabla 1: Ventajas y limitaciones de la MIS.

[2]MIS: Minimally Invasive Surgery.
[3]Endoscopio: Instrumento en forma de tubo que contiene una luz y una óptica que permite visualizar el interior de un órgano.

Tipos de cirugía MIS hoy en dia [9]:

Existen casi tantos tipos como especialidades quirúrgicas. Entre ellos:

Angioscopia, permite la visión endoscópica del interior de los vasos sanguíneos (arterias y venas), permite pequeñas operaciones como identificar y eliminar pequeños coágulos y placas de grasa. Realizada por primera vez en el año 1993. Artroscopia, permite examinar el interior de una articulación y realizar operaciones como la operación de menisco y de ligamentos. La primera fue en el año 1982. Colposcopia, examen de la vagina mediante un endoscopio especialmente diseñado que se introduce en la misma, permite cirugías de tumores o infecciones de órganos. Culdoscopia, examen de los órganos pélvicos femeninos (trompas, ovarios y útero). Con el uso de un endoscopio que se introduce en la cavidad abdomino-pélvica a través de la pared posterior de la vagina. Cistoscopia, examen endoscópico de la vejiga de la orina, permite tratar enfermedades de próstata, vejiga y uréteres sin abrir desde hace muchos años. Endoscopia digestiva, desde 1991 se usa para tratamiento y diagnostico mini-invasivos de un sinfín de enfermedades. Y la Laparoscopia, en este procedimiento se introduce en la cavidad abdominal un tubo largo y fino con una camera de video a través de una pequeña incisión en los alrededores del ombligo. Esto permite ver los órganos internos, e interactuar quirúrgicamente sobre ellos.

Entre muchos otros tipos, cabe mencionar la cirugía video-asistida, que permite realizar el procedimiento quirúrgico con visión trasmitido por endoscopio estando el cirujano y el paciente en diferentes ubicaciones espaciales. La combinación de video en tiempo real se combina con las imágenes de una tomografía axial computarizada (T.A.C.[4]) o de resonancia magnética[5], técnicas practicadas por primera vez en el año 2000.

Los procedimientos en la cavidad abdominal son tratados con la **laparoscopia [10]**, hay que decir que esta técnica permite la visión de la cavidad pélvica-abdominal con la ayuda de una lente, la cual permite observar imágenes del interior. Con el mismo principio que se realiza esto también se pueden realizar intervenciones quirúrgicas, por lo que también se considera un sistema de cirugía mínimamente invasivo para curar o prevenir enfermedades. El aparato usado para la toma de imágenes se llama torre de laparoscopia y por medio de una pequeña incisión (de 0.5 a 1.5 cm) ingresa al cuerpo.

[4]T.A.C.: También llamada escáner, que utiliza radiación X para obtener cortes o secciones de objetos anatómicos con fines de diagnóstico.
[5]Imagen por resonancia magnética: técnica no invasiva para obtener información sobre la estructura y composición del cuerpo.

2.2. Laparoscopia.

Originalmente este procedimiento fue descrito en 1901, y se popularizó a partir de la primera colecistectomía laparoscópica en 1987 [11]. Hay que resaltar que la laparoscopia, siendo una técnica mini-invasiva presenta las ventajas mencionadas en la Tabla 1.

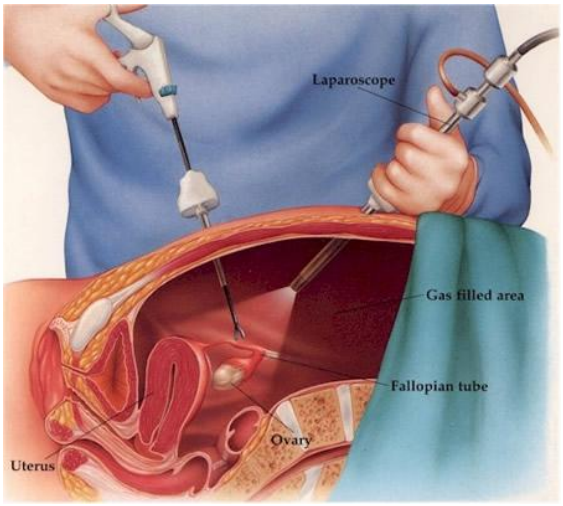

Figura 10: Laparoscopia [11].

En laparoscopia es necesario separar la pared abdominal de los órganos, elevando el abdomen, para que el cirujano tenga acceso al interior del paciente y pueda manipular los instrumentos con relativa facilidad. La elevación se realizó en un principio, con insuflación por aire, después se uso óxido nitroso y actualmente se emplea CO_2, dado que suprime la combustión y el cuerpo lo absorbe con mayor rapidez [12]. Un ejemplo del uso de laparoscopia es la colecistectomía, que se analizara más adelante.

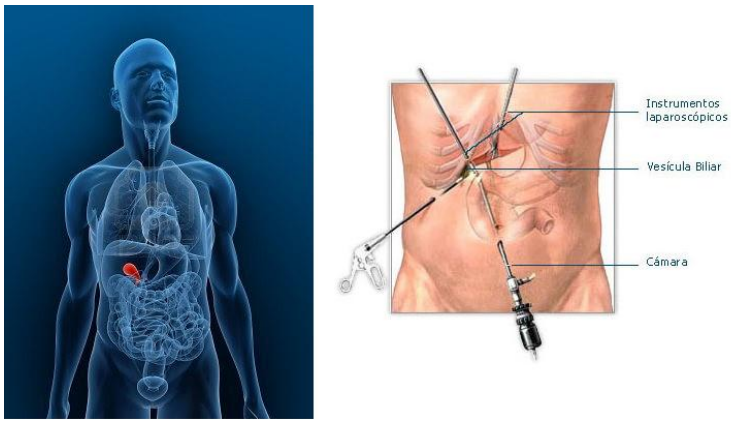

Figura 11: Vesícula biliar (izquierda) y disposición de los instrumentos para su extracción (derecha) [13].

Existen intervenciones laparoscópicas-quirúrgicas de extirpación de órganos (o ablativos) tales como los mostrados en la Tabla 2:

Nombre:	Descripción
Colecistectomía	Intervención que se realiza para quitar una vesícula biliar enferma, que esta inflamada u obstruida [13].
Apendicectomía	Extracción del apéndices (causado por la apendicitis). La intervención se realiza con una pequeña incisión en el cuadrante inferior derecho del abdomen y se extirpa el apéndice.
Histerectomía	Intervención para extirpar la matriz.
Coletomía Asistida	Extirpación del colon.
Ligadura de trompas	Forma permanente de evitar el embarazo cerrando las trompas de Falopio.
Gastrectomía	Extirpación total o parcial del estómago, en caso de problemas gástricos como úlceras o cáncer.
Prostatectomía	Extirpación total o parcial de la próstata.
Hepatectomía	Procedimiento para diagnostico, tratar o evacuar lesiones y tumores en el hígado.
Vasectomia	Procedimiento para la esterilización masculina.

Tabla 2: Procedimientos laparoscópicos [12].

2.2.1. Colecistectomía:

Siendo este procedimiento del interés de este proyecto, la Figura 12 presenta los órganos relacionados en el mismo.

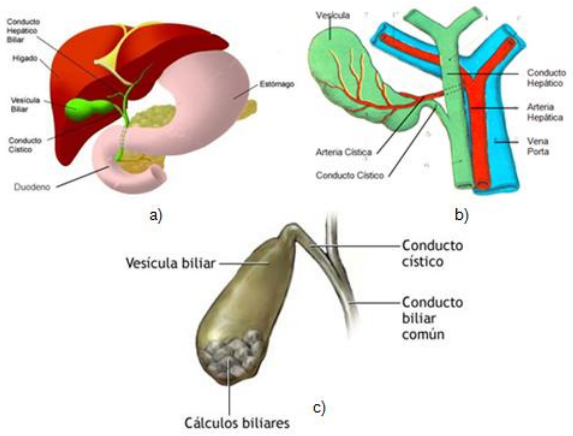

Figura 12: Órganos biliares (a), vesícula biliar (b) y vesícula enferma (c).

La bilis es una secreción elaborada en el hígado y evacuada por el conducto hepático hacia el duodeno, pasando por un elemento biliar llamado vesícula que está conectada al conducto hepático por el conducto cístico (ver el lado izquierdo de la Figura 12 c.). La excreción de bilis es discontinua y depende del tránsito digestivo, así que se acumula en la vesícula cuando no hay alimento pasando por el duodeno [14].

La presencia de partículas sólidas en la vesícula, llamadas cálculos, obstaculiza el paso de la bilis generando dolor e inflamación en el paciente. La colecistectomía es un método frecuente para tratar esta enfermedad [14].

En el quirófano, el cirujano se ubica a la izquierda del paciente, a su lado el auxiliar encargado del manejo de la cámara, y frente a ellos o a la derecha del paciente se ubican el asistente principal y la enfermera (Figura 13).

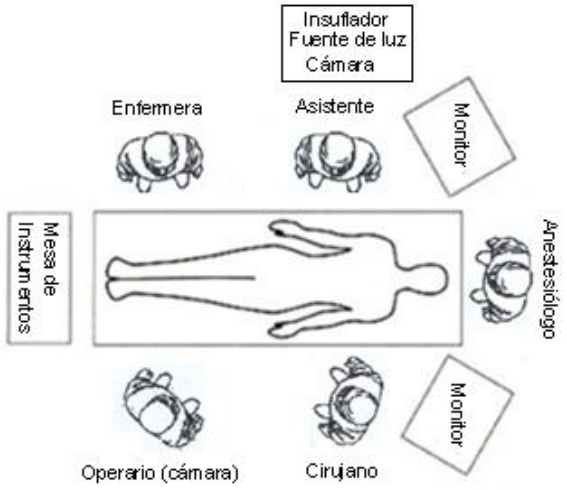

Figura 13: Distribución en el quirófano [15].

Para llevar a cabo el procedimiento como tal, se hacen 3 pequeñas incisiones de acuerdo a la distribución presentada en la figura anterior, de la siguiente manera:

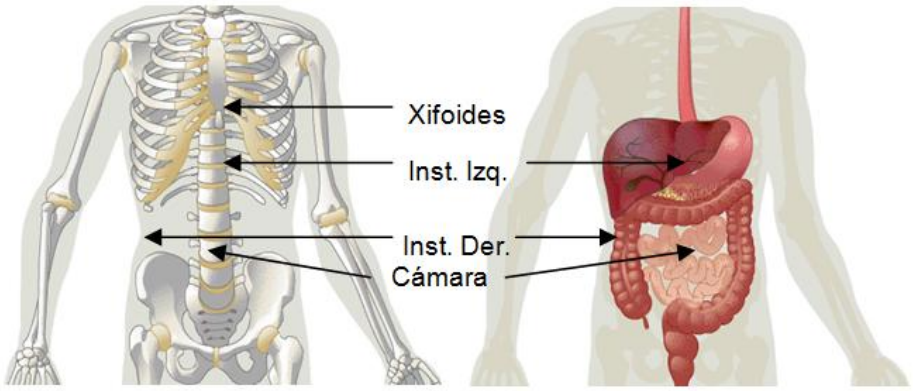

Figura 14: Ubicación de las incisiones para realizar la colecistectomía [15].

Cabe anotar que:

1. La primera incisión es de 0.01m (10 mm), cercana al ombligo o sobre éste, para introducir el sistema de visión.

2. La segunda incisión de 0.01 m (10 mm) se realiza 0.05 m por debajo del xifoides[6] en la parte superior del abdomen, por la cual se introducen los instrumentos para cortar, disecar y coagular durante la intervención.

La tercera de entre 0.005 m (5 mm) y 0.01 m (10 mm) se realiza 0.05 m debajo de las costillas del lado derecho del paciente, esta se utiliza para introducir pinzas que mueven el hígado, la vesícula y sus conductos.

2.2.2. Etapas quirúrgicas en una colecistectomía.

Generalmente la colecistectomía se divide en cinco pasos o etapas, descritas a continuación [15]:

I. Exposición del conducto y arteria císticos.

Para esta etapa se utilizan dos pinzas de retracción (instrumentos a la izquierda de la Figura 15) introducidas por la parte derecha del abdomen. Una pinza se usa para agarrar y mover la vesícula hacia arriba, en dirección del hombro derecho del paciente y la otra hala hacia la región llamada Bolsa de Hartman [15].

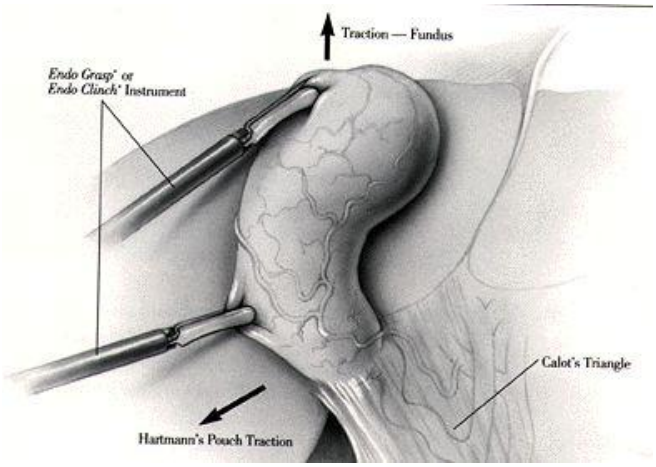

Figura 15: Exposición del conducto y arteria císticos [15].

[6]Xifoides: Indica el límite inferior de la cavidad torácica.

II. Disección del triángulo de Calot.

Se introduce una pinza de retracción adicional por la incisión realizada en la parte superior del abdomen, con la cual se expone el conducto y la arteria císticos por fuera del tejido como lo muestra la Figura 16. Para esta exposición es necesario realizar movimientos de apertura y cierre sobre los conductos tratando de dilatar la zona y mejorar la visualización en el área de trabajo.

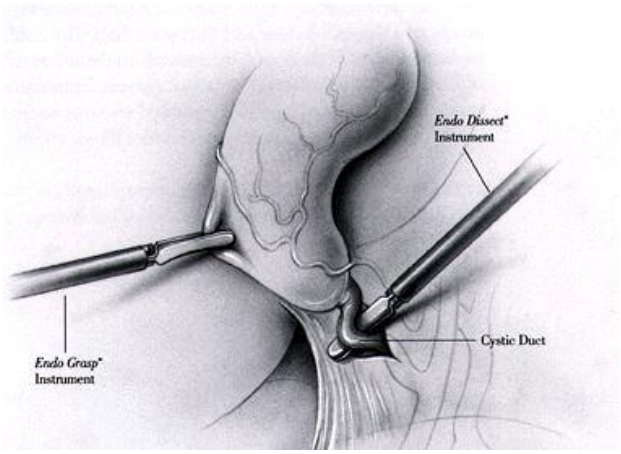

Figura 16: Disección del conducto cístico[15].

III. Sección del conducto y la arteria císticos.

Se introduce por la incisión superior una pistola de endoclips que grapa con tres clips el conducto cístico, colocando dos a la derecha del conducto y uno a la izquierda (cercano a la vesícula). Luego se reemplaza la pistola por una tijera quirúrgica que corta en medio de los clips. Y se repite el procedimiento para la sección de la arteria (ver Figura 17).

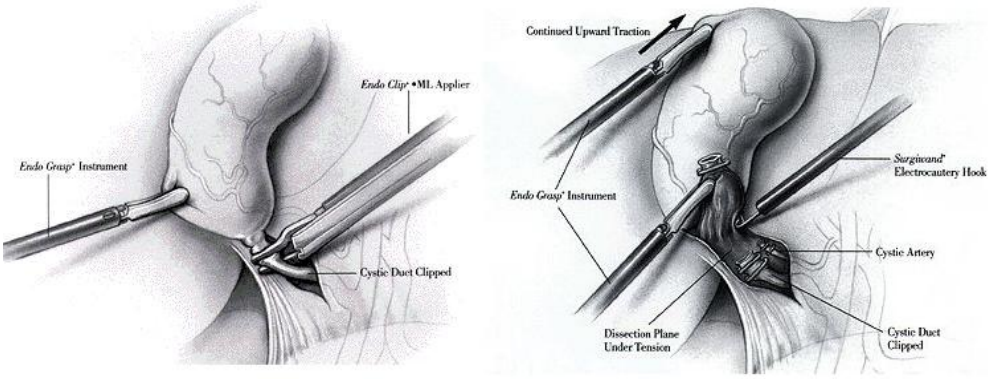

Figura 17: Sección del conducto y arteria císticos[15].

IV. Disección de la vesícula del lecho hepático.

Como la vesícula se encuentra ligada al lecho hepático por una serie de adherencias, se procede a disecar la vesícula del lecho hepático (despegar de la membrana del hígado) utilizando un instrumento de electrocauterio (instrumento a la derecha de la Figura 18) que se introduce por la incisión superior y que tiene la capacidad de disecar y cauterizar la sangre producida.

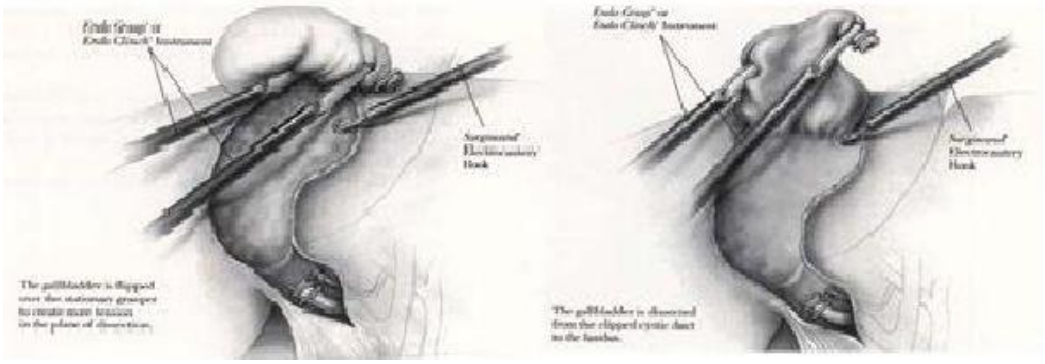

Figura 18: Disección de la vesicular del lecho hepático[15].

V. Extracción de la vesícula biliar.

Finalmente se extrae la vesícula, pero esta etapa se omitió ya que depende de factores como la cantidad de cálculos o tamaño de la inflamación, la experiencia del cirujano y las técnicas del grupo médico.

Así se ve el procedimiento en una etapa intermedia:

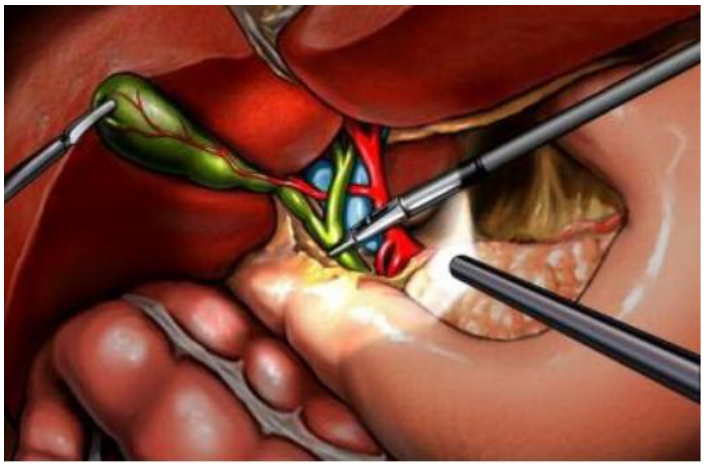

Figura 19: Vista ampliada del procedimiento de extracción de vesícula biliar [16].

2.3. Robots utilizados en este proyecto.

En la Universidad del Cauca, la temática en torno a la laparoscopia empezó con el trabajo de maestría “Modelado, simulación en 3D y control de un robot para cirugía laparoscópica” por el ingeniero Sergio Alexander Salinas, en el año 2009.Y el año siguiente con el proyecto de pregrado “Diseño y simulación en 3D de un robot porta endoscopio para cirugía laparoscópica”, realizado por Cristian Méndez Rodríguez y Víctor Gabriel Torres Muñoz.

En el primer trabajo se diseñó un robot quirúrgico llamado LAPBOT, y en el segundo trabajo se diseñó el robot porta-endoscopio HIBOU.

2.3.1. Robot asistente para cirugía laparoscópica LAPBOT

El robot (LapBot) se utiliza para efectuar el procedimiento quirúrgico como tal, permite posicionar y orientar el instrumento quirúrgico en un espacio tridimensional; manteniendo el punto fijo de movimiento sobre la incisión donde se encuentra el trocar, con el fin de que el robot sea intrínsecamente seguro [17].

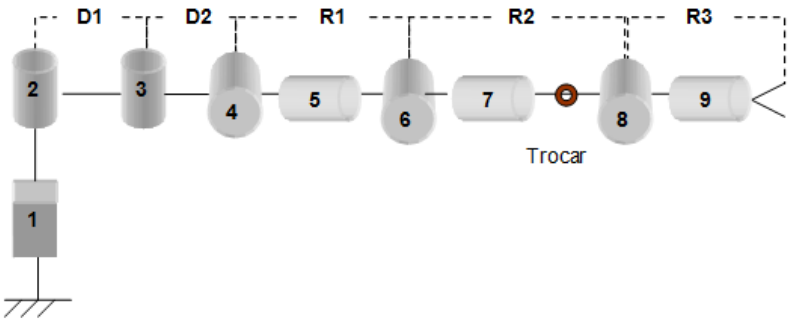

Figura 20: Estructura cinemática del robot LAPBOT [17].

En la **¡Error! No se encuentra el origen de la referencia.** se observan los parámetros geométricos de este robot.

j	μ_j	σ_j	α_j	d_j	θ_j	r_j
1	1	1	0	0	0	r_1
2	1	0	0	0	θ_2	0
3	1	0	0	$D1$	θ_3	0
4	0	0	90º	$D2$	θ_4	0
5	0	0	90º	0	θ_5	$R1$
6	0	0	-90º	0	θ_6	0
7	1	0	90º	0	θ_7	$R2$
8	1	0	-90º	0	θ_8	0
9	1	0	90º	0	θ_9	0
10	0	1	0	0	0	$R3$

Tabla 3: Parámetros geométricos de LapBot [18].

Dónde:

- **j:** Número de la articulación.
- **σ:** Indica si la articulación es rotoide (0) o prismática (1).
- **μ:** Indica si la articulación es activa (motorizada = 1) o pasiva (no motorizada=0).
- **α:** Ángulo entre los ejes Zj-1 y Zj, correspondiente a una rotación alrededor de Xj-1.
- **d:** Distancia entre Zj-1 y Zj a lo largo de Xj-1.
- **θ:** Ángulo entre los ejes Xj-1 y Xj correspondiente a una rotación alrededor de Zj.
- **r:** Distancia entre Xj-1 y Xj a lo largo de Zj.

Finalmente el modelado del robot LapBot debe hacerse respetando el paso por el trocar y se obtiene entonces una estructura como la mostrada en la Figura 21:

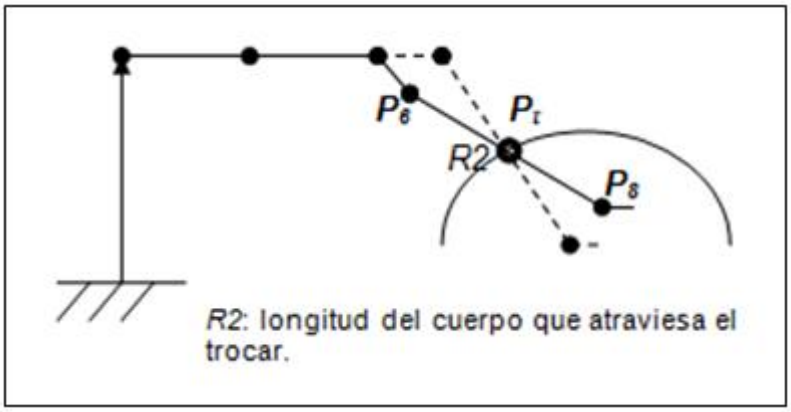

Figura 21: Movimiento del LAPBOT através del trocar [18].

2.3.2. Robot porta endoscopio HIBOU.

El robot porta endoscopio HIBOU es un robot serie que permite posicionar y orientar la cámara dentro del paciente a través de una pequeña incisión y respetando el paso por esta misma.

Para lograr una estructura que pueda posicionar y orientar correctamente el efector final del robot en cualquier punto del espacio cartesiano al interior del abdomen y respetando el paso por el trocar[7], se hace necesario siete articulaciones, de las cuales cinco son motorizadas y dos son no motorizadas (articulaciones pasivas). La técnica de articulaciones pasivas es utilizada para asegurar el paso por el trocar u orificio abdominal [19]. La estructura cinemática del robot HIBOU se puede observar en la Figura 22.

[7]Trocar: Punto o coordenada donde se realiza la incisión en el paciente para introducir los instrumentos necesarios para una intervención quirúrgica.

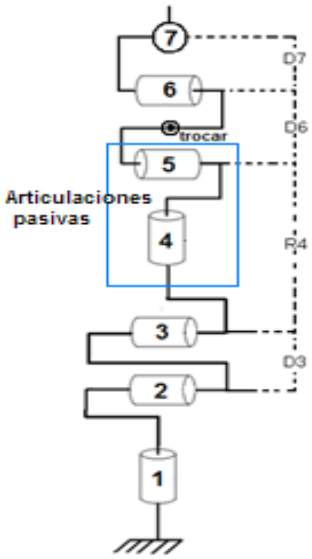

Figura 22: Estructura Cinemática del HIBOU [19].

En la Tabla 4 se observan los parámetros geométricos de este robot.

j	ant	M	σ	A	D	θ	R
1	0	1	0	0	0	t1	0
2	1	1	0	-90	0	t2	0
3	2	1	0	0	D3	t3	0
4	3	0	0	90	0	t4	R4
5	4	0	0	-90	0	t5	0
6	5	1	0	0	D6	t6	0
7	6	1	0	-90	D7	t7	0

Tabla 4: Parámetros Geométricos del HIBOU [19].

A partir de la información presentada en la Tabla 4, se obtiene el modelo geométrico directo (MGD) que permite calcular la posición y orientación del efector final del robot en el espacio cartesiano, a partir de las posiciones articulares de cada grado de libertad. El MGD permite saber la posición exacta de la cámara sin hacer uso de un sistema de visión o sensores.

Por otro lado el modelo geométrico inverso (MGI) se usa para encontrar el valor de las variables en cada articulación conociendo la posición y la orientación en el sistema de coordenadas cartesianas de la cámara. Es importante mencionar que para hallar el MGI es necesario tener en cuenta la restricción por el paso en el trocar, lo cual limitaría los movimientos de posicionamiento de tal forma que la sección del robot dentro del paciente tendrá un punto de rotación fijo en el abdomen o trocar [20].

Los vectores de orientación las articulaciones se observan en la Figura 23:

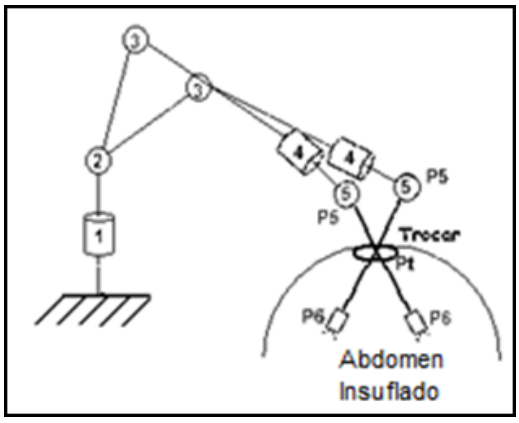

Figura 23: Movimiento del LapBot con restricción por el trocar [20].

2.4. Herramientas de desarrollo software utilizadas [21].

Qt [22]:

Es un framework[8] para desarrolladores, que se usa para hacer aplicaciones con GUI[9]. Esta biblioteca incluye un conjunto de widgets (“controles” en la terminología de Windows) encargados de proporcionar las funciones estándar de una interfaz o GUI. QT presenta una alternativa para la comunicación inter-objeto denominada “signals y slots” que remplaza al antiguo sistema de callbacks utilizado por otros sistemas, además de proporcionar un modelo de eventos convencional para la captura de las pulsaciones del

[8] *Framework*: herramienta software para desarrolladores.
[9] GUI: *Graphical User Interface* (Interfaz Grafica de Usuario).

ratón, teclas u otras entradas del usuario haciendo posible la creación de aplicaciones multiplataforma mediante funciones estándar, incluyendo drivers para Linux, Windows, e incluso drivers nativos para bases de datos como Oracle ®, Microsoft® SQL Server, IBM DB2®, Borland.

La versión usada en este proyecto es qt-creator-win-opensource-2.3.0 y qt-win-opensource-4.7.4-vs2008.

Principios de Qt (Signals and slots) [23]:

"Señales y ranuras" es un mecanismo desarrollado para la comunicación inte-objeto con el fin de reemplazar a los conocidos callbacks. Este sistema es flexible, totalmente orientado a objetos esta implementado en C++.

Una señal es emitida por un objeto cuando cambia su estado de alguna forma. Sólo aquellas clases que definan una señal y las subclases que deriven de ésta pueden emitir la señal. Cuando se emite una señal, el slot conectado a ella es ejecutado de forma inmediata como si se tratara de una función de llamada normal. Si varios slots están conectados a una misma señal todos ellos serán ejecutados uno después de otro pero en un orden arbitrario.

Los slots son funciones en C++ y por lo tanto también podrán ser llamadas de forma normal. Su única característica especial es que las señales pueden ser conectadas a una señal. En comparación con los callbacks, los signals y slots son mucho más sencillos y rápidos de utilizar ya que la diferencia con las aplicaciones reales es insignificante. La emisión de una señal es de cuatro a diez veces más lenta que trabajar directamente con los callbacks debido al tiempo requerido para localizar el objeto, pero su velocidad relativa aumenta al necesitar muchas menos operaciones de tipo new y delete. Además QT trabaja con un conjunto de widgets que poseen funciones típicas para el diseño de interfaces de usuario, los widget son capaces de recibir los eventos del teclado o ratón junto a otros eventos del sistema [21].

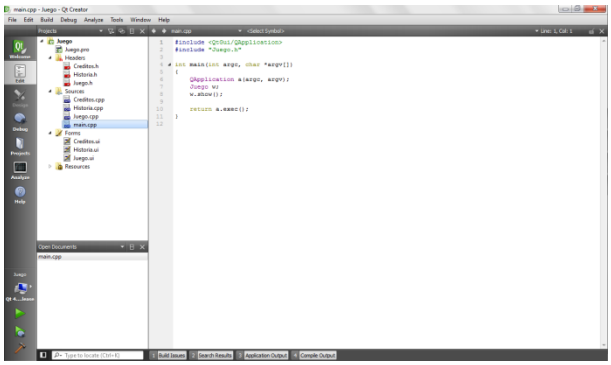

Figura 24:Interfaz de desarrollo de Qt.

Qt Designer [22]:

Es un diseñador de interfaces gráficas para aplicaciones basados en las bibliotecas QT. Es posible escribir aplicaciones completamente en código fuente, o utilizando Qt Designer para un desarrollo más rápido. La arquitectura basada en componentes hace posible a los desarrolladores ampliar Qt Designer con widgets personalizados y extensiones. Qt Designer presenta el inconveniente de no poderse programar directamente sobre él y tener que recurrir a entornos de desarrollo para poder conectar los eventos del computador con otras aplicaciones.

Diseñar un archivo con Qt Designer es un proceso simple, consiste en arrastrar los ítems desde un menú a un formulario, utilizando herramientas de edición para seleccionar, cortar, pegar y cambiar el tamaño de los diferentes componentes. A la salida de este programa se obtiene un fichero con la extensión .ui que contiene una descripción de la interfaz indicando sus características, componentes y forma de comportamiento.

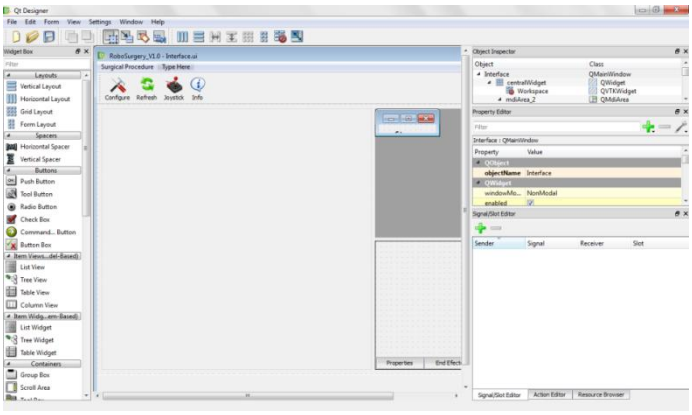

Figura 25: Interfaz de Qt Designer (desde Visual Studio).

VTK [24]:

VTK (Visualzation Toolkit) es un conjunto de bibliotecas que permite la visualización de geometrías 3D, soportando una amplia variedad de algoritmos de visualización y modelado. Es una herramienta open source que extiende su aplicación a prácticamente todos los campos en los que se emplean objetos 3D entre los que se destacan la medicina y las herramientas industriales.

Las funciones que integran VTK se basan en los principios de orientación a objetos conformando un completo sistema jerárquico de filtros y fuentes que dotan de gran flexibilidad a todo el sistema. VTK no consiste únicamente en un sistema de visualización sino que aporta una amplia gama de algoritmos para tratar las imágenes y objetos creados, permitiendo entre otras cosas el añadir texturas, aplicar tensores, reducir el número de polígonos, además de proporcionar una gran variedad de clases de datos para trabajar datos poligonales, imágenes, volúmenes y mallas.

VTK posee un modelo basado en el paradigma del flujo de datos adoptado por la mayoría de los sistemas comerciales. Según este paradigma, los diferentes módulos son conectados formando una red. La ejecución de la red de visualización es controlada en respuesta a las demandas de datos (conducción por demanda) o en respuesta a una entrada del usuario (conducción por eventos). La gran ventaja de este sistema es la flexibilidad, y puede ser rápidamente adaptado a diferentes tipos de datos o a nuevas implementaciones algorítmicas. La parte de visualización es la encargada de generar los datos que serán presentados por el modelo gráfico.

La versión usada en este proyecto es vtk-5.8.0.

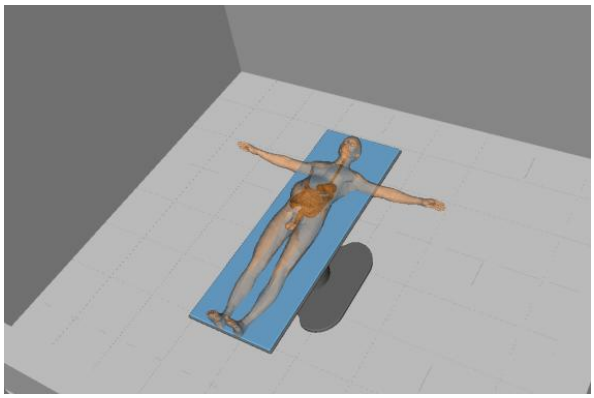

Figura 26: Renderizado[10] obtenido usando VTK.

Blender [25]:

Es un programa informático multiplataforma, dedicado especialmente al modelado, animación y creación de gráficos tridimensionales.

El programa fue inicialmente distribuido de forma gratuita pero sin el código fuente, con un manual disponible para la venta, aunque posteriormente pasó a ser Herramienta software libre. Actualmente es compatible con todas las versiones de Windows, Mac OS X, Linux, Solaris, FreeBSD e IRIX.

Tiene una muy peculiar interfaz gráfica de usuario, que se critica como poco intuitiva, pues no se basa en el sistema clásico de ventanas; pero tiene a su vez ventajas importantes sobre éstas, como la configuración personalizada de la distribución de los menús y vistas de cámara.

La versión usada de esta es blender-2.6.1-release-windows32

[10]Renderizado: Proceso donde se imita un espacio 3D formado por polígonos, luces, texturas, materiales y animación, simulando ambientes y estructuras físicas.

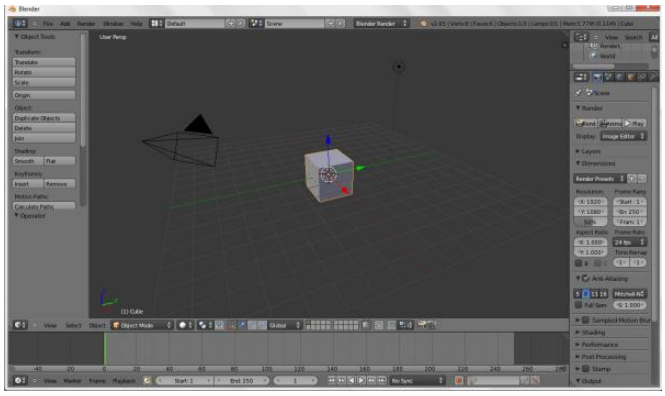

Figura 27: Interfaz de trabajo en Blender.

CMake [26]:

Cmake (Cross Platform Makefile Generator) es un generador de proyectos que funciona en diferentes plataformas. Su función es la de configurar y gestionar el proceso de construcción de un proyecto en un archivo de texto simple, (CmakeLists.txt) donde se describe el proceso mediante código. Posee gran variedad de comandos que permiten básicamente vincular bibliotecas, crear ejecutables, cargar archivos al proyecto y la generación de instaladores.

La versión usada es cmake-2.8.5-win32-x86

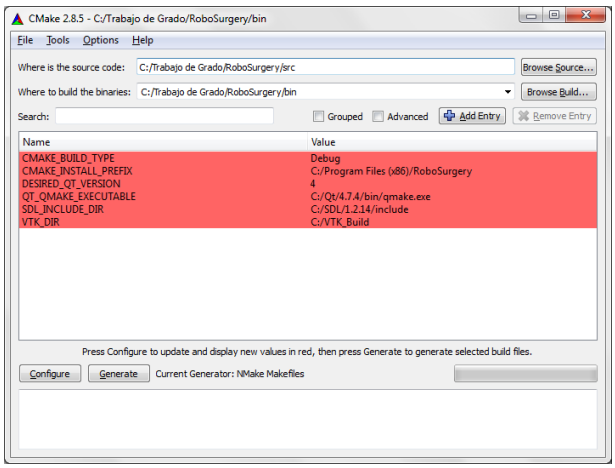

Figura 28: Interfaz de CMake.

Microsoft Visual Studio 2008 [21]:

Es un entorno de desarrollo integrado (IDE, por sus siglas en inglés) para sistemas operativos Windows. Soporta varios lenguajes de programación tales como Visual C++, Visual C#, Visual J#, ASP.NET y Visual Basic .NET, aunque actualmente se han desarrollado las extensiones necesarias para muchos otros.

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

La versión usada es Microsoft Visual Studio 2008

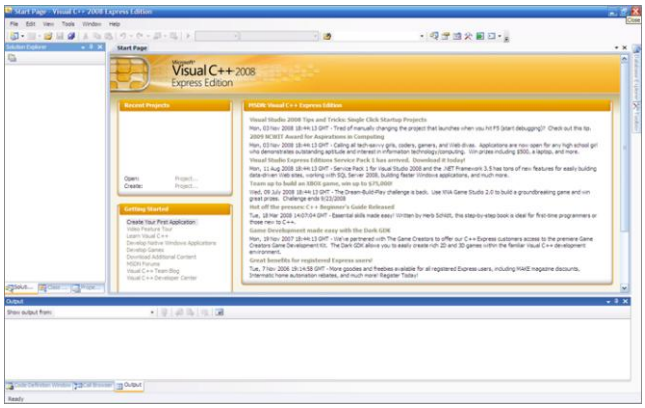

Figura 29: Interfaz deVisual Studio 2008.

Make Human [27]:

Es un freeware[11] open source[12] para la creación rápida de prototipos humanos, desarrollado por programadores, artistas y entusiastas del modelado 3D en computadora. MakeHuman™ se utiliza para crear un modelo 3D de un ser humano a partir de unas características controladas por el usuario como la raza, el sexo, el tamaño, la musculatura y otros atributos.

[11]Freeware: Es software que está disponible para uso sin costo.
[12]Open Source: Permite el acceso al diseño y los detalles de la implementación de un software.

Este exporta en múltiples formatos como: Blender, Maya, XSI, Cinema 4D, Zbrush, etc., con el fin de ser incluidos en escenas complejas (como en este proyecto de grado), representaciones o juegos.

MakeHuman™ está desarrollado principalmente en Python y OpenGL para ser fácilmente portado a otros sistemas operativos diferentes (compatible con Windows, Linux y OS X).

La versión usada es makehuman-nightly-win32

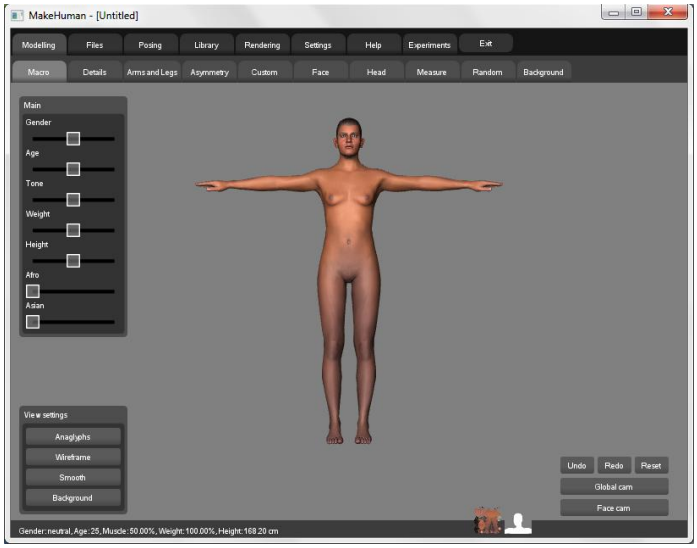

Figura 30: Diseño de un cuerpo humano con Make Human.

En resumen las herramientas software anteriormente mencionadas se usaron para realizar este proyecto, para las respectivas guías y archivos de instalación ver anexo A.

3. DESARROLLO E IMPLEMENTACIÓN SOFTWARE.

Es importante conceptualizar como se lleva a cabo el proceso de desarrollo de un software. Como metodología de desarrollo se utilizó el **modelo evolutivo**, con un enfoque metodológico que ordena rigurosamente las etapas del **proceso**, de tal forma que el inicio de cada etapa debe esperar a la finalización de la etapa anterior. Esto se explicará más adelante, por lo que primero se presentan conceptos importantes en el desarrollo software.

3.1. Ciclos de vida del software.

Cualquier producto a desarrollar tiene un principio y un fin, el ciclo de vida determina qué tipo de producto se tendrá. Este ciclo varía según lo que se quiere construir, el tamaño, presupuesto, capacidad del equipo, experiencia de los desarrolladores, etc.

Los sistemas son desarrollados de mejor manera mediante el uso de un ciclo específico de actividades del analista, el desarrollador y el usuario. Las etapas de este ciclo son:

- Planeación: Especificaciones iníciales, factibilidad.
- Desarrollo: Construcción del software.
- Operación: Uso del sistema.
- Mantenimiento: Corrección y actualización.

El proyecto de desarrollo de RoboSurgery espera ser una primera versión que en proyectos de grado posteriores se “mejore” y adicione nuevas funciones a la herramienta. Por dicha razón uno de las etapas más importantes en el ciclo de vida de este proyecto es la de mantenimiento.

Mantenimiento de herramientas software, en esta etapa se llevan a cabo ciertas actividades que realimentan y enriquecen el proyecto, por ejemplo:

- Corregir errores (mantenimiento correctivo).

- Adaptación (revisión).
- Perfectivo (mejoras o modificaciones).
- Preventivo (reingeniería).

EL nivel de importancia de estas actividades se observa en la Figura 31.

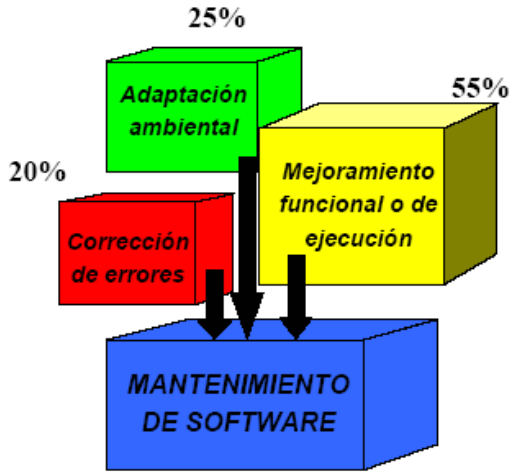

Figura 31: Mantenimiento del software [27].

Esto implica que dentro del mantenimiento software la mayor parte del trabajo se centra en el mejoramiento funcional. Aun así es de tener en cuenta que para los trabajos futuros el reto se centra en que:

- Es difícil o imposible seguir el proceso de evolución del Herramienta software a través de varias versiones, si los cambios no están documentados adecuadamente.
- Es muy difícil entender el programa de “alguien” más.
- Ese “alguien” tal vez no se encuentre para explicarlo, en el caso de este proyecto, pues una vez culminada la primera versión culmina la participación del desarrollador en el mismo.
- La documentación no existe o es inadecuada.
- Muchas herramientas software no están diseñadas para el cambio.

Es importante tener en cuenta estas observaciones para diseñar estrategias que puedan solventarlos en el futuro. Se espera que la documentación y modelado de la herramienta sea suficiente para apoyar los futuros desarrollos sobre esta herramienta software.

3.2. Modelos de ciclo de vida software.

El modelo “clásico” conocido como build and fix, desorganizado y cuyo resultado tiende al fracaso, es el de construir y corregir. En este el desarrollador se sienta a programar sin hacer diseños o análisis, corrigiendo continuamente hasta llegar con “suerte” al producto terminado, difícilmente mantenible y/o modificable.

Es poco recomendable trabajar de esta manera, se ha comprobado que incluso un diseño burdo en papel o cualquier diagrama, ayudan más que no haber hecho nada.

Durante los años 80, el ciclo de vida más difundido era el cascada [28]. Consiste fundamentalmente de las etapas de análisis de requerimientos, diseño, construcción, pruebas e implantación. Cada nueva etapa se inicia una vez se valide el final de la anterior, este está basado en documentos y obliga a la formalidad, atacando las desventajas del build and fix.

Sin embargo tiende a ser muy largo, dado que una nueva etapa no se inicia hasta que termine la anterior, la retroalimentación con el cliente es tardía y muchos proyectos fallan por no haber entendido los requerimientos del cliente. Otro problema es que los requerimientos no son estáticos, y cambian rápidamente y el producto generado debe irse ajustando a dichos cambios.

Con el fin de solventar los anteriores problemas de estos modelos, se creó otro ciclo de vida, ajustado a este proyecto, llamado ciclo de vida evolutivo.

El ciclo de vida evolutivo consiste de una serie de iteraciones cortas y fijas en duración (entre dos y seis semanas). Al final de cada iteración se tiene una parte del producto que se ha probado e integrado a la anterior. Cada iteración a su vez, sigue las fases del ciclo de cascada. En las primeras iteraciones se buscará implementar los requerimientos centrales del sistema o los que impliquen mayor riesgo [28].

Ya que las iteraciones tienen duración fija, las fechas de entrega son inamovibles así que si no se puede cumplir con los requerimientos establecidos, se cambiará el alcance dejando requerimientos para una iteración posterior [28].

Los principales beneficios de este modelo de desarrollo son:

- Menor probabilidad de que el proyecto falle.
- Mayor productividad.
- Mitigación de riesgos en iteraciones tempranas.
- Progreso visible en un corto tiempo.
- Retroalimentación oportuna.
- En cada iteración se genera conocimiento que permitirá mejorar las subsecuentes iteraciones.

Un proceso evolutivo popular actualmente es el UP[13]. Esta metodología es implementada comercialmente en el RUP[14], un refinamiento del Proceso Unificado que provee las herramientas para realizar y documentar las fases del proceso.

3.3. Modelado usando UML.

Desde el punto de vista de la ingeniería, el proceso de modelado es una de las partes más importantes en un desarrollo, la razón de esto es que mientras una idea se tenga estructurada en un modelo, esta se puede llevar a cabo usando cualquier conjunto de herramientas para apoyar el desarrollo, por otro lado el modelado es el medio estándar para trasmitir ideas y unificar la manera de comunicarse entre ingenieros o con personas de múltiples disciplinas.

[13] UP: *Unified Process*.
[14] RUP: *Rational Unified Process*.

Los diagramas UML son un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema Herramienta software. Es un “plano” (modelo), que incluye aspectos conceptuales como procesos y funciones de un sistema.

Siguiendo una metodología de desarrollo, la estructura o esqueleto de este proyecto de grado se definió previamente a la codificación, haciendo uso de diagramas de “caso de uso” para definir los requerimientos de la herramienta software y diagrama de clases para definir la estructura del mismo, este inicialmente sin mucho nivel de detalle, pues en el proceso de desarrollo se vio sujeto a modificaciones necesarias, pero proporcionó un marco o vía del camino a seguir.

Además en los trabajos que sirvieron de referencia para este proyecto los diagramas UML facilitaron el entendimiento de la codificación usada en otros programas.

3.3.1. Diagrama de casos de uso.

Este diagrama define el “qué” no él “cómo”, a partir de los requerimientos funcionales definidos y la relación sistema/usuario. Sirve de guía para definir los requerimientos no funcionales, aspectos de calidad, desempeño, usabilidad, etc.

Estos diagramas documentan los requerimientos funcionales de un sistema (o se usan para documentar los procesos de negocio de una organización), describen el comportamiento de cómo un sistema responde a las solicitudes de uno de los actores primarios [29].

Estos diagramas pueden ser:

- Un documento de texto.
- Diagramas de flujo.
- Diagramas de secuencia.
- Redes de Petri.
- Un lenguaje de programación.

El modelado de Casos de Uso es la técnica más efectiva y a la vez la más simple para modelar los requisitos del sistema desde la perspectiva del usuario. Define el comportamiento de un sistema de manera clara, coherente y fácil de comunicar. Los Casos de Uso se utilizan para modelar cómo un sistema o negocio funciona actualmente, o cómo los usuarios desean que funcione. Es realmente una forma de modelar procesos. Los modelos (o diagramas) se usan para comunicar y estandarizar.

Los casos de uso definen:

- Alcance: Qué es el sistema en discusión.
- Actor: Quién tiene la meta.
- Nivel: Qué tan alto o bajo es el nivel de esa meta.

La Figura 32 presenta el diagrama de casos de uso propuesto, donde se definen los requisitos del usuario al usar la herramienta software:

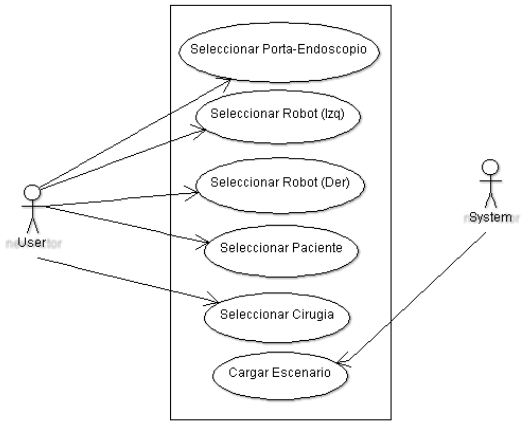

Figura 32: Diagrama de casos de uso para de RoboSurgery.

Las elipses son los casos de uso, que indican una funcionalidad, el recuadro identifica las partes del sistema y los casos de uso que lo conforman.

Estos diagramas se definen según los requerimientos funcionales del sistema y la relación sistema/usuario esperada, para definir aspectos como desempeño y usabilidad entre otros.

3.3.2. Diagrama de relaciones entre casos de uso.

Describe la estructura del sistema mostrando sus clases, atributos y las relaciones entre ellos. Es utilizado para el análisis y diseño del sistema, se describe la lógica del negocio, la información que se transporta, en resumen las entradas al sistema, los procesos que ocurren dentro de él, y las salidas que se generan.

Relaciones entre casos de uso llevan una flecha de navegación que apunta a quién inicializa la interacción con el sistema, como se observa en la Figura 33. Si no hay flecha indica que no hay interacción entre el actor y el caso de uso.

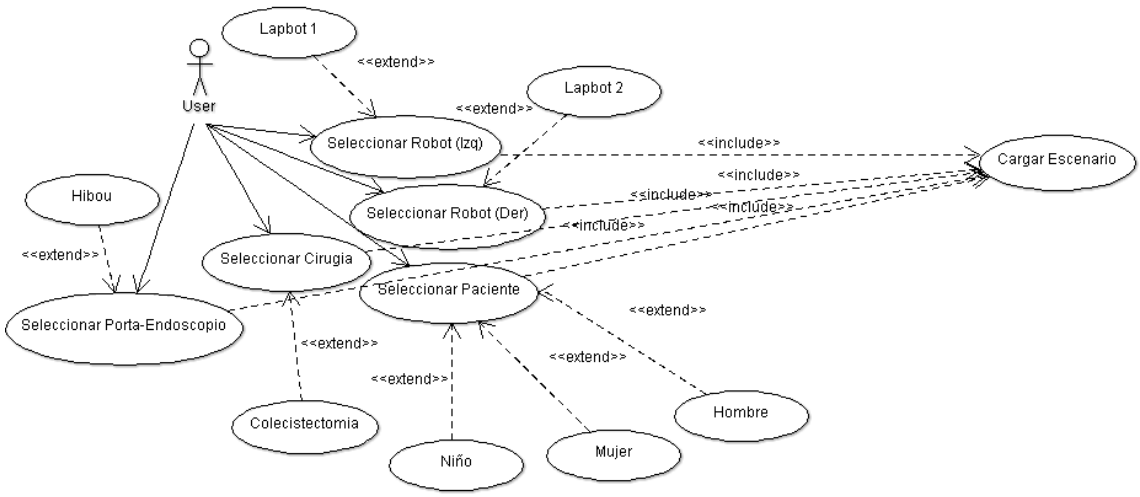

Figura 33: Diagrama de relaciones entre casos de uso de RoboSurgery.

Este diagrama nos muestra cómo se relacionan los casos de uso, se observa que el actor solo interactúa sobre los casos de uso que se definieron como funcionalidades del programa, para realizar las respectivas selecciones y de estas a su vez se desprenden las mismas. Esto da a entender que el sistema propuesto es modular, y como tal la expansión

de sus funciones implicaría incluir nuevos casos de uso que son transparentes para el usuario.

3.3.3. Diagrama de clases

Este diagrama muestra las relaciones de las clases que involucran el sistema, las relaciones pueden ser asociativas, de herencia, o de uso [30].

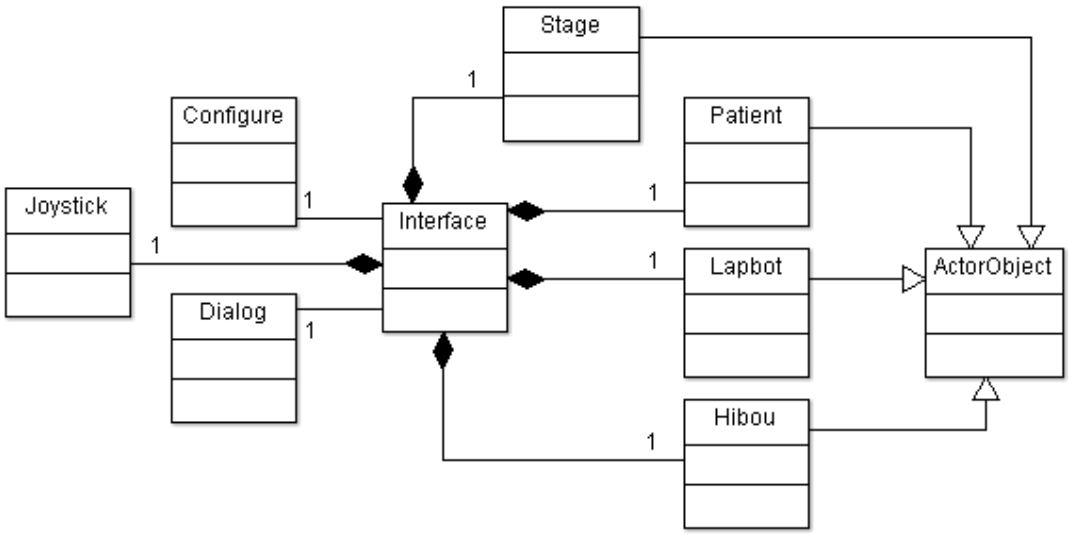

Figura 34: Diagrama de clases de Robo-Surgery.

Este diagrama muestra como se pretende codificar el programa, la estructura está compuesta como se muestra en la Tabla 5, así:

<Nombre de la clase>
<Atributos>
<Métodos>

Tabla 5: Estructura de un bloque en el diagrama de clases.

Los métodos y los atributos pueden ser prívate, donde solo desde la misma clase se puede acceder, public, donde cualquier clase puede acceder, o protected, donde su acceso es restringido. Esto afecta la accesibilidad a los métodos y atributos desde otras clases, por lo que hay que ser cuidadoso durante la codificación de las mismas, pues esto influirá en el desempeño del programa.

Lo anterior implica que si por ejemplo, es necesario imprimir el valor de una variable en una consola, y la variable es privada, no es posible llamarla desde el programa principal para conocer su valor. Por lo que hay dos soluciones, una es declarar la variable como una variable 'public', la otra forma es crear un método que se puede llamar 'getVariableX' pero este método debe ser 'public' y retornar el valor de la variable que deseamos conocer,

3.4. Conceptos importantes de la programación orientada a objetos.

A lo largo del desarrollo de este proyecto de grado, se codificaron las clases usando la metodología de la POO[15], se definieron algunas clases (normalmente en los archivos de cabecera), compuestas por ciertos atributos y métodos presentados en los diagramas.

Algunos conceptos importantes a definir de la POO se observan en la Tabla 6:

Nombre:	Descripción
Encapsulamiento	Desde fuera del objeto solo se puede acceder a los métodos e identificadores que permita el creador del objeto.
Herencia	Permite reutilizar y extender código. Se usa para hacer nuevas clases a partir de unas ya existentes.
Polimorfismo	Permite tratar de manera genérica objetos de distintas clases, ahorrando y haciendo simple el código.
Constructores	Este método se ejecuta automáticamente cuando se crea un objeto, su función es inicializar el objeto, y preparar para realizar las funciones necesarias.
Destructores	Se ejecuta automáticamente cuando un objeto queda fuera de uso, su función es destruir el objeto, liberando cualquier asignación dinámica de memoria.

[15]POO: Programación Orientada a Objetos.

Sobrecarga	Es la redefinición de un método, otra forma de decirlo, es un mismo método con varias definiciones diferentes, lo que distingue uno de otro va a ser el tipo y el numero de parámetros que recibe.
Métodos	Al igual que en la programación estructurada los métodos son funciones que contienen una serie de acciones a ejecutar cuando esta es llamada
Atributos	Son las características o las variables que componen a un objeto

Tabla 6: Conceptos asociados a la POO [31].

Una vez contextualizadas y conceptualizadas todas estas ideas, se procede a codificar el software. RoboSurgery tiene ciertas características que se analizarán a continuación.

3.5. Clases importantes.

Desde el punto de vista de la POO, hay 2 maneras de codificar una clase, uno es el método en línea, donde en el archivo ".h" declara y codifica los métodos. El otro método es el fuera de línea, en el cual se usa el archivo ".h" para declarar los métodos y se crea un archivo ".cpp" para codificar los métodos como tal.

Ejemplo del método en línea:

```
inlinevoid                on_actionInfo_triggered(){Dialog.setModal(true);
Dialog.exec();};
```

Como podemos ver este método "simplifica" la cantidad del lineas de codigo, pero no es muy conveniente en métodos con muchas acciones a ejecutar en él.

Ejemplo del método fuera de línea:

Este ejemplo presenta el metodo fuera de línea y la estructura básica de una clase, en el archivo ".h":

Figura 35: Método fuera de línea y estructura del archivo "ActorObject.h".

3.5.1. La clase "ActorObject".

En el programa esta clase sirve para crear los objetos que forman parte del renderizado (escenario).

```
public:
    ActorObject(constchar *,double R = 0.5,double G = 0.5, double B =
0.5, double x = 0, double y = 0, double z = 0, double O = 1);
```

Lo mas relevante en esta clase es el constructor "ActorObject", es quien recibe los parámetros necesarios para crear el objeto para el renderizado. Estos parametros son:

- Char *: entrega la dirección o ubicación del archivo ".obj" a cargar en el render.
- R, G y B: Son los parametros para determinar el color de las mallas que componen al objeto. Estos van de 0 a 1, equivalente al formato de color que va de 0 a 255.

- X, Y y Z: Son las variables que definen la pocision en el espacio de la escena donde se debe cargar el objeto, la orientación de los ejes se observa en la siguiente figura, donde se muestra el escenario usado en RoboSurgery.

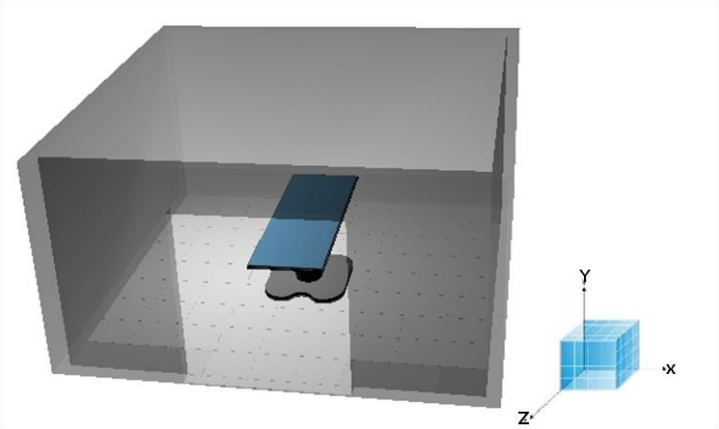

Figura 36: Disposición de los ejes X,Y y Z en RoboSurgery.

- O: Este parámetro define qué tan opaco es el objeto cargado en la escena, este valor varía entre 0 y 1, donde 0 es trasparente y 1 es totalmente opaco.

Es de tener en cuenta que esta es una clase propia, o creada para el programa RoboSurgery, por lo que el nombre “ActorObject” o los parámetros que esta clase recibe pueden ser cambiados o editados según el programador.

Un ejemplo de cómo es llamada o usada esta clase es el siguiente:

En la clase donde se cree un objeto del tipo “ActorObject”, se crea en el archivo “.h” un objeto como por ejemplo así: `ActorObject *Human.` Una vez hecho esto, en el lugar donde vamos a llamar (y crear como tal el objeto para la escena) se usa el siguiente comando:

```
Human = new ActorObject("../../3d-
objects/Patients/MAN.obj",1,0.709803921,0.482352941,0.4,-0.55,0.05,0.2);
```

Se puede observar cómo se crea el objeto, y se entregan a la clase “ActorObject” los parámetros necesarios para cargar un archivo “.obj” en la escena (render).

Finalmente para que el objeto se pueda observar en la escena:

```
render->AddActor(Escenario[0]->retorno_Actor());
```

Esto “Adiciona” un nuevo actor al objeto de la clase que contiene el renderizado (en este caso se llama “render”).

3.5.2. La clase “RobotLAPBOT”:

Esta clase es un poco diferente al resto, ya que fue modificada para que permitiera crear varios objetos de esta clase en diferentes posiciones sin afectar su comportamiento, por esta razón el constructor de esta clase recibe 7 parámetros del tipo “double” así:

```
class RobotLAPBOT
{
private:

	double PIx, PIy, PIz, Ry;
	double TrocarX, TrocarY, TrocarZ;
	ActorObject *RobotL[11];
	void AssemblingRobotL();
	vtkAssembly *JointsL[9];
	vtkAssembly *robotAssembly;
	void Move_Robot(double, double, double, double ,double,double,
double, double, double);

public:

	RobotLAPBOT();
	RobotLAPBOT(double, double, double, double, double, double,
double);
	~RobotLAPBOT();
	void MGI(double, double, double);
```

```
        inline vtkAssembly *Return_Robot(){ return (robotAssembly);};
};
```

De estos siete parámetros, los tres primeros indican la posición de la base del robot, normalmente el segundo parámetro (que seria Y) vale cero, pues la base del robot esta a la altura del suelo de la escena y la base se posiciona en el plano X, Z del escenario. El cuarto parámetro es para indicar el ángulo de giro en grados de la primera articulación del LapBot, para rotar toda la estructura en dirección al paciente. Los últimos tres parámetros son para entregar el valor de la posición espacial del trocar (sus valores en X, Y y Z respectivamente).

Este procedimiento se hizo sacando provecho de la sobrecarga de métodos (específicamente sobre el constructor de la clase), pues esta se usa cuando un método posee el mismo nombre, pero lo que cambia en sí son los parámetros que esta función recibe. En este caso el compilador entiende que el método requerido o llamado es el que recibe los parámetros previamente definidos, la clase tiene estos dos métodos:

- `RobotLAPBOT();`
- `RobotLAPBOT(double, double, double, double, double, double, double);`

Así siempre que se cree un objeto de esta clase, si la función recibe los parámetros correctos, el compilador escogerá el método adecuado para crear el objeto.

El trocar correspondiente al robot LapBot es entregado en los parámetros del constructor sobrecargado, y estos a la vez actualizados dentro de la clase para poder crear múltiples objetos de esta clase.

Esta clase posee un método llamado MGI, el cual tiene codificado las ecuaciones necesarias para calcular según los parámetros recibidos desde el programa principal, la posición (en longitud) y orientación (en grados) para cada articulación y así poder alcanzar la posición requerida.

Finalmente este método entrega a un método llamado “Move_Robot” los parámetros necesarios para posicionar las articulaciones con el fin de alcanzar la posición requerida, de esta manera:

```
void RobotLAPBOT::Move_Robot(double r1, double t2, double t3,double
t4,double t5,double t6,double t7,double t8,double t9)
{
	JointsL[0]->SetPosition(0,r1,0);
	JointsL[1]->SetOrientation(0,-t2,0);
	JointsL[2]->SetOrientation(0,-t3,0);
	JointsL[3]->SetOrientation(0,0,t4);
	JointsL[4]->SetOrientation(0,t5,0);
	JointsL[5]->SetOrientation(0,0,t6);
	JointsL[6]->SetOrientation(0,t7,0);
	JointsL[7]->SetOrientation(0,0,t8);
	JointsL[8]->SetOrientation(0,t9,0);
}
```

Donde r1 es el desplazamiento calculado por el MGI para la primera articulación del robot (la cual es prismatica), mientras t2, t3, t4, t5, t6, t7, t8 y t9 son los ángulos para orientar cada una de las otras articulaciones.

3.5.3. La clase “RobotHIBOU”:

Esta clase funciona igual que la clase “RobotLAPBOT”, asi que miraremos mas en detalle como se relaciona con la clase “ActorObject” (tambien aplica al robotLAPBOT).

```
class RobotHIBOU
{
private:

	double PIx, PIy, PIz, Ry;
	double TrocarX, TrocarY, TrocarZ;
	void AssemblingRobotH();
	vtkAssembly *JointsH[7];
	vtkAssembly *robotAssembly;
	void Move_Robot(double, double, double, double ,double,double,
double);

public:
```

```
        ActorObject *RobotH[13];
        RobotHIBOU(double , double , double , double, double, double,
double);
        ~RobotHIBOU();
        void MGI(double, double, double);
        inline vtkAssembly *Return_Robot(){ return (robotAssembly);};
};
```

En esta clase se observa cómo cargar un objeto en la escena, pero aquí se observan otras funciones útiles para el ensamble de robots:

```
void RobotHIBOU::AssemblingRobotH()
{
        RobotH[0] = new ActorObject("../../3d-
objects/Robots/HIBOU/Base.obj",1,1,1,0,0,0,1);
        RobotH[1] = new ActorObject("../../3d-
objects/Robots/HIBOU/Art1.obj",1,1,1);
…
```

Cada uno de los objetos llamados en esta función, carga cada una de las piezas correspondientes del robot, llendo desde la base hasta la punta del mismo.

```
…
        RobotH[12] = new ActorObject("../../3d-
objects/Robots/HIBOU/MotorPD.obj",0,0,0,0.3,0,-0.0457,1);
```

El vector RobotH[ ] caga todos los archivos “.obj” que forman la estructura del robot Hibou, cada vez que se crea el objeto de la clase “ActorObject” este recibe los parámetros necesarios.

```
        JointsH[6] = vtkAssembly::New();
        JointsH[6]->AddPart(RobotH[8]->objActor);
        JointsH[6]->SetOrigin(0.59608,-0.4,0);
```

Estas funciones son un ejemplo de cómo conectar estructuras, con la lógica padre-hijo, por lo que el robot (serie) se empieza a ensamblar desde la punta hacia la base.

```
        JointsH[5] = vtkAssembly::New();
        JointsH[5]->AddPart(RobotH[7]->objActor);
        JointsH[5]->AddPart(JointsH[6]);
        JointsH[5]->SetOrigin(0.588,-0.4,0);
```

```
…

JointsH[1] = vtkAssembly::New();
JointsH[1]->AddPart(RobotH[2]->objActor);
JointsH[1]->AddPart(RobotH[11]->objActor);
JointsH[1]->AddPart(RobotH[12]->objActor);
JointsH[1]->AddPart(JointsH[2]);
JointsH[1]->SetOrigin(0,0,0);
```

Esta forma de ensamblado implica que al ordenar el posicionamiento de alguna articulación especifica, se verán afectados los respectivos hijos de la pieza.

```
JointsH[0] = vtkAssembly::New();
JointsH[0]->AddPart(RobotH[1]->objActor);
JointsH[0]->AddPart(RobotH[9]->objActor);
JointsH[0]->AddPart(RobotH[10]->objActor);
JointsH[0]->AddPart(JointsH[1]);
```

Este objeto ‘robotAssembly’ es quien contiene la estructura completa del robot ensamblada, y finalmente es lo que se ve en la escena del render.

```
robotAssembly = vtkAssembly::New();
robotAssembly->AddPart(JointsH[0]);
robotAssembly->AddPart(RobotH[0]->objActor);
robotAssembly->RotateY(Ry);
robotAssembly->AddPosition(PIx, PIy, PIz);
}
```

La combinación de las funciones “vtkAssembly::new”, “addpart”, “setOrigin”, “AddPosition” y el vector JointsH[] (objeto de la clase “vtkAssembly”) permite simular el comportamiento de las articulaciones de un robot, ya que convierte cada articulación en la referencia necesaria para que el objeto haga el movimiento en el espacio. Esto significa que el objeto siguiente a la articulación sea el hijo y el objeto anterior a la articulación el padre, esto con el fin de que cualquier movimiento efectuado en el padre solo afecte a los hijos siguientes a él.

3.5.4. La clase “m2c”

Esta clase es bastante diferente a las implementadas en la herramienta, y se recomienda ampliamente no modificarla, se utiliza para encontrar las variables cartesianas del MGI del robot HIBOU, de esta manera:

Archivo “m2c.h”

```
#define PI 3.141592
#define D3 0.3
#define R4 0.4
#define D6 0.3

usingnamespace std;

double  f_solve(double t[5],double x,double y, double z);
void nelmin ( double fn ( double x[], double x1, double y, double z), int
n, double start[], double xmin[], double *ynewlo, double reqmin, double
step[], int konvge, int kcount,   int *icount, int *numres, int *ifault ,
double COR[]);
void timestamp ( void );
```

‘f_solve’ y ‘nelmin’ son los metodos que se implementarón para calcular las soluciones de las ecuaciones presentadas en el MGI del robot.

Archivo “m2c.cpp”

En este archivo al principio del método, “f_solve” se declara y da valor a los puntos del trocar del robot Hibou, así:

```
// TROCAR Hibou
double TX = 0.4;
double TY = 0.18;
double TZ = 0.2;
```

Por lo que siempre que hayan cambios en las coordenadas del trocar del robotHibou, esas modificaciones también deben hacerse en este archivo de manera manual.

3.5.5. La Clase "Joystick"

En cuanto a esta clase no es necesario detenerse a mirar los archivos ".h" y ".cpp" pues estos ya están codificados para que funcionen tal y como es requerido, aunque si es necesario realizar unos procedimientos previos de instalación para que Visual pueda compilar el proyecto con las funciones de la librería joystick, que son las librerías SDL (ver anexo B para la instalación).

Otro aspecto importante de esta clase es la habilitación del joystick y la asignación de los botones del mismo para realizar determinadas funciones definidas por el programador:

```
Joystick dispHardware;
double x;
double y;
double z;
bool hX;
bool hY;
bool hZ;
int sRobot;
bool hR;
```

Estos son los atributos necesarios para habilitar el joystick, se ponen en el archivo de cabecera ("Interface.h") del programa principal.

```
sRobot = 0;
hX = false;
hY = false;
hZ = false;
hR = false;
if (dispHardware.joystickNames.count()> 0)
{
      QMessageBox::warning(this, tr("WARNING"),tr("The Joystick is
ready for use"));
```

```
            dispHardware.open(0);

      connect(&dispHardware,SIGNAL(axisValueChanged(int,int)),SLOT(Change
_Axis(int,int)));
            connect(&dispHardware,SIGNAL(buttonValueChanged(int,
bool)),SLOT(Change_Button(int,bool)));
      }
      else
      {
            QMessageBox::warning(this, tr("WARNING"),tr("¡No device
found!"));
      }
```

Esta porción de código habilita el joystick, se coloca en la función principal del "Interface.cpp" donde se encuentran las funciones del programa principal. También retorna mensajes que sirven para saber si el dispositivo funciona o no.

La manera para asignar funciones o tareas a cada botón del joystick es:

```
void Interface::Change_Button(int boton, bool valor)
{
      sRobot=0;
      switch(boton)
      {
      case 0: cout <<"boton 0 del joystick"<< endl;
            break;
      case 1: cout <<"HIBOU"<< endl;
            sRobot = 0;
            text->SetInput("HIBOU");
                  hX = false;
                  hY = false;
                  hZ = false;
            break;
      case 2: cout <<"LAPBOT_1"<< endl;
            sRobot = 1;
            text->SetInput("LAPBOT_1");
                  hX = false;
                  hY = false;
                  hZ = false;
            break;
      case 3: cout <<"LAPBOT_2"<< endl;
            sRobot = 2;
```

```
                text->SetInput("LAPBOT_2");
                      hX = false;
                      hY = false;
                      hZ = false;
                break;
        }

}
```

Donde cada ‘case’ representa cada botón del joystick y a cada uno se le asigna el método (o función) que se desea llamar.

Para este trabajo de grado la herramienta software puede ser manejada desde un solo joystick y con el mismo asignar qué robot se encuentra activo y también ordenar acciones sobre algunos objetos. Por ejemplo durante el procedimiento de extracción de la vesícula, uno de los botones del joystick es el que afirma que se realiza la acción de cortar sobre la respectiva arteria. Para esto se utilizó un control usb para PC joystick Genius Maxfighter F-17 Turbo.

Figura 37: Joystick usado en el proyecto.

Finalmente los robots, órganos y el paciente cargados en la escena se observan en la Figura 38, donde aparece la interfaz desarrollada en VTK usando todas las herramientas

software de apoyo para crear iconos y menús, ventanas y sub-ventanas, y múltiples escenas de renderizado.

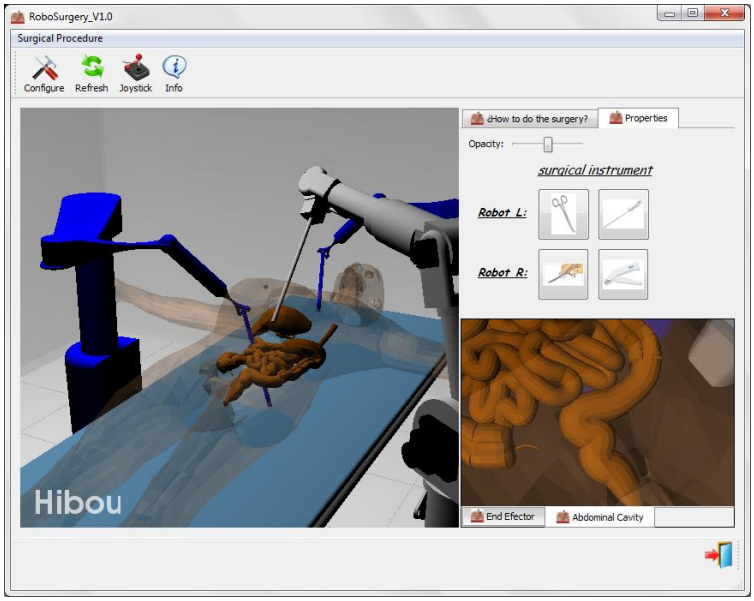

Figura 38: Objetos cargados en dos escenarios de renderizado.

3.6. Cómo usar las herramientas software.

Aquí se presenta una metodología de desarrollo software que aplica a cualquier proyecto de desarrollo que involucre el uso de Qt, Vtk y Visual Studio.

3.6.1. Manejo de Qt, CMake y Visual Studio.

En la metodología planteada para el desarrollo de este proyecto se propone iniciar el desarrollo desde qt-creator.

Uso de Qt-Creator.

Para poder crear botones o insertar imágenes en una interfaz de Qt a través de Visual primero hay que crear los “recursos” en Qt. Esto es indispensable para evitar errores futuros (ver anexo B).

Uso de CMake

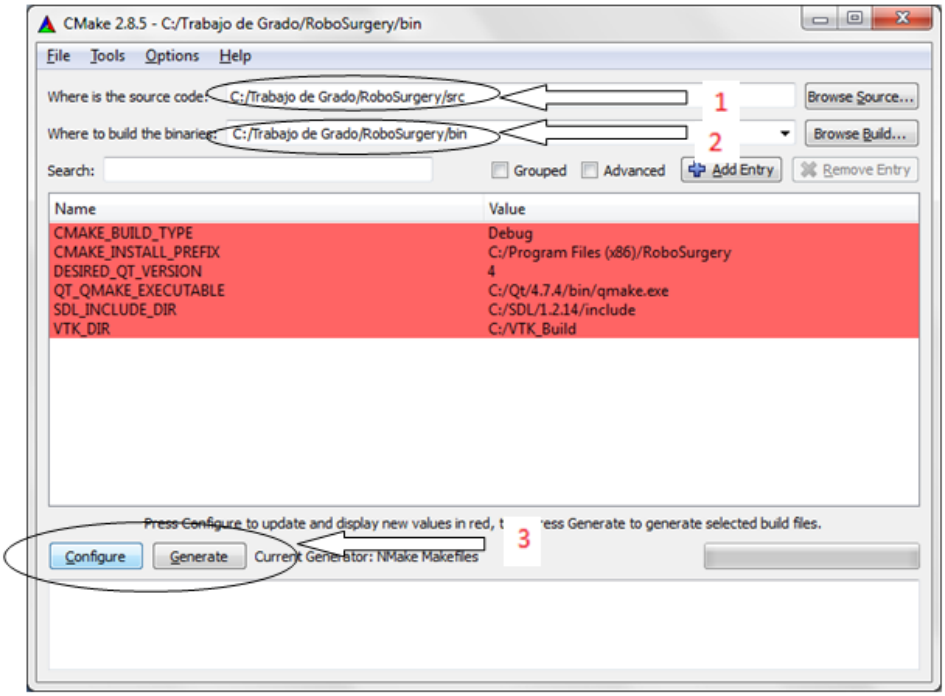

Figura 39: Pasos en CMake para llevar el proyecto a Visual Studio.

1. Abrir Cmake (cmake-gui).
2. Asignar el directorio correspondiente a la carpeta con el código fuente.
3. Asignar el directorio correspondiente a la carpeta donde se pondrán los archivos de Visual Studio
4. Se debe presionar Configurar, esperar a que muestre tal como en la figura anterior, si encuentra las bibliotecas necesarias para codificar. Luego se debe

presionar nuevamente configurar pero esta vez ya no debería aparecer ningún directorio marcado en rojo. Finalmente se presiona “Generate”.

Terminados estos procesos se busca en la carpeta llamada “bin” (se recomienda llamar la carpeta de esta forma). Debe haber un archivo “nombre.sln” como el que se muestra en la siguiente figura:

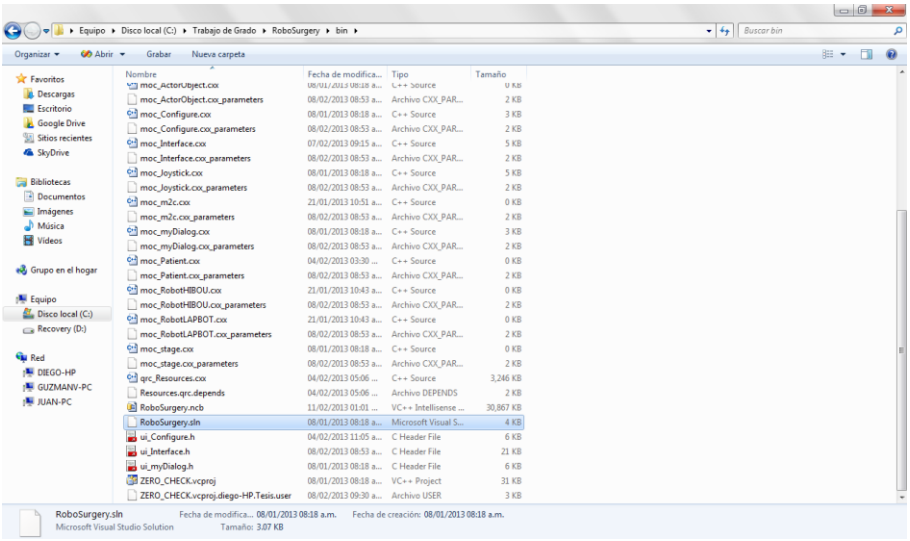

Figura 40: Vista de los archivos generados por CMake en la carpeta bin.

A partir de aquí ya se puede trabajar en Visual Studio, usando Qt y añadiendo todas las bibliotecas necesarias para el programa que se requiera realizar.

Es importante tener en cuenta, cada vez que se añada una nueva clase al código, crear los nuevos archivos “.h”, “.cpp” y “.ui” (si lo tiene) en la carpeta del código fuente y volver a repetir los tres pasos anteriores de **“Usando CMake”**.

3.6.2. Uso de blender para el modelado 3D.

El uso de Blender es de gran ayuda e indispensable para relacionar los objetos en un entorno 3D, por ello para el redimensionamiento y ubicación espacial de los objetos primero se utilizó Blender como intermediario.

Importando-Exportando archivos en Blender:

El proceso de importar y exportar es realmente sencillo, basta con seleccionar los directorios y los formatos adecuados. Lo realmente interesante de este apartado es el uso de Blender para distribuir adecuadamente los objetos 3D en la escena. Para ello como se observa en la siguiente figura, se importan todos los archivos correspondientes a la escena que el programador quiere hacer:

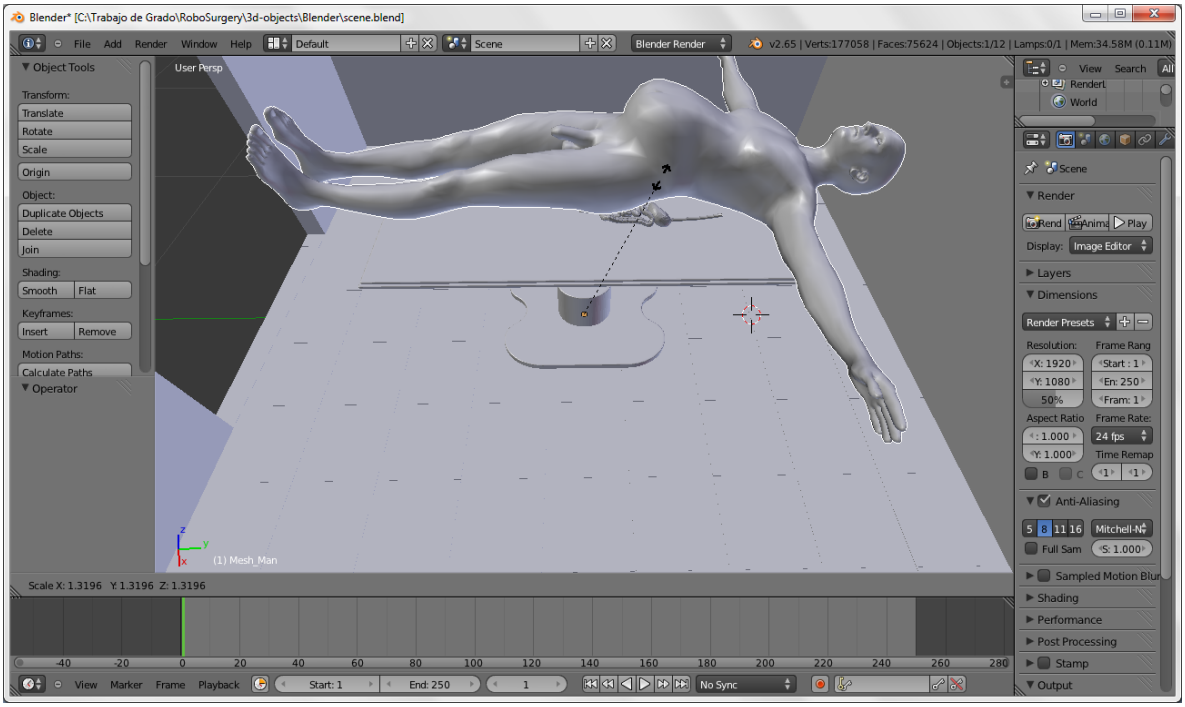

Figura 41: Uso de Blender para redimensionar.

En este caso se importa el archivo “MAN.obj” el cual es entregado por MakeHuman, pero la relación en tamaño del objeto dentro de la escena es extremadamente grande y la posición no es realmente la que corresponde. Por ello se utilizan las funciones de teclado rápido de Blender como:

- G: Permite trasladar el objeto en el espacio para llevarlo a la posición deseada.
- R: Permite rotar al objeto sobre cualquier eje.
- S: Permite escalar el objeto para llevarlo al tamaño deseado.

Con estas tres funciones es suficiente para acomodar los objetos en la escena.

Por último es importante resaltar que Blender también puede actuar de “intermediario” o convertir los archivos de otros formatos 3D, al formato “.obj” que es el usado en la biblioteca de VTK.

3.6.3. Manejo de cámaras y luces en VTK.

Este importante y “olvidado” tema, entra en la sección de detalles para mejorar el desempeño gráfico de VTK, pero aun así es importante para garantizar al usuario final el producto deseado. Por lo tanto:

En el archivo “Interface.h”:

Es necesario incluir estas librerías que trae por defecto VTK:

```
#include"vtkCamera.h"
#include"vtkLight.h"
```

Y se crean los objetos de estas clases (con el nombre que el programador desee):

```
vtkLight *lightStage;
vtkLight *lightStageC;
vtkCamera *cam;
vtkCamera *camC;
```

En el archivo “Interface.cpp”:

```
	//Light
	lightStage = vtkLight::New();
	lightStage->PositionalOn();
	lightStage->SetDiffuseColor(1,1,1);
	lightStage->SetPosition(0,1,0);
	lightStage->SetFocalPoint(0,0,0);
	lightStage->SetIntensity(1);
	lightStage->SetConeAngle(90);
	lightStage->SetLightTypeToSceneLight();
	lightStage->SetColor(1,1,1);
	//Se crea y edita las propiedades de las luces
	lightStageC = vtkLight::New();
	lightStageC->PositionalOn();
	lightStageC->SetDiffuseColor(1,1,1);
	lightStageC->SetPosition(0,1,0);
	lightStageC->SetFocalPoint(0,0,0);
	lightStageC->SetIntensity(1);
	lightStageC->SetConeAngle(160);
	lightStageC->SetLightTypeToSceneLight();
	lightStageC->SetColor(1,1,1);
```

Se crearon los respectivos objetos y se asignaron las propiedades para usarlos en la escena.

```
	//Text
	text = vtkTextActor::New();
	text->SetInput("Academic Tool");
	text->GetTextProperty()->SetFontFamilyToArial ();
	text->GetTextProperty()->SetLineSpacing(1.0);
	text->GetTextProperty()->BoldOn();
	text->GetTextProperty()->SetFontSize(38);
	text->GetTextProperty()->SetOpacity(0.7);
	text->GetTextProperty()->SetColor(1,1,1);
	text->SetDisplayPosition( 15, 15 );
	//Se crea y edita las propiedades de los textos dentro de una
escena de renderizado
	textC = vtkTextActor::New();
	textC->SetInput("NO SIGNAL");
	textC->GetTextProperty()->SetFontFamilyToArial ();
	textC->GetTextProperty()->SetLineSpacing(1.0);
	textC->GetTextProperty()->BoldOn();
	textC->GetTextProperty()->SetFontSize(24);
	textC->GetTextProperty()->SetOpacity(0.7);
```

```
textC->GetTextProperty()->SetColor(1,0,0);
textC->SetDisplayPosition( 15, 15 );
```

Aquí se crearon los objetos correspondientes a los textos que se obervan sobre la escena.

```
//cameras
//Workspace
cam = vtkCamera::New();
cam->SetPosition(1,0.6,1.1);
cam->SetFocalPoint(0.4,0.15,0);
//0.4,0,0.13
//Abdominal Cavity
camC = vtkCamera::New();
camC->SetPosition(0.2, 0.18, 0.2);
camC->SetFocalPoint(0.6, -1.20, -0.25);
camC->Azimuth(5);
camC->Elevation(10);
camC->Roll(0);
camC->SetViewAngle(30);
```

Aquí se crearon las dos cámaras con sus respectivas propiedades para enfocar la escena del render.

Hay dos aspectos importantes aquí, primero se crean dos objetos de cada tipo con el fin de poder usarlos en dos escenas de renderizado diferentes, por esta razón la estructura lógica del llamado a cada método es tal y como se muestra. El segundo aspecto observables que en este archivo cada objeto debe crearse, y asignársele el valor a las diferentes propiedades. Por ejemplo:

camC->SetPosition(0.2, 0.18, 0.2);

camC->SetFocalPoint(0.6, -1.20, -0.25);

Estos 2 métodos permiten gestionar la posición de la cámara dentro de la escena de render.

Finalmente dentro de este mismo archivo se debe hacer lo siguiente:

```
render->SetActiveCamera(cam);

renderC->SetActiveCamera(camC);

render->AddLight(lightStage);

renderC->AddLight(lightStageC);
```

Estas funciones sirven para asignar a cada escena de renderizado los objetos respectivos, sin esto, a pesar de crear los objetos no se verá ningún cambio en la escena.

3.6.4. Manejo de interfaces, para crear mensajes o menús.

Crear interfaces es una de las actividades más importantes pues permiten mejorar las prestaciones de un desarrollo, para crear interfaces ver anexo B.2.

3.6.5. Uso y aprovechamiento de la consola

Uno de los elementos propios de la interfaz es la consola (pantalla negra) que se ejecuta paralelamente al programa, este espacio puede ser aprovechado de múltiples maneras con funciones básicas como:

cout << "opacity= " << Opacity << endl;

Donde,

- cout > función de c++, que indica que el contenido siguiente debe imprimirse en la consola.
- “opacity= “ > muestra el contenido escrito entre las comillas.
- Opacity > variable que queremos se imprima.
- Endl > indica que debe hacerse un salto de línea.

Desde el punto de vista del programador le permite conocer el valor de algunas variables durante la ejecución del programa, o comprobar que el programa está entrando en

determinados métodos. Se puede utilizar también para apoyarse en el proceso de compilación, pues hay ocasiones donde no hay error de sintaxis[16] pero la lógica en el código está mal manejada

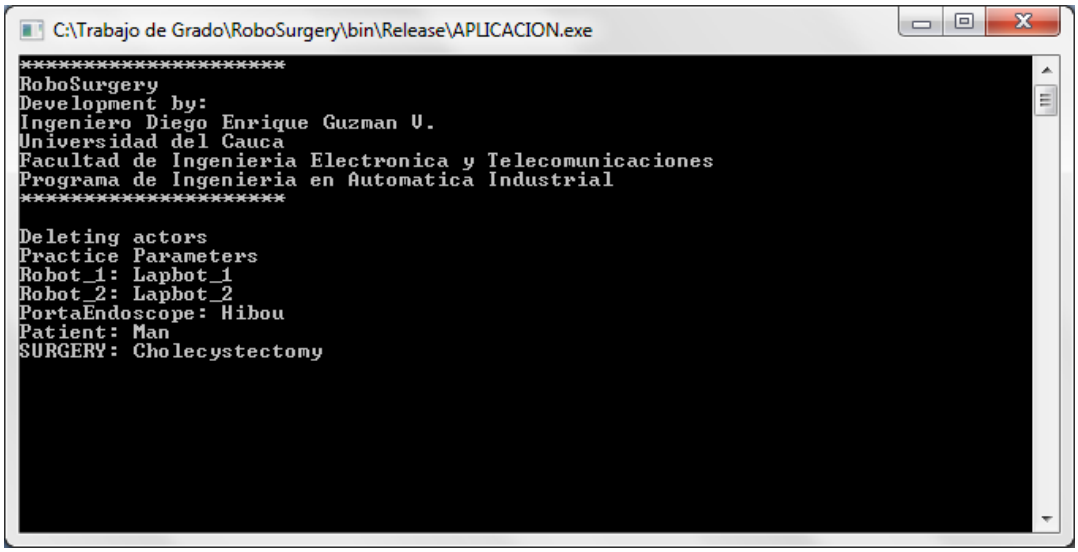

Figura 42: Vista de la consola de RoboSurgery presentando alguna información extra.

[16]Sintaxis: Conjunto de reglas y principios que definen el lenguaje usado para dar una orden o acción al programa codificado.

4. ROBOSURGERY, RESULTADOS OBTENIDOS.

En este capítulo se presentan los resultados según los objetivos planteados en el proyecto, las características de RoboSurgery y los componentes de este.

4.1. Componentes de la herramienta desarrollada.

Al iniciar RoboSurgery desde el ejecutable, luce como se muestra en la Figura 43.

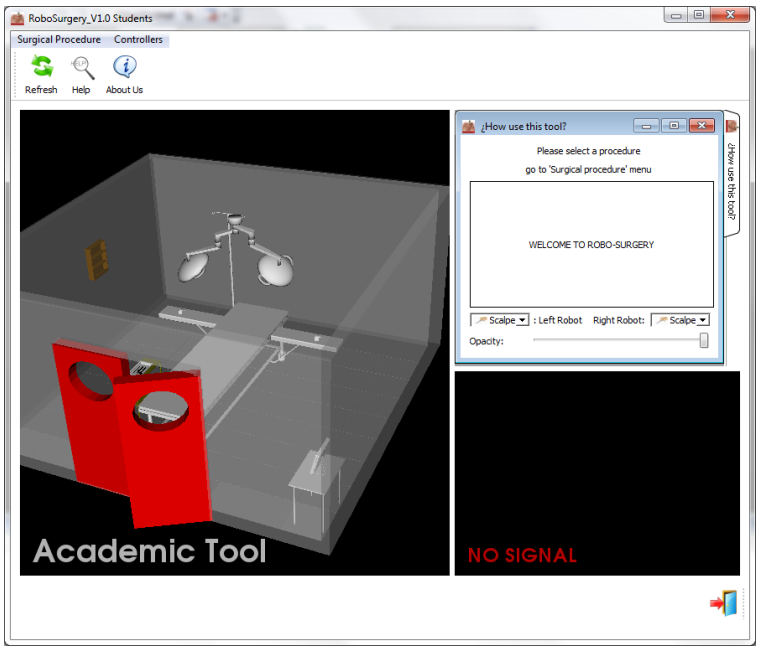

Figura 43: RoboSurgery.

RoboSurgery se desarrolló con propósitos académicos, orientado a los robots quirúrgicos usados en procedimientos laparoscópicos.

La Figura 44: Opciones del menú ‘Surgical Procedure’ (derecha), Opciones menú ‘Controllers’ (izquierda).Figura 44 muestra la barra de herramientas, esta presenta dos opciones, ‘Surgical Procedure’ y ‘Controllers’.

Figura 44: Opciones del menú 'Surgical Procedure' (derecha), Opciones menú 'Controllers' (izquierda).

Surgical Procedure muestra las prácticas que se pueden hacer en la herramienta. Se desarrolló la opción de realizar una laparoscopia diagnóstica, donde no se puede hacer un procedimiento quirúrgico como tal si no explorar el interior del paciente con propósitos diagnósticos. También se puede realizar un procedimiento quirúrgico en esta primera versión, la colecistectomía, en esta se pueden dirigir los robots implementados en el software para que realicen la extracción de la vesícula.

Controllers permite habilitar el 'joystick' para la manipulación de los robots, este funciona como se presenta en la Figura 45:

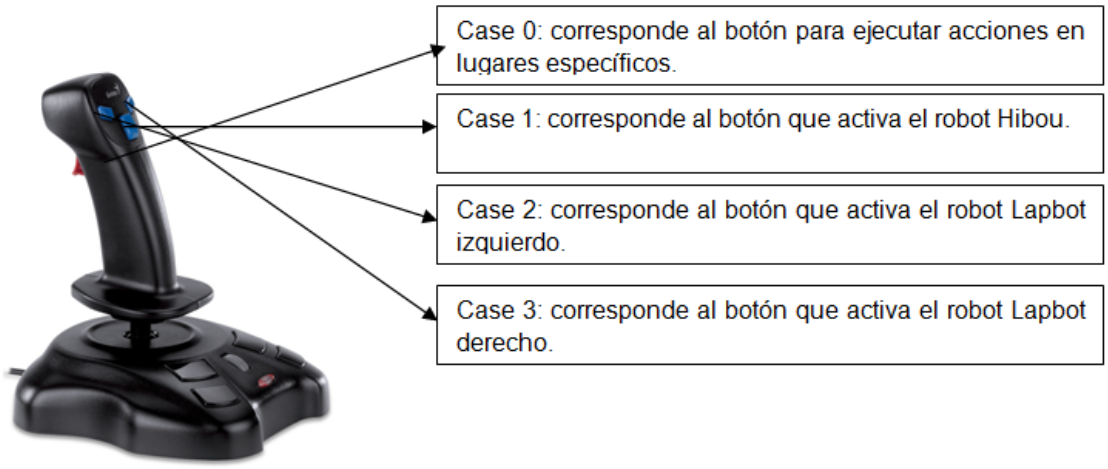

Figura 45: Distribución de los botones en el 'joystick'.

La barra de herramientas:

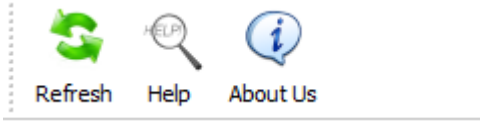

Figura 46: Barra de herramientas.

Esta presenta tres opciones sobre la interfaz, ‘Refresh’ ajusta la cámara del espacio de trabajo a su posición inicial. Ya que la cámara del espacio de trabajo puede ser manipulada por medio del ‘mouse’ es fácil perder la posición inicial para realizar el procedimiento.

‘Help’ presenta la interfaz observada en la Figura 47, esta es una guía de usuario donde se puede encontrar respuesta a las posibles dudas sobre el manejo de la herramienta.

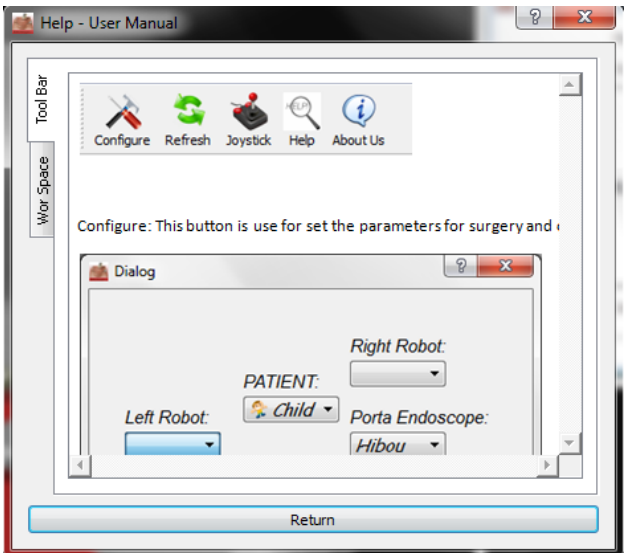

Figura 47: Interfaz de ‘Help’.

Finalmente la opción ‘About Us’ presenta una interfaz como la observada en la Figura 48, la cual muestra información sobre el desarrollador de este proyecto.

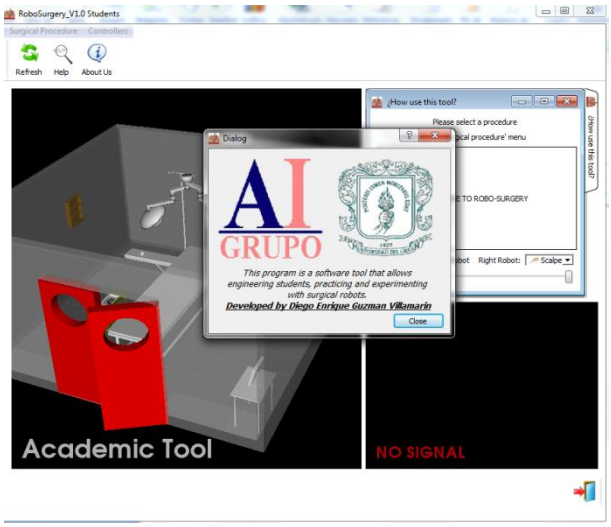

Figura 48: Interfaz 'About Us'.

Este proyecto requiere el renderizado en múltiples ventanas usando VTK, estos se observan en la Figura 49, ya que una presenta el escenario donde se encuentra el paciente y las respectivas herramientas para llevar a cabo el procedimiento. La ventana más pequeña muestra la vista interna del paciente, manipulada por el robot porta-endoscopio Hibou.

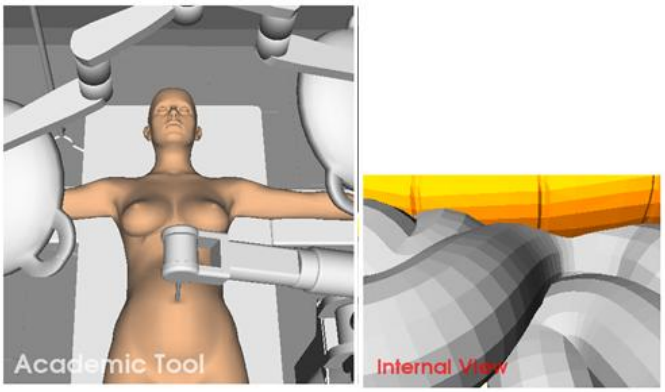

Figura 49: Escenario del paciente (izquierda), vista interna del paciente (derecha).

El menú de Controles, observado en la Figura 50, tiene 3 áreas importantes:

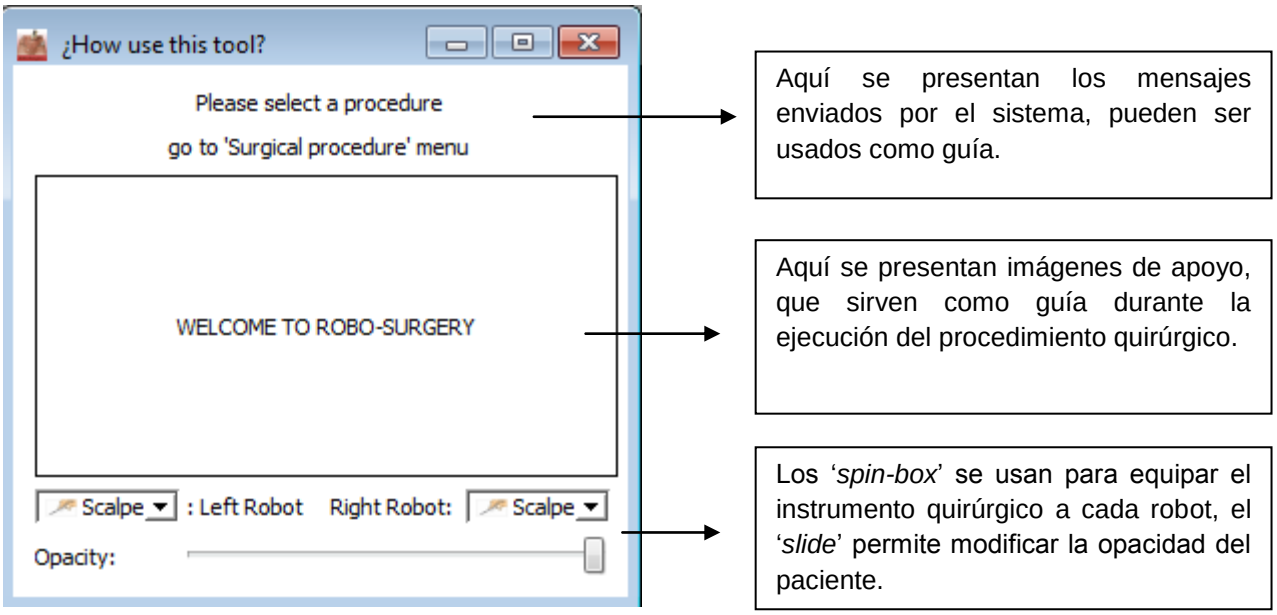

Figura 50: Menú de controles.

Es importante tener en cuenta que para realizar correctamente el procedimiento quirúrgico se debe equipar los instrumentos correctos a cada robot.

4.2. La colecistectomía.

Para llevar exitosamente a cabo la extracción de la vesícula, usando RoboSurgery el procedimiento se desarrolla así:

1. Iniciar el programa.

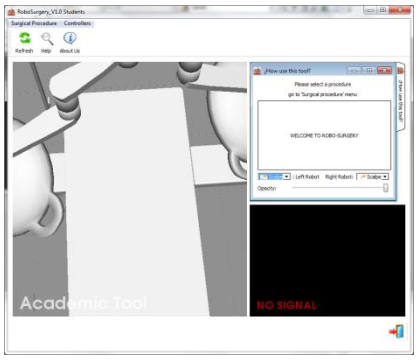

Figura 51: Abriendo RoboSurgery.

2. Ir al menú y escoger el procedimiento a realizar.

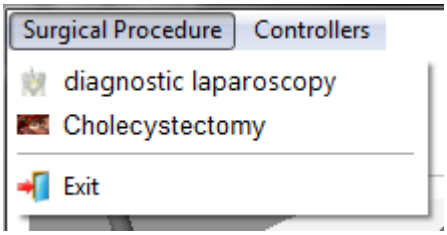

Figura 52: Menú de herramientas de RoboSurgery.

3. Seleccionar el paciente que se desee usar.

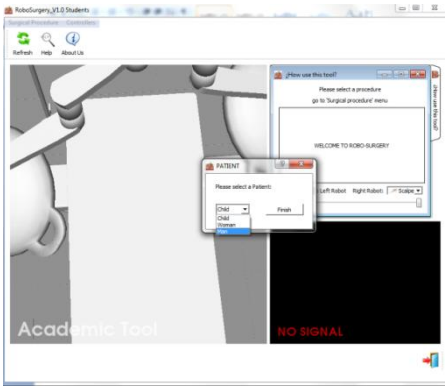

Figura 53: Interfaz de selección de pacientes en RoboSurgery.

4. Ir al menú y escoger el control que se usará para manipular los robots.

Figura 54: Menú de selección de control de RoboSurgery.

Desde este punto comienzan a ejecutarse las etapas quirúrgicas.

5. Presionar el botón central del joystick activar el robot Hibou.

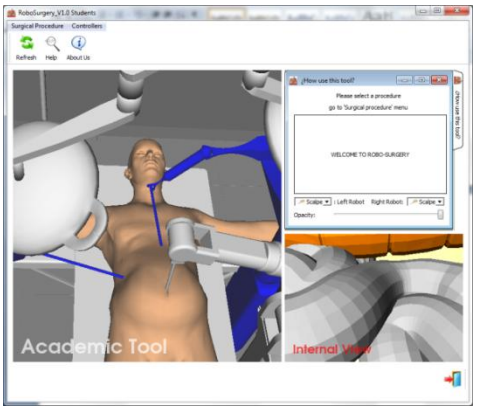

Figura 55: Vista de RoboSurgery para iniciar el procedimiento quirúrgico.

6. Mover el robot hasta que la cámara alcance la posición necesaria para observar correctamente la vesícula biliar, desde una perspectiva clara.

Figura 56: Posicionando la cámara del endoscopio.

7. Activar el robot Lapbot izquierdo con el botón izquierdo del joystick.

Figura 57: Tomando la vesícula para moverla.

8. Llevar el Lapbot izquierdo hasta la vesícula y cambiar el instrumento quirúrgico por pinzas, para poder tomar la vesícula hay que presionar el gatillo al frente del joystick y moverla de su posición hacia arriba. **Esto es equivalente a hacer el paso 1, exposición del conducto y arteria císticos.**

Figura 58: Equipando el robot LapBot izquierdo con pinzas.

9. Activar el Lapbot derecho con el botón derecho del joystick y llevarlo hasta la parte inferior de la arteria. **Esto es equivalente a la disección del conducto cístico.**

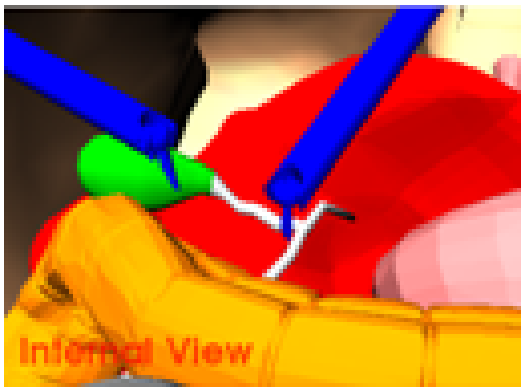

Figura 59: Llevando el robot LapBot derecho a la arteria.

10. Presionar el gatillo del joystick y equipar la grapadora para cortar el flujo de sangre en la parte inferior de la arteria. **Equivalente a la sección del conducto y la arteria císticos.** Ir a la parte superior de la arteria, y poner otra grapa igual que en el punto 9.

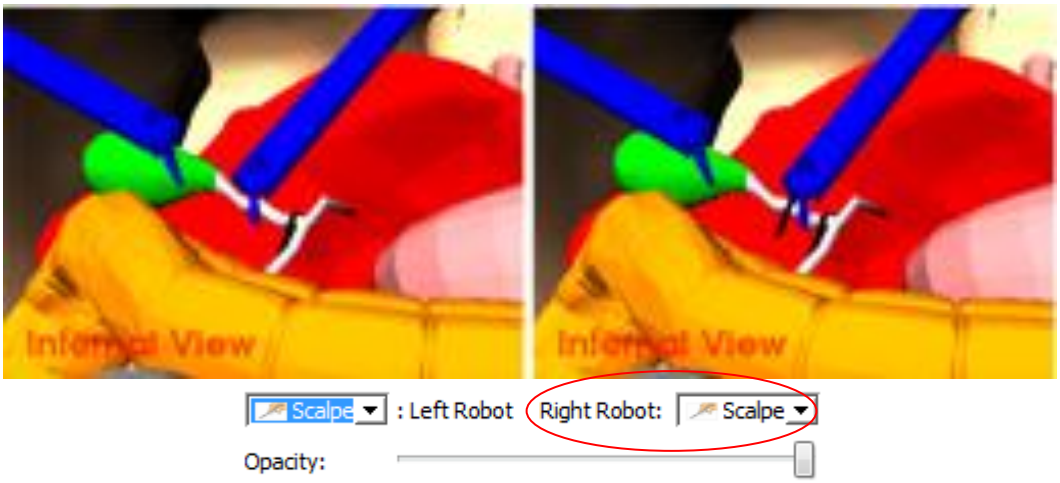

Figura 60: Poniendo grapas en la arteria.

11. Cambiar el instrumento en el LapBot derecho por bisturí, cuando la arteria se vea roja, significa que esta ha sido cortada. **Equivalente a la disección de la vesícula del lecho hepático.**

Figura 61: Cortando la arteria para la extracción de la vesícula.

12. Finalmente se extrae la vesícula biliar.

Figura 62: Vista desde el porta-endoscopio de la vesícula extraída.

Además en cualquier momento puede ser usado el 'slide' opacity para cambiar el nivel de opacidad del paciente, por si se quisiera ver desde el Workspace el procedimiento desde una vista más amplia.

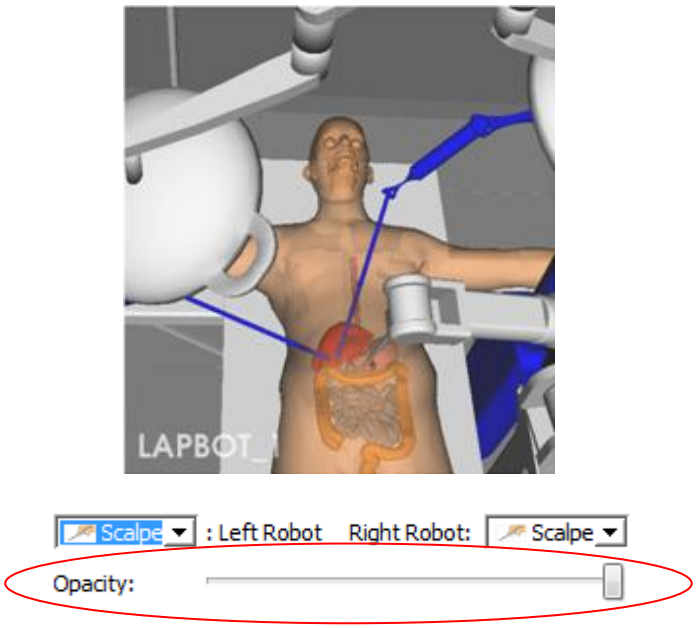

Figura 63: Paciente con menor grado de opacidad.

La cámara del Workspace puede ser movida a gusto por el usuario utilizando el mouse.

- Ventana de dialogo del ‘About Us’.

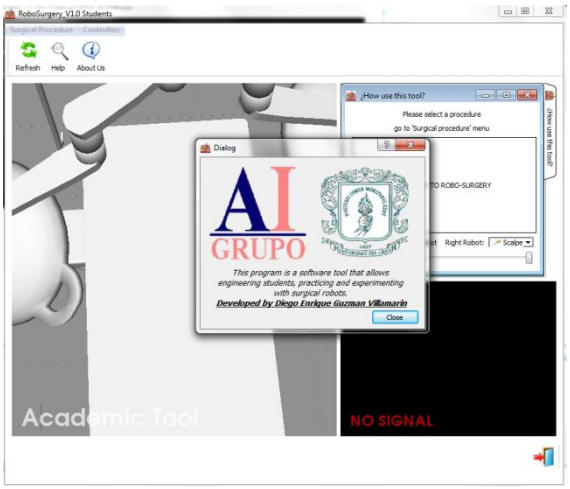

Figura 64: Interfaz About Us.

- Ventana de dialogo ‘Help’.

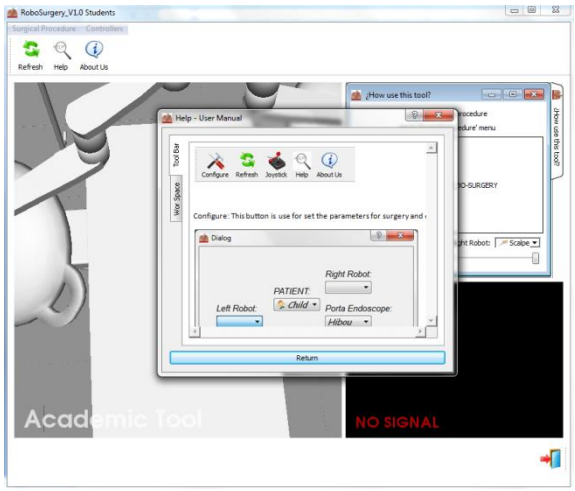

Figura 65: Interfaz Help.

4.3. Laparoscopia diagnostica.

Este procedimiento adicional, permite utilizar el robot porta-endoscopio en un paciente y mover la cámara internamente. Su propósito es netamente diagnostico, no permite realizar ningún procedimiento quirúrgico como tal.

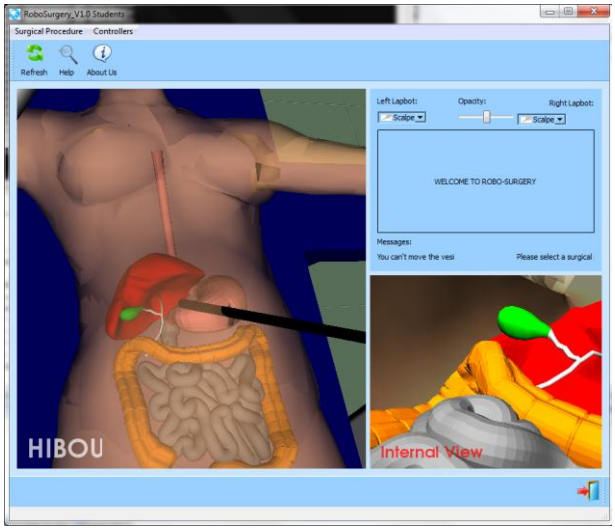

Figura 66: RoboSurgery en modo “diagnostical laparoscopic”.

4.4. Los pacientes.

De acuerdo al objetivo propuesto, se incluyó la opción de realizar cualquier procedimiento con un hombre, mujer o niño, como los observados en la Figura 67.

Figura 67: Pacientes usados en la herramienta.

Estos pacientes se crearon en ‘MakeHuman’ y fueron importados en Blender para adecuar las proporciones y finalmente se llevaron a RoboSurgery.

Es importante señalar de acuerdo a los resultados obtenidos en el estudio “Colecistectomía de urgencia laparoscópica versus abierta” [32], la edad mínima para realizar el procedimiento de la laparoscopia es de 20 años y máximo hasta 87 años. Por lo que se puede considerar “incorrecto” hablar de niño como paciente en el software.

5. CONCLUSIONES Y TRABAJOS FUTUROS.

5.1. Conclusiones.

Se usaron los robots Hibou y LapBot pues estos fueron previamente modelados y simulados en la Facultad en anteriores proyectos de grado, su fortaleza radica en el respeto por el paso por el trocar y su diseño permite alcanzar cualquier posición dentro del abdomen del paciente.

La herramienta software se construyó con diversas herramientas libres: por medio de Qt se diseñó la interfaz, por medio de Blender se modelaron los objetos 3D, por medio de Make-Human se modelaron los humanos, por medio de CMake se exportaron los archivos generados en Qt para poder codificar en Visual Studio, herramienta en la cual se integra todo el desarrollo de RoboSurgery. Sin embargo los robots sobre los cuales se construyó la herramienta fueron diseñados y simulados lo que implicaba al momento de integrarlos considerar la forma en que fueron desarrollados, buscando que funcionen bajo un mismo espacio. Cada robot fue necesario considerarlo como una clase.

Al contar con un medio virtual que permita a través de prueba y ensayo aprender la respuesta que se genera sobre los controles a un robot quirúrgico, el cual pretende de una manera mínimamente invasiva, generar el menor riesgo, con los menores recursos, permitiéndole al estudiante aprender con sus errores.

Luego de un análisis previo en el estado del arte, se llegó a la conclusión de que no existe software de simulación en robótica quirúrgica. En la actualidad existen paquetes de simulación de robótica en entornos industriales, y software de simulación en medicina pero nunca apoyados por robots. Por lo que se vio necesario el desarrollo de una herramienta que pudiera hacer prácticas quirúrgicas usando robots y le permitiera al ingeniero conocer el comportamiento de uno de estos robots al ser controlados sobre el cuerpo humano.

El tener una herramienta libre que no genera costos adicionales para su implementación permite que ese costo que se ahorra en la virtualización del procedimiento como tal sea multiplicado, al tampoco generar costos en el manejo de la herramienta software.

Este proyecto de grado propuso un método diferente de desarrollo en VTK, que permitió que al codificar el software se optimizara la cantidad de líneas de códigos y la cantidad de consumo computacional por parte del software RoboSurgery, mejorando el tiempo de respuesta y simulación.

5.2. Trabajos futuros.

Es necesario en futuras versiones ir aumentando las funcionalidades de este desarrollo, por ejemplo, aumentar el número de robots usados e implementar más cirugías.

El uso de algoritmos de reconstrucción 3D y funciones para el manejo de deformaciones, permitirían potenciar la herramienta ya que harían que la herramienta software se acerque mucho más a la realidad, permitiendo una práctica más realista y de la cual se puede sacar provecho en muchas áreas.

Otro aspecto sería adicionar una interfaz háptica para la realimentación de fuerzas, lo que haría que la práctica fuera mucho más realista, haciendo de la herramienta software una herramienta de simulación capaz de mostrar la respuesta real de los robots en el entorno.

6. REFERENCIAS BIBLIOGRAFICAS.

[1] “Retro informática”. [En línea] Disponible en: http://www.fib.upc.edu/retro-informatica/avui/salut.html. Consultado: [Febrero de 2013].

[2] “Simbionix™”. [En línea] Disponible en: http://simbionix.com/simulators/lap-mentor/. Consultado: [Marzo de 2013].

[3] “Limb & Things UK”. [En línea] Disponible en: http://limbsandthings.com/uk/products/lapsim-laparoscopic-trainer/. Consultado: [Marzo de 2013].

[4] “slideshare”. [En línea] Disponible en: http://www.slideshare.net/Haptica/haptica-promis-sages-2009. Consultado: [Marzo de 2013].

[5] Freund E.; “Cosimir® Getting Started”. Institute of Robotics Research, Alemania. 2000.

[6] “User Manual”. [En línea] Disponible en: http://www.intelitekdownloads.com/Manuals/Robots/100346-b-RoboCell-usb-v47(0210).pdf. Consultado: [Marzo de 2013].

[7] “Easy-Rob”. [En línea] Disponible en: http://www.easy-rob.com/easy-rob/ Consultado: [Marzo de 2013].

[8] Franco A.; Rodríguez C. “Modelado y Animación con el ROBOWORKS”. Universidad EAFIT – Departamento de Ingeniería de Producción, 2005.

[9] Carol E.; Scoot-Conner. “Laparoscopic gastrointestinal surgery”. Medical Clinics of North America, Vol. 86, No. 6, pp 1401-1422, 2002.

[10] Khaitan L.; Holzman M. “Laparoscopic advances in general surgery”. The Journal of the American Medical Association, Vol 287, No. 2, pp 1502-1506, 2002.

[11] “Laparoscopia ginecológica”. [En línea] Disponible en: http://www.proyecto-bebe.es/la_laparoscopia.htm Consultado: [Enero de 2013].

[12] Medina M. Patient Information: About laaroscopy and endoscopy. Society of laparoscopic Surgeons.

[13] “Salutaris Medical Center”. [En línea]. Disponible en: http://salutarismedicalcenter.blogspot.com/2012/11/cirugia-vesicula-biliar-colecistectomia-laparoscopica-piedras-calculos-guadalajara.html Consultado: [Enero de 2013].

[14] “Enfermedades de la vesícula biliar”. [En línea]. Disponible en: http://www.nlm.nih.gov/medlineplus/spanish/gallbladderdiseases.html Consultado: [Enero 2013].

[15] “Cirugia digestiva y endocrina”. [En línea]. Disponible en: http://drmarin.galeon.com/cctlap.htm Consultado: [Febrero de 2013].

[16] “Aurora Health Care®”. [En Línea]. Disponible en: http://www.aurorahealthcare.org/yourhealth/healthgate/getcontent.asp?URLhealthgate=%22177901.html%22 Consultado: [Enero de 2013].

[17] Salinas S.; Vivas A. “Modelado, simulación y control del robot para cirugía laparoscopica ‘Lapbot’”. Revista Chilena de Ingeniería, Vol. 17, No. 3, pp 317-328, 2009.

[18] Salinas S.; Vivas A. “Simulación 3D de Movimientos Quirúrgicos de una Colecistectomía asistida por el Robot ‘LapBot’”, VIII Congreso de la Asociación Colombiana de Automática, Cartagena, Colombia, 2009.

[19] Méndez C.; Torres V.; Salinas S.; Vivas A. “Diseño y simulación en 3D de un robot porta endoscopio para cirugía laparoscópica”. V Seminario Internacional de Ingeniería Electrónica, Bucaramanga, Colombia, 2011.

[20] Muñoz C.; Ordoñez J. “Software manipulador del robot porta endoscopio Hibou para cirugía laparoscópica”. Trabajo de grado (Ingeniería en Automática Industrial). 2012.

[21] Mayorca H.; Torres A. “Entorno gráfico de un simulador quirúrgico 3D”. Trabajo de grado (Ingeniería en Automática Industrial). 2011.

[22] “Qt Project”. [En línea]. Disponible en: http://qt-project.org/resources/getting_started Consultado: [Noviembre 2012].

[23] “Qt Project”. [En línea]. Disponible en: http://qt-project.org/doc/qt-4.8/signalsandslots.html Consultado: [Noviembre 2012].

[24] “Visualización Toolkit”. [En línea]. Disponible en: http://www.vtk.org/ Consultado: [Noviembre 2012].

[25] “Blender”. [En línea]. Disponible en: http://www.blender.org/education-help/quickstart/ Consultado: [Noviembre 2012].

[26] “CMake”. [En línea]. Disponible en: http://www.cmake.org/ Consultado: [Noviembre 2012].

[27] “Make Human”. [En línea]. Disponible en: http://www.makehuman.org/sectionview/291 Consultado: [Noviembre 2012].

[28] Craig L. “Applying UML and Patterns: an introduction to Object.oriented análisis and design and interative development”. Prentice Hall. 3er ed. 2005.

[29] “Casos de Uso”. [En línea]. Disponible en: http://users.dcc.uchile.cl/~psalinas/uml/casosuso.html Consultado: [Diciembre 2012].

[30] “Modelo de clases”. [En línea]. Disponible en: http://users.dcc.uchile.cl/~psalinas/uml/modelo.html Consultado: [Diciembre 2012].

[31] “Conceptos y principios orientado a objetos”. [En línea]. Disponible en: http://www.elguille.info/colabora/NET2005/Percynet_Conceptosyprincipiosorientadoaobjetos.htm Consultado: [Diciembre 2012].

[32] Chávez J; Amezcua J. “Colecistectomía de urgencia laparoscópica versus abierta”. Cirujano General, Vol. 34, No. 3, pp 174-178, 2012.

Anexo A: Instalación de herramientas software.

Para complementar la guía: “Introducción al Qt y Qt-creator” del profesor Oscar Andres Vivas Albán, se redacto esta guía de instalación de vtk, para asi tener la biblioteca de VTK para codificar usando Visual Studio.

A.1. PASOS PARA INSTALAR VTK.

Una vez terminado de instalar las herramientas según la guía “Introducción al Qt y Qt-creator” se procede asi:

1. Crear 3 carpetas

- VTK.
- VTK_Source.
- VTK_Build.

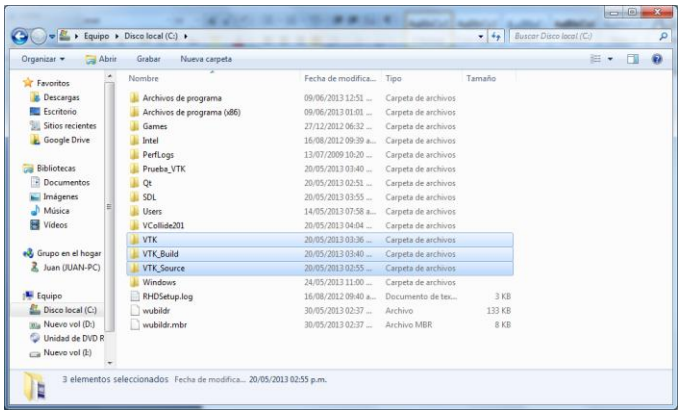

Figura A.1. Creación de directorios para VTK.

Recomendación: Estas carpetas deben crearse en el disco c, por ejemplo, con la siguiente dirección:

"C:\VTK"

2. Iniciar "Cmake" (el gui, se encuentra en "bin"), apuntar los path a las carpetas correspondientes en source y build, respectivamente. Presionar configure:

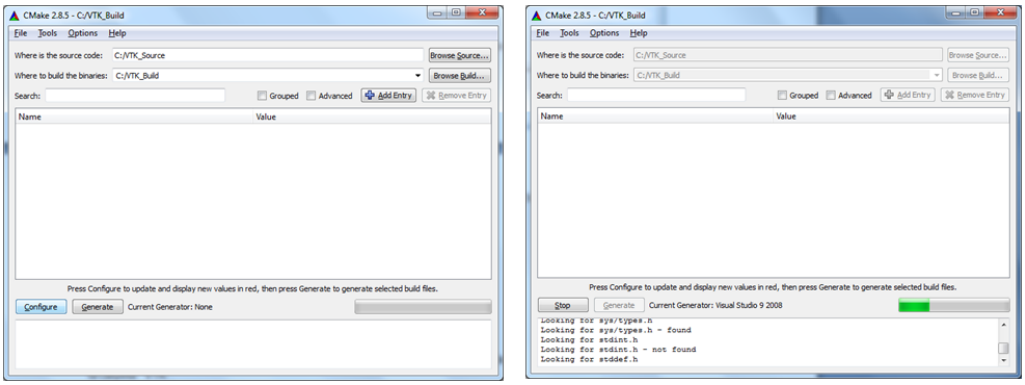

Figura A.2. Configurar path en CMake.

3. En las opciones en rojo:

- Cambiar el path de "C:/Program Files/VTK", al correspondiente a la ubicación de la carpeta "VTK".

- Marcar las casillas de verificación.
  - vtk_use_guisuport
  - vtk_use_qt
  - vtk_build_Examples

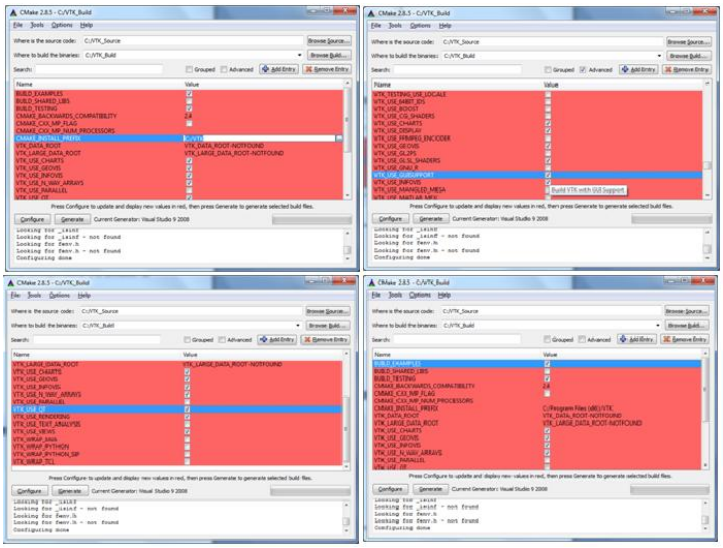

Figura A.3. Configurar CMake.

Presionar "configure".

4. En la segunda ronda, marcar en las opciones en rojo:

- vtk_to_opengl

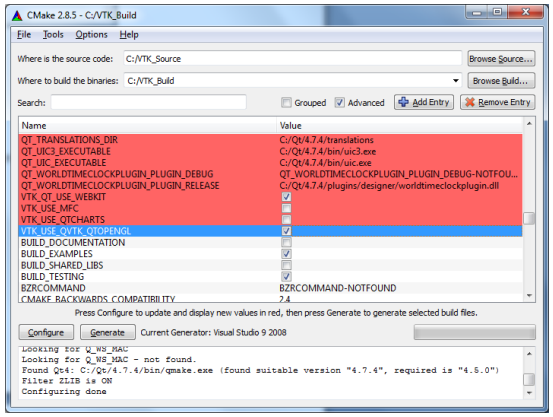

Figura A.4. Configuración CMake después del primer configure.

Presionar "configure".

5. Presionar "generate". (al terminar cerrar cmake)

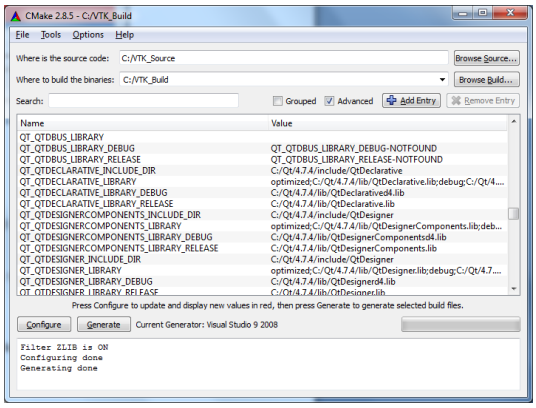

Figura A.5. CMake al terminar configuración.

6. Buscar en la carpeta VTK_Build, el archivo VTK.sln (archivo de visual estudio), he iniciarlo.

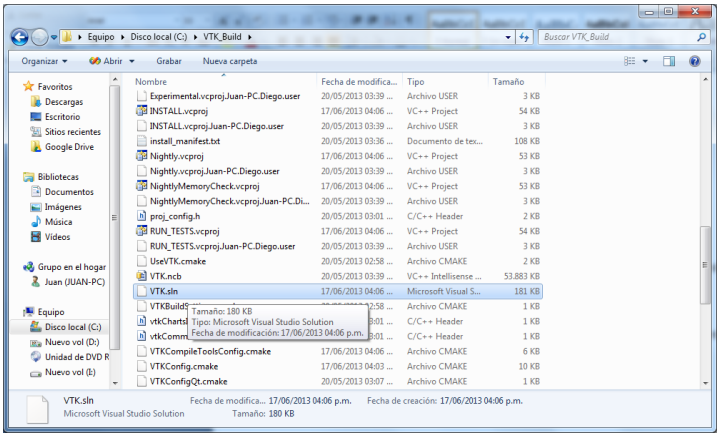

Figura A.6. Archivos generados en la carpeta bin.

7. En la parte superior, en la barra de herramientas baja, buscar "debug" y cambiarlo por "release".

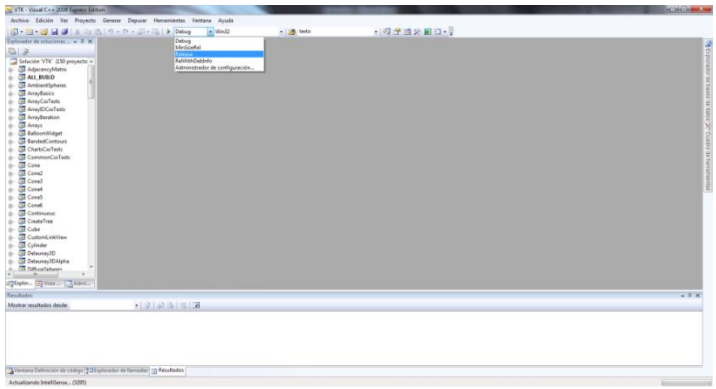

Figura A.7. Modo de funcionamiento de Visual Studio.

8. En el lado izquierdo, señalar "all build", ir al menú de "Generar", y seleccionamos "Generar ALL BUILD".

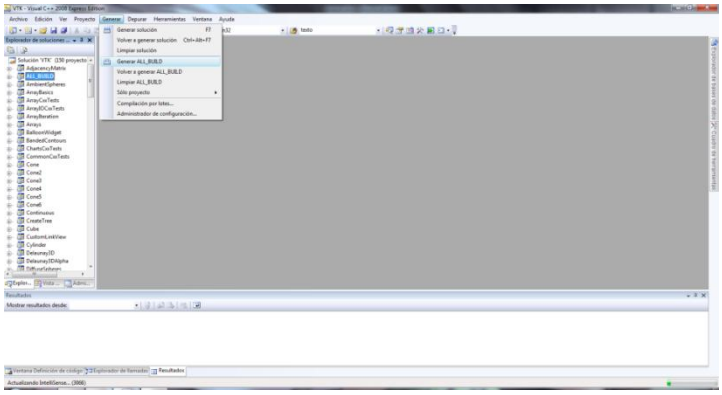

Figura A.8. Generar “ALL BUILD”.

(el procedimiento de compilación demora aproximadamente 2 horas).

9. En el lado izquierdo, buscamos "INSTALL", vamos al menú "Generar", y seleccionamos "Generar INSTALL".

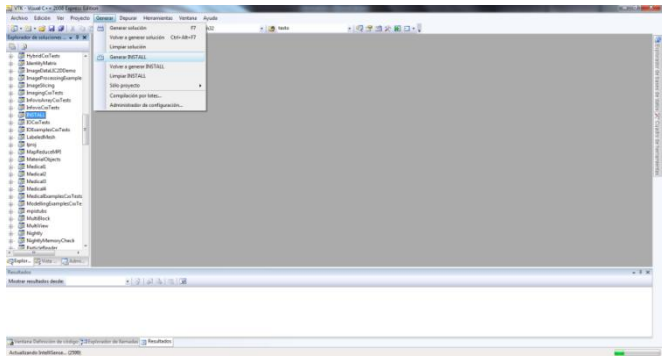

Figura A.9. Generar “INSTALL”.

10. Una vez concluyan las operaciones de compilación, nos dirigimos al menú "Herramientas", "Opciones", en las opciones al lado izquierdo, encontramos un submenú "Proyectos y soluciones", verificamos en "Directorios de VC++".

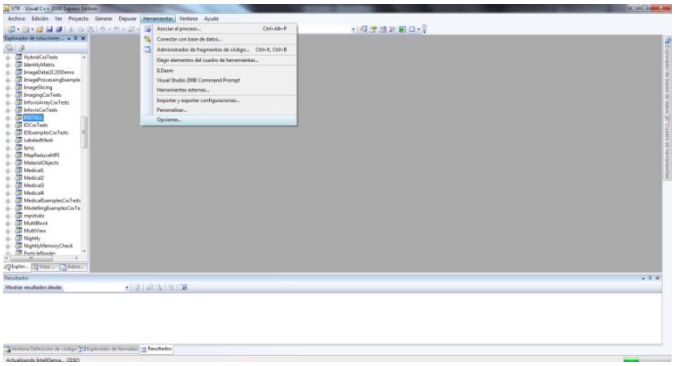

Figura A.10. Buscando “Proyectos y soluciones”.

11. En la lista de "Mostrar directorios para:", verificamos que en "archivos de inclusión" se encuentre el siguiente path:

"C:\VTK\include\vtk-5.8". Si no se encuentra, lo añadimos seleccionando la opción de "añadir nueva línea", y una vez agregada, damos en la opción "Comprobar entradas".

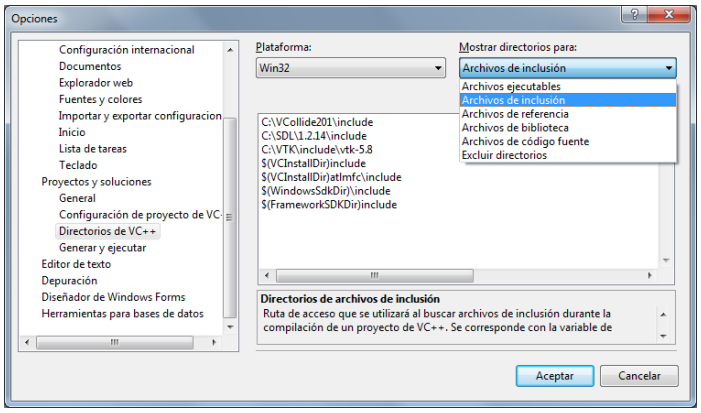

Figura A.11. Verificando archivos de inclusión.

12. En la lista de "Mostrar directorios para:", verificamos que en archivos de inclusión se encuentre el siguiente path:

"C:\VTK\lib\vtk-5.8". En caso contrario, lo añadimos seleccionando la opción de "añadir nueva línea", y una vez agregada, damos en la opción "Comprobar entradas".

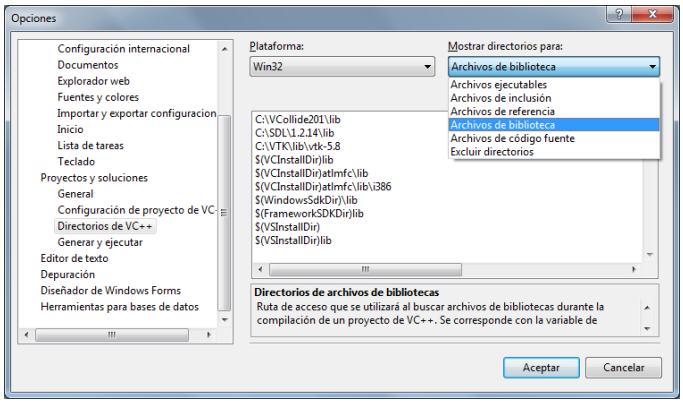

Figura A.12. Verificando archivos de biblioteca.

A.2. COMO VERIFICAR EL CORRECTO FUNCIONAMIENTO DE VTK.

1. Descomprimimos el archivo "Lectura" (una carpeta con un código para introducir archivos .obj en una interfaz de Qt), y creamos una carpeta (vacía) llamada "Lectura_Build".

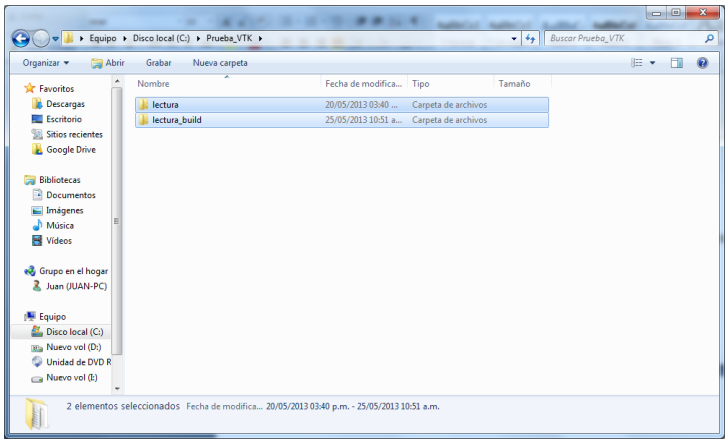

Figura A.13. Creando directorios de el programa “Lectura”.

2. Abrimos el gui "Cmake", y en source apuntamos a la dirección del archivo "Lectura", y en build, apuntamos a la carpeta "Lectura_Build".

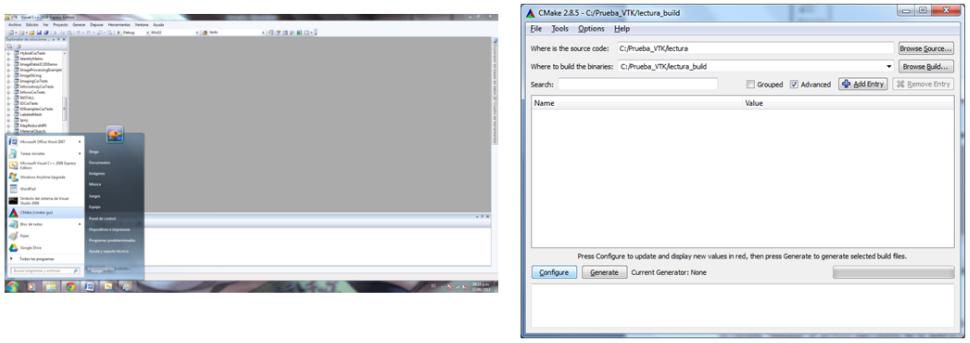

Figura A.14. Abriendo y configurando path de CMake.

3. Presionamos "configure" 2 veces.

4. Cerramos "Cmake", y nos dirigimos a la carpeta, "Lectura_Build", se busca dentro el archivo Lectura.sln.

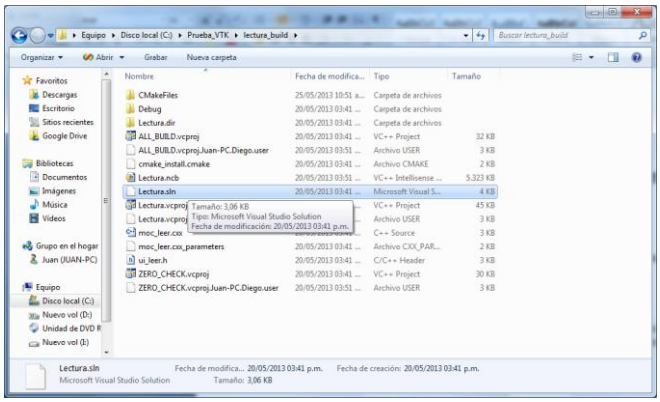

Figura A.15. Archivos contenidos en la carpeta Lectura_build.

5. Se compila el código así:

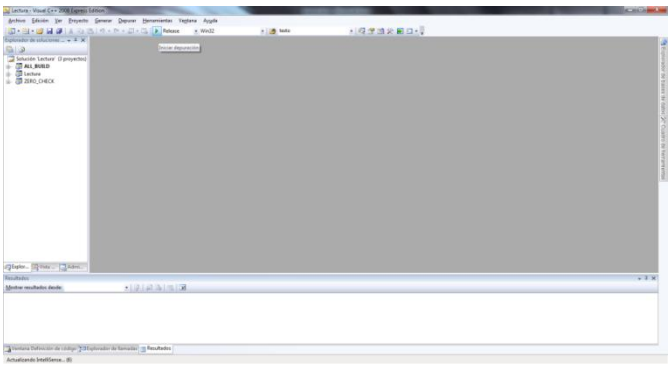

Figura A.16. Compilar código en Visual Studio.

6. En la carpeta Lectura_Build se genera una carpeta llamada "Release", dentro de esta debe haber un archivo llamado Lectura.exe

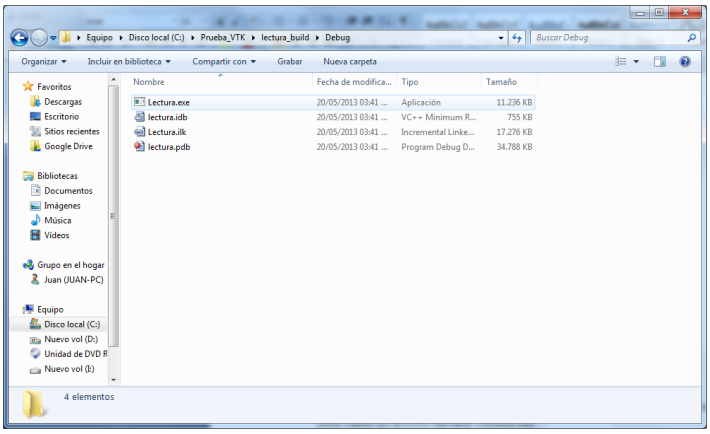

Figura A.17. Abriendo el archivo ejecutable.

7. Si el programa de "VTK" quedo bien instalado, al presionar "ARCHIVO", buscamos un archivo con extensión ".obj", y debe mostrar, el objeto dentro del ejecutable sin problema.

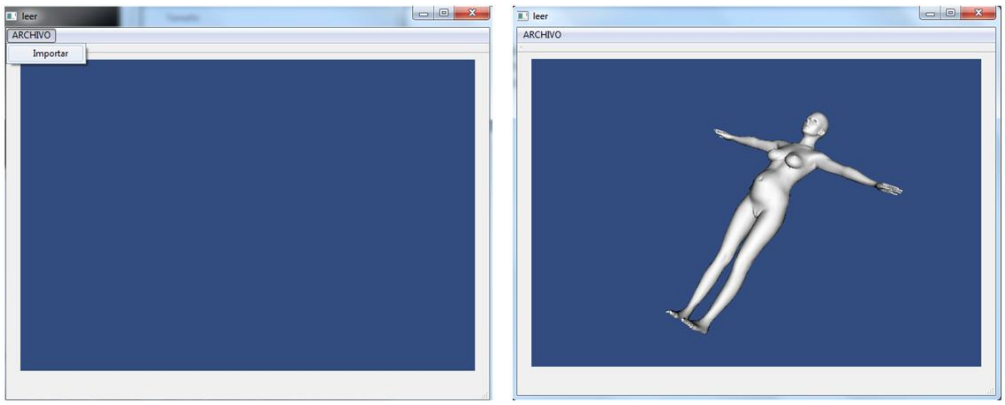

Figura A.18. Probando el programa “Lectura.exe”.

Adicionalmente se incluye en el paquete de entregables el programa “Lectura” para probar vtk, todos los paquetes de instalación de las herramientas Qt, Visual Studio, CMake, vtk. Finalmente se incluye algunos programas desarrollados para apoyarse en el desarrollo, como un programa para probar cámaras y un programa para probar posiciones de objetos dentro de la escena.

A.3. INSTALACIÓN DE BIBLIOTECAS.

Para instalar tanto SDL, VCollide y Rapid (o para cualquier otra) se realiza el siguiente procedimiento:

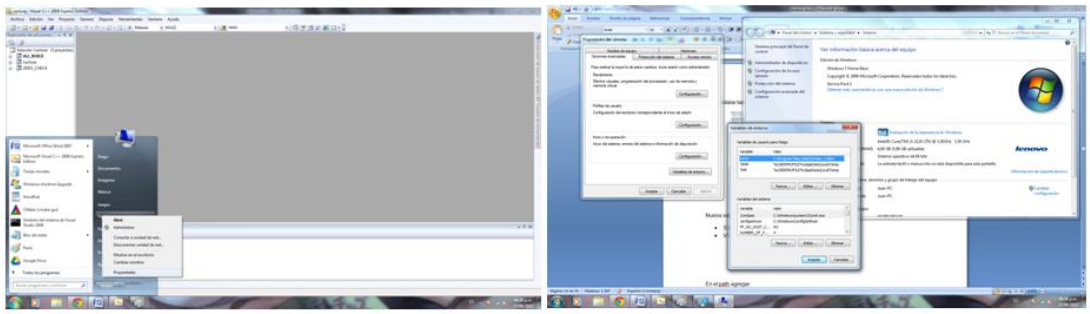

Figura A.19. Variables de entorno de Windows 7.

Nueva variables de entorno:

- SDLDIR C:\SDL\1.2.14
- VCOLLIDEDIR C:\VCollide201

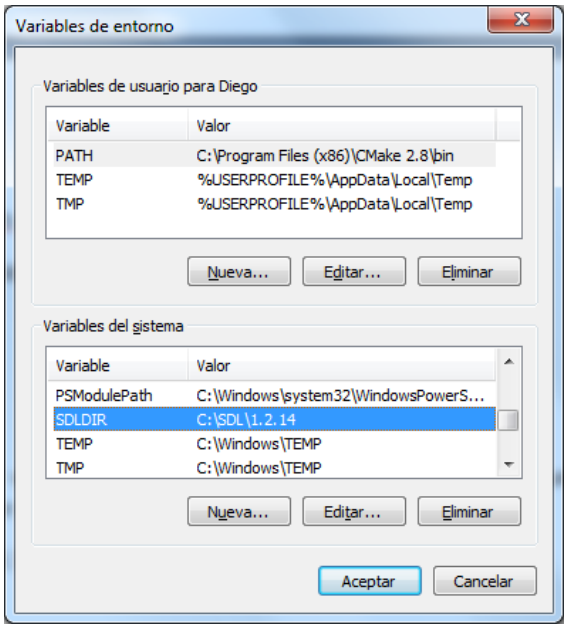

Figura A.20. Creando variables de entorno.

En el path agregar:

- ;C:\VCollide201\lib
- ;C:\SDL\1.2.14\lib

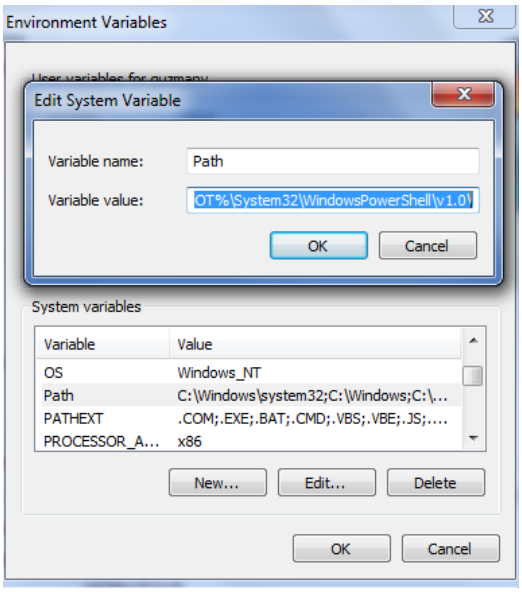

Figura A.21. Agregando directorios al path.

Reiniciar el computador, pues Windows no reconoce las nuevas variables de entorno hasta que se reinicie el sistema.

En visual, Proyectos y soluciones:

Archivos de inclusión:

C:\SDL\1.2.14\include

C:\VCollide201\include

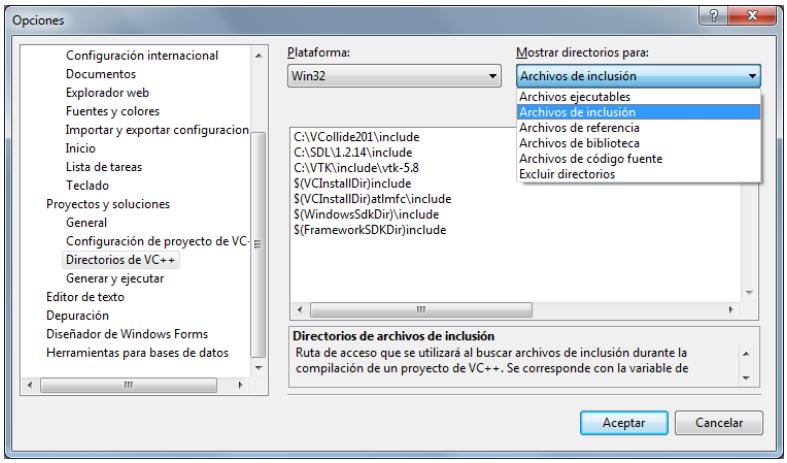

Figura A.22. Agregando archivos de inclusión.

Archivos de biblioteca:

C:\SDL\1.2.14\lib

C:\VCollide201\lib

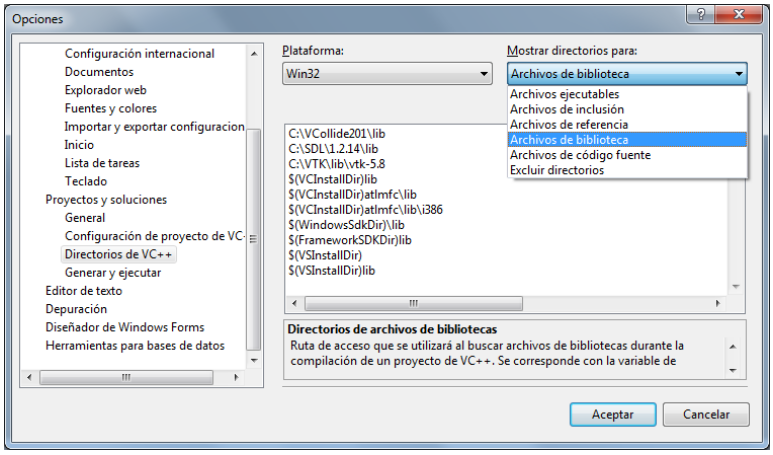

Figura A.23. Agregando archivos de biblioteca.

Los respectivos programas de instalación de cada herramienta, se incluyen en los entregables del trabajo.

Anexo B: Creación de recursos e interfaces.

Esta es una de las tareas iníciales o previas a empezar a desarrollar un proyecto, no es necesario tener todas las imágenes para recursos elegidas, pero es importante configurar primero esto ya que en fases posteriores del proyecto sería realmente problemático hacer esto desde visual.

B.1. CREACIÓN DE RECURSOS.

1. Dentro de la carpeta con el código fuente se crea una carpeta para almacenar los recursos a usar. Por recursos hace referencia a todas las imágenes e iconos que se usaran para desarrollar el software, asi:

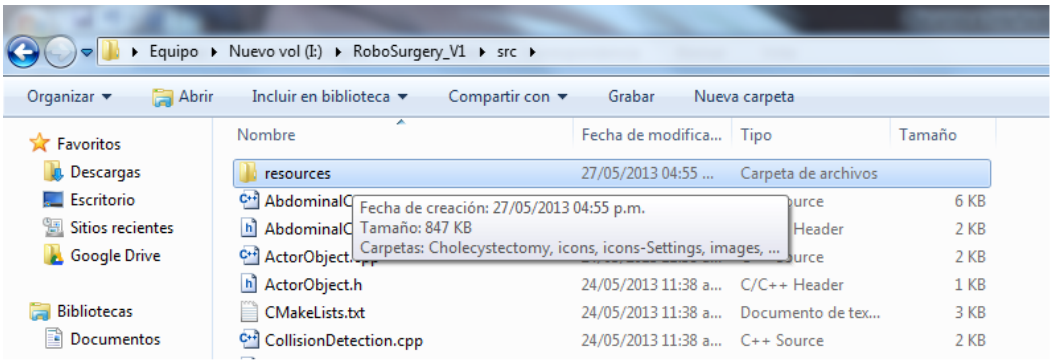

Figura B.1. Creando carpeta de recursos dentro de la carpeta “src”.

2. Dentro de la misma se guardan los archivos a utilizar, preferiblemente en formato '.png' ya que este permite guardar la opacidad de una imagen y así quitarle el fondo blanco (usando un programa de apoyo como por ejemplo GIMP).

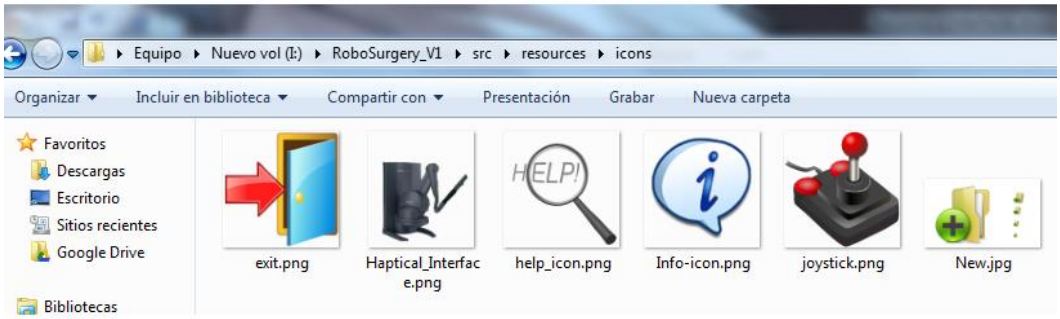

Figura B.2. Guardando recursos para usar en el proyecto.

3. Abrimos (o creamos) el proyecto en Qt-Creator, clic derecho en el proyecto, y agregamos nuevo:

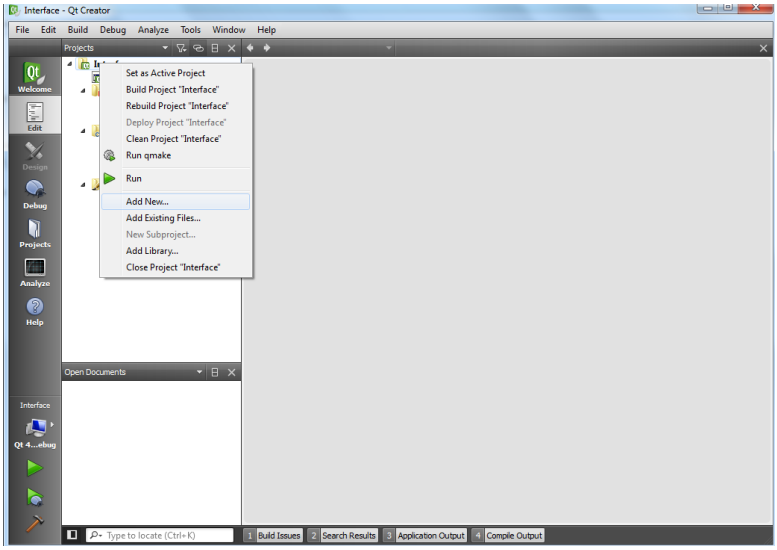

Figura B.3. Agregando nuevos archivos al proyecto.

4. Seleccionamos 'Qt Resource file'.

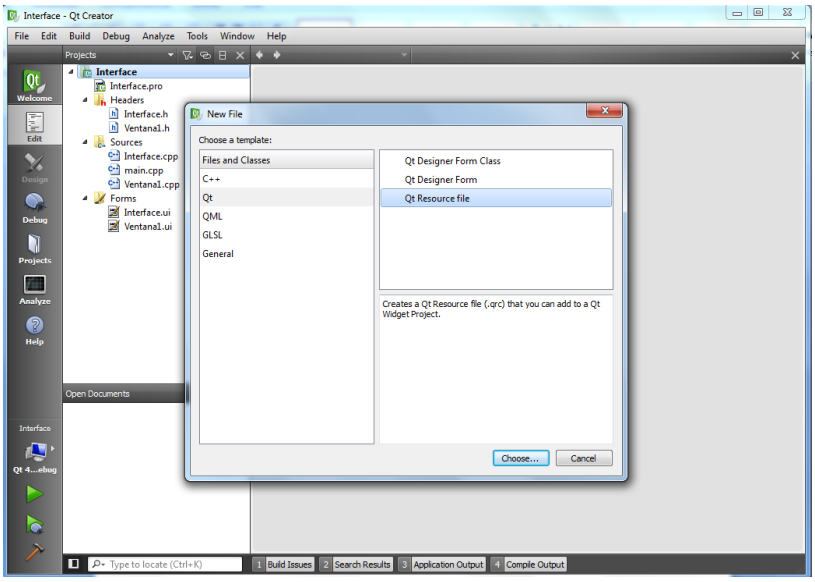

Figura B.4. Agregamos un archivo de tipo “resource”.

5. Le asignamos un nombre.

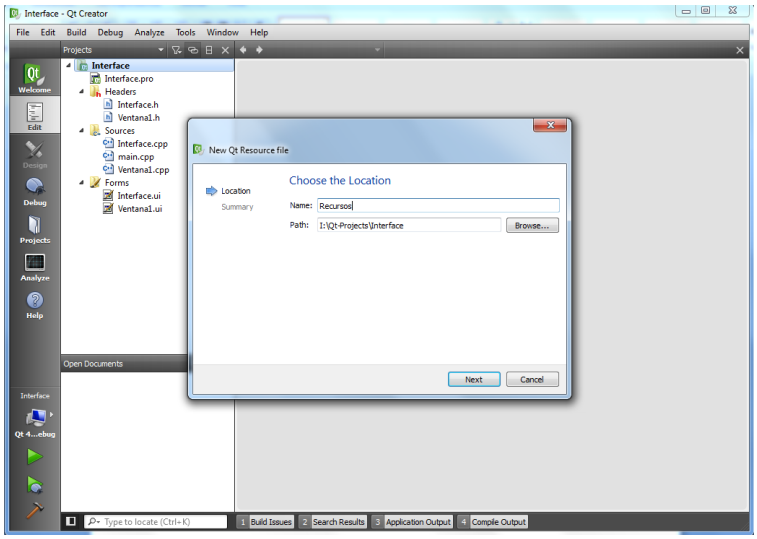

Figura B.5. Asignando nombre al archivo de recursos.

6. Ahora se da clic en 'add' y luego 'add prefix'.

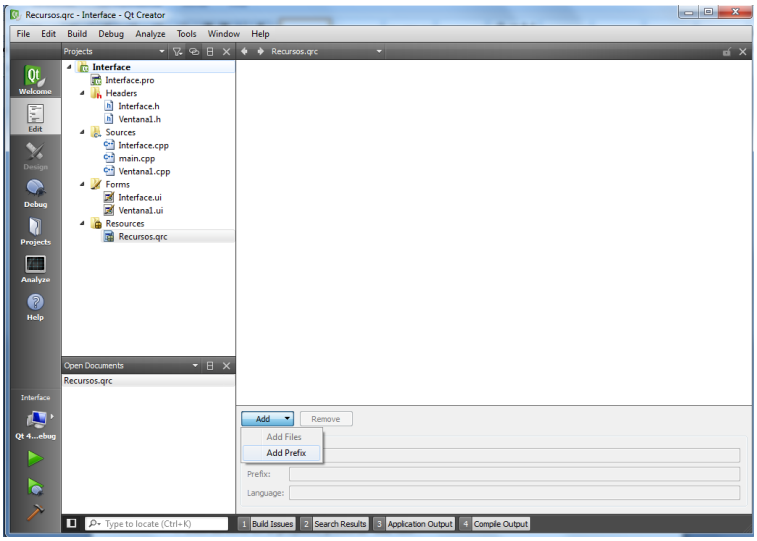

Figura B.6. Creando directorio de recursos del proyecto.

7. Se asigna el nombre deseado, preferiblemente sencillo y fácil de recordar para cuando haya que asignar recursos desde el código. Luego se presiona en 'add files'.

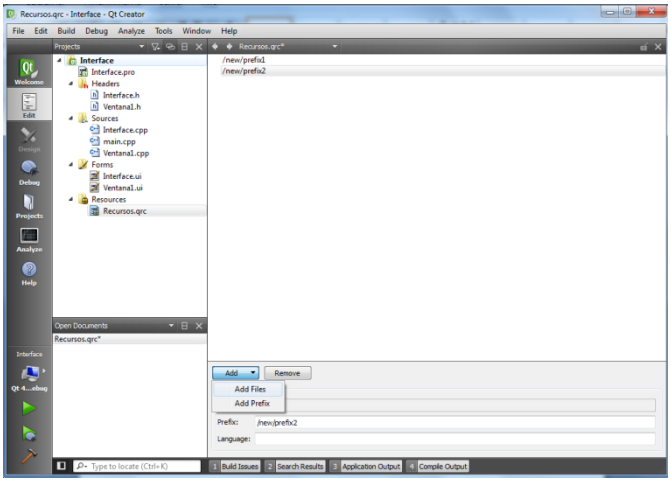

Figura B.7. Adicionando archivos para usar como recursos.

8. Finalmente se pegan los archivos de Qt en la carpeta del código fuente del proyecto en visual, y se repiten todos los pasos de "Usando CMake".

9. Abrimos el archivo .sln generado por CMake dentro de la carpeta “bin”.

10. Se procede a abrir en Qt-Dessigner la interfaz, y se da clic en 'edit resources'.

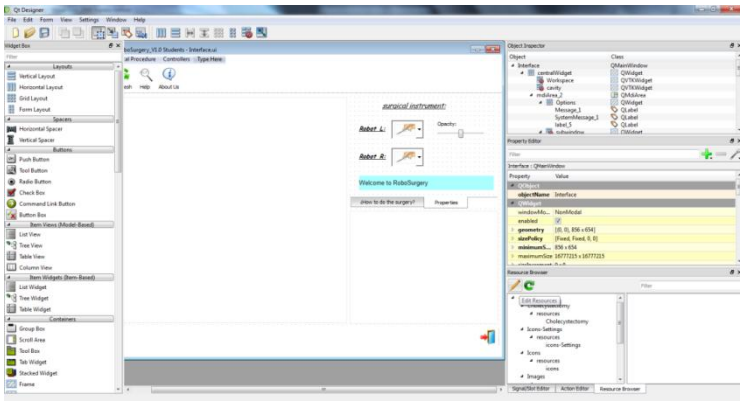

Figura B.8. Asignando recursos en la interfaz.

11. Se abre el archivo creado en Qt, y automáticamente carga todos los recursos.

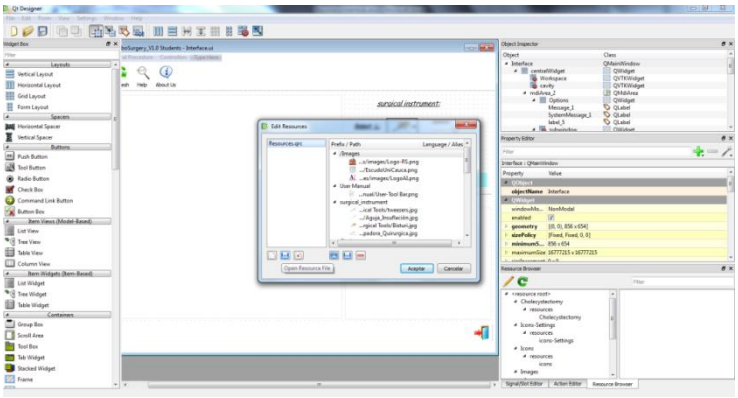

Figura B.9. Vista de los recursos asignados desde Qt.

El proyecto ya está listo para usar imágenes y crear iconos (recursos).

B.2. CREACIÓN DE INTERFACES.

1. Clic derecho en el proyecto, presionar 'add new'.

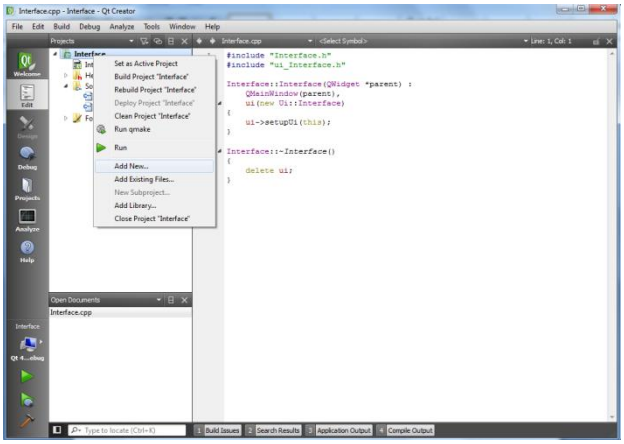

Figura B.10. Adicionando nuevos archivos.

2. En la siguiente ventana seleccionar 'Qt Designer Form Class'

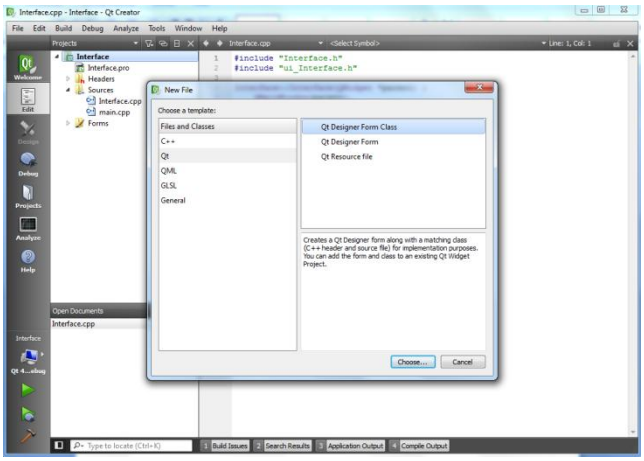

Figura B.11. Creando archivo tipo "Qt Designer Form Class".

3. Seleccionamos el tipo de widget que se desea crear, por ejemplo 'Dialog without Buttons'.

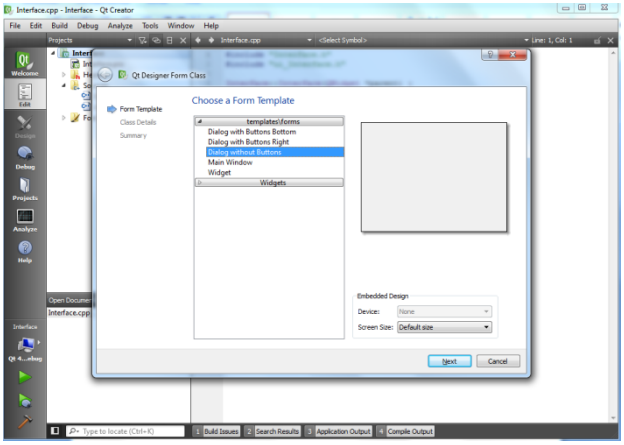

Figura B.12. Seleccionando tipo de widget para crear.

4. Le asignamos un nombre a la clase, se recomienda sea tal cual como la llamaremos dentro de nuestro proyecto en visual.

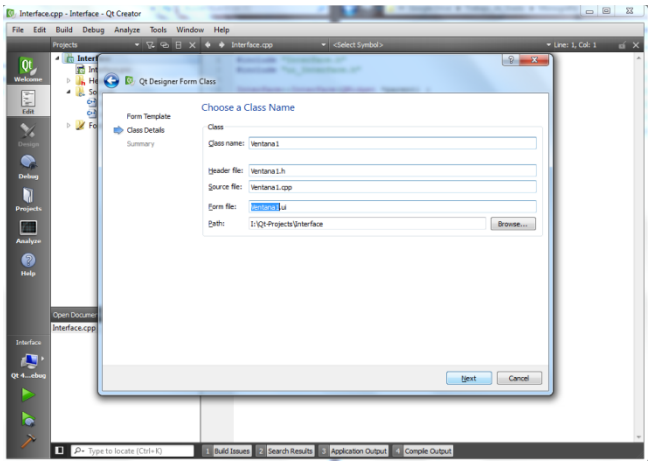

Figura B.13. Dando nombre al nuevo widget.

5. Se abre la carpeta donde se encuentra el código fuente del proyecto en Qt, se elige los archivos corres a la nueva clase, se copian.

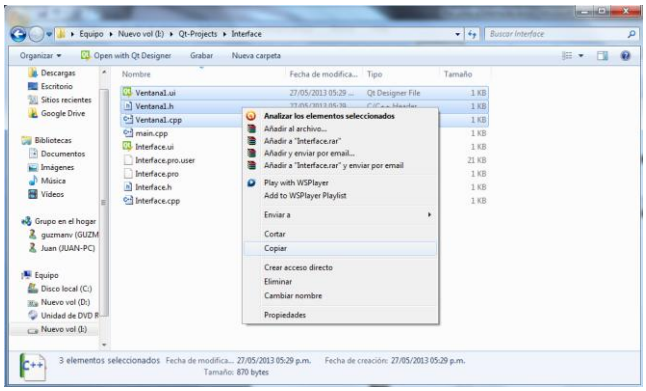

Figura B.14. Seleccionando los archivos que corresponden a la nueva interfaz.

6. Finalmente se pegan los archivos en la carpeta del código fuente del proyecto en visual, y se repiten todos los pasos de "Usando CMake".

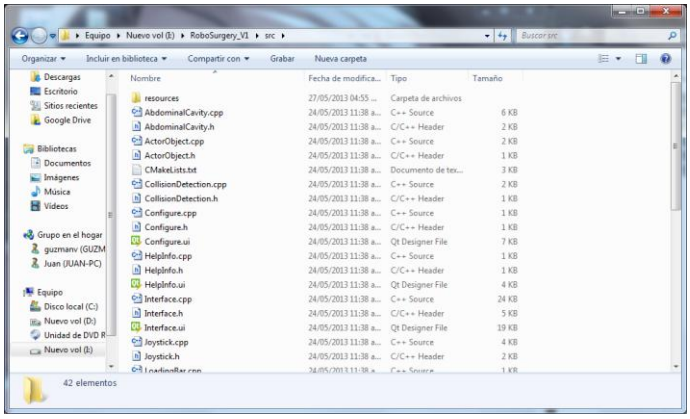

Figura B.15. Adicionando nuevos archivos al código fuente.

7. Se abre el archivo del proyecto '.sln', en el archivo '.h' de la clase principal, en la sección de archivos de cabecera se incluyen los archivos '.h' de las nuevas interfaces.

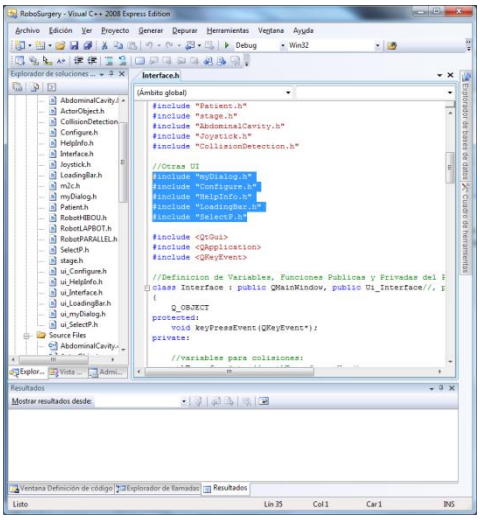

Figura B.16. Agregando cabeceras de nueva interfaz.

8. Luego entre los atributos privados se crean los objetos respectivos.

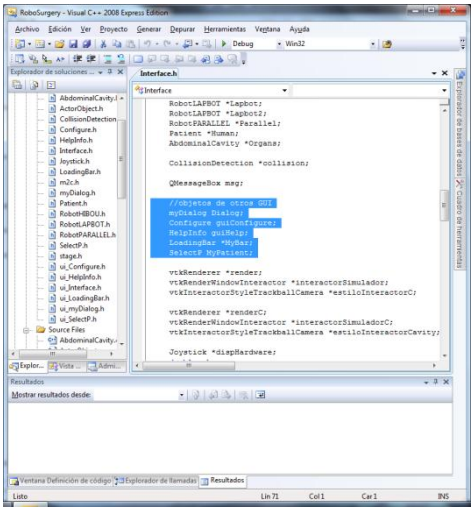

Figura B.17. Creando objetos correspondientes a las nuevas interfaces.

9. Es importante señalar una diferencias, hay objetos que son puntero, y otros que no. Los que son punteros se utilizan en casos donde queremos mostrar una interfaz pero la interfaz de atrás debe seguir funcionando.

Estos se codifican así:

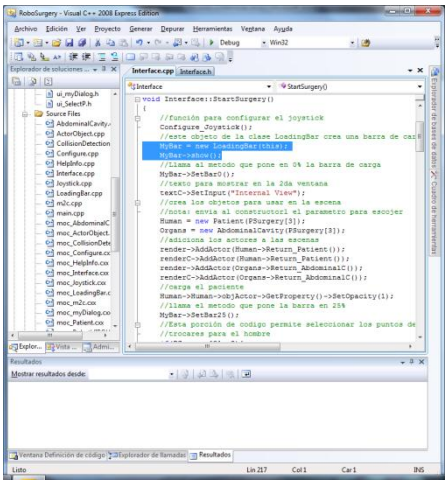

Figura B.18. Llamando interfaz con puntero.

Los que no llevan puntero, son por ejemplo ventanas de mensaje, que deben 'detener' la ejecución del programa, y dejar solo una activa.

El código va así:

```
void on_actionInfo_triggered(){

Dialog.setModal(true);

Dialog.exec();

}
```

y finalmente se ve asi:

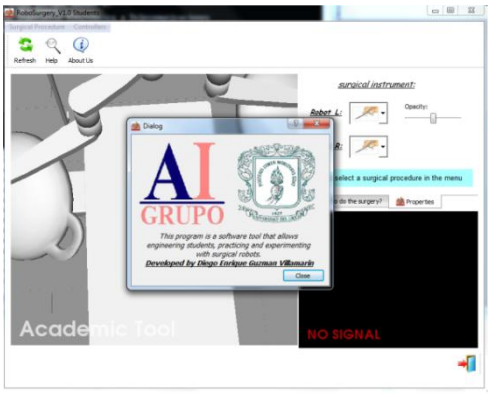

Figura B.19. Vista de una interfaz adicional.