Ambiente quirúrgico virtual para operaciones de cirugía endonasal

Universidad
del Cauca

Luisa Fernanda Mejía Palomino

Lina María Ramírez Palomino

Director: Mg. Hermes Fabián Vargas Rosero

Co-Director: PhD. Oscar Andrés Vivas Albán

Universidad del Cauca

Facultad de Ingeniería Electrónica y Telecomunicaciones

Facultad de Ciencias Naturales, Exactas y de la Educación

Ingeniería en Automática Industrial e Ingeniería Física

Popayán, marzo de 2019

Ambiente quirúrgico virtual para operaciones de cirugía endonasal

Luisa Fernanda Mejía Palomino

Lina María Ramírez Palomino

Trabajo de grado presentado a la Facultad de Ingeniería Electrónica y Telecomunicaciones y a la Facultad de Ciencias Naturales, Exactas y de la Educación para la obtención del título de Ingeniero en Automática Industrial e Ingeniero Físico.

Director: Mg. Hermes Fabián Vargas Rosero

Co-Director: PhD. Oscar Andrés Vivas Albán

Universidad del Cauca

Facultad de Ingeniería Electrónica y Telecomunicaciones

Facultad de Ciencias Naturales, Exactas y de la Educación

Ingeniería en Automática Industrial e Ingeniería Física

Popayán, marzo de 2019

Nota de aceptación: _____

_____

_____

_____

_____

_____

Director: _____

Mg. Hermes Fabián Vargas Rosero

Co-Director: _____

PhD. Oscar Andrés Vivas Albán

_____

Firma del Jurado

_____

Firma del Jurado

Popayán, marzo de 2019

Agradecimientos

Primero que todo, queremos dar gracias a Dios, sin Él nada de esto hubiera sido posible, gracias por iluminarnos en este camino, por permitirnos vivir oportunidades muy grandes, y por todas las personas que ha puesto en nuestro camino y nos permiten crecer profesionalmente y como personas.

A nuestros padres, por ser nuestra inspiración, por darnos su apoyo incondicional en cada paso de nuestra vida, y por motivarnos siempre a cumplir nuestros sueños.

A nuestros amigos y compañeros, gracias por cada una de las experiencias vividas, lo más bonito de esta etapa universitaria fue compartir tantos momentos valiosos con ustedes.

A nuestra familia, gracias por tantos consejos y apoyo en cada momento de la carrera.

A nuestro director Fabián Vargas por todos los conocimientos y experiencia brindados en el proceso de realización de este proyecto.

A nuestro Co-Director Oscar Andrés Vivas por su gran contribución a este proyecto, por darnos la confianza y la oportunidad de realizar este trabajo.

Al profesor Jairo Alfonso Vásquez por brindarnos su conocimiento y apoyo en relación a los temas médicos.

Un especial agradecimiento a la Universidad de Málaga y al laboratorio IA^{3}R por permitirnos hacer uso de sus instalaciones y recursos.

A Víctor Muñoz y Enrique Bauzano, por la grata acogida en la Universidad de Málaga, por acompañarnos, supervisarnos y compartir su tiempo y conocimiento con nosotras.

A la Universidad del Cauca, especialmente a nuestros profesores, por todo el apoyo y conocimientos brindados a lo largo de toda la carrera.

Resumen

El objetivo de este trabajo es la realización de una propuesta de un ambiente quirúrgico virtual para operaciones de cirugía endonasal basada en robots. Para ello, se desarrolló una herramienta virtual haciendo uso del software Unity 3D que cuenta con los componentes básicos de una cirugía real. Esta herramienta permite practicar y experimentar con los robots quirúrgicos, que en este caso son 2 Universal Robots (UR5), obteniendo las respuestas de comportamiento de estos y sin poner en riesgo a ningún cuerpo humano.

La manipulación de cada uno de los robots se realizó de manera diferente, para el primer robot se utilizó el sistema operativo para robots, ROS, el cual da facilidad para el manejo de este, la visualización se da mediante el visualizador gráfico Rviz y la manipulación haciendo uso del nodo de ROS *"joint_state_publisher"*. Para el segundo robot, el cual es semiautónomo, se hizo una codificación en Unity 3D en lenguaje C#.

La comunicación de la simulación realizada en Unity y la planificación realizada en ROS se hace a través de un paquete llamado Ros Bridge Server, que permite el flujo de datos mediante una conexión Ethernet.

El ambiente quirúrgico virtual fue evaluado con diferentes trayectorias neurológicas, en donde se evidencia que no existe error entre el movimiento realizado en ROS y el que se obtiene en el ambiente de Unity.

Palabras clave: Cirugía endonasal, Unity 3D, ROS, UR5.

Abstract

The objective of this work is the realization of a proposal of a virtual surgical environment for operations of endonasal surgery based on robots. For that reason, a virtual tool was developed using the Unity 3D software that has the basic components of a real surgery. This tool allows to practice and experiment with surgical robots, which in this case are 2 Universal Robots (UR5), obtaining the behavioral responses of these and not putting any human body at risk.

The manipulation of each one of the robots was done differently, for the first robot it was use the operating system for robots, ROS, which facilitates the handling of this, the visualization is given by the graphic display Rviz and the manipulation making use of the ROS node *"joint_state_publishe*r". For the second robot, which is autonomous, an encoding was made in Unity 3D in C # language.

The communication of the simulation carried out in Unity and the planning carried out in ROS is done through a package called Ros Bridge Server, which allows the flow of data through an Ethernet connection.

The virtual surgical environment was evaluated with different neurological trajectories, where it is evident that there is no error between the movement performed in ROS and the one obtained in the Unity environment.

Keywords: Endonasal surgery, Unity 3D, ROS, UR5.

TABLA DE CONTENIDO

Lista de figuras

Lista de tablas

1. INTRODUCCIÓN

Un ambiente virtual es aquél que “traslada” objetos del mundo real a un espacio creado artificialmente. Los elementos que se pueden representar en este tipo de mundos son muy variados, pueden ser personas, animales, arboles, coches, hasta fenómenos naturales. Una característica importante de los ambientes virtuales es que tienen una interactividad en tiempo real. En este caso el tiempo real significa que la computadora es capaz de procesar los datos del usuario y transformar el mundo virtual, garantizando la interacción de manera fluida [1].

La simulación quirúrgica se define como una técnica y no una tecnología, para sustituir o ampliar las experiencias reales a través de experiencias guiadas, que evocan o replican aspectos sustanciales del mundo real de una forma totalmente interactiva [2], el ambiente quirúrgico ideal debe ser predecible, consistente, estandarizado, seguro y reproducible [3].

Al utilizar simuladores quirúrgicos se crean condiciones en las cuales cometer errores no es perjudicial o peligroso para los pacientes, lo cual proporciona la oportunidad de practicar y recibir realimentación constructiva que permitirá evitar que se cometan errores al intervenir a pacientes reales [4].

Un buen simulador quirúrgico debe contar con ciertas características como son: la velocidad de trabajo, la precisión, la facilidad de uso, este debe mostrar los órganos internos de manera realista, que estos órganos respondan a las interacciones con el cirujano, por ejemplo, deformándose y que respondan mediante modificaciones estructurales a acciones quirúrgicas habituales como cortes, cauterización o sutura [5].

El problema actual es que conforme los simuladores son más complejos la imagen y la interrelación con el usuario es menos realista y no traducen el comportamiento real de los tejidos a nuestra actuación ya que estos simuladores requieren ordenadores y programas informáticos más grandes y complejos [6].

La cirugía endonasal representa un gran desafío para los neurocirujanos, quienes requieren tener mucha experiencia para implementar este tipo de técnicas. La curva de aprendizaje de esta es algo compleja debido a la dificultad que la misma implica, por lo que el desarrollo de una herramienta basada en simuladores quirúrgicos para cirugía endonasal, que cuente con las características mencionadas anteriormente, constituye un gran aporte para el campo de la medicina.

Lo anterior representa un reto muy grande para los científicos e ingenieros, porque se tiene la necesidad de diseñar y desarrollar nuevos entornos e interfaces en los que se incluyan todos los componentes de una cirugía real robotizada, también,

modelos anatómicos que sean lo más parecidos a los reales y que incluyan algoritmos que permitan realizar la simulación del comportamiento de los robots.

La herramienta ROS es un sistema operativo de código abierto, la cual está basada en una arquitectura distribuida donde el procesamiento toma lugar en los nodos que pueden recibir, mandar y multiplexar mensajes de sensores, control, estados, planificaciones, actuadores, entre otros, brindando los servicios de un sistema operativo [7], [8]. Adicionalmente se agrega el potencial del motor gráfico Unity 3D que permite crear una amplia gama de aplicaciones en conjunto con ROS.

Con el fin de aportar a la línea de investigación de la robótica médica se hace importante la realización de este proyecto, para así contribuir a los retos antes mencionados.

1.1 Estructura de la monografía

Este documento se divide en 6 capítulos cuyo contenido trata de las temáticas más relevantes para la realización del proyecto. El capítulo 1 (ya abordado) presenta una introducción para contextualizar al lector. En el capítulo 2 se presenta el estado del arte, seguido por el capítulo 3 que contiene el diseño del ambiente quirúrgico virtual en la herramienta Unity 3D, el capítulo 4 muestra la inclusión del módulo de comunicación con ROS al proyecto, la evaluación y resultados se muestra en el capítulo 5, seguido por el capítulo 6 que indica conclusiones y trabajos futuros, respectivamente.

Este proyecto contó con el apoyo del PhD. Víctor Muñoz y PhD. Enrique Bauzano del Instituto Andaluz de Automática Avanzada y Robótica de la Universidad de Málaga, España, en donde se realizó una pasantía de 5 meses (Febrero – Julio 2018).

2. MARCO TEÓRICO Y ESTADO DEL ARTE

Este capítulo busca dar una introducción conceptual a la temática de este proyecto, haciendo una revisión bibliográfica de conceptos relacionados con la cirugía endonasal, el software y el robot usado.

2.1 MARCO TEÓRICO

2.1.1 Adenoma Hipofisario (AH)

La hipófisis es una glándula de secreción interna ubicada en la base del cráneo por encima de los conductos nasales, encargada de controlar ciertas actividades hormonales tanto de ella como de otras glándulas del cuerpo. Un adenoma hipofisario (AH) consiste en una neoplastia (formación anormal de tejidos celulares) benigna que puede desarrollar un comportamiento localmente agresivo sin extenderse más allá del cráneo [9]. Aún se desconoce la causa de esta patología, pero en algunos casos se relaciona con una enfermedad genética llamada Síndrome de Neoplasia Endocrina Múltiple [10].

Las alternativas de abordaje para dicha patología incluyen fármacos apropiados para bloquear la producción excesiva de hormonas, la radio terapia con el fin de disminuir el tamaño del tumor y la cirugía endoscópica endonasal transesfenoidal para remover el tumor.

En la Figura 1 se observa un adenoma hipofisario por medio de una resonancia magnética.

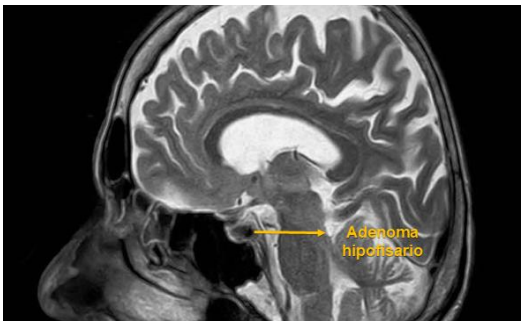


Figura 1. Resonancia magnética con visualización del adenoma hipofisario [14].

2.1.2 Cirugía endoscópica endonasal transesfenoidal

Es una técnica mínimamente invasiva que utiliza el endoscopio como fuente de visión sin la necesidad de un retractor transesfenoidal. Ha generado gran comodidad en los cirujanos de hipófisis a pesar de que se debe tener un amplio conocimiento anatómico y estar habituado a la utilización de endoscopios, los cuales permiten un mejor control de la visión interna en la intervención [11] [12].

2.1.2.1 Técnica quirúrgica general

El instrumental utilizado para la intervención incluye: Endoscopio nasal Storz de 30° y de 0° de 4 mm, pinza de coagulación bipolar, pinzas de Kerrison, fresa de alta velocidad, curetas, disectores quirúrgicos y cuchillas neuroquirúrgicas como base de la operación, como se observa en la Figura 2, pero también existe otro instrumental secundario para complicaciones en la cirugía [13].

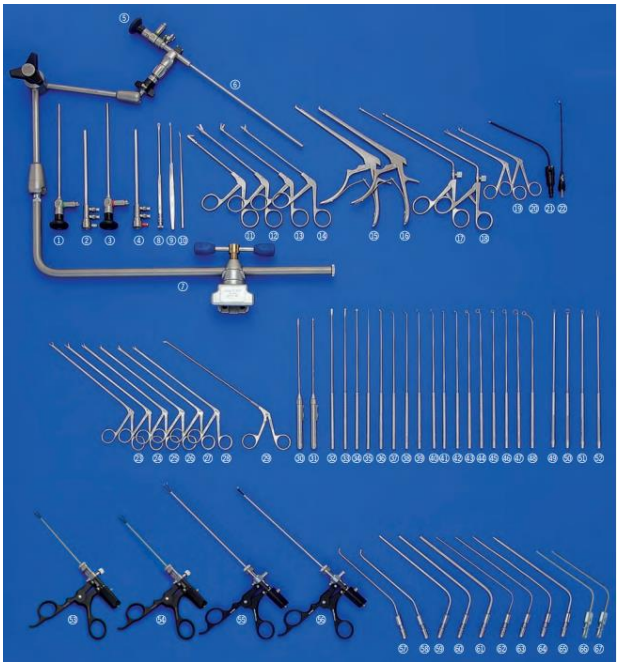


Figura 2. Instrumental quirúrgico de la cirugía hipofisaria [14].

2.1.2.2 Pasos de la cirugía endoscópica endonasal transesfenoidal

La intervención quirúrgica consiste en tres fases, la nasal, la esfenoidal y la selar, dividiéndose en una serie de pasos para el logro del objetivo el cual es la extracción del tumor [11].

- Paso 1: Aplicación de epinefrina y anestesia local con el fin de evitar sangrado en la mucosa local.

- Paso 2: El primer obstáculo encontrado al ingresar a la fosa nasal será el cornete inferior, por consiguiente, se procede a luxarlo con el fin de habilitar la vía nasal.

- Paso 3: Se luxa el cornete medio por encima del cornete inferior para ampliar la vía de la intervención quirúrgica.

- Paso 4: Se identifica el hueso esfenoidal y se retira la mucosa, después se procede a luxar el hueso con el fin de llegar a la hipófisis y así proceder al último paso.

- Paso 5: Identificación y extracción del tumor.

2.1.2.3 Configuración de la sala de operación para la cirugía

Un quirófano bien organizado, ayuda a la optimización del trabajo grupal desarrollado por los integrantes del equipo médico y también a una mejor atención al paciente.

La fuente de luz, el monitor de las cámaras de video y el sistema de grabación del video se encuentran ubicados ergonómicamente detrás de la cabeza del paciente y de frente al primer cirujano, quien se encuentra al lado derecho del paciente. El anestesiólogo se coloca con su equipo en el lado izquierdo del paciente al nivel de la cabeza. El segundo cirujano está en el lado izquierdo del paciente, y la enfermera se coloca al nivel de las piernas del paciente. El instrumental quirúrgico de la figura 2 se localiza en la mesa de instrumental o en la mesa de mayo (mesa quirúrgica). Ver Figura 3.

También existen otros equipos que se encuentran en una sala de quirófano, como lo son las vitrinas, los porta sueros, el negatoscopio,

oxigeno, bancos y tarimas, los cuales son organizados dependiendo de su necesidad [14].

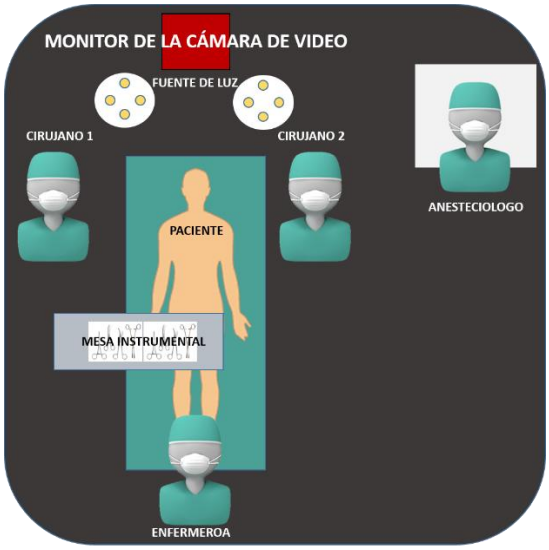


Figura 3. Configuración del quirófano.
Fuente: Propia

2.2 ESTADO DEL ARTE

2.2.1 Unity 3D

En la actualidad se han recreado entornos virtuales de simulación quirúrgica los cuales han sido simulados y ejecutados en un motor de videojuegos llamado Unity 3D, el cual ha generado un gran impacto en los últimos años debido a su facilidad de uso en el desarrollo de videojuegos, pero no solamente posee esta aplicación sino que también ha permitido que gran variedad de compañías implementen sus ideas en este entorno con el fin de recrear de forma interactiva sus conocimientos en las distintas áreas [15].

Para el área de la medicina, se han implementado distintos entornos que aportan a la educación del personal médico como programas de entrenamiento y simuladores especializados en diferentes patologías, los cuales están basados en motores gráficos de juegos donde se les puede incorporar modelos de deformación que permiten hacer los entornos más próximos a la realidad, cuyo objetivo es contribuir al aprendizaje y al análisis del ambiente en el que se trabaja [16] [17], entre los distintos aportes que

se han ejecutado, se encuentran prototipos de simulación virtual para cirugía laparoscópica [18] un entrenador virtual diseñado para enseñar cómo atender a un paciente en shock por paro cardiaco [15], Clinispace Dialysis diseñado para la práctica de los procesos de la diálisis [15], *Gyn Trainer* plataforma virtual para aprendizaje de técnicas de ginecología, entre otros [19] incluyendo motores de juegos que incorporan técnicas neuroquirúrgicas [17].

Un simulador de neurocirugía en tiempo real maneja la perforación del cráneo y la interacción quirúrgica con el cerebro. Esto implica el desarrollo y la combinación de áreas tales como la simulación física y la visualización volumétrica. Los datos de entrada del simulador provienen de imágenes de tomografía computarizada y de resonancia magnética de los pacientes [20] [21].

Unity 3D también aporta la posibilidad de usar realidad aumentada con el fin de navegar en un modelo tridimensional sin necesidad de acceder físicamente al paciente [22].

2.2.2 ROS: Aplicaciones robóticas

Entre los principales trabajos realizados en esta herramienta, se tiene la integración de dispositivos en un robot quirúrgico tele-operado mediante ROS. Dicho robot consiste en tres brazos manipuladores, cada uno de ellos instalado sobre estructuras móviles con ruedas denominadas unidades robotizadas, el cual puede fijarse mediante unas extremidades manipuladas en cualquier posición alrededor del paciente. Cada una de estas estructuras tendrá una misión concreta y actuará como cámara, pinza izquierda o pinza derecha. La consola consta de dos dispositivos denominados *haptics* y dos pedales además de un monitor 3D [23].

El proyecto de [24], sistema de supervisión no invasivo de signos vitales con un robot, realiza el diseño e integración en ROS de un sistema para monitorización de signos vitales mediante la técnica fotopletismográfica (técnica óptica ampliamente utilizada en clínica para la monitorización periférica de la frecuencia cardiaca). Esta es capaz de medir la frecuencia cardíaca y la frecuencia respiratoria. La información obtenida de cada signo vital es muestreada mediante una cámara digital y procesada en línea, por diferentes nodos, respetando la plataforma multimodal de ROS.

El proyecto BROCA consiste en el diseño e integración de robots miniaturizados en un sistema modular e inteligente para cirugías laparoscópicas de una sola incisión. Este incorpora tres brazos robóticos UR5 además de una pantalla de visión 3D que posibilita al cirujano operar

sentado, ampliando su campo de visión y permitiendo controlar la intervención a través de unos mandos que emulan la instrumentación de la laparoscopia [25] [26]. Este proyecto se realiza en ROS y contiene *scripts* escritos en lenguaje C para manipular el brazo robótico [27].

El proyecto CRANEEAL, es un proyecto que relaciona distintas universidades de España: Universidad de Málaga, Universidad Miguel Hernández de Elche, y Universidad de Valladolid, donde se dispone de equipos de personas con diferentes enfoques y especialidades que realizan aportes para la complementación del proyecto, basada en las experiencias adquiridas en trabajos previos. En concreto, el equipo de investigadores se ha tomado el trabajo de realizar la integración de diferentes equipos robóticos utilizando ROS (*Robot Operating System*), donde se integren las diferentes piezas de software generadas, como son el simulador dinámico (programado en C++ / Ogre / Physx e integrado en ROS) como el módulo cognitivo (bajo arquitectura SOAR y ROS) [28].

2.2.3 Universal Robot 5 (UR5)

El UR5 de Universal Robots presentado en la Figura 4 es un robot ligero, que puede programarse para mover distintas herramientas, y puede comunicarse con otro tipo de máquinas por medio de señales eléctricas [29]. Está constituido por juntas y tubos de aluminio las cuales pueden manipular cargas útiles de 5kg (11 lb), un alcance de hasta 850 mm y sus grados de libertad son 6 articulaciones giratorias, flexibles y ligeras [30].

Posee una interfaz gráfica llamada PolyScope la cual genera una facilidad en la programación del robot y así logra los movimientos de las herramientas en las trayectorias deseadas [29] [30]. Ver Figura 5.

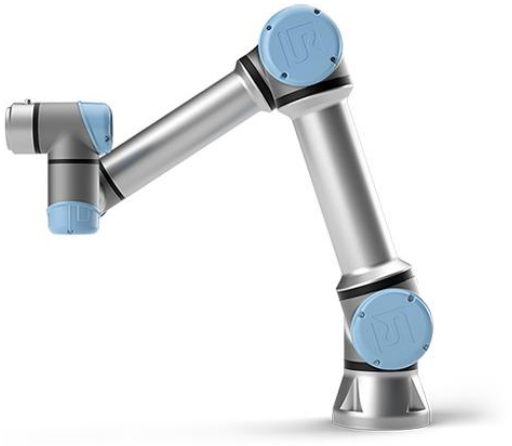


Figura 4. UR5 Universal Robot [30].

Figura 5. Caja de control y consola de programación [30].

3. DISEÑO DE UN AMBIENTE QUIRÚRGICO VIRTUAL UTILIZANDO LA HERRAMIENTA UNITY

En este capítulo se realiza una descripción del software Unity 3D, y de los diferentes programas utilizados para el diseño de todos los modelos que hacen parte de la simulación.

3.1 Unity 3D

Unity es un motor de videojuegos creado por Unity Technologies el cual permite el diseño, la creación y la representación de múltiples ambientes, tanto en 2D como en 3D para el desarrollo de juegos y simulaciones en distintas plataformas como: Xbox, PlayStation, Windows, Linux, OS y Android. Este motor permite una aproximación interactiva y visual a los proyectos desarrollados en él.

Unity puede usarse a la vez con motores gráficos como Blender, 3ds Max, Maya, SketchUp, y demás herramientas de desarrollo de modelos 3D, al igual que cuenta con diferentes lenguajes de programación.

Este cuenta con dos versiones, Unity Pro, el cual necesita una licencia profesional para la creación de proyectos, y Unity Personal que genera una licencia gratuita para usos educativos y básicos [31].

3.1.1 Herramientas software

3.1.1.1 SketchUp

SketchUp es un programa de diseño gráfico y modelado en tres dimensiones (3D) de fácil uso, donde se pueden realizar entornos arquitectónicos, civiles, cinematográficos y entornos de videojuegos. Permite la creación de modelos en 3D partiendo de formas y volúmenes de un espacio y también permite la adaptación visual de los modelos, como son los colores y las texturas [32].

3.1.1.2 Blender

Blender es un software de creación 3D de código abierto y gratuito, es utilizado para la creación de videojuegos, modelos 3d, películas animadas, arte, modelos para impresiones 3d, aplicaciones interactivas y efectos visuales, cuenta con un motor de juego integrado en las últimas versiones y también posee amplias características como: texturizado, escultura, animación, edición de gráficos, simulación de fluido y humo, edición de video, entre otras [33].

3.1.1.3 Makehuman

Makehuman es un software utilizado para la realización de prototipos humanos fotorealísticos en 3D. En este programa se pueden definir diferentes características para el humanoide que se desea realizar, como la edad, peso, sexo, raza, proporción de la cara, brazos y piernas entre una amplia serie de parámetros definidos con el fin de permitir la creación de humanos extremadamente realistas [34].

3.2 Diseño del ambiente

El desarrollo del ambiente quirúrgico en Unity 3D, se encuentra enfocado en la creación de un banco de modelos y animaciones 3D haciendo uso de las herramientas software mencionadas anteriormente, con las cuales se logre satisfacer el primer objetivo.

3.2.1 Diseño de la estructura del quirófano

Principalmente para el diseño de este ambiente se realizó el modelo 3D de la estructura de la sala de cirugía o quirófano. Debido a que SketchUp cuenta con una facilidad de uso para desarrollar modelos arquitectónicos se optó por realizar el modelo en este software.

Ya creado el modelo en SketchUp se le agregaron los colores y texturas correctas para darle una mejor vista digital al quirófano, así como se puede observar en la Figura 6.

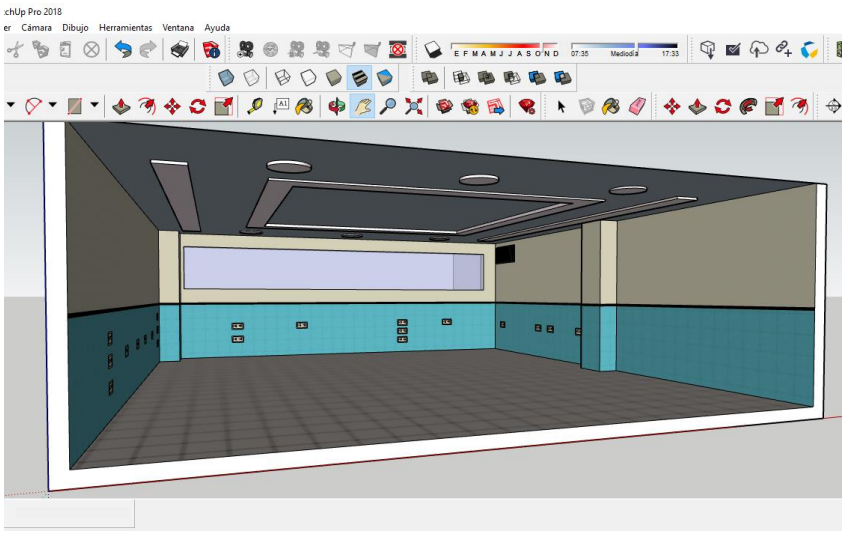


Figura 6. Modelo del quirófano en SketchUp.
Fuente: Propia

Posteriormente se exporta el modelo anterior a un formato FBX (formato de archivos de contenido digital), con el fin de importar el quirófano al motor de Unity 3D para realizarle los respectivos cambios que llevarán del modelo de SketchUp básico, a un modelo más real en Unity (ver Figura 7).

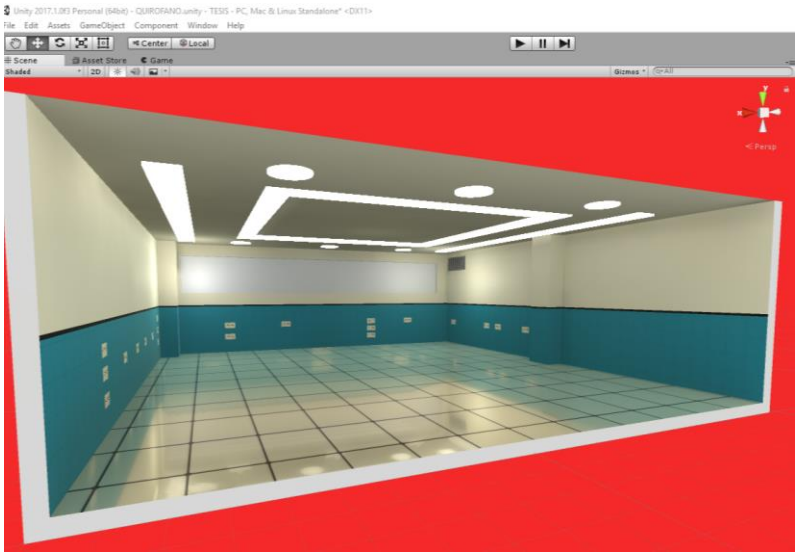


Figura 7. Modelo del quirófano en Unity.
Fuente: Propia

3.2.2 Diseño de los equipos del quirófano

Para todos los modelos de los equipos se realiza el mismo procedimiento de exportación e importación que se hizo para la sala del quirófano. En las imágenes que se presentarán a continuación se encuentran los equipos utilizados en la configuración de la sala de cirugía. Estos modelos se encuentran exportados en formato FBX (imágenes de la izquierda) e importados en Unity (imágenes de la derecha) con su respectivo cambio.

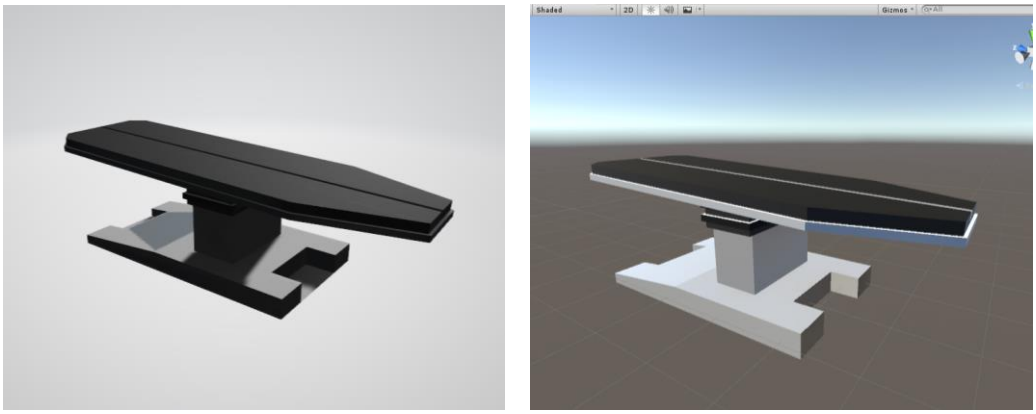


Figura 8. Mesa de operación. Modelo FBX y modelo en Unity.
Fuente: Propia

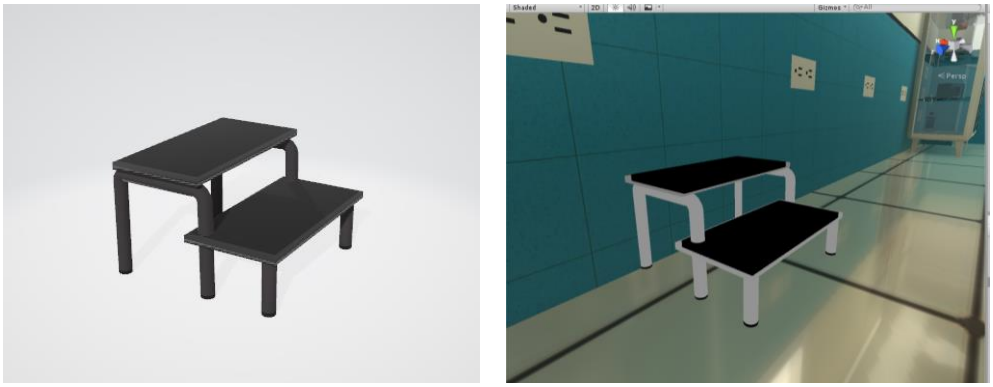


Figura 9. Banca. Modelo FBX y modelo en Unity.
Fuente: Propia

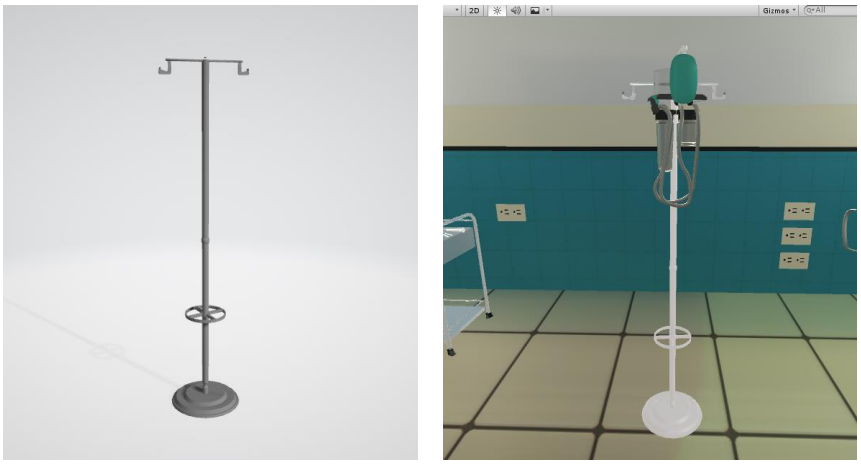


Figura 10. Porta sueros. Modelo FBX y modelo en Unity.
Fuente: Propia

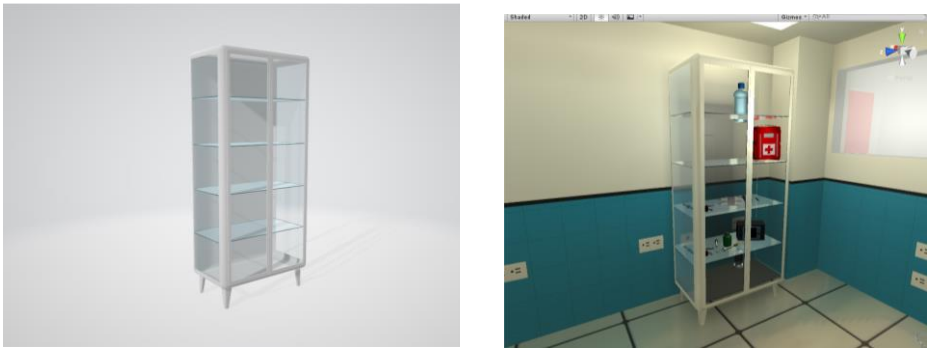


Figura 11. Vitrina. Modelo FBX y modelo en Unity.
Fuente: Propia

Figura 12. Mesa instrumental. Modelo FBX y modelo en Unity.
Fuente: Propia

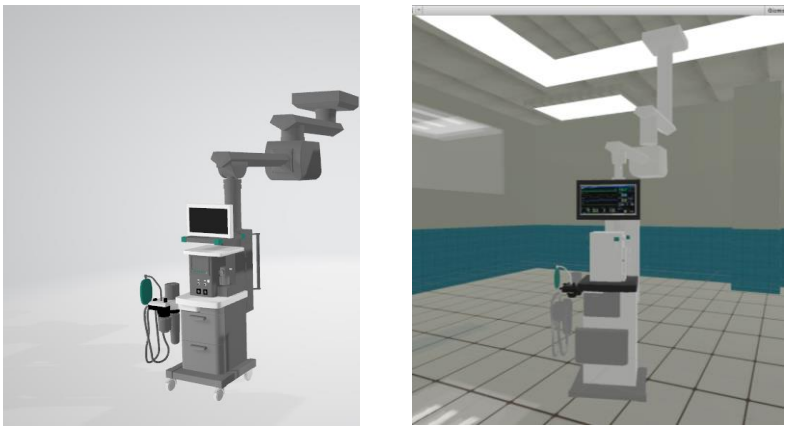


Figura 13. Mesa de anestesia. Modelo FBX y modelo en Unity.
Fuente: Propia

A cada objeto se le agregó en Unity la iluminación correcta con el objetivo de hacer más real el ambiente.

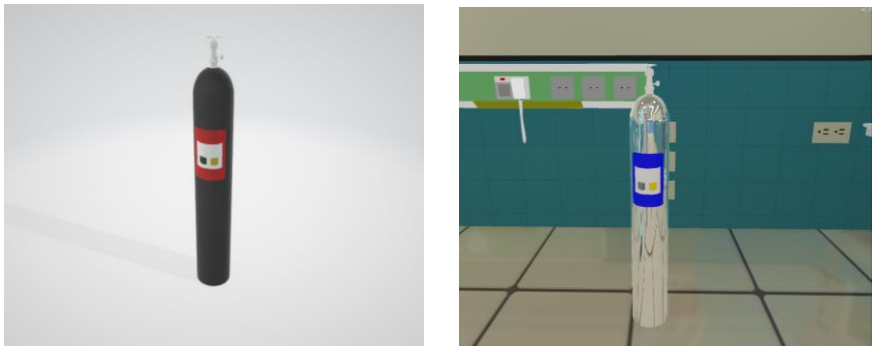


Figura 14. Oxigeno. Modelo FBX y modelo en Unity.
Fuente: Propia

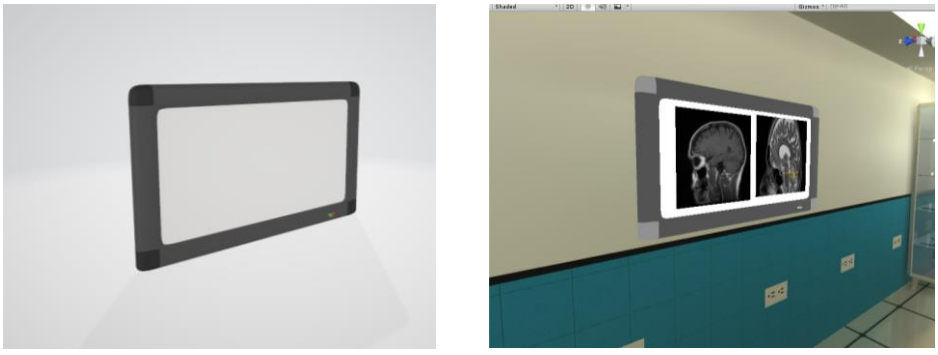


Figura 15. Negatoscopio. Modelo FBX y modelo en Unity.
Fuente: Propia

Figura 16. Mesa de mayo. Modelo FBX y modelo en Unity.
Fuente: Propia

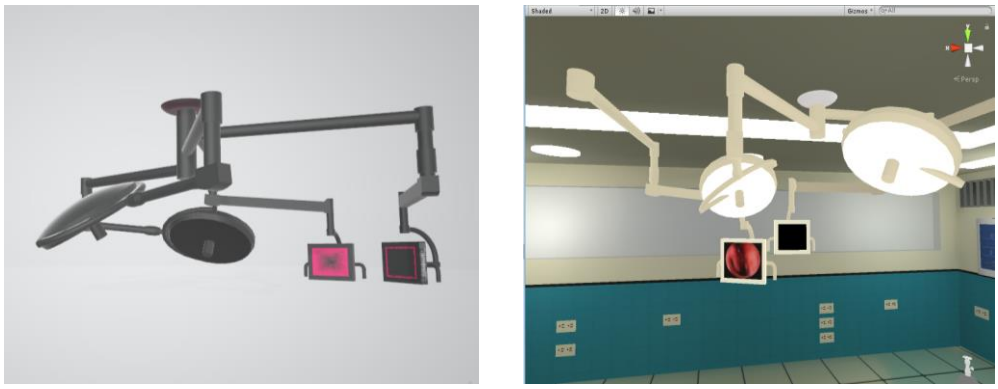


Figura 17. Fuente de luz y Monitores. Modelo FBX y modelo en Unity.
Fuente: Propia

3.2.3 Diseño del instrumental quirúrgico

El instrumental necesario para la intervención quirúrgica de la figura 2 fue representado en modelos tridimensionales realizados en Blender. El diseño de estos modelos fue basado en el instrumental quirúrgico de [14], a continuación, se presentará la imagen junto con su diseño 3D.

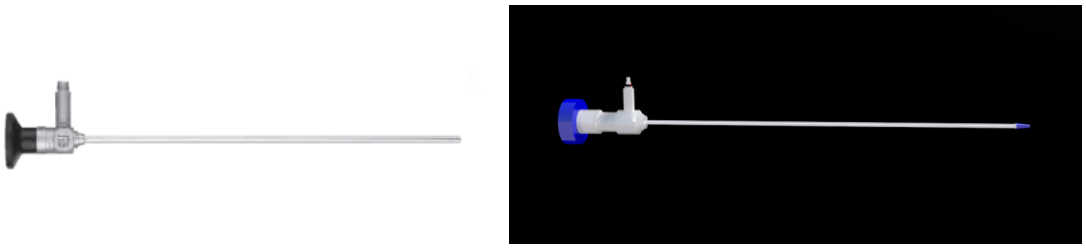


Figura 18. Endoscopio rígido [14] y su modelo 3D.

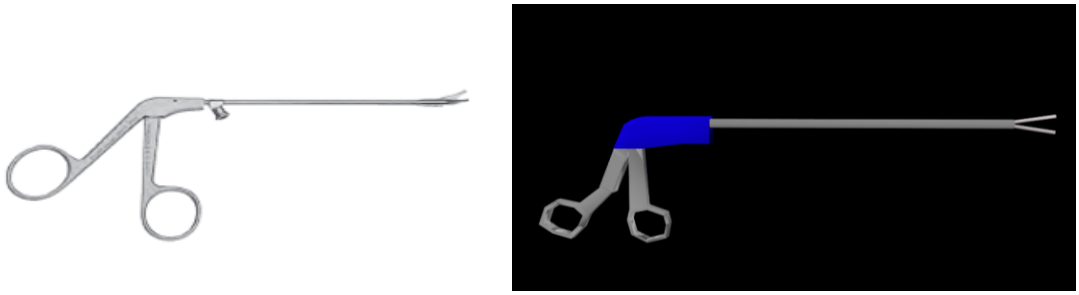


Figura 19. Pinza de coagulación bipolar [14] y su modelo 3D.

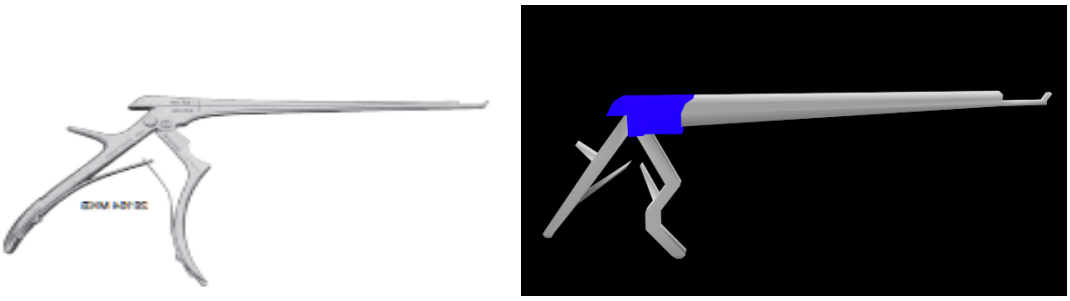


Figura 20. Pinzas de Kerrison [14] y su modelo 3D.

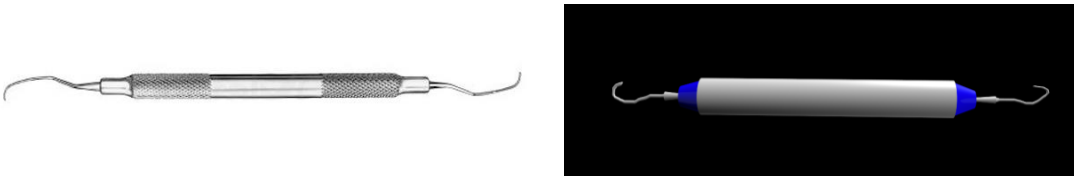


Figura 21. Cureta [14] y su modelo 3D.

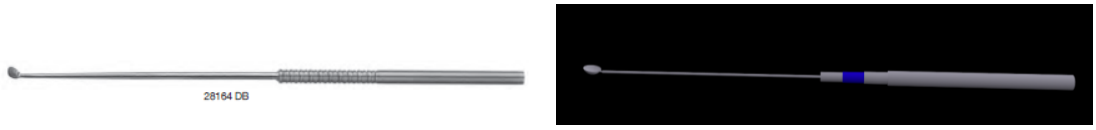


Figura 22. Disector quirúrgico [14] y su modelo 3D.

3.2.4 Diseño del personal médico

Para la creación del personal médico en el quirófano, se usó el software Makehuman, donde se realiza el prototipo del humano agregándole la ropa y los accesorios adecuados como se ve en la Figura 23, para que así al importarlos en Unity se le puedan hacer modificaciones con el objetivo de que se vea más real, como se observa en la Figura 24.

Figura 23. Modelo del humanoide en Makehuman.
Fuente: Propia

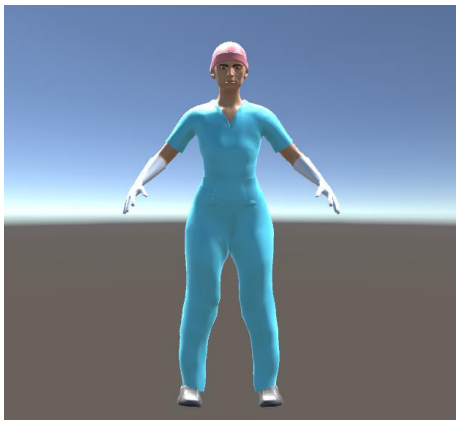


Figura 24. Modelo del humanoide en Unity.
Fuente: Propia

3.2.5 Diseño de la zona de operación

Para el modelamiento de la zona de operación se descargaron dos modelos importantes de *"3d warehouse"* [35], una página de modelos creados en SketchUp. Los modelos fueron el cerebro (Figura 25) y la cabeza (Figura 26). Se exportaron a un archivo FBX y se le realizaron los respectivos cambios al importarlo en Unity.

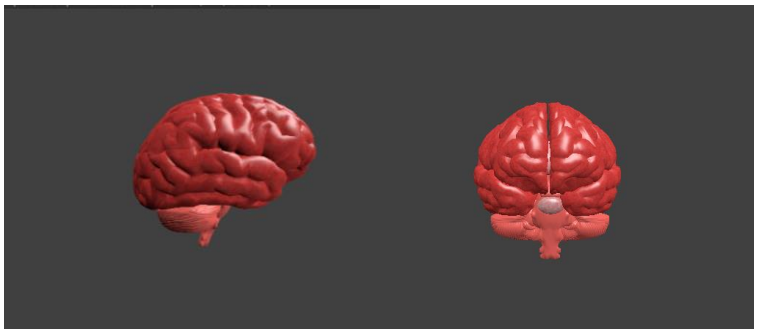


Figura 25. Modelo del cerebro en vista frontal y lateral.
Fuente: Propia.

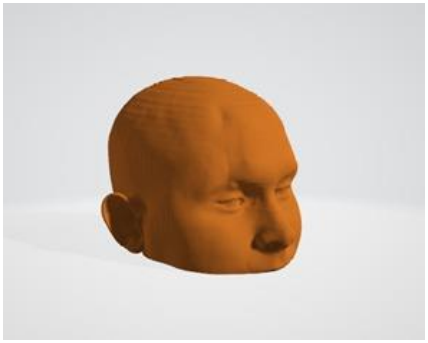


Figura 26. Modelo de la cabeza del paciente.
Fuente: Propia.

El adenoma hipofisario, fue realizado directamente desde Unity con tres esferas, y se les agregó una textura y colores adecuados como se observa en la Figura 27.

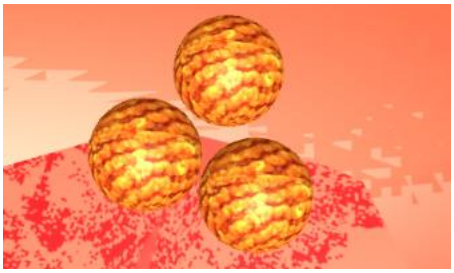


Figura 27. Modelo del adenoma hipofisario (AH).
Fuente: Propia

Ya creado el tumor, se importa el modelo del cerebro a Unity con el propósito de ubicar el adenoma dentro de la glándula hipofisaria como se puede ver en la Figura 28, con la ubicación correspondiente se procede importar el modelo de la cabeza con el fin de hacerle los cambios necesarios para una mejor visualización.

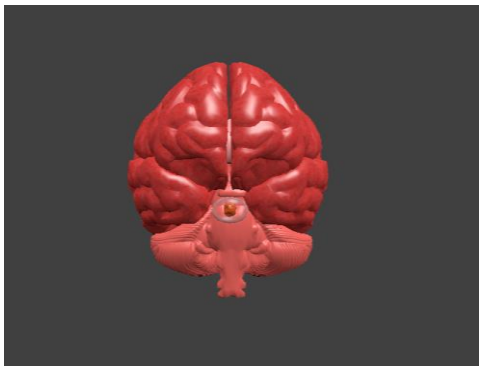


Figura 28. Modelo del adenoma hipofisario (AH) ubicado en el cerebro.
Fuente: Propia**.**

Posteriormente a la importación de los modelos en Unity se realiza la ubicación correspondiente del cerebro dentro de la cabeza.

3.3 Configuración del ambiente quirúrgico en Unity

La configuración del ambiente quirúrgico está basada en la Figura 3 con el fin de cumplir el protocolo de organización de la intervención y hacer de esta una aproximación más real (ver Figura 29).

Al personal médico se le agregaron ciertas animaciones que permite realizar Unity, para que así sus movimientos fueran adecuados al comportamiento real de un humano que se encuentra en una sala de cirugía. También se detalló con cuidado que los objetos encontrados en la sala no fueran a interrumpir los movimientos de los médicos produciendo choques.

El instrumental quirúrgico utilizado en la intervención fue ubicado sobre la mesa de mayo, al lado derecho de la cirujana como se observa en la Figura 30.

Figura 29. Configuración del ambiente quirúrgico en Unity.
Fuente: Propia.

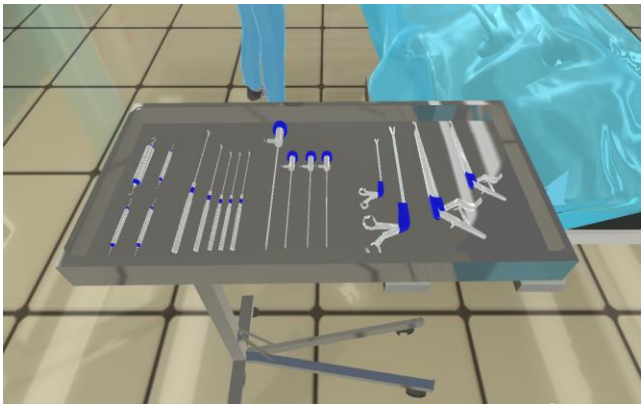


Figura 30. Instrumental quirúrgico en Unity.
Fuente: Propia.

4. INCLUSIÓN DEL MÓDULO DE COMUNICACIÓN CON ROS AL AMBIENTE DISEÑADO EN UNITY

En este capítulo se realiza una descripción de la estructura y funcionamiento de ROS, ya que es esencial conocer sus elementos principales y su dinámica.

Es necesario describir el espacio de trabajo (*catkin*), y el método de programación implementado para controlar el robot UR5.

El entorno ROS es un conjunto de librerías y herramientas que permiten crear aplicaciones para robots cuyo objetivo es simplificar la tarea de crear el comportamiento complejo de un robot. ROS provee los servicios estándares de un sistema operativo tales como abstracción de hardware, control de dispositivos de bajo nivel, implementación de funcionalidades usadas habitualmente, paso de mensajes entre procesos y mantenimiento de paquetes. También ofrece herramientas y librerías para escribir y ejecutar código mediante múltiples computadoras con el fin de fomentar el desarrollo de software de robótica colaborativa [36] [37].

Está compuesto por una colección de herramientas, bibliotecas y convenciones que tienen como objetivo simplificar la tarea de crear un comportamiento complejo y robusto para un robot, en una amplia variedad de plataformas robóticas. Su estructura se basa en una arquitectura distribuida, donde el procesamiento toma lugar en nodos, los cuales pueden recibir, enviar y multiplexar mensajes de sensores, control, estados, planificaciones y actuadores, entre otros [38].

Gracias al conjunto de bibliotecas que presenta ROS y su código abierto, se han podido realizar suficientes trabajos en los que se crean aplicaciones para robots, desde los controladores hasta los algoritmos de última generación con potentes herramientas de desarrollo [37], desde construcciones de algoritmos y aplicaciones para robots abogados, hasta manos robóticas en contribución con la medicina con excelentes resultados en cuanto a facilidad de integración, con lo que se considera la opción ideal en un proyecto de gran magnitud [36].

4.1 Niveles de ROS

Los niveles básicos de ROS son 3, el nivel de sistemas de archivos, el nivel de computación gráfica, y el nivel comunitario.

4.1.1 Nivel de sistema de archivos

En el nivel de sistema de archivos se encuentran los recursos principales que están en el disco, tales como: paquetes, metapaquetes, manifiestos, repositorios, mensajes y servicios [39].

.

- Paquetes: Son la unidad principal para organizar el software de ROS. Un paquete puede contener procesos (nodos) de tiempo de ejecución de ROS, una biblioteca dependiente de ROS, conjuntos de datos y archivos de configuración.

- Metapaquetes: Son paquetes especializados que solo sirven para representar un grupo de otros paquetes relacionados.

- Manifiestos: Los manifiestos (package.xml) proporcionan datos sobre un paquete, incluidos su nombre, versión, descripción, información de licencia, dependencias y otra información como paquetes exportados.

- Repositorios: Son una colección de paquetes que comparten una versión en común.

- Mensajes: Los mensajes (msg) son estructuras de datos compuestos. Los tipos de mensajes definen las estructuras de los mensajes enviados en ROS.

- Servicios: Los servicios (srv) definen la estructura de los datos de solicitud y respuesta de ROS.

4.1.2 Nivel de computación gráfica

Los conceptos básicos de este nivel son nodos, maestro, servidor de parámetros, mensajes, servicios, tópicos y bolsas, los cuales proporcionan datos de diferentes maneras, como se observa en la Figura 31 [39].

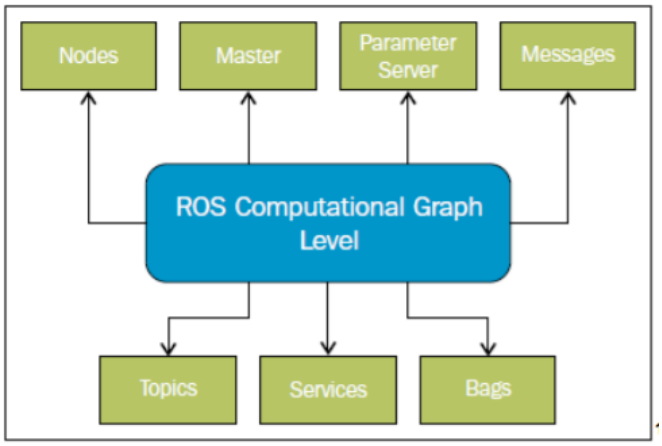


Figura 31. Nivel de computación gráfica [39].

- Nodos (*nodes*): Un nodo es un proceso que realiza algún tipo de computación en el sistema. Los nodos se combinan dentro de un grafo, compartiendo información entre ellos, para crear ejecuciones complejas. Un nodo puede controlar un sensor láser, otro los motores de un robot y otro la construcción de mapas.

- Maestro (*master*): El maestro es el nodo principal, el cual proporciona servicios de denominación y registro al resto de los nodos del sistema. Sin el maestro, los nodos no se pueden encontrar, intercambiar mensajes o invocar servicio.

  El maestro también proporciona el servidor de parámetros y se ejecuta comúnmente con el comando *roscore.*

- Servidor de parámetros: El servidor de parámetros permite almacenar los datos por clave en una ubicación central. Actualmente forma parte del maestro.

- Mensajes: Los nodos se comunican entre sí pasando mensajes. Un mensaje es simplemente una estructura de datos, que comprende campos escritos. Se admiten los tipos estándar (entero, flotante, booleano, etc.).

- Servicios: El modelo publicador/suscriptor es un paradigma de comunicación muy flexible, aunque el medio de transporte no es muy apropiado para las interacciones de pregunta respuesta, que a menudo se requieren en un sistema distribuido. La

pregunta/respuesta se realiza mediante servicios que son definidos mediante un par de estructuras de mensajes, uno para la pregunta y otro para la respuesta. Un nodo ofrece un servicio con un nombre y un cliente utiliza el servicio enviando un mensaje de solicitud, y espera la respuesta.

- Tópicos (*topics*): Los tópicos son un medio de comunicación pensado en la transmisión asíncrona de mensajes. Todos los datos que tengan que ser publicados constantemente y estar disponibles para otros nodos se publican a través de tópicos.

- Bolsas: Las bolsas son un formato para guardar y reproducir datos de mensajes ROS [40].

4.1.3 Nivel comunitario [39]

La comunidad de ROS está basada en distintos recursos que permiten a distintos grupos el intercambio de software y conocimientos. Entre estos recursos están: distribuciones, repositorios, ROS wiki y lista de email.

- Distribuciones: Son colecciones de metapaquetes que se pueden instalar, son utilizadas principalmente para mantener versiones estables del software.

- Repositorios: ROS se basa de una red de repositorios de código, donde diferentes instituciones pueden desarrollar y aportar su propio software.

- ROS wiki: Es el principal foro para documentación e información sobre ROS. Cualquier persona puede aportar nueva documentación, corregir errores o escribir tutoriales.

- Lista de email: Es el principal canal de comunicación sobre actualizaciones de ROS, así como un método de debate sobre temas relacionados con ROS.

4.2 *Catkin* (espacio de trabajo)

Catkin es el sistema de compilación oficial de ROS y el sucesor del sistema de compilación ROS original, *rosbuild*. *Catkin* combina macros de CMake (herramienta multiplataforma de generación o automatización de código) y scripts de Python para proporcionar alguna funcionalidad además del flujo de trabajo normal de CMake. Permite una mejor distribución de paquetes, mejor soporte de compilación cruzada y mejor portabilidad.

El principal objetivo es la creación de un espacio de trabajo que permita la ejecución de código ROS de manera sencilla mediante el uso de herramientas y convenciones que simplifiquen el proceso [41].

4.3 Ros Industrial

ROS-Industrial es un proyecto de código abierto que extiende las capacidades avanzadas de ROS a la automatización industrial y a la robótica.

El repositorio ROS-Industrial incluye interfaces para manipuladores industriales comunes, pinzas, sensores y redes de dispositivos. También proporciona bibliotecas de software para la calibración automática de sensores 2D / 3D, la ruta del proceso y/o la planificación del movimiento, aplicaciones como Scan-N-Plan, herramientas para desarrolladores como el complemento Qt Creator ROS y un currículo de capacitación que es específico para las necesidades de los fabricantes. Ros Industrial cuenta con el apoyo de un consorcio internacional de miembros de la industria e investigación [42].

La arquitectura de ROS-Industrial se muestra a continuación en la Figura 32:

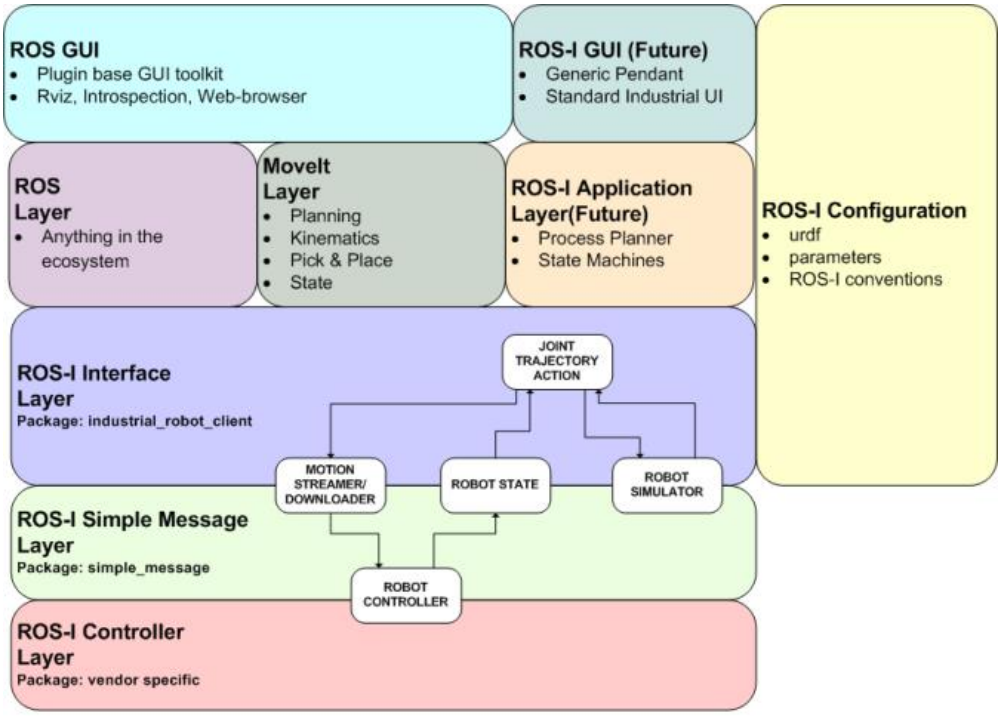


Figura 32. Arquitectura de Ros Industrial [42].

4.4 RosBridge

RosBridge es un componente que permite comunicar programas externos a ROS, como Unity, con nodos de ROS de forma directa. Los mensajes estándar de ROS son convertidos en mensajes JSON que pueden ser enviados a través de una interfaz web. RosBridge establece un protocolo de comunicaciones que permite suscribirse o publicar en un tópico de ROS desde los nodos creados en programas externos. Además, incluye una capa de comunicación regida por WebSockets que permite que el agente externo acceda al entorno de ROS desde cualquier localización [43].

El proceso de comunicación ROS-Unity consiste en una transmisión de mensajes codificados como cadenas “yaml” (*YAML Ain’t Markup Lenguage*) [44] a través de WebSockets. Estas cadenas yaml podrían ser tratadas de forma directa por los agentes externos, pero resulta más cómodo utilizar nodos JSON [45] para mapear dichas cadenas a objetos instanciados en el entorno de Unity [43].

La arquitectura de este proyecto se muestra a continuación en la Figura 33:

Figura 33. Arquitectura del proyecto.
Fuente: Modificado de [43]

4.5 Herramientas software

Como se observa en la arquitectura del proyecto, en ROS se despliega el modelo del robot que se importa mediante Rosbridge a Unity. Es por esto que para el desarrollo de la aplicación se requieren dos computadores con el fin de establecer la arquitectura del proyecto mencionada anteriormente, uno con sistema operativo Ubuntu 16.04.3, donde se ejecuta ROS y Rviz (visualizador 3D), y otro con Windows 7, donde se ejecutará Unity.

4.5.1 ROS.

Es el sistema operativo que permite la interacción con uno de los robots UR5, de diferentes paquetes disponibles en la wiki de ROS se tienen todas las características del robot, el modelo URDF, que contiene el algoritmo de la estructura robótica dentro del entorno ROS.

Ya que en este caso se cuenta con Ubuntu 16.04, se instalará la versión compatible de ROS llamada ROS Kinetic.

4.5.2 RViz

Es un visualizador 3D que permite representar datos de diferentes sensores y los distintos estados de información de ROS. Mediante Rviz se puede visualizar modelos virtuales robóticos y mostrar "*online*" los datos recibidos por los diferentes sensores de robots reales, como la visualización de imágenes obtenidas a partir de una cámara, nube de puntos de sensores láseres, mapas, o las diferentes "transformaciones" que forman el modelo virtual del robot, entre otros [46].

4.5.2.1 URDF

Unified Robot Descripción Format (URDF) es una especificación XML para describir un robot. La especificación incluye:

- Descripción cinemática y dinámica del robot.
- Representación visual del robot.
- Modelo de colisiones del robot.

La descripción de un robot consiste en un conjunto de elementos de eslabones (links) y un conjunto de elementos de unión (joints) que conectan los eslabones juntos [47].

4.5.2.2 *joint _state _publisher*

Es un paquete de ROS el cual publica mensajes de tipo *sensor_msgs/JointState* a un robot. Este paquete lee el parámetro *robot_description,* encuentra las articulaciones no fijas y publica un mensaje JointState con todas esas articulaciones definidas.

Cuenta con una herramienta GUI que despliega una ventana con *sliders*, cada *slider* se ajusta a los límites máximo y mínimo de cada articulación [48].

La caja de control y programación del robot UR5 se puede observar en la Figura 34, el movimiento del robot se realiza mediante las flechas indicadas, es por esto que se decidió hacer el movimiento con los *sliders* haciendo uso del nodo *"joint_state_publisher".*

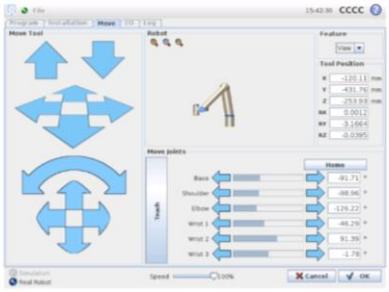


Figura 34. Caja de control y programación del UR5 [30].

4.6 Programación en ROS para el robot UR5

Para el robot UR5 que se tiene en ROS se implementa un código el cual permite visualizar el robot en la plataforma Rviz, con los *sliders* correspondientes a cada articulación; también permite la conexión con ROS a través de Rosbridge, el archivo *principal.launch* se muestra a continuación:

```
<launch>

   <include file="$(find
rosbridge_server)/launch/rosbridge_websocket.launch">
            <param name="port" value="9090"/>
   </include>

   <arg name="urdf_file" default="$(find xacro)/xacro.py '$(find
ur_description)/urdf/UR5_robot.urdf.xacro'"/>

   <param name="robot/name" value="UR5" />
   <param name="robot_description" command="$(arg urdf_file)" />

   <node name="joint_state_publisher" pkg="joint_state_publisher"
type="joint_state_publisher">
            <param name="use_gui" value="TRUE"/>

   </node>


   <node name="robot_state_publisher" pkg="robot_state_publisher"
type="state_publisher"/>

   <node name="rviz" pkg="rviz" type="rviz" />
   <node name="rqt_graph" pkg="rqt_graph" type="rqt_graph"
output="screen"/>


</launch>
```

A continuación, se explicará cada línea del código para su mayor comprensión. Inicialmente se tiene:

```
<include file="$(find
rosbridge_server)/launch/rosbridge_websocket.launch">
<param name="port" value="9090"/>
</include>
```

Aquí se incluye el archivo rosbridge_websocket.launch el cual permite la conexión de programas externos, en este caso Unity, con el programa ROS. Se incluye el parámetro “port” con el valor “9090” que es el que se tiene por defecto.

A continuación, se tiene:

```
<arg name="urdf_file" default="$(find xacro)/xacro.py '$(find
ur_description)/urdf/UR5_robot.urdf.xacro'"/>

  <param name="robot/name" value="UR5" />
  <param name="robot_description" command="$(arg urdf_file)" />
```

Aquí se agrega el modelo URDF del robot UR5, que se toma del repositorio Ros Industrial [42], este es el archivo que se importa a Unity a través de Rosbridge.

Ahora:

```
<node name="joint_state_publisher" pkg="joint_state_publisher"
type="joint_state_publisher">
        <param name="use_gui" value="TRUE"/>

</node>
```

Como se observa, aquí se llama el nodo “joint_state_publisher” y se activa la ventana con los sliders correspondientes a cada articulación, con los cuales se realizarán los movimientos del UR5.

Por último, se tiene:

```
<node name="robot_state_publisher" pkg="robot_state_publisher"
type="state_publisher"/>
  <node name="rviz" pkg="rviz" type="rviz" />
```

```
  <node name="rqt_graph" pkg="rqt_graph" type="rqt_graph"
output="screen"/>
```

El nodo “robot_state_publisher toma el modelo URDF y los ángulos de cada articulación del robot del nodo “joint_state_publisher” y publica los resultados en Rviz vía tf, el cual es un paquete de ROS que permite que el usuario haga un seguimiento del movimiento de las articulaciones del robot.

El nodo “rviz” permite visualizar el robot en la plataforma Rviz, el nodo “rqt_graph” permite visualizar las gráficas de computación de ROS, como la que se observa en la Figura 35, en donde se pueden observar los diversos nodos de ROS que están activos.

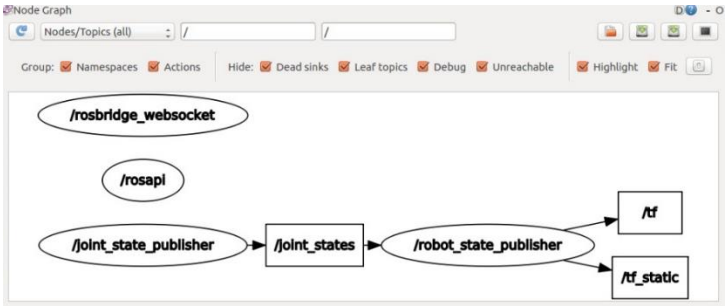


Figura 35. Nodos activos en ROS.
Fuente: Propia

4.7 Simulación

La simulación de la aplicación se realiza en Rviz mediante la integración ROS-Unity. Para su desarrollo es necesario inicializar el entorno y establecer la comunicación con Unity.

Para llevar a cabo la simulación de la aplicación se requieren dos sesiones de terminal (T1, T2), así:

T1

Ejecución del *roscore*

T2

Ejecución del archivo *principal.launch*

4.7.1 UR5 en ROS

Para iniciar la simulación, lo primero que se debe hacer es inicializar el *roscore*, por lo que en T1 se ejecuta:

```
$ source /home/luisa/catkin_ws/devel/setup.bash
$ roscore
```

El resultado se observa en la Figura 36:

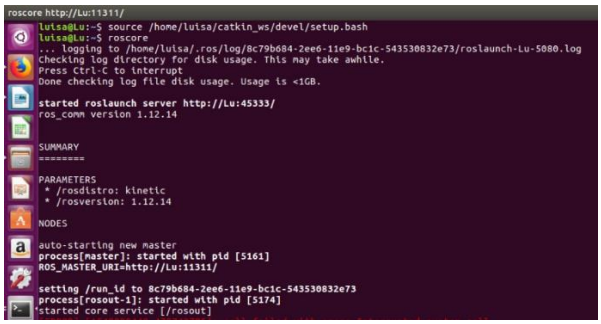


Figura 36. Inicialización exitosa del *roscore*.
Fuente: Propia

A continuación, en T2 se inicializa el archivo *principal.launch*, de la siguiente manera:

```
$ source /home/luisa/catkin_ws/devel/setup.bash
$ roslaunch file_server principal.launch
```

El resultado se obtiene en la Figura 37:

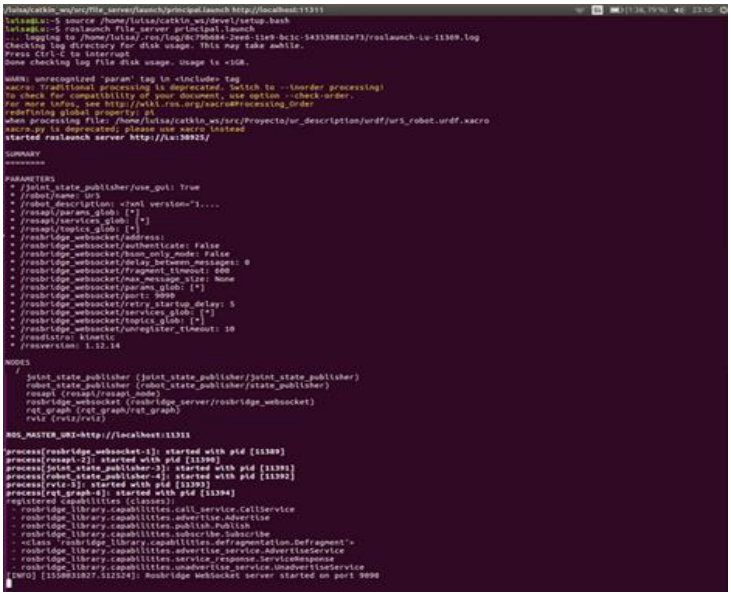


Figura 37. Ejecución del archivo *principal.launch*.
Fuente: Propia

A continuación, en la Figura 38 se puede visualizar el robot UR5 en la herramienta Rviz, con los *sliders* que corresponden a cada articulación.

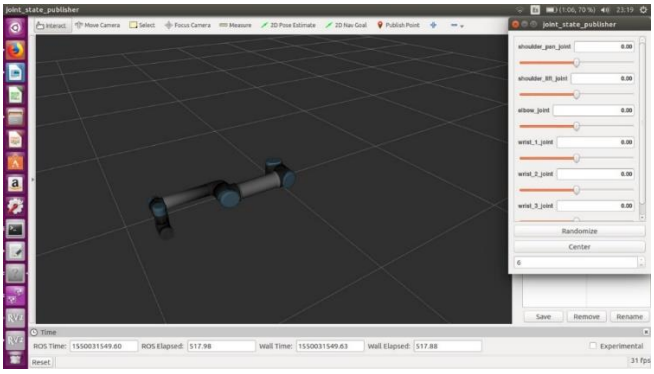


Figura 38. UR5 en Rviz.
Fuente: Propia

4.7.2 Importación del robot UR5 a Unity

La importación de modelo de robot UR5 a Unity, se hace con el fin de poder implementar los módulos de comunicación con la versión del robot ya adecuado en el motor de juegos Unity 3D. A continuación, se presentarán los pasos que se deben realizar para la importación del modelo.

Principalmente se realiza la descarga del repositorio *RosSharp* [49] (conjunto de herramientas y bibliotecas que permiten la comunicación de ROS con Unity), como se observa en la Figura 39, la descarga es un archivo *.unitypackage* (archivo de Unity) el cual puede abrirse mediante el *editor de Unity* (ver Figura 40).

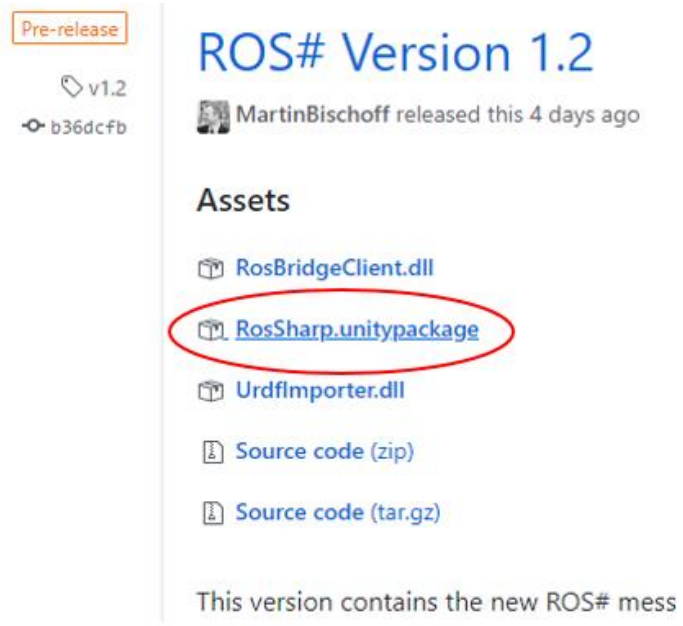


Figura 39. Repositorio RosSharp.
Fuente: Propia

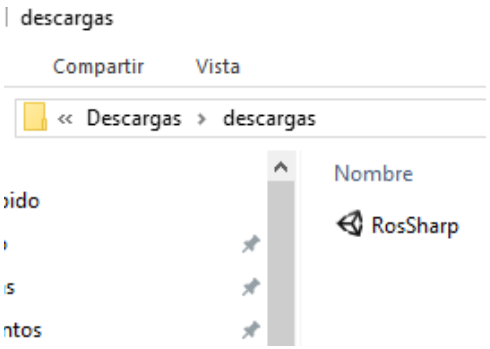


Figura 40. Formato de trabajo en Unity.
Fuente: Propia

Al abrir el archivo que se descargó se obtiene una escena en Unity donde se encontrará el *RosBridgeClient* en la barra de herramientas como se observa en la Figura 41, el cual nos permite generar la conexión de Unity con Ros realizando una serie de pasos que se nombrarán posteriormente.

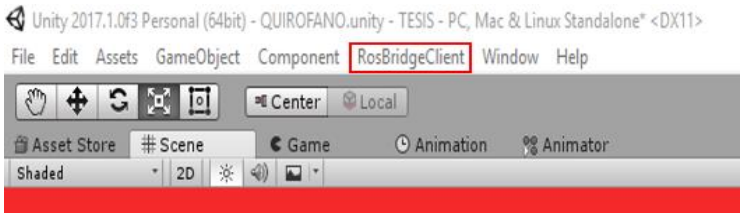


Figura 41. *RosBridgeClient* en la barra de Unity.
Fuente: Propia

Al hacer click en el *RosBridgeClient*, se genera una nueva opción (ver Figura 42).

Se procede a realizar un click en esta opción, para que así despliegue una nueva pestaña llamada *RosSharp.Urdf* (Figura 43), en la cual ingresaremos algunos datos como la correspondiente dirección IP del computador, con el puerto que está por defecto, que es el “9090” y el lugar donde se desea importar el modelo. Esto se realiza con el fin de generar la conexión de los

dos computadores y para ello estos deben estar enlazados con un cable *Ethernet*.

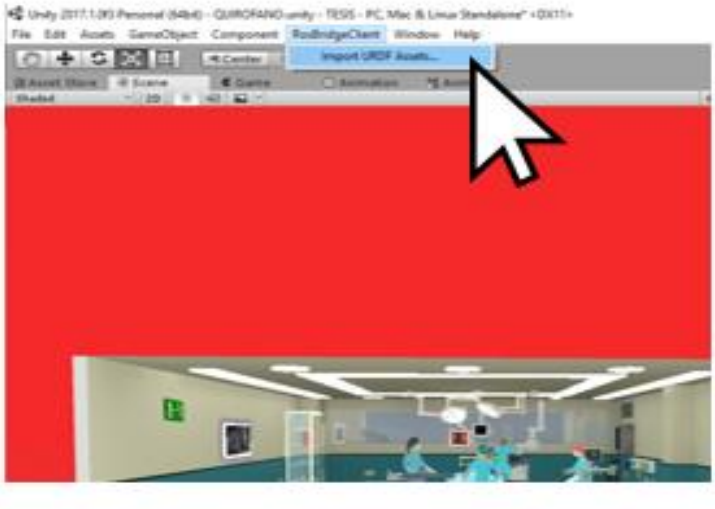


Figura 42. Pestaña de importación del modelo en el *RosBridgeClient*.
Fuente: Propia

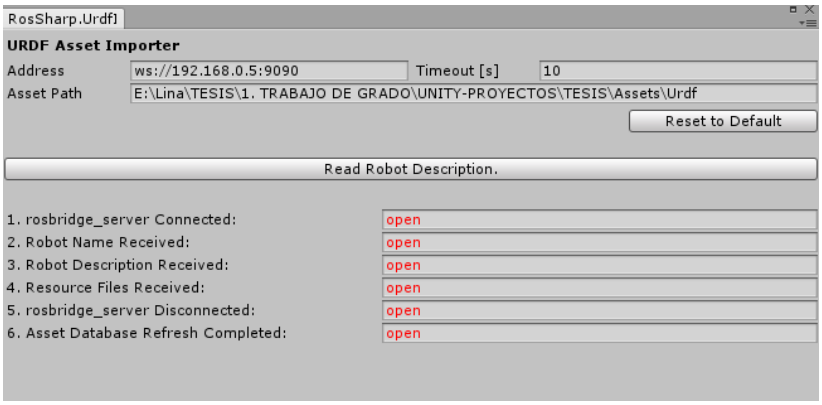


Figura 43. Pestaña del *RosSharp.Urdf*.
Fuente: Propia

Con el procedimiento anterior se procede a leer la descripción del robot, para que así las características principales de este se encuentren habilitadas y en letras verdes (*done*), como se observa en la Figura 44.

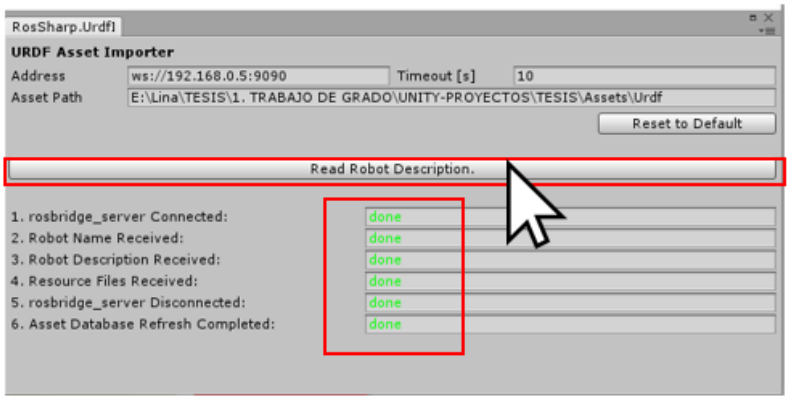


Figura 44. Lectura de la descripción del robot.
Fuente: Propia

Cuando las caracteristicas se encuentran habilitasdas en color verde (*done),* aparece una pestaña que nos da la opción de generar el modelo que se desea importar en la escena de Unity, para ello de procedió a dar click en la opción "*yes"* como se ve en la Figura 45.

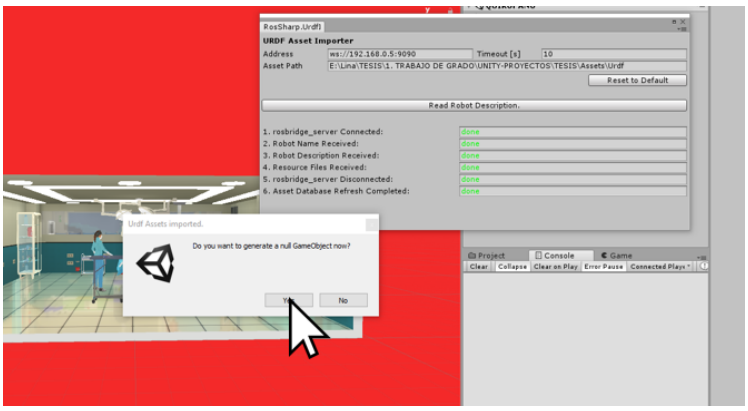


Figura 45. Importación del modelo en la escena de Unity.
Fuente: Propia

Con el procedimiento anterior, se obtuvo el *gameobject* en la *Hierarcy* de Unity llamado UR5 donde se puede observar el robot, con sus diferentes características, importado en la escena (Figura 46). Las únicas características que no son apropiadas en el robot, son los colores y las

texturas, por consiguiente, se procede a hacer un cambio en la visualización del robot basándose en el robot original de la Figura 4 para así obtener el resultado que se observa en la Figura 47.

Figura 46. Modelo del robot UR5 importado desde ROS a Unity.
Fuente: Propia

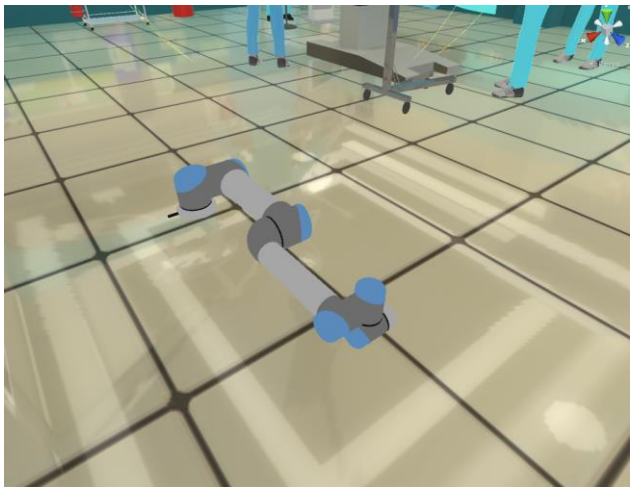


Figura 47. UR5 en la escena de Unity.
Fuente: Propia

Teniendo ya el robot importado, se realizaron una serie de arreglos e implementaciones de códigos, para que así este pudiera ser manejado desde ROS, tales como cambios de velocidad, anexos de herramientas y códigos en el inspector.

Aplicando todos los cambios necesarios se ejecuta el programa. Con la conexión de los dos computadores ya establecida, se pueden realizar los movimientos requeridos por medio de *sliders*.

5. RESULTADOS

5.1 Robot semiautónomo en Unity

Este robot proporcionará una asistencia en el quirófano la cual controla el movimiento de la cámara y su navegación (ver Figura 48). Es controlado por medio de una trayectoria punto a punto establecida, con el fin de llegar al tumor de la hipófisis abarcando todos los obstáculos de la vía nasal tales como el cornete inferior, cornete medio, hueso esfenoidal e hipófisis.

El modelo de este robot está basado en el modelo importado por ROS, con la diferencia de que a este no se le agregaron los mismos scripts debido a que por ser un robot semiautónomo sus movimientos fueron realizados directamente desde el motor de Unity, también se le fue agregado a su ultima articulación un endoscopio el cual cuenta con una cámara visualizada en el *game* donde se pueden observar los movimientos que el robot está ejecutando.

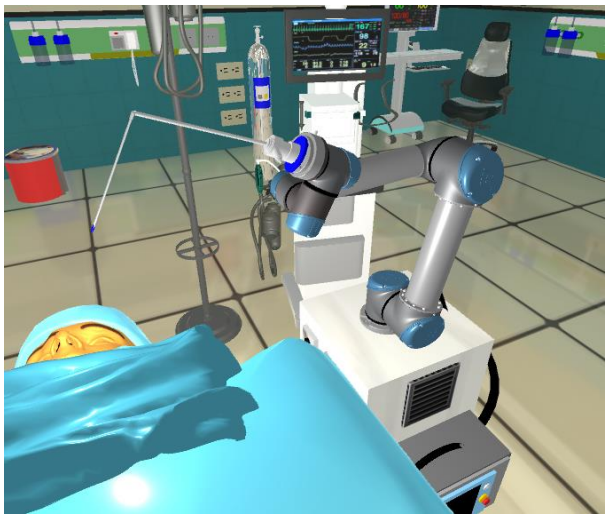


Figura 48. Robot semiautónomo.
Fuente: Propia

Para la realización de la trayectoria de este robot, fue necesario la identificación de cada punto que este abarcaría. Esto se logró con la variación de la rotación de cada articulación del robot a una velocidad establecida en Unity, con el fin de que el *script* realizado nos mostrará unas variables públicas las cuales se pueden

modificar según los puntos de cada trayectoria y así cambiar los valores de la rotación de cada articulación según el punto al que se desea llegar, tal como se muestra en la Figura 49, donde se muestran variables como *speed* que es la velocidad que se le desee establecer al movimiento del robot , *los targets* en los cuales se añade el gameobject al que se le realizará el movimiento en este caso las articulaciones del robot (2) y los *rotation vectors* que son los vectores de rotación modificados según cada punto.

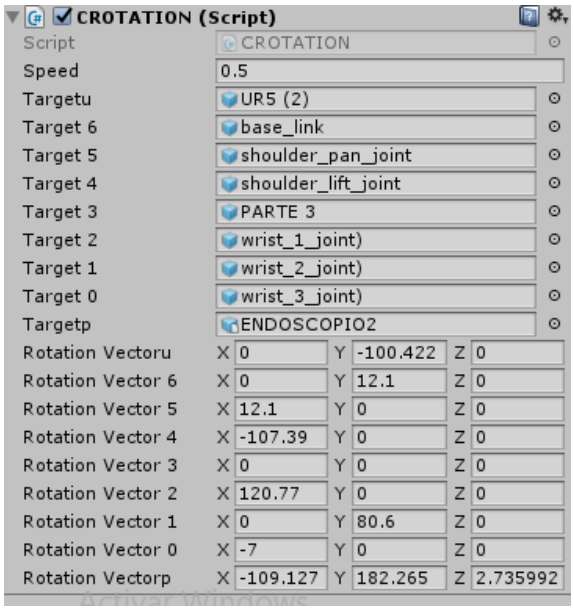


Figura 49. Variables públicas de modificación en Unity.
Fuente: Propia

Se realizaron cinco códigos en C# estableciendo las cinco trayectorias punto a punto.

5.1.1 Puntos de la trayectoria

- Posición inicial

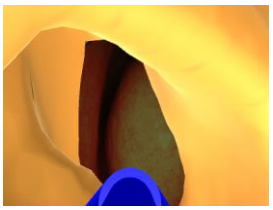
Articulación	Rotación X [º]	Rotación Y [º]	Rotación Z [º]
UR5 (2)	0	-100,422	0
base_link	0	12,1	0
shoulder_pan_joint	0	0	0
shoulder_lift_joint	12,1	0	0
elbow_joint	-107,39	0	0
wrist_1_joint	120,77	0	0
wrist_2_joint	0	80,6	0
wrist_3_joint	0	0	-7
ENDOSCOPIO2	-109,127	182,265	2,735992

- Vista de la cámara en la posición inicial

- Cornete inferior

Articulación	Rotación X [º]	Rotación Y [º]	Rotación Z [º]
UR5 (2)	0	-100,422	0
base_link	0	12,1	0
shoulder_pan_joint	0	0	0
shoulder_lift_joint	-4,7	0	0
elbow_joint	-107,39	0	0
wrist_1_joint	120,77	0	0
wrist_2_joint	0	80,6	0
wrist_3_joint	0	0	-7
ENDOSCOPIO2	-109,127	182,265	2,735992

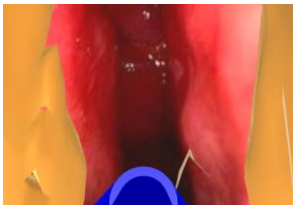
- Vista de la cámara en el cornete inferior



- Cornete medio

Articulación	Rotación X [º]	Rotación Y [º]	Rotación Z [º]
UR5 (2)	0	-100,422	0
base_link	0	13,4	0
shoulder_pan_joint	0	0	0
shoulder_lift_joint	-9,2	0	0
elbow_joint	-110,7	0	0
wrist_1_joint	124,2	0	0
wrist_2_joint	0	80,6	0
wrist_3_joint	0	0	-7
ENDOSCOPIO2	-109,127	182,265	2,735992

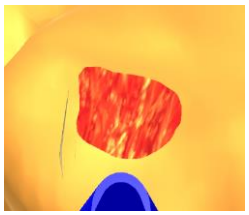
- Vista de la cámara en el cornete medio



- Hueso esfenoidal

Articulación	Rotación X [º]	Rotación Y [º]	Rotación Z [º]
UR5 (2)	0	-100,422	0
base_link	0	14,4	0
shoulder_pan_joint	0	0	0
shoulder_lift_joint	-12,6	0	0
elbow_joint	-111,5	0	0
wrist_1_joint	125	0	0
wrist_2_joint	0	80,6	0
wrist_3_joint	0	0	-7
ENDOSCOPIO2	-109,127	182,265	2,735992

- Vista de la cámara en el hueso esfenoidal



- Hipófisis (vista del tumor)

Articulación	Rotación X [°]	Rotación Y [°]	Rotación Z [°]
UR5 (2)	0	-100,422	0
base_link	0	14,9	0
shoulder_pan_joint	0	0	0
shoulder_lift_joint	-15,6	0	0
elbow_joint	-113,9	0	0
wrist_1_joint	129,2	0	0
wrist_2_joint	0	80.6	0
wrist_3_joint	0	0	-7
ENDOSCOPIO2	-109,127	182,265	2,735992

- Vista de la cámara dentro de la hipófisis (vista del tumor)

Tabla 1. Puntos de la trayectoria para el robot semiautónomo en Unity.
Fuente: Propia

5.2 Robot UR5 en ROS

Como se mencionó en el apartado 4.6 el movimiento del robot UR5 en ROS se realiza con la ayuda del nodo "*joint_state_publisher*", el cual permite visualizar una ventana con *sliders* que deja llevar al robot a las posiciones deseadas.

A continuación, se exponen las diferentes posiciones para el robot ingresando por la primera fosa nasal con la herramienta del taladro.

5.2.1 Puntos de la trayectoria

- Posición inicial

Articulación	Rotación [rad]
base_link	0
shoulder_pan_joint	0
shoulder_lift_joint	0
elbow_joint	0
wrist_1_joint	0
wrist_2_joint	0
wrist_3_joint	0

- Vista de la cámara en la posición inicial

- Cornete inferior

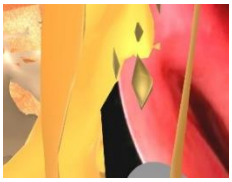
Articulación	Rotación [rad]
base_link	0
shoulder_pan_joint	0
shoulder_lift_joint	0
elbow_joint	0.56
wrist_1_joint	-0.12
wrist_2_joint	-0.02
wrist_3_joint	0

- Vista de la cámara en el cornete inferior

- Cornete medio

Articulación	Rotación [rad]
base_link	0
shoulder_pan_joint	0
shoulder_lift_joint	0
elbow_joint	0.66
wrist_1_joint	-0.24
wrist_2_joint	-0.05
wrist_3_joint	0

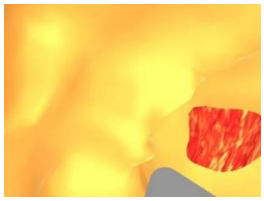
- Vista de la cámara en el cornete medio



- Hueso esfenoidal

Articulación	Rotación [rad]
base_link	0
shoulder_pan_joint	0
shoulder_lift_joint	0
elbow_joint	0.75
wrist_1_joint	-0.38
wrist_2_joint	-0.06
wrist_3_joint	0

- Vista de la cámara en el hueso esfenoidal



- Hipófisis (vista del tumor)

Articulación	Rotación [rad]
base_link	0
shoulder_pan_joint	0
shoulder_lift_joint	0
elbow_joint	0.87
wrist_1_joint	-0.56
wrist_2_joint	-0.09
wrist_3_joint	0

- Vista de la cámara dentro de la hipófisis (vista del tumor)

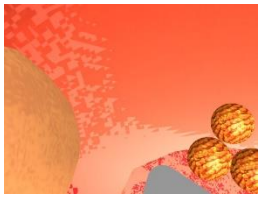


Tabla 2. Puntos de la trayectoria para el robot UR5 en ROS ingresando por la primera fosa nasal con la herramienta del taladro.
Fuente: Propia

A continuación, se exponen las diferentes posiciones para el robot ingresando por la segunda fosa nasal con la herramienta del taladro.

- Posición inicial

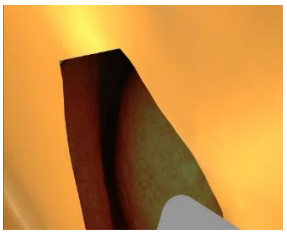
Articulación	Rotación [rad]
base_link	0
shoulder_pan_joint	0
shoulder_lift_joint	0
elbow_joint	0
wrist_1_joint	0
wrist_2_joint	0
wrist_3_joint	0

- Vista de la cámara en la posición inicial

- Cornete inferior

Articulación	Rotación [rad]
base_link	0
shoulder_pan_joint	0
shoulder_lift_joint	0
elbow_joint	0.58
wrist_1_joint	-0.09
wrist_2_joint	-0.02
wrist_3_joint	0

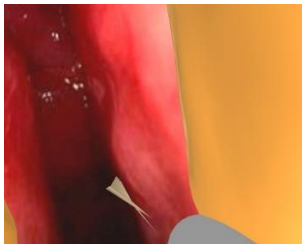
- Vista de la cámara en el cornete inferior



- Cornete medio

Articulación	Rotación [rad]
base_link	0
shoulder_pan_joint	0
shoulder_lift_joint	0
elbow_joint	0.65
wrist_1_joint	-0.19
wrist_2_joint	-0.03
wrist_3_joint	0

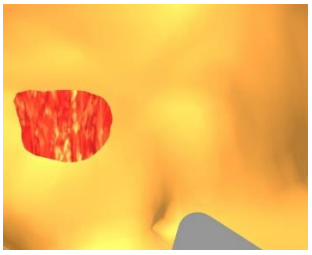
- Vista de la cámara en el cornete medio



- Hueso esfenoidal

Articulación	Rotación [rad]
base_link	0
shoulder_pan_joint	0.05
shoulder_lift_joint	0
elbow_joint	0.79
wrist_1_joint	-0.47
wrist_2_joint	-0.07
wrist_3_joint	0

- Vista de la cámara en el hueso esfenoidal



- Hipófisis (vista del tumor)

Articulación	Rotación [rad]
base_link	0
shoulder_pan_joint	0.10
shoulder_lift_joint	0
elbow_joint	1.00
wrist_1_joint	-0.91
wrist_2_joint	-0.10
wrist_3_joint	0

- Vista de la cámara dentro de la hipófisis (vista del tumor)

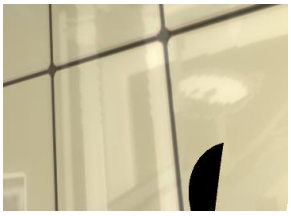
Tabla 3. Puntos de la trayectoria para el robot UR5 en ROS ingresando por la segunda fosa nasal con la herramienta del taladro.
Fuente: Propia

A continuación, se exponen las diferentes posiciones para el robot ingresando por la primera fosa nasal con la herramienta del disector.

- Posición inicial

Articulación	Rotación [rad]
base_link	0
shoulder_pan_joint	0
shoulder_lift_joint	0
elbow_joint	0
wrist_1_joint	0
wrist_2_joint	0
wrist_3_joint	0

- Vista de la cámara en la posición inicial



- Cornete inferior

Articulación	Rotación [rad]
base_link	0
shoulder_pan_joint	0
shoulder_lift_joint	0
elbow_joint	0.48
wrist_1_joint	-0.04
wrist_2_joint	-0.01
wrist_3_joint	0

- Vista de la cámara en el cornete inferior

- Cornete medio

Articulación	Rotación [rad]
base_link	0
shoulder_pan_joint	0
shoulder_lift_joint	0
elbow_joint	0.69
wrist_1_joint	-0.30
wrist_2_joint	-0.05
wrist_3_joint	0

- Vista de la cámara en el cornete medio

<table>
<tr>
<td>

- Hueso esfenoidal

Articulación	Rotación [rad]
base_link	0
shoulder_pan_joint	0
shoulder_lift_joint	0
elbow_joint	0.77
wrist_1_joint	-0.38
wrist_2_joint	-0.08
wrist_3_joint	0

</td>
<td>

- Vista de la cámara en el hueso esfenoidal

</td>
</tr>
<tr>
<td>

- Hipófisis (vista del tumor)

Articulación	Rotación [rad]
base_link	0
shoulder_pan_joint	0
shoulder_lift_joint	0
elbow_joint	0.89
wrist_1_joint	-0.57
wrist_2_joint	-0.10
wrist_3_joint	0

</td>
<td>

- Vista de la cámara dentro de la hipófisis (vista del tumor)

</td>
</tr>
</table>

Tabla 4. Puntos de la trayectoria para el robot UR5 en ROS ingresando por la primera fosa nasal con la herramienta del disector.
Fuente: Propia

Como se observa con las trayectorias utilizadas se logra acceder a la totalidad del tumor para su posterior extracción y no se evidencia una gran diferencia entre una herramienta y otra.

Es importante tener en cuenta que estos datos corresponden a una cabeza en específico, si se quisiera cambiar de cabeza todos los datos cambiarían debido a la anatomía y morfología correspondiente a cada persona.

5.3 Procedimiento de la simulación

Para realizar el correcto procedimiento de la simulación se planteó una serie de pasos que se observan en el diagrama de la Figura 50.

Es importante tener en cuenta que para poder iniciar con el primer acceso se deben realizar 2 pasos fundamentales que los ejecuta el cirujano de manera manual: aplicar la epinefrina, con el fin de evitar el sangrado en la mucosa nasal y seccionar cornetes, con el objetivo de abrir la fosa nasal para permitir el paso del robot y las herramientas.

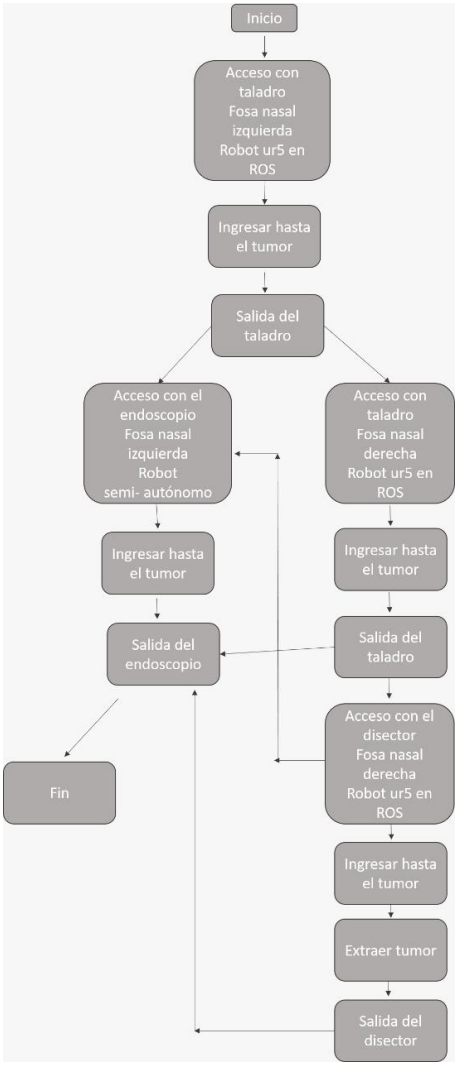


Figura 50. Diagrama de flujo del procedimiento de la simulación.
Fuente: Propia

Después de que el cirujano realice los primeros pasos, se procede a ingresar con el robot UR5 desde ROS, que tiene el taladro en su efector final, por la fosa nasal izquierda. Se debe pasar por el cornete inferior, cornete medio y hueso esfenoidal hasta llegar a la hipófisis en donde se encuentra el tumor. Una vez realizado esto, el robot sale y se procede a que el robot semiautónomo, que tiene el endoscopio, ingrese por la misma fosa nasal que ya está abierta.

Se realiza el mismo procedimiento del taladro por la fosa nasal derecha, hasta que se llega al tumor. Una vez realizado esto se efectúa un cambio de herramienta, el disector que será el encargado de la extracción del tumor ingresa de la misma manera; se realiza la extracción del tumor, posteriormente, el cirujano debe realizar el taponamiento nasal y con esto se finaliza la simulación de la cirugía.

5.4 Configuración del quirófano

Para realizar el anterior procedimiento de manera correcta se realizaron diferentes pruebas de cuál sería la posición correcta de los robots en el quirófano, con el objetivo de que no se den choques entre ellos y de que el cirujano tenga también un espacio de trabajo adecuado.

Inicialmente, se planteó la configuración de las siguientes figuras:

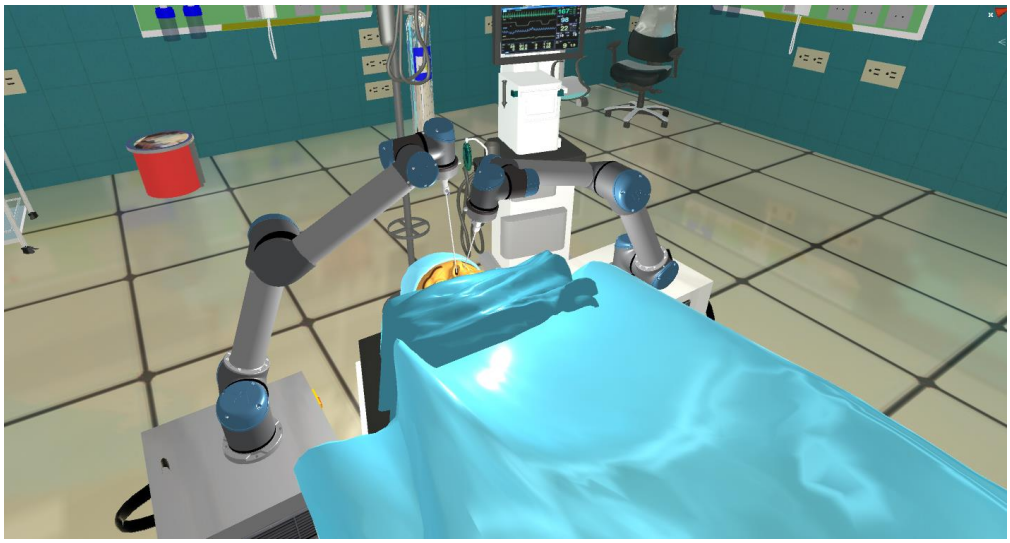


Figura 51. Configuración 1, vista 1.
Fuente: Propia

Figura 52. Configuración 1, vista 2.
Fuente: Propia

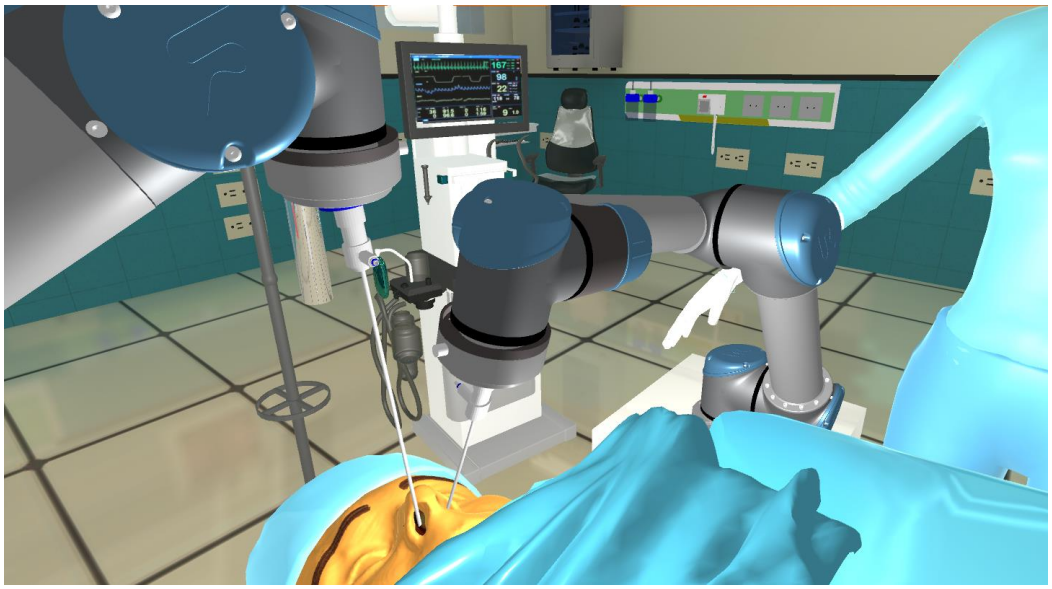


Figura 53. Configuración 1, vista 3.
Fuente: Propia

Como se puede observar esta ubicación de los robots no es adecuada debido a que el cirujano no cuenta con un espacio de trabajo óptimo para realizar los procedimientos de la cirugía, adicionalmente existe un riesgo de colisión entre los 2 robots.

El segundo intento de configuración se puede observar en las siguientes figuras:

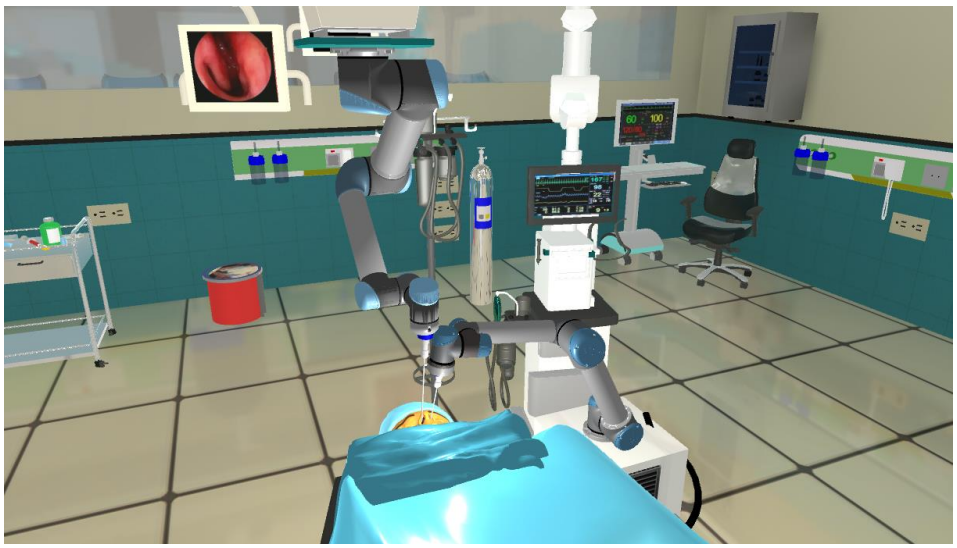


Figura 54. Configuración 2, vista 1.
Fuente: Propia

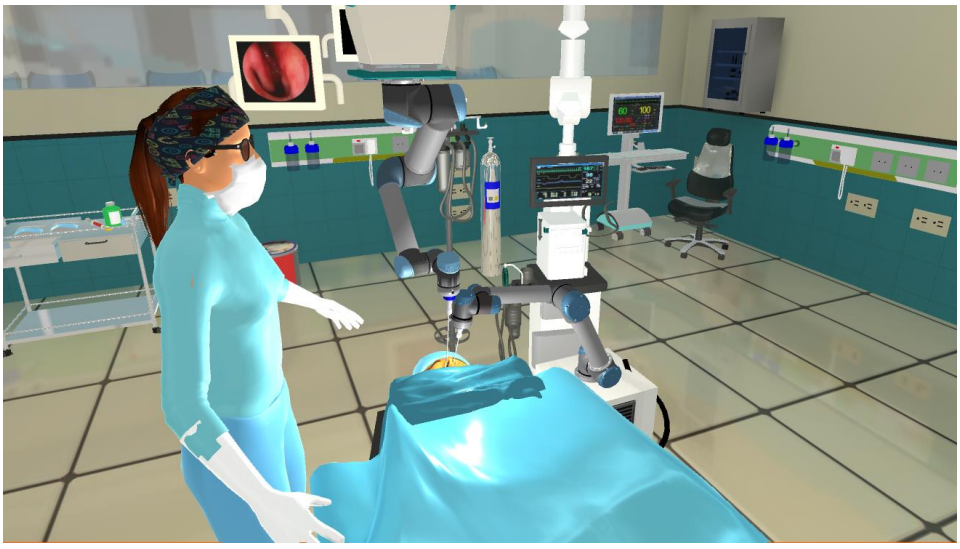


Figura 55. Configuración 2, vista 2.
Fuente: Propia

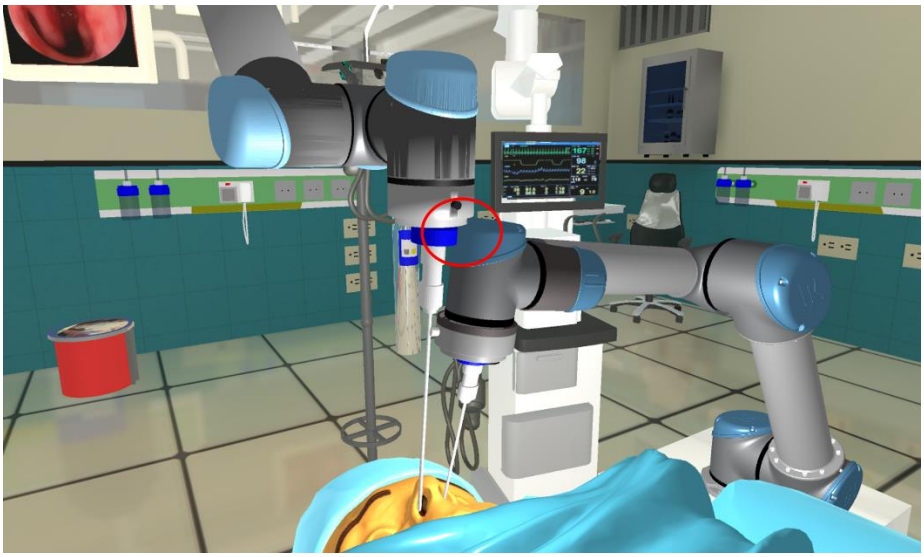


Figura 56. Configuración 2, vista 3.
Fuente: Propia

En este caso se puede observar que el cirujano cuenta con un espacio de trabajo adecuado, pero que al ubicar los robots de tal manera se da una colisión entre ellos, por lo que esta configuración no es correcta.

Por último, se tiene la configuración utilizada que se puede observar en las siguientes figuras:

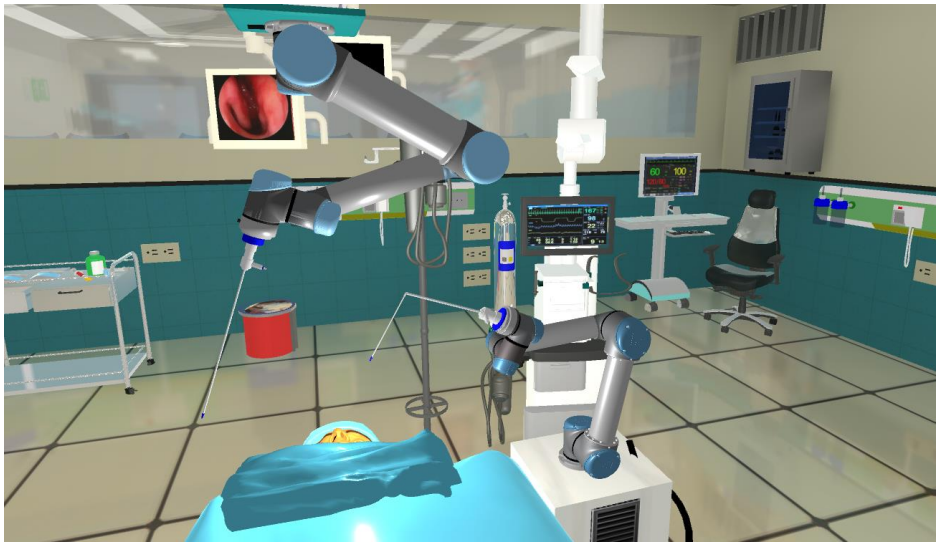


Figura 57. Configuración 3, vista 1.
Fuente: Propia

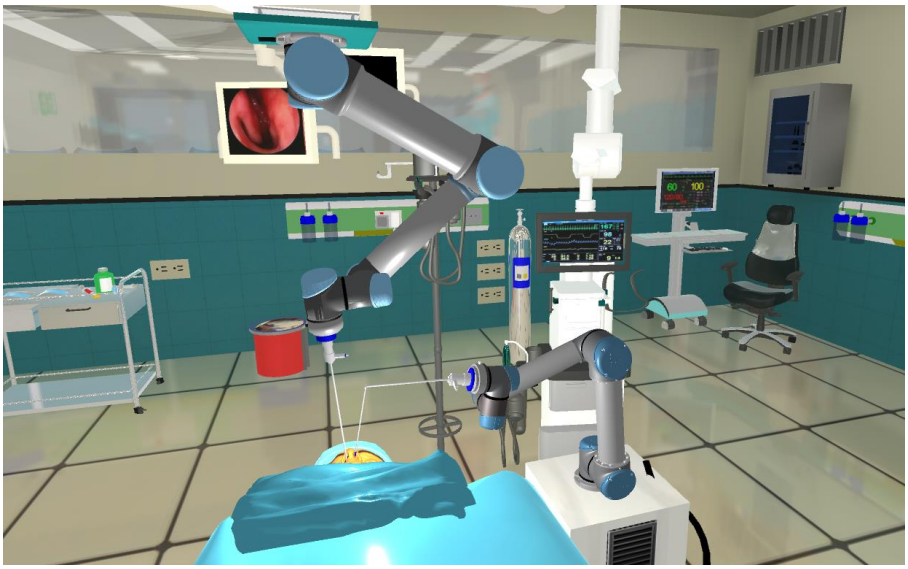


Figura 58. Configuración 3, vista 2.
Fuente: Propia

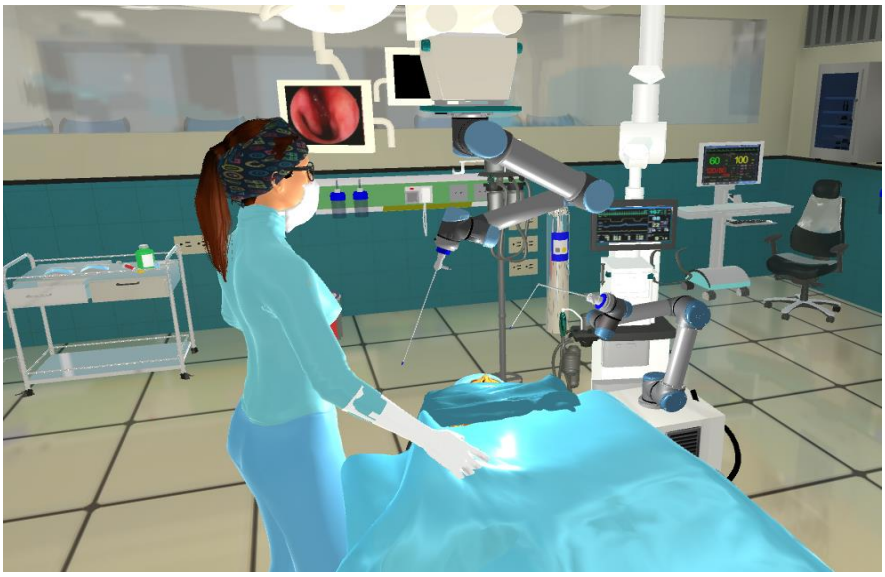


Figura 59. Configuración 3, vista 3.
Fuente: Propia

En este caso fue importante realizar un cambio en el modelo del endoscopio que se usa (ver Figura 60), ya que de esta manera se obtiene que los 2 robots pueden

estar activos sin que exista riesgo de colisión entre ellos y que el cirujano tenga espacio suficiente para realizar sus procedimientos.

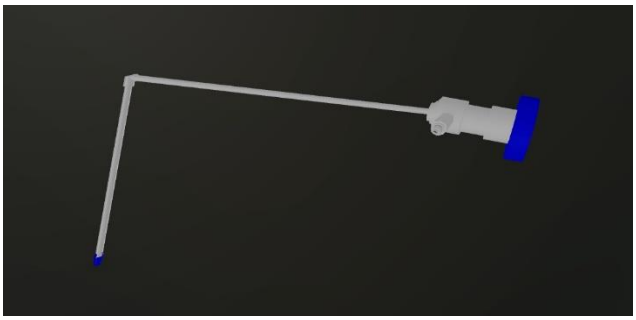


Figura 60. Endoscopio modificado.
Fuente: Propia

5.5 Secuencia de desplazamientos de los robots

Con la configuración correcta de los robots en el quirófano también se obtuvo que cada uno de los robots debe realizar pequeños desplazamientos para llegar a cada posición deseada (ver Figura 61), esto representa una gran ventaja debido a que el área de trabajo (nariz) es muy pequeña, por lo tanto, los movimientos realizados por los robots no deben ser bruscos porque podrían llegar a causar daño interno.

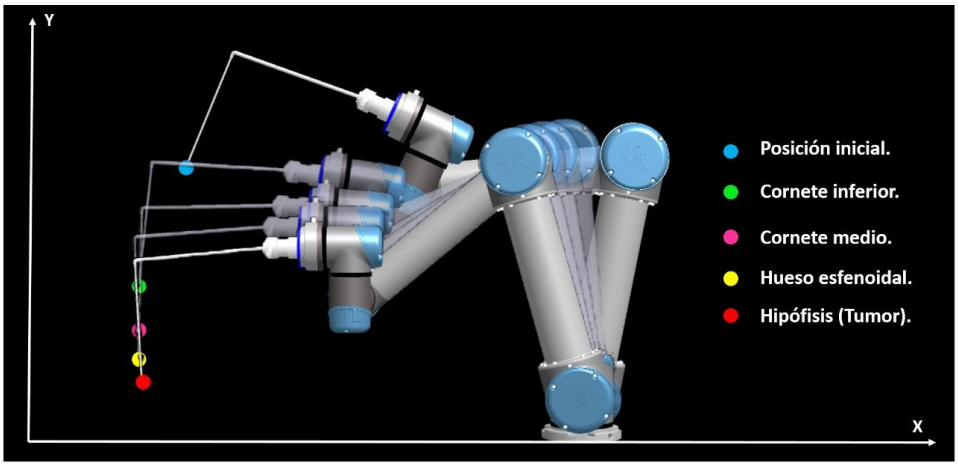


Figura 61. Secuencia de desplazamientos del robot semiautónomo en el plano XY.
Fuente: Propia

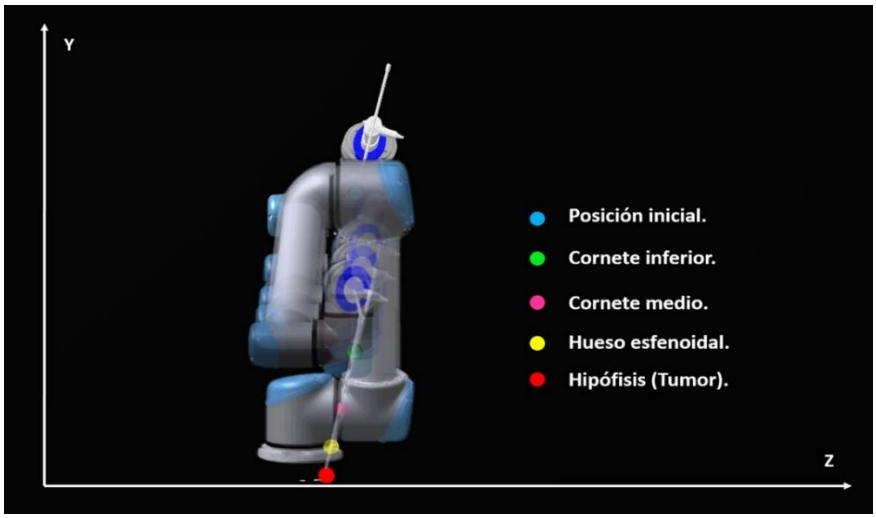


Figura 62. Secuencia de desplazamientos del robot semiautónomo en el plano YZ.
Fuente: Propia

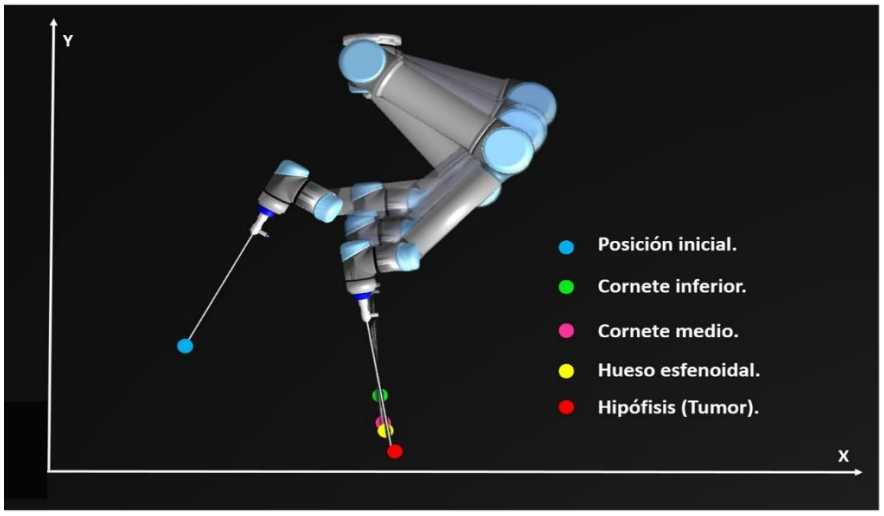


Figura 63. Secuencia de desplazamientos del robot UR5 en ROS en el plano XY.
Fuente: Propia

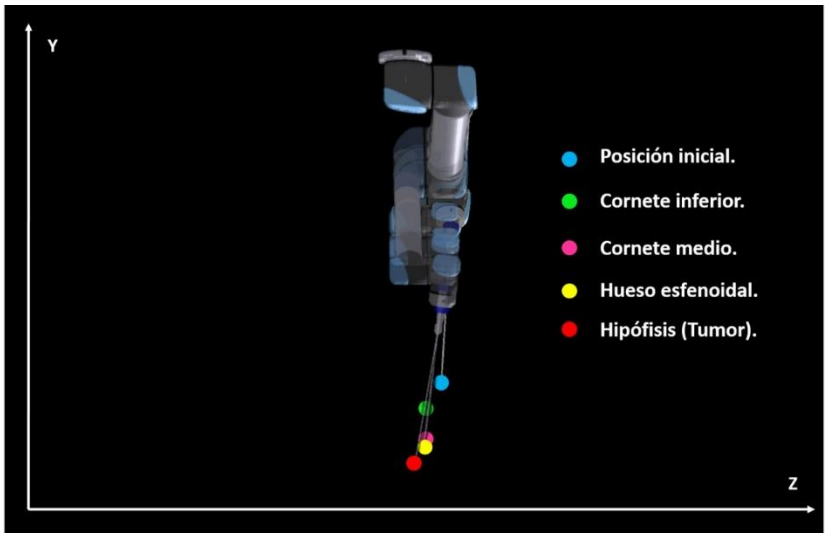


Figura 64. Secuencia de desplazamientos del robot UR5 en ROS en el plano YZ.
Fuente: Propia

5.6 Cálculo del error

En la Figura 65 se presenta la anatomía involucrada en la intervención quirúrgica real, donde se encuentran los cuatro puntos fundamentales de esta cirugía, los cuales están ubicados en el conducto nasal que lleva a la glándula hipofisiaria con el fin de abordar el tumor.

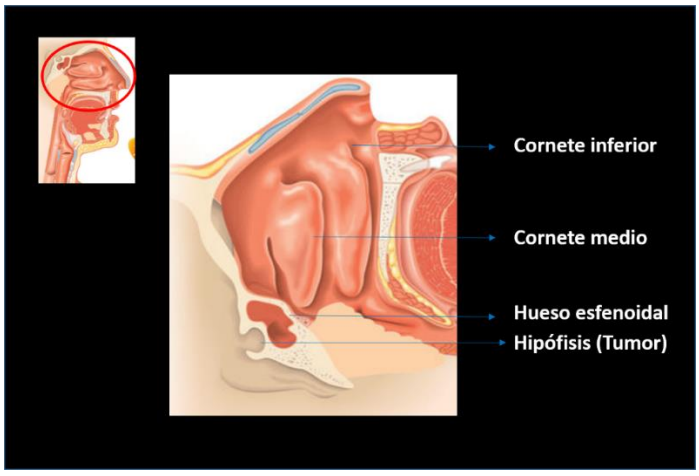


Figura 65. Anatomía involucrada en la intervención
Fuente: Propia

Para realizar la comprobación de que la herramienta de los robots accediera correctamente por el conducto nasal hasta llegar al tumor, se elaboraron tres *gameobject* en forma de planos, los cuales fueron ubicados en el conducto nasal con una posición en los ejes (X, Y, Z) dados en Unity del modelo 3D de la cabeza e involucrando también la glándula hipofisiaria del modelo del cerebro con su debida posición, ver Figura 66.

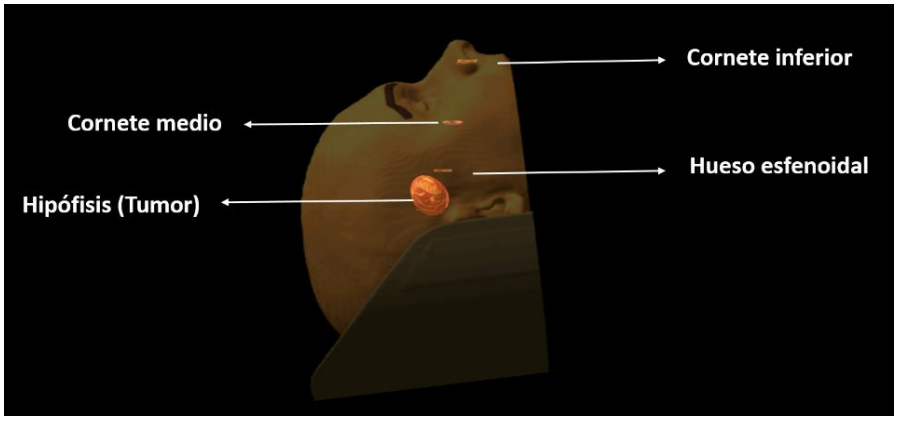


Figura 66. Anatomía en el ambiente quirúrgico.
Fuente: Propia

- Robot semiautónomo

A la punta del endoscopio del robot semiautónomo se le agregó un *gameobject* en forma de esfera, con unas dimensiones acordes a la herramienta, con el fin de analizar la posición de la esfera en los ejes (X, Y, Z) en el ambiente de Unity cuando la herramienta acceda al conducto nasal de la fosa izquierda abordando cada obstáculo de este.

El primer obstáculo al que se enfrenta la herramienta es el plano ubicado en el cornete inferior como se puede observar en la Figura 67.

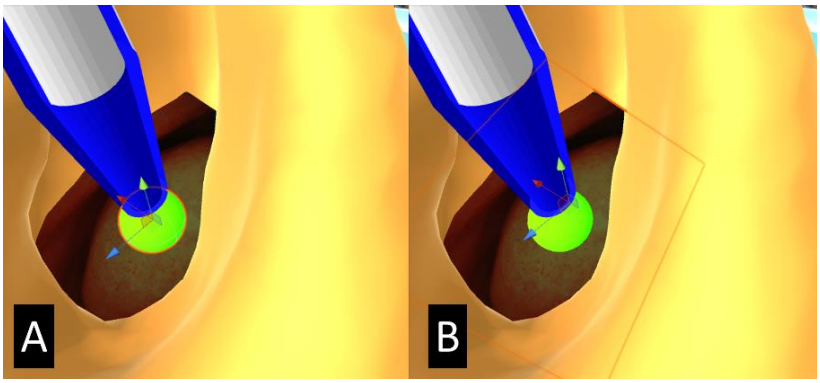


Figura 67. Eje de coordenadas (X, Y, Z) de la esfera (A) y del plano ubicado en el cornete inferior (B) en la fosa nasal izquierda.
Fuente: Propia

Posiciones	X	Y	Z
PLANO CORNETE INFERIOR	-10.099	2.629	-11.90
ESFERA	-10.101	2.650	-11.90

Tabla 5. Posiciones del plano ubicado en el cornete inferior y de la esfera con respecto al eje de coordenadas de Unity en la fosa nasal izquierda.
Fuente: Propia

El siguiente obstáculo al que se enfrenta la herramienta es el cornete medio, ver Figura 68.

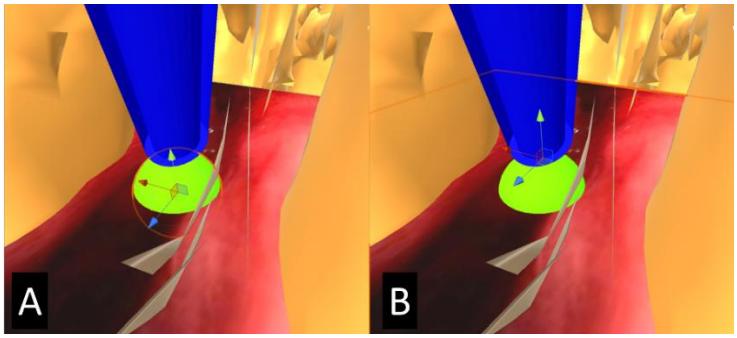


Figura 68. Eje de coordenadas (X, Y, Z) de la esfera (A) y del plano ubicado en el cornete medio (B) en la fosa nasal izquierda.
Fuente: Propia

Posiciones	X	Y	Z
PLANO CORNETE MEDIO	-10.094	2.519	-11.94
HERRAMIENTA	-10.096	2.519	-11.92

Tabla 6. Posiciones del plano ubicado en el cornete medio y de la esfera con respecto al eje de coordenadas de Unity en la fosa nasal izquierda.
Fuente: Propia

Posteriormente, la herramienta se encuentra con el hueso esfenoidal, ver Figura 69.

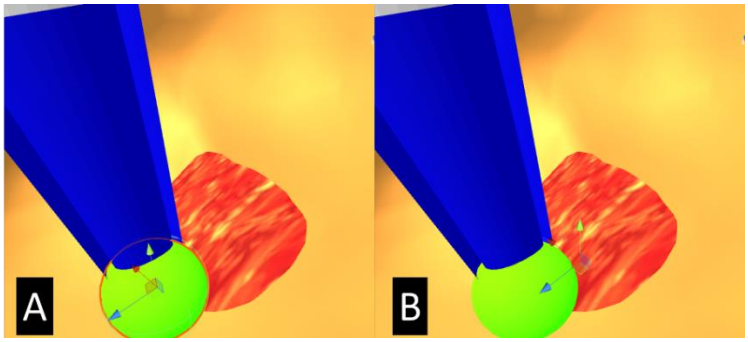


Figura 69. Eje de coordenadas (X, Y, Z) de la esfera (A) y del plano ubicado en el hueso esfenoidal (B) en la fosa nasal izquierda.
Fuente: Propia

Posiciones	X	Y	Z
PLANO HUESO ESFENOIDAL	-10.096	2.41	-11.96
HERRAMIENTA	-10.094	2.40	-11.95

Tabla 7. Posiciones del plano ubicado en el hueso esfenoidal y de la esfera con respecto al eje de coordenadas de Unity en la fosa nasal izquierda.
Fuente: Propia

El último obstáculo al que se enfrenta la herramienta del robot semiautónomo es la glándula hipofisiaria de del modelo 3D del cerebro, donde se puede visualizar el tumor dentro de ella.

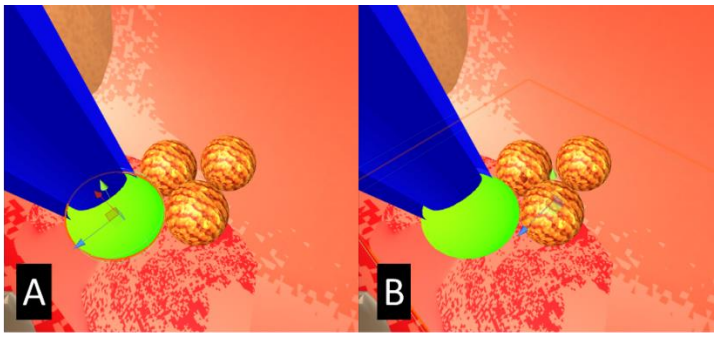


Figura 70. Eje de coordenadas (X, Y, Z) de la esfera (A) y del interior de la gandula hipofisiaria (B) en la fosa nasal izquierda.
Fuente: Propia

Posiciones	X	Y	Z
HIPÓFISIS	-10.094	2.35	-11.97
HERRAMIENTA	-10.097	2.37	-11.96

Tabla 8. Posiciones de la hipófisis y de la esfera con respecto al eje de coordenadas de Unity en la fosa nasal izquierda.
Fuente: Propia

A continuación, se presenta el esquema del movimiento del robot dentro de la cabeza, abordando cada obstaculo presentado dentro del conducto nasal hasta abordar el tumor dentro de la hipofisis (ver Figura 71) y en la Figura 72 el movimiento realizado para el abordaje de cada obstaculo con una vista exterior.

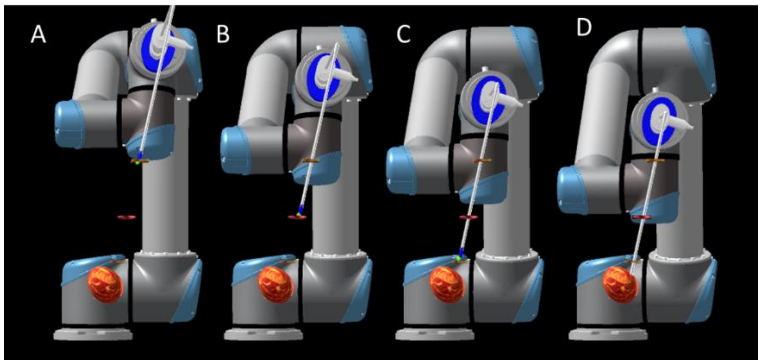


Figura 71. Ubicación de la herramienta del robot semiautónomo en cada obstáculo abordado.
Fuente: Propia

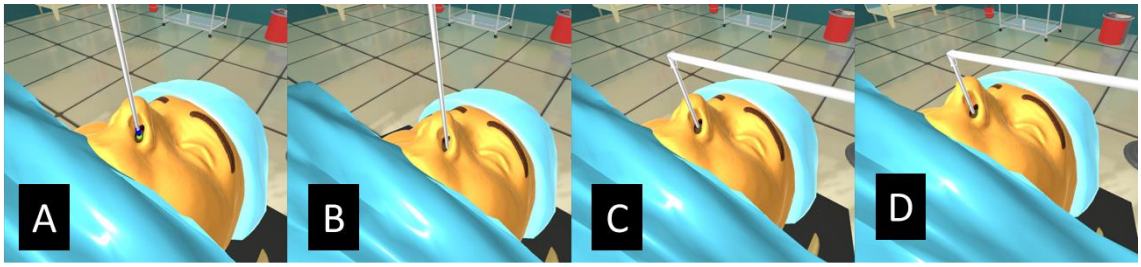


Figura 72. Vista exterior del endoscopio en cada obstáculo abordado para el robot semiautónomo.
Fuente: Propia

A: Cornete inferior.
B: Cornete medio.
C: Hueso esfenoidal.
D: Hipófisis.

- Robot UR5 en ROS

Para este robot se realizó el mismo procedimiento del robot semiautónomo con la diferencia de que la herramienta de este robot accede por la fosa nasal derecha. Como primer obstáculo se encuentra el cornete inferior (ver Figura 73).

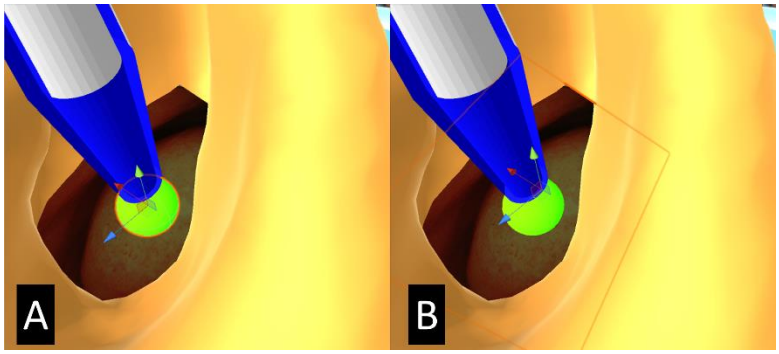


Figura 73. Eje de coordenadas (X, Y, Z) de la esfera (A) y del plano ubicado en el cornete inferior (B) en la fosa nasal derecha.
Fuente: Propia

Posiciones	X	Y	Z
CORNETE INFERIOR	-10.074	2.639	-11.906
HERRAMIENTA	-10.065	2.64	-11.912

Tabla 9. Posiciones del plano ubicado en el cornete inferior y de la esfera con respecto al eje de coordenadas de Unity en la fosa nasal derecha.
Fuente: Propia

A continuación, el obstáculo encontrado es el cornete medio, ver Figura 74.

.

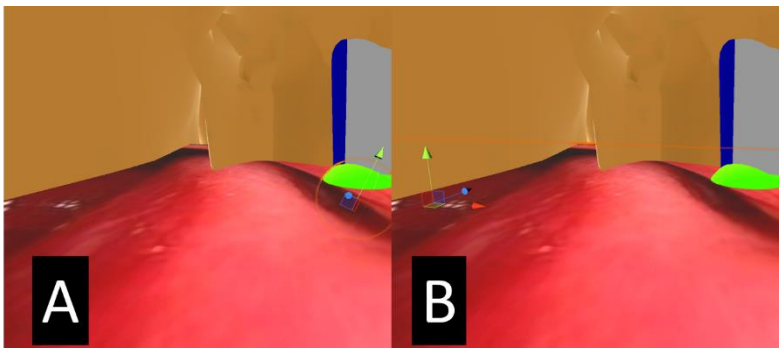


Figura 74. Eje de coordenadas (X, Y, Z) de la esfera (A) y del plano ubicado en el cornete medio (B) en la fosa nasal derecha.
Fuente: Propia

Posiciones	X	Y	Z
CORNETE MEDIO	-10.094	2.519	-11.94
HERRAMIENTA	-10.089	2.521	-11.932

Tabla 10. Posiciones del plano ubicado en el cornete medio y de la esfera con respecto al eje de coordenadas de Unity en la fosa nasal derecha.
Fuente: Propia

Posteriormente, el obstáculo al que se enfrenta la herramienta es el hueso esfenoidal, ver Figura 75.

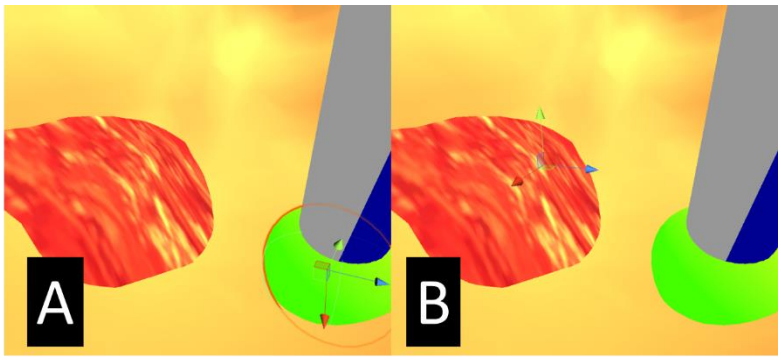


Figura 75. Eje de coordenadas (X, Y, Z) de la esfera (A) y del plano ubicado en el hueso esfenoidal (B) en la fosa nasal derecha.
Fuente: Propia

Posiciones	X	Y	Z
HUESO ESFENOIDAL	-10.096	2.41	-11.96
HERRAMIENTA	-10.085	2.407	-11.95

Tabla 11. Posiciones del plano ubicado en el hueso esfenoidal y de la esfera con respecto al eje de coordenadas de Unity en la fosa nasal derecha.
Fuente: Propia

Por último, la herramienta se encuentra con el tumor ubicado en la hipófisis, ver Figura 76.

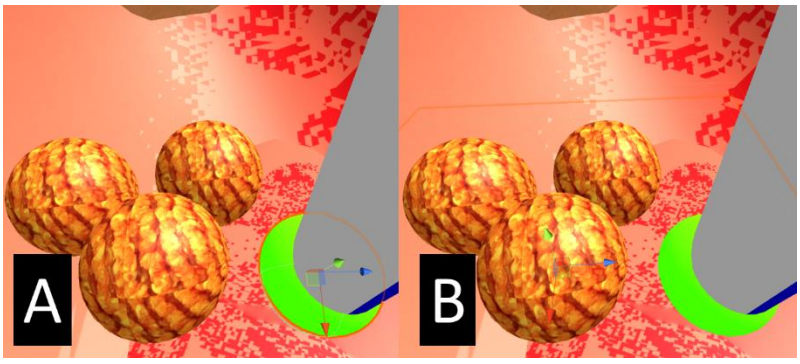


Figura 76. Eje de coordenadas (X, Y, Z) de la esfera (A) y del interior de la hipófisis (B) en la fosa nasal derecha.
Fuente: Propia

Posiciones	X	Y	Z
HIPÓFISIS	-10.094	2.35	-11.97
HERRAMIENTA	-10.091	2.362	-11.95

Tabla 12. Posiciones de la hipófisis y de la esfera con respecto al eje de coordenadas de Unity en la fosa nasal derecha.
Fuente: Propia

De la misma manera que con el robot semiautónomo, se presenta el esquema del movimiento del robot dentro de la cabeza, abordando cada obstaculo presentado dentro del conducto nasal hasta abordar el tumor dentro de la hipofisis como se ve en la Figura 77 y en la Figura 78 el movimiento realizado para el abordaje de cada obstaculo con una vista exterior.

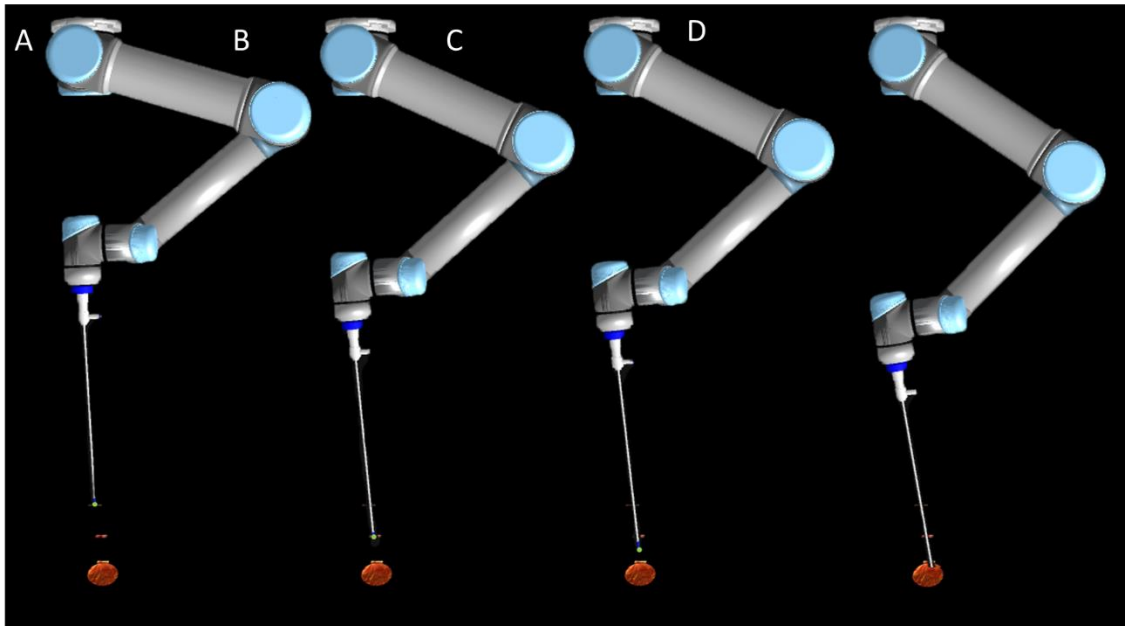


Figura 77. Ubicación de la herramienta del robot UR5 en ROS en cada obstáculo abordado.
Fuente: Propia

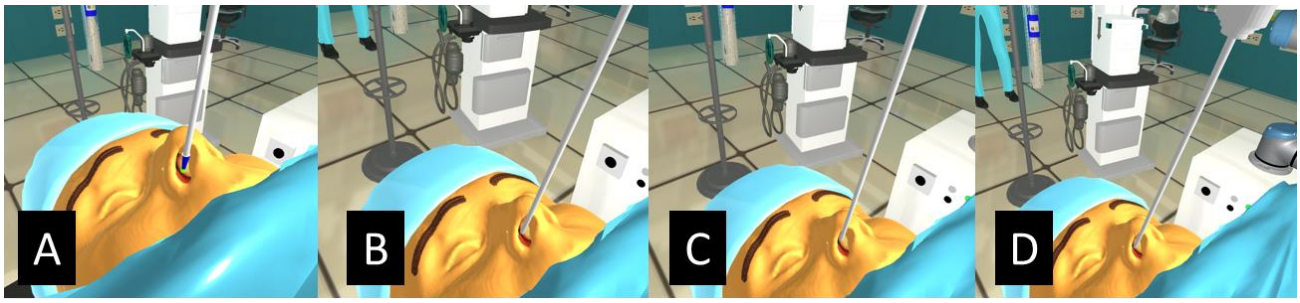


Figura 78. Vista exterior del endoscopio en cada obstáculo abordado para el robot UR5 en ROS.
Fuente: Propia

Por último, se realiza el cálculo del error de posición de la herramienta en cada posición deseada. La posición deseada es el plano (X,Y,Z) en cada obstáculo del conducto nasal, y la posición alcanzada el plano (X,Y,Z) de la esfera.

Se debe tener en cuenta que Unity trabaja con escalas y las unidades de medida pueden ser relativas, estas pueden ser tomadas como metros, centímetros, kilómetros, etc.

- Error robot semiautónomo

Posición	POSICIÓN DESEADA	POSICIÓN ALCANZADA	ERROR (%)
X	-10.099	-10.101	0.02
Y	2.629	2.650	0.7
Z	-11.90	-11.90	0

Tabla 13. Error de posición del robot semiautónomo en el cornete inferior.
Fuente: Propia

Posición	POSICIÓN DESEADA	POSICIÓN ALCANZADA	ERROR (%)
X	-10.094	-10.096	0.02
Y	2.519	2.519	0
Z	-11.94	-11.92	0.16

Tabla 14. Error de posición del robot semiautónomo en el cornete medio.
Fuente: Propia

Posición	POSICIÓN DESEADA	POSICIÓN ALCANZADA	ERROR (%)
X	-10.096	-10.094	0.02
Y	2.41	2.40	0.4
Z	-11.96	-11.95	0.08

Tabla 15. Error de posición del robot semiautónomo en el hueso esfenoidal.
Fuente: Propia

Posición	POSICIÓN DESEADA	POSICIÓN ALCANZADA	ERROR (%)
X	-10.094	-10.097	0.03
Y	2.35	2.37	0.85
Z	-11.97	-11.96	0.08

Tabla 16. Error de posición del robot semiautónomo en el tumor.
Fuente: Propia

- Error robot UR5 en ROS

Posición	POSICIÓN DESEADA	POSICIÓN ALCANZADA	ERROR (%)
X	-10.074	-10.065	0.02
Y	2.639	2.64	0.7
Z	-11.906	-11.912	0

Tabla 17. Error de posición del robot UR5 en ROS en el cornete inferior.
Fuente: Propia

Posición	POSICIÓN DESEADA	POSICIÓN ALCANZADA	ERROR (%)
X	-10.094	-10.089	0.04
Y	2.519	2.521	0.08
Z	-11.94	-11.932	0.06

Tabla 18. Error de posición del robot UR5 en ROS en el cornete medio.
Fuente: Propia

Posición	POSICIÓN DESEADA	POSICIÓN ALCANZADA	ERROR (%)
X	-10.096	-10.094	0.02
Y	2.41	2.407	0.12
Z	-11.96	-11.95	0.08

Tabla 19. Error de posición del robot UR5 en ROS en el hueso esfenoidal.
Fuente: Propia

Posición	POSICIÓN DESEADA	POSICIÓN ALCANZADA	ERROR (%)
X	-10.094	-10.091	0.02
Y	2.35	2.362	0.5
Z	-11.97	-11.95	0.16

Tabla 20. Error de posición del robot UR5 en ROS en el tumor.
Fuente: Propia

De lo anterior se puede evaluar que los errores obtenidos en cada posición son mínimos, aproximadamente 0. Con lo que se puede concluir que el paso de las herramientas por el conducto nasal se está realizando de manera adecuada en cada uno de los obstáculos que se encuentran en el camino, sin causar ningún daño interno y sin que la herramienta se desvíe hacia el exterior de este.

En este caso se observa que el robot más preciso es el robot UR5 en ROS, con lo que se tiene que realizar el control desde esta herramienta permite obtener errores casi nulos, brindando seguridad y precisión a los cirujanos en la intervención.

6. CONCLUSIONES Y TRABAJOS FUTUROS

6.1 Conclusiones

- Se propuso un ambiente quirúrgico virtual diseñado para una intervención de cirugía endonasal basada en robots, en el cual se buscó una configuración adecuada con el fin de adaptar el sistema robotizado al ambiente. El ambiente conto con diversas características básicas de un simulador virtual de cirugía endonasal tales como el Instrumental quirúrgico, los equipos médicos, el personal médico y también se involucró la anatomía correcta del paciente la cual debió verse de una manera clara y realista. Pero también se tuvo que tener en cuenta que el ambiente incluía dos robots que debían adaptarse a la ergonomía de la intervención sin generar obstáculos, debían actuar de manera que proporcionaran seguridad y precisión en los movimientos realizados para evitar daños en las fosas nasales y sus desplazamientos debían ser acordes con la información obtenida en el preoperatorio las cuales son características fundamentales del ambiente con el sistema robótico incluido.

- Se diseñó un quirófano virtual haciendo uso de la herramienta Unity 3D ya que esta posee una interfaz de fácil uso pero con una gran potencia en los entornos que se desee utilizar. Se diseñó la estructura del quirófano en un software llamado SketchUp el cual brinda una facilidad de uso en cuanto a diseños arquitectónicos, a partir del modelo de la estructura se anexaron los modelos de los equipos médicos realizados en una herramienta Blender que permite la importación a Unity de los objetos diseñados en ella y para la importación de los humanoides como personal médico se utilizó el software llamado MakeHuman, finalmente se obtuvo la configuración final de prototipo de quirófano, realizando variaciones de posición en el sistema robótico con el fin de evitar colisiones con su alrededor.

- Se incluyó un módulo de comunicación entre Unity y ROS, llamado Rosbridge server el cual permite que ROS se comunique con programas externos a este, el movimiento del robot se realiza en ROS y se visualiza en Unity de manera instantánea. Esto se realizó con el objetivo de que se pueda compartir información con los grupos del proyecto español CRANEEAL, lo que permite que haya estandarización en los procesos y que se pueda seguir trabajando en este proceso de una manera más fácil.

- En el ambiente virtual se evaluaron dos trayectorias neuronales, para el robot importado desde ROS la trayectoria fue evaluada manualmente en la fosa nasal derecha del modelo del paciente y para el robot semiautónomo se evaluó una trayectoria punto a punto manejada directamente desde Unity en la fosa nasal

izquierda del modelo. Para la evaluación de estas trayectorias se analizó la anatomía de las fosas nasales del modelo del paciente, con el fin de diseñar un conducto nasal por donde debía conducirse la herramienta en la intervención. Se obtuvo un error para las dos trayectorias cuyos valores fueron aproximadamente cero, indicando que los robots poseen un alto grado de precisión ya que la herramienta que el robot manejo no se desvió del conducto afectando zonas externas.

- Este proyecto brinda un aporte al proyecto español CRANEEAL, cuyo propósito fue realizar el primer prototipo de simulación con el fin de que pueda ser utilizado a futuro como base para la realización de la simulación real de la intervención, también brinda una ayuda para adaptar el sistema robótico a la ergonomía de la intervención real para evitar inconvenientes futuros.

6.2 Trabajos futuros

- Implementar un prototipo de simulación física, que permita verificar los resultados obtenidos.

- Implementar este prototipo de simulación para diferentes tipos de patologías e intervenciones.

- Incluir al sistema realidad virtual haciendo uso de la herramienta Unity 3D.

- Realizar diferentes pruebas de planificación de trayectorias utilizando el robot UR5 real.

- Realizar el diseño de un endoscopio rígido con ángulo de 90º, con el fin de realizar la intervención real con el sistema robótico.

REFERENCIAS BIBLIOGRÁFICAS:

[1] F. Flores, “Implementación de partículas físicas en un ambiente gráfico de cirugía de próstata,” UNAM, vol. 1, pp. 1–14, 2011.

[2] D. M. Gaba, “The future vision of simulation in health care,” Quality and safety in Health care, vol. 13, no. 1, pp. i2–i10, 2004.

[3] Y. Okuda, E. O. Bryson, S. DeMaria, L. Jacobson, J. Quinones, B. Shen, and A. I. Levine, “The utility of simulation in medical education: What is the evidence?” Mount Sinai Journal of Medicine, vol. 76, no. 4, pp. 330–343, 2009.

[4] J. García, M. Arias, and É. Valencia, “Diseño de prototipo de simulador para entrenamiento en cirugía laparoscópica,” Revista Ingeniería Biomédica, vol. 5, no. 9, pp. 13–19, 2011.

[5] J. L. Molina, E. Silveira, D. Heredia, D. Fernández, L. Bécquer, T. Gómez, Y. González, and M. Castro, “Los simuladores y los modelos experimentales en el desarrollo de habilidades quirúrgicas en el proceso de enseñanza-aprendizaje de las ciencias de la salud,” Revista Electrónica de Veterinaria, vol. 13, no. 6, pp. 1–23, 2012.

[6] G. Galisteo, J. D. R. Samaniego, B. González, and S. G. Baquero, “Aprendizaje de la cirugía laparoscópica en pelvitrainer y en simuladores virtuales,” Actas Urológicas Españolas, vol. 30, no. 5, pp. 451–456, 2006.

[7] J. A. Millán, “Reconocimiento gestual para interacción humano-robot basado en ROS”, Tesis de pregrado en Ingeniería en Tecnologías Industriales, Universidad de Sevilla, Sevilla, España, 2015.

[8] L. Rodríguez, “Desarrollo de un sistema de teleoperación maestro-esclavo utilizando ROS y Matlab (Aplicación a la teleoperación del robot ATENEA)”, Tesis de pregrado en Ingeniería Electrónica y Automática Industrial, Universidad Miguel Hernández de Elche, Elche, España, 2016.

[9] F. I. Aranda López, M. Niveiro de Jaime, G. Peiró Cabrera, C. Alenda González, and A. Picó Alfonso, “Adenoma hipofisario: estudio de la actividad proliferativa con Ki-67,” Rev. Española Patol., vol. 40, no. 4, pp. 225–231, 2007.

[10] N. Wohllk and R. Díaz, “Neoplasia Endocrina Multiple,” vol. 24, no. 5, pp. 778–783, 2013.

[11] P. Ajler, S. Hem, E. Goldschmidt, F. Landriel, A. Campero, C. Yampolsky and A. Carrizo, “Cirugía transnasal endoscópica para tumores de hipófisis,” Surgical Neurology International, vol. 3, no. 7, p. 389, 2012.

[12] P. Cappabianca and E. De Divitiis, “Back to the Egyptians: Neurosurgery via the nose – A five-thousand year history and the recent contribution of the endoscope,” Neurosurg Rev, vol. 30, no. 1, pp. 1–7, 2007.

[13] F. M. del Castillo, A. Jurado, P. López and A. De la Riva-Aguilar, “Cirugía endoscópica nasal de procesos selares,” Neurocirugia, vol. 14, no. 6, pp. 512–516, 2003.

[14] P. Cappabianca and L. M. Cavallo, “Endoscopic pituitary and skull base surgery.” Napoly, Italy, p. 91.

[15] Virtual Colors Blog Mayo 27, 2017 “Novedades de Unity en la educación - Virtual Colors Blog”, [online], Available: http://virtualcolors.com/blog/novedades-de-unity-en-la-educacion/

[16] A. Constaín, K. Torres, J. Arango, and A. Vivas, “Modelado, identificación paramétrica y control del robot scorbot-er 5 plus,” in VIII Congreso de la Asociación Colombiana de Automática, Cartagena, Colombia, 2009.

[17] Unity, “Unity 3d games,” [online], Available: http://www.unity3dgames.eu [Accessed: 12-2018]

[18] T. D. Moreno, “Diseño y construcción de un prototipo de simulador con realidad virtual para cirugía laparoscópica” , Tesis de pregrado en Ingeniería Mecatrónica, Universidad Autónoma de México, Ciudad de México, México, 2017.

[19] K. Storz, “Simuladores,” storz , El Mundo de La Endoscopia., vol. 1, pp. 85, 2018.

[20] G. Echegaray, I. Herrera, I. Aguinaga, C. Buchart, and D. Borro, “A brain surgery simulator,” *IEEE computer Graphics and Applications*, vol. 34, no. 3, pp. 12–18, 2014.

[21] J. Martínez, I. Muñoz, D. Pineda, R. Avendaño, J. Domínguez, and R. Alfonso, “Uso de simuladores para entrenamiento en neurocirugía: cambio en el paradigma de entrenamiento quirúrgico,” *Anales Médicos*, vol. 62, no. 2, pp. 106–113, 2017.

[22] P. L. Solarte, J. M. Sabater, E. M. Aguilar, O. A. Vivas, V. Samper, and J. M. Vicente, “Uso de realidad aumentada como apoyo a un sistema de navegación en neurocirugía”, Actas de las XXXIX Jornadas de Automática, Badajoz, 2018.

[23] E. Bauzano, A. Fernández, M. C. López, J. Klein, A. Rentería, and V. F. Muñoz, “Integración de dispositivos en un robot quirúrgico teleoperado mediante ROS,” Actas de las XXXVI Jornadas de Automática, Bilbao, 2015.

[24] J. C. Cobos, and M. Abderrahim, “Sistema de Supervisión no invasivo de Signos Vitales con un robot”, XXXVI Jornadas de Automática, Bilbao, 2015.

[25] ICE, “Proyecto BROCA”, [Online], Available: http://www.proyecto-broca.es/. [Accessed: 12-2018]

[26] Universal Robots, “Robots colaborativos UR5”, [Online], Available: http://www.infoplc.net/index.php?option=com_k2&view=item&id=102244:universal-robots-robomatica-robots-colaborativos-UR5-ur10&Itemid=2. [Accessed: 11-2018]

[27] A. García, N. Garcia, and J. Sabater, “Automatic Detection of Surgical Gauzes Using Computer Vision”, 23rd Mediterranean Conference on Control and Automation (MED), Torremolinos, España, 2015.

[28] “Proyecto craneeal,” [online], Available: http://www.uma.es/medical-robotics/info/107696/proyecto-dpi2016-80391-C3-1-R/, [Accessed: 12-2018]

[29] Universal Robots, “Manual de Usuario UR5,” [Online], Available: https://s3-eu-west-1.amazonaws.com/ur-support-site/22085/UR5_User_Manual_es_Global.pdf [Accessed: 12-2018]

[30] Universal Robots, “Caja de control,” [Online], Available: https://www.universal-robots.com/media/1801300/esp_semea_199912_ur5_tech_spec_web_a4.pdf [Accessed: 12-2018]

[31] “Unity 3d”, [Online], Available: https://unity3d.com/es, [Accessed: 12-2018]

[32] “SketchUp”, [online], Available: http://www.sketchup-la.com/?geocode=CO&gclid=EAIaIQobChMI0PnB_aC14AIVRh6GCh0-7AihEAAYASAAEgIHTPD_BwE, [Accessed: 12-2018]

[33] “Blender”, [online], Available: https://www.blender.org/, [Accessed: 12-2018]

[34]” Makehuman”, [online], Available: http://www.makehumancommunity.org/, [Accessed: 12-2018]

[35] “3d warehouse”, [online], Available: https://3dwarehouse.sketchup.com/, [Accessed: 12-2018]

[36] B. Estebanez, “Diseño e implantación de un sistema multimodal para un asistente robótico.” Ph.D. dissertation, Universidad de Málaga, España, 2013.

[37] “Wiki ros”, [Online], Available: http://wiki.ros.org [Accessed: 12 -2018].

[38] M. Díaz, and J. Escobar, “Interfaz Háptica Tipo Guante Con Realimentación Vibratoria”, Tesis de pregrado en Ingeniería Automática Industrial, Universidad del Cauca, Popayán, Colombia, 2013.

[39] J. Gutiérrez, “Introducción a ROS en Raspberry Pi,” Tesis de Máster Universitario Ingeniería de Telecomunicación, Universitat Oberta de Catalunya, Barcelona, Cataluña, España, 2017.

[40] ROS, “Concepts”, [Online], Available: http://wiki.ros.org/ROS/Concepts [Accessed: 12-2018].

[41] ROS, “CATKIN”, [Online], Available: http://wiki.ros.org/catkin/conceptual_overview [Accessed: 11-2018].

[42] ROS, “ROS Industrial”, [Online], Available: http://wiki.ros.org/Industrial. [Accessed: 11-2018]

[43] E. Peña, “Interfaz inmersiva para misiones robóticas basada en realidad virtual”, Tesis de pregrado en Ingeniería en Tecnologías Industriales, Universidad Politécnica de Madrid, Madrid, España, 2017.

[44] O. Ben, C. Evans, and B Ingerson. “YAML Ain’t Markup Language (YAML ^TM)”, Versión 1.1. yaml.org, Tech. Rep, 2005.

[45] D. Crockford. “The apliccation/json media type for javascript object notation(json)” [Online], Available: https://tools.ietf.org/pdf/rfc4627.pdf [Accessed: 11-2018]

[46] ROS, “Rviz”, [Online], Available: http://wiki.ros.org/rviz [Accessed: 12-2018].

[47] R. Merino, “Creación de modelo URDF del robot MANFRED”, Tesis de pregrado en Ingeniería Técnica Industrial, Universidad Carlos III de Madrid, Madrid, España, 2017.

[48] ROS, “joint_state_publisher”, [Online], Available: http://wiki.ros.org/joint_state_publisher [Accessed: 12-2018].

[49] “ros-sharp”, [Online], Available: https://github.com/siemens/ros-sharp [Accessed: 12-2018].

ANEXOS

A1. Instalación de Unity

La instalación de Unity 3D se realiza en el sistema operativo Windows, en este caso Windows 10. Los pasos de la instalación son:

1. El primer paso para llevar a cabo la instalación de Unity es dirigirse a la página web oficial de Unity [31] , se da click en “Obtener Unity”, donde este redireccionará a una página que muestra la última versión de Unity. Ver Figura A1.

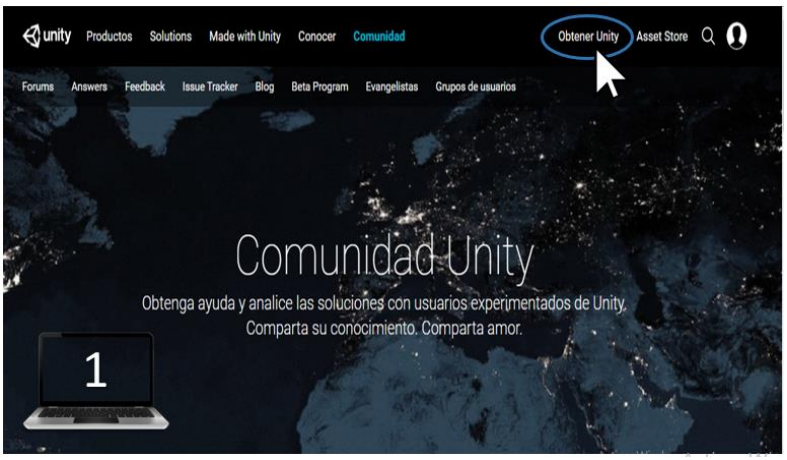


Figura A1: Página principal de Unity.
Fuente: Propia

2. La página siguiente muestra los tres planes de licencia que ofrece Unity. Para principiantes la mejor opción es el plan personal la cual es la única que es gratuita y posee características similares que los otros. Ver Figura A2.

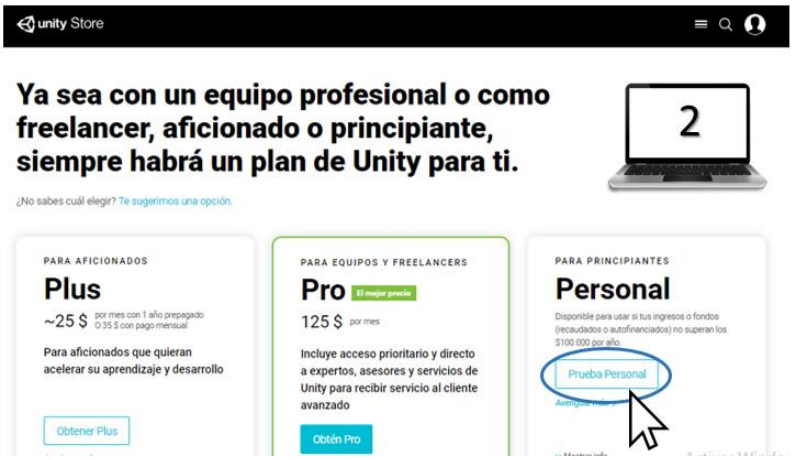


Figura A2. Planes de licencia de Unity.
Fuente: Propia

3. Al hacer click en “Prueba Personal” redireccionará a una página donde se debe aceptar los términos. Ver Figura A3.

4. Después se debe hacer click en la descarga del instalador, ya sea para Windows o para el sistema operativo que se necesite. Ver Figura A3.

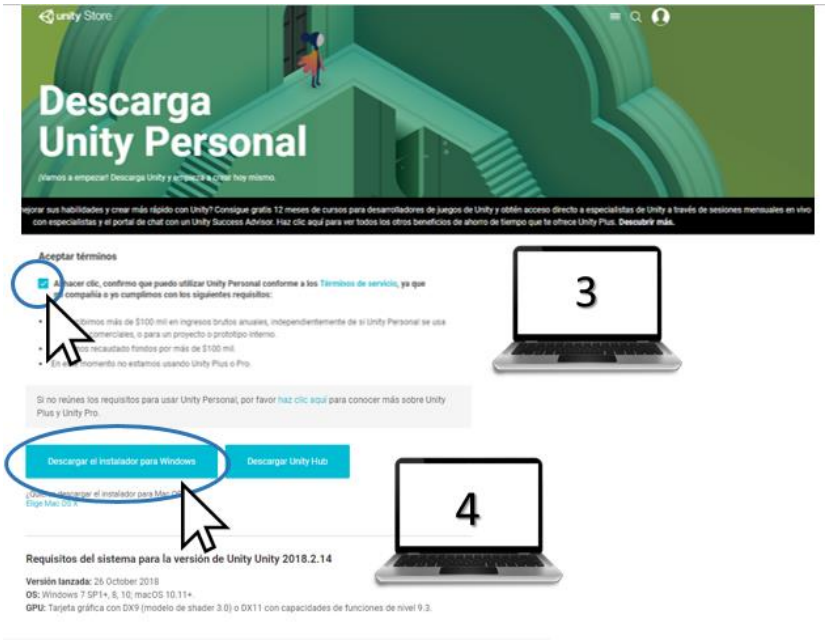


Figura A3. Términos y condiciones de descarga.
Fuente: Propia

5. Ejecuta el asistente de descarga de Unity. Este archivo no es el instalador de Unity, sino un asistente el cual descargará el instalador de Unity y lo ejecutará. Ver Figura A4.

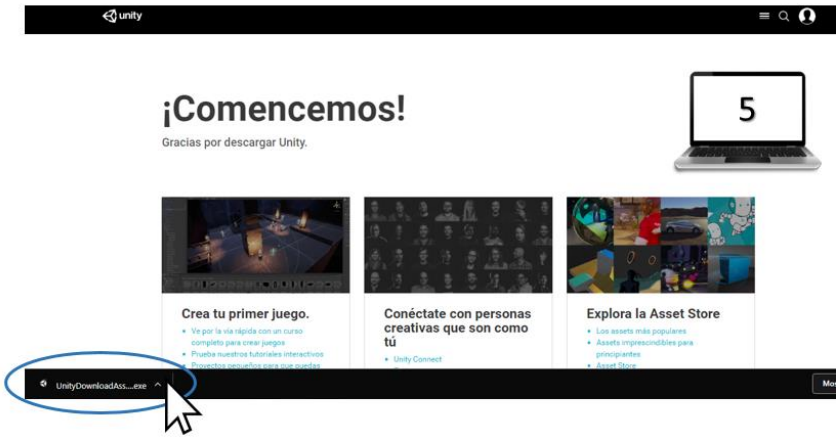


Figura A4. Asistente de descarga Unity.
Fuente: Propia

6. Las imágenes que se presentan a continuación (A5-A8), son los pasos a seguir después de ejecutar el instalador de Unity. Realizar todo el procedimiento hasta llegar a la finalización donde ya se obtendrá el motor de Unity en el computador.

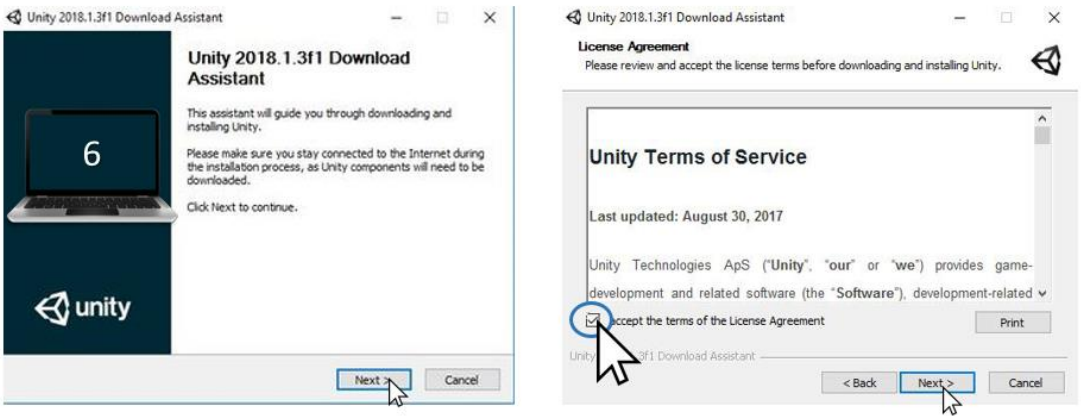


Figura A5. Paso 1 y 2 de la ejecución del software.
Fuente: Propia

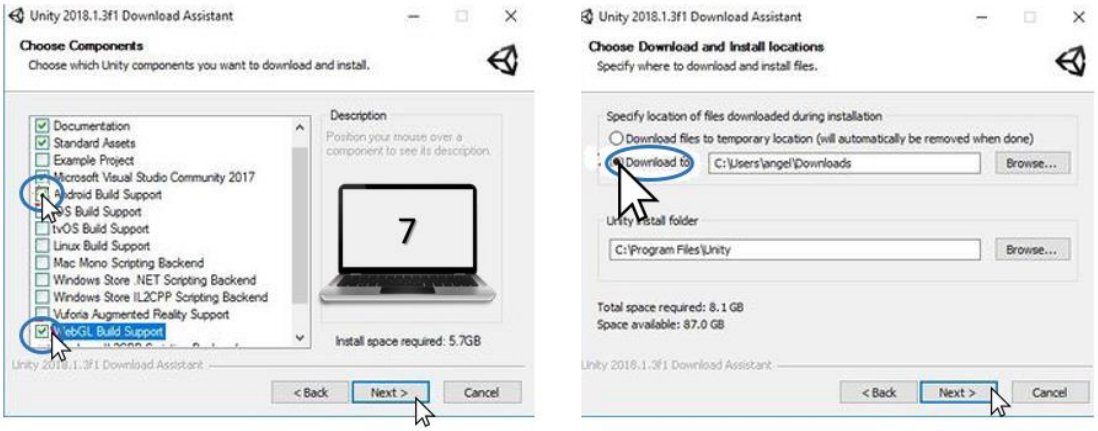


Figura A6. Paso 3 y 4 de la ejecución del software.
Fuente: Propia

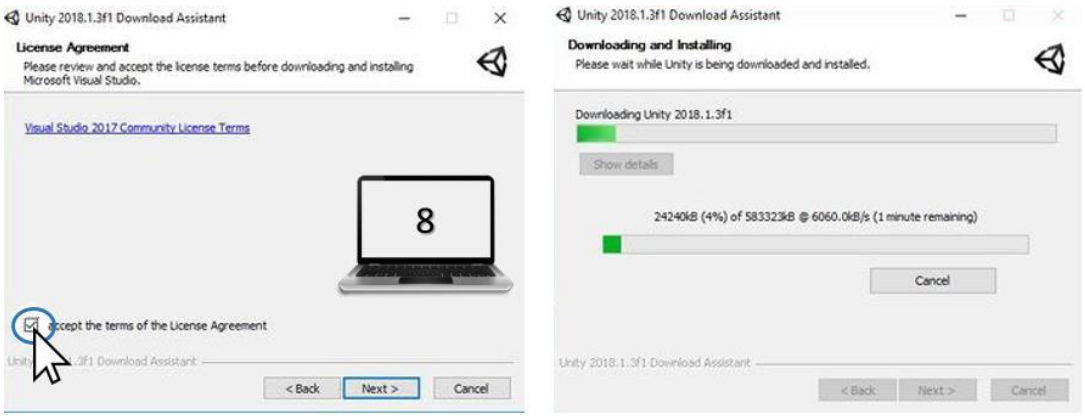


Figura A7. Paso 5 y 6 de la ejecución del software.
Fuente: Propia

Figura A8. Paso 7 y 8 de la ejecución del software.
Fuente: Propia

A1.1 Primer acceso

Una vez instalado el programa, la primera ventana que se encontrará será la de creación de un nuevo proyecto o un proyecto ya creado como se ve en la Figura A9.

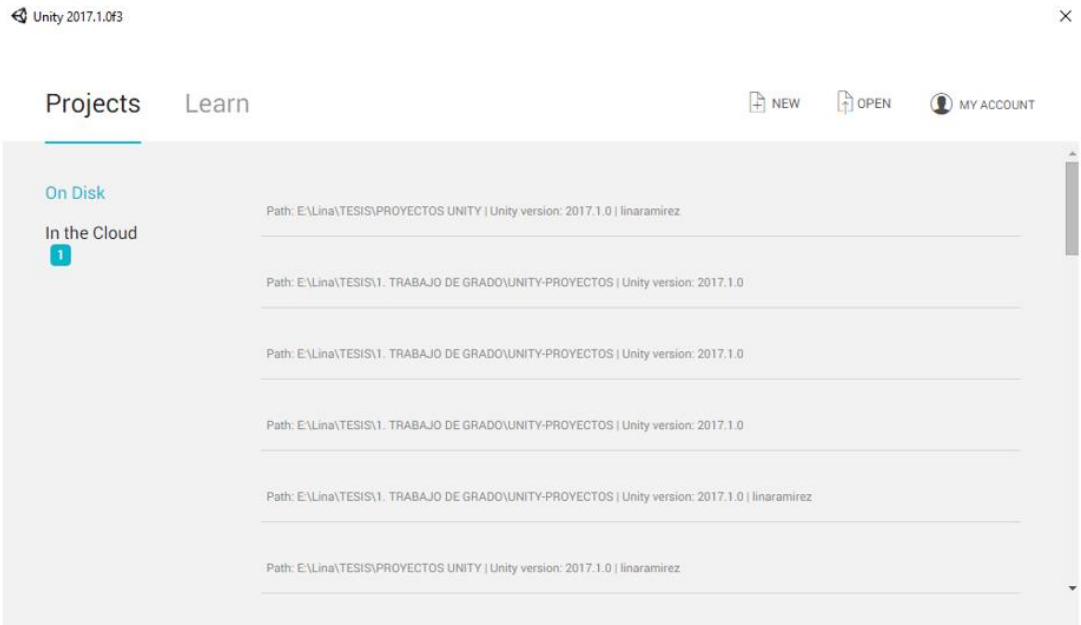


Figura A9. Primera ventana de Unity.
Fuente: Propia.

Para crear un nuevo proyecto es necesario clickear en la pestaña “NEW”, realizado el procedimiento anterior, se despliega una pantalla donde se establece el nombre, la ubicación y el entorno deseado (2D, 3D) del proyecto que se creará tal como se muestra en la Figura A10.

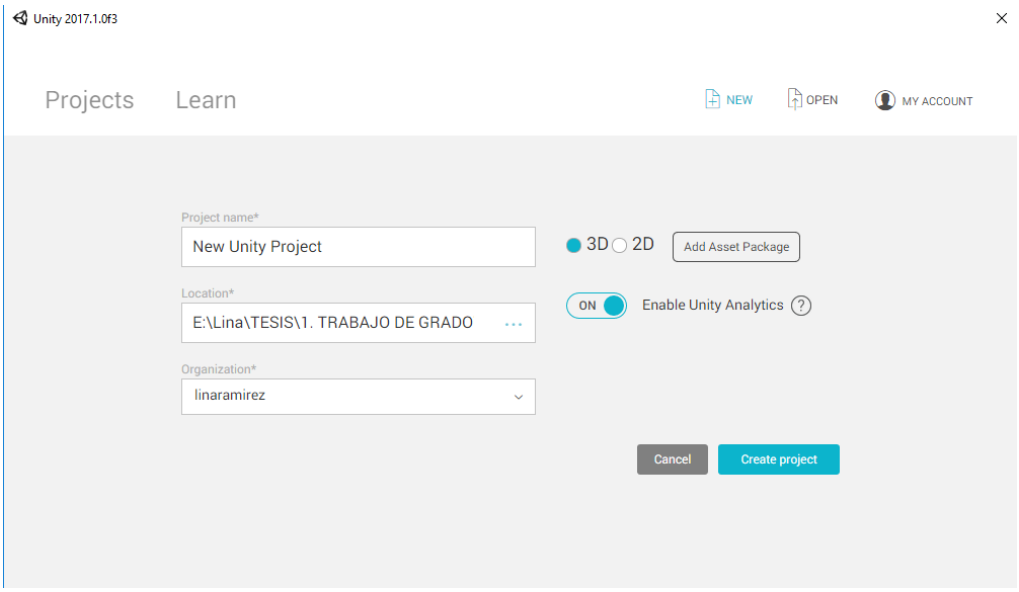


Figura A10. Creación del proyecto.
Fuente: Propia.

Cuando el procedimiento anterior se encuentre realizado, la primera ventana que se observa es la de la Figura A11, a partir de este punto se empieza con el desarrollo de la aplicación que se desee.

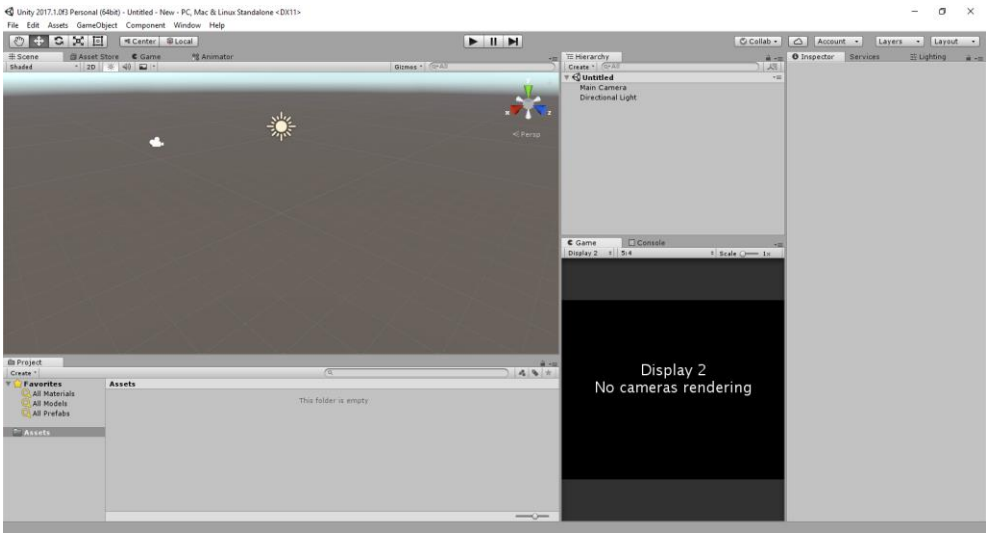


Figura A11. Primera ventana del proyecto.
Fuente: Propia

A1.2 Interfaz de usuario.

La interfaz de usuario se divide principalmente en 5 secciones importantes, las cuales se pueden observar en la Figura A12 describiéndose cada una posteriormente a ella.

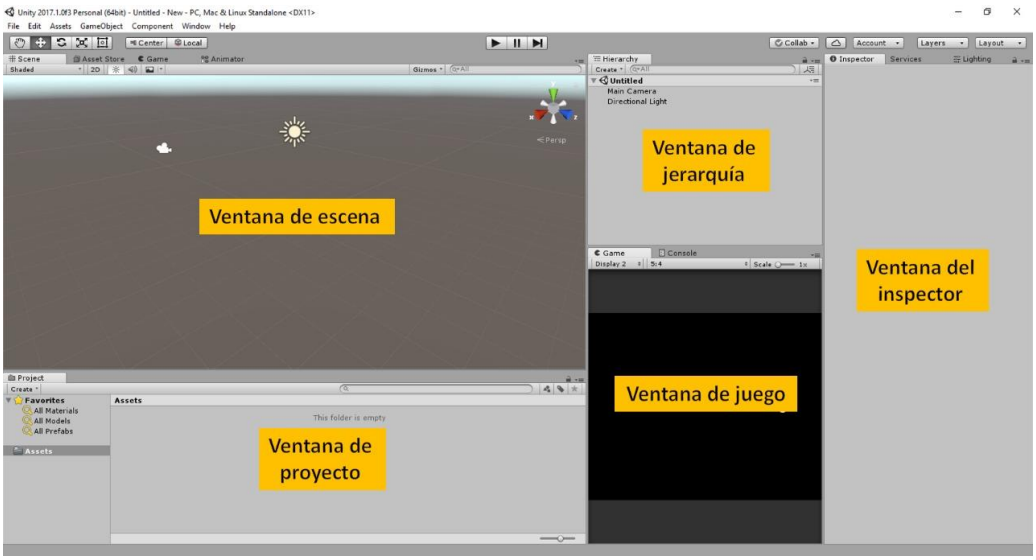


Figura A12. Interfaz de usuario de Unity.
Fuente: Propia

A1.3 Ventana de escena

La *scene* (escena) es la ventana donde se realizará visualmente la aplicación ya sea 3D o 2D. En esta ventana se pueden realizar distintos tipos de modificaciones a los objetos que se encuentran en ella, como mover, rotar y escalar. En esta ventana es donde se editan las luces, las cámaras, los terrenos, el canvas y demás objetos permitiendo así tener una visualización muy similar al que se obtendrá al finalizar la aplicación o juego.

A1.4 Ventana de jerarquía

La ventana de *Hierarchy* es una representación textual de cada objeto encontrado en la escena, una herramienta útil para facilitar al usuario la selección de cada objeto.

A1.5 Ventana de inspector

El inspector muestra las características añadidas a un objeto seleccionado y permite así la modificación de estas según la necesidad del usuario.

A1.6 Ventana de juego

El *game* (la cámara) muestra la aplicación al ser compilada. Así es posible probar diferentes resoluciones y adecúa la visualización final.

Para el acceso a esta ventana, basta con dar "*play*" al icono de los controladores de reproducción y ejecución del proyecto, los cuales pertenecen a la barra de tareas. Ver Figura A13.

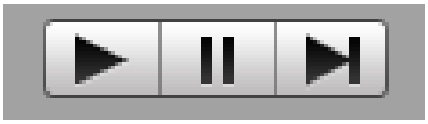


Figura A13. Controladores de ejecución del proyecto.
Fuente: Propia.

A1.7 Ventana de proyecto

Muestra la organización del proyecto en Unity, donde se visualizan *Assets* que incluyen recursos a usar en las escenas, como los materiales, las texturas, los *scripts*, entre otros.

Dentro de esta ventana existen diferentes carpetas, con el fin de darle una mejor organización a los proyectos que se crean, también cuenta con un buscador que permite al usuario acceder de forma más rápida a los componentes deseados.

A1.8 Botones de control

En la barra de control se encuentran 5 botones elementales los cuales se pueden apreciar en la Figura A14.

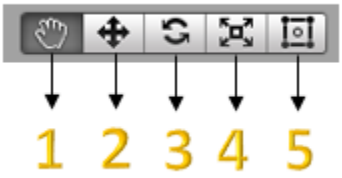


Figura A14. Botones de control.
Fuente: Propia.

A1.8.1 *Hand tool*

Permite el movimiento alrededor de la escena, se puede acceder a este cliqueando sobre él o presionando la tecla Q.

La combinación del botón *hand too*l con otras teclas permite realizar distintos movimientos como:

- *Hand tool* + Shift: Permite rotar la escena.
- *Hand tool* + Alft: Realiza movimientos a una mayor velocidad.
- *Hand tool* + Ctrl + Scroll (Mouse): Permite hacer zoom sobre la escena.

A1.8.2 *Translate tool*

Permite el movimiento de un objeto en los planos X, Y o Z, se puede acceder cliqueando en él o presionando la tecla W.

A1.8.3 *Rotate tool*

Permite el movimiento rotacional en los ejes X, Y o Z, se puede acceder haciendo click en él o presionando la tecla E.

A1.8.4 *Scale tool*

Este botón permite dos diferentes movimientos, el primero es escalar un objeto seleccionado según la necesidad y el segundo movimiento es que junto con el *scroll* del mouse hace un zoom de la escena. Se puede acceder mediante el teclado presionando la tecla R.

A1.8.5 *Position and scale tool*

El último botón es el encargado de modificar tanto la posición como la escala de un objeto seleccionado. La forma de acceder a él por medio del teclado es presionando la tecla T.

A1.9 *Gameobject*

Los *gameobject*, son conocidos como los objetos del juego. Se considera *gameobject* a cualquier elemento que se encuentre en la escena. Unity cuenta con una variedad de objetos predefinidos los cuales pueden agregarse dando click en la pestaña "*GameObject*", así como se ve en la Figura A15.

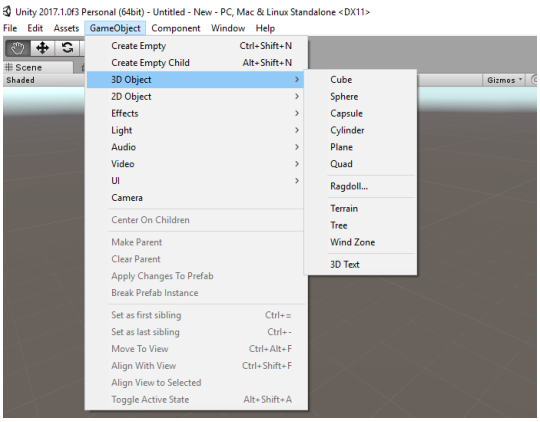


Figura A15. Creación de *gameobject.*
Fuente: Propia.

A.10 Programación

Unity emplea un lenguaje de programación orientado a objetos, este motor gráfico permite el desarrollo de proyectos en los lenguajes tales como C#, *UnityScript* o *Boo*.

UnityEngine es una librería con la que cuenta Unity, la cual facilita la programación de los objetos deseados. Posee una amplia cantidad de funciones y clases que permite la facilidad de programar además permite el uso de diferentes tipos de variables.

Cuando el *script* se encuentra programado y compilado se arrastra desde la ventana del proyecto hasta el inspector donde se encuentran las características del objeto que se desea programar. Si las variables del programa se han declarado públicas se pueden manipular los atributos desde el editor mientras el programa se encuentre en ejecución.

A2. Instalación de ROS

La distribución de ROS que se instalará en este caso es ROS 16.04 Kinetic.
Los pasos a seguir son:

1. Configuración de repositorios de Ubuntu: Para ello se utiliza la aplicación “Gestor de paquetes synaptic” de Ubuntu. Si esta aplicación no se encuentra, se instala desde el “Centro de software de Ubuntu”. El proceso correspondiente a la configuración de los repositorios de Ubuntu se realiza para permitir los repositorios de tipo *restricted, universe* y *multiverse*.

2. Configuración del archivo sources.list: Ejecutar en el terminal el siguiente comando:

```
$ sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc) main" > /etc/apt/sources.list.d/ros-latest.list'
```

3. Configuración de las llaves: Con esto nos aseguramos de que el paquete que se está descargando es correcto y no uno corrupto. Se debe ejecutar:

```
$ sudo apt-key adv --keyserver hkp://ha.pool.sks-keyservers.net:80 --recv-key 421C365BD9FF1F717815A3895523BAEEB01FA116
```

4. Actualizar todos los paquetes: Se debe ejecutar el siguiente comando:

```
$ sudo apt-get update
```

5. Instalación completa de ROS, que incluye rqt, rviz, bibliotecas genéricas, simuladores 2D/3D. Utilizar en el terminal:

```
$ sudo apt-get install ros-kinetic-desktop-full
```

6. Inicialización del Rosdep: Este permite instalar fácilmente dependencias del sistema para los archivos fuente que se desean compilar y es requerido para ejecutar algunos componentes importantes de ROS. Ejecutar los siguientes comandos:

```
$ sudo rosdep init
$ rosdep update
```

7. Configuración del entorno: Se deben agregar las variables de entorno de ROS automáticamente a la sesión de *bash* cada vez que se abre una nueva terminal.

```
$ echo "source /opt/ros/kinetic/setup.bash" >> ~/.bashrc
source ~/.bashrc
```

Si solo se desea cambiar el entorno de la terminal actual se usa:

```
$ source /opt/ros/kinetic/setup.bash
```

Este proceso permite instalar varias distribuciones de ROS (por ejemplo, Kinetic y Lunar) en el mismo equipo y cambiar entre ellas.

8. Obtener RosInstall: Permite descargar fácilmente muchos árboles de código fuente de paquetes de ROS con un solo comando. Ejecutar en la terminal:

```
$ sudo apt install python-rosinstall python-rosinstall-generator python-wstool build-essential
```

A2.1 Instalación de Rosbridge

Para instalar Rosbridge se debe ejecutar en la terminal el siguiente comando:

```
$ sudo apt-get install ros-<rosdistro>-rosbridge-server
```

La distribución en este caso es Kinetic, por lo que se tiene:

```
$ sudo apt-get install ros-kinetic-rosbridge-server
```

A2.2 Instalación del paquete UNIVERSAL_ROBOT

Para instalar este paquete se ejecuta en la terminal:

```
$ sudo apt-get install ros-kinetic-universal-robot
```

A3. Manual de usuario para el ambiente

Descripción

Este ambiente quirúrgico cuenta con una cierta cantidad de pasos a realizar en el momento de la intervención virtual, tales como la aplicación de la epinefrina que evitará el sangrado masivo en el interior de la nariz, seccionar con el bisturí los cornetes, con el fin de abrir paso hasta el tumor por la vía nasal, el taponamiento nasal al concluir la cirugía y la trayectoria neurológica del robot semiautónomo.

Para la realización de los pasos mencionados anteriormente, se desarrollaron códigos C# en el motor de Unity para accionar correctamente los pasos por medio del teclado, enviando así la información al ambiente y ejecutando el proceso necesario en el momento indicado.

A3.1 Aplicación de la epinefrina

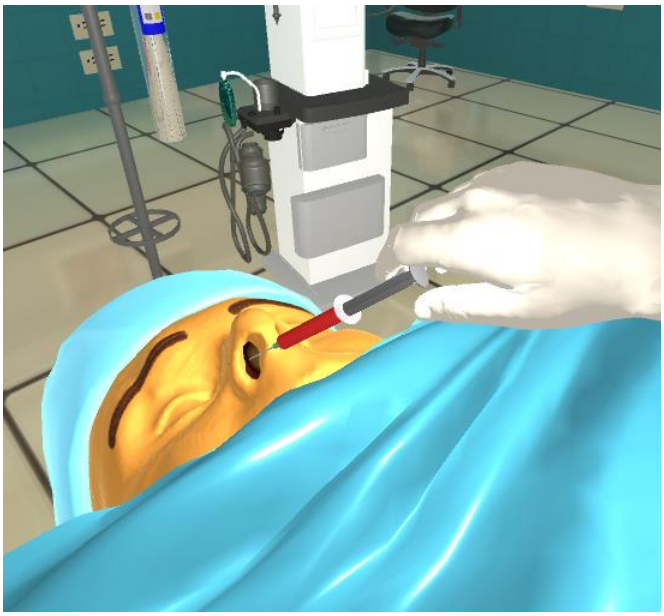


Figura A16. Aplicación de la epinefrina en el ambiente quirúrgico.
Fuente: Propia

A3.1.1 Código implementado para la simulación de la aplicación de la epinefrina

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class epinefrina : MonoBehaviour {

    public GameObject epi;
    public GameObject mano;


public void Start () {
        epi = GameObject.Find ("epi");
        mano = GameObject.Find ("mano");

    }

    public void Update () {
        if (Input.GetKeyDown ("p"))
            epi.SetActive (true);
        if (Input.GetKeyDown ("p"))
            mano.SetActive (true);
        if (Input.GetKeyDown ("o"))
            epi.SetActive (false);
        if (Input.GetKeyDown ("o"))
            mano.SetActive (false);

    }
}
```

A3.1.2 Modo de uso

Cuando la escena se encuentra ejecutada, lo primero que se debe hacer antes de intervenir con los robots es la simulación de la aplicación de la epinefrina, para ello al hundir la tecla “P” del teclado la epinefrina y la mano del cirujano aparecerán en la escena inmediatamente y al hundir la tecla “O” desaparecerán.

A3.2 Seccionar los cornetes

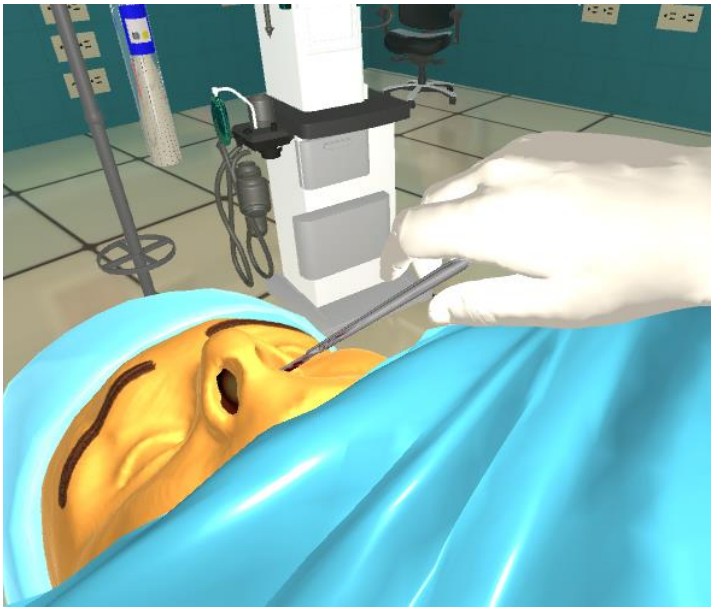


Figura A17. Seccionar los cornetes en el ambiente quirúrgico.
Fuente: Propia

A3.2.1 Código implementado para la simulación de seccionar los cornetes

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class bisturi : MonoBehaviour {

    public GameObject Scal;
    public GameObject mano;

    public void Start () {
        Scal = GameObject.Find ("Scal");
        mano = GameObject.Find ("mano");

    }

    public void Update () {
        if (Input.GetKeyDown ("m"))
```

```
            Scal.SetActive (true);
        if (Input.GetKeyDown ("m"))
            mano.SetActive (true);
        if (Input.GetKeyDown ("n"))
            Scal.SetActive (false);
        if (Input.GetKeyDown ("n"))
            mano.SetActive (false);

    }
}
```

A3.2.2 Modo de uso

Cuando la aplicación de la epinefrina ya se encuentre ejecutada se procede a simular el paso de seccionar los cornetes, para ello al hundir la tecla “M” se presentará en la escena el bisturí y la mano del cirujano y al hundir la tecla “N” estas desaparecerán inmediatamente.

A3.3 Taponamiento nasal

Figura A18. Taponamiento nasal en el ambiente quirúrgico.
Fuente: Propia

A3.3.1 Código implementado para la simulación del taponamiento nasal

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class algodon : MonoBehaviour {

    public GameObject alg;
    public GameObject mano;


    public void Start () {
        alg = GameObject.Find ("alg");
        mano = GameObject.Find ("mano");

    }

    public void Update () {
        if (Input.GetKeyDown ("l"))
            alg.SetActive (true);
        if (Input.GetKeyDown ("l"))
            mano.SetActive (true);
        if (Input.GetKeyDown ("k"))
            alg.SetActive (false);
        if (Input.GetKeyDown ("k"))
            mano.SetActive (false);
        if (Input.GetKeyDown ("j"))
            mano.SetActive (false);

    }
}
```

A3.3.2 Modo de uso

Debido a que en la intervención las fosas nasales quedan con lesiones temporales se procede a hacer la simulación del taponamiento nasal que se basa en añadir dos algodones dentro de la nariz. Para ello al hundir la tecla “L” aparecerá en la escena la mano del cirujano, con los tapones nasales ya introducidos en la nariz, y para quitar solamente la mano del cirujano, se hunde la tecla “J” debido a que los tapones se deben quedar en la nariz para el postoperatorio.

A3.4 Trayectoria neurológica del robot semiautónomo

La trayectoria neurológica de este robot fue una trayectoria punto a punto, en la cual se implementaron cinco códigos en los que se incluían las 6 articulaciones del robot, el endoscopio que se encuentra anexado a él y el modelo completo del Robot.

Para la realización de cada punto de la trayectoria se utilizaron cinco teclas las cuales al ser presionadas constantemente llevan al robot de un punto a otro con una velocidad apropiada para la intervención.

A3.4.1 Modo de uso

Las teclas utilizadas para la realización de esta trayectoria con el robot semiautónomo serán mencionadas posteriormente con su respectiva ejecución.

"Z": Posición inicial del robot semiautónomo**. Figura A19.**

"X": Posición del robot semiautónomo en el cornete inferior. **Figura A20.**

"C": Posición del robot semiautónomo en el cornete medio. **Figura A21.**

"V": Posición del robot semiautónomo en el Hueso esfenoidal. **Figura A22.**

"B": Posición del robot semiautónomo dentro de la hipófisis. **Figura A23.**

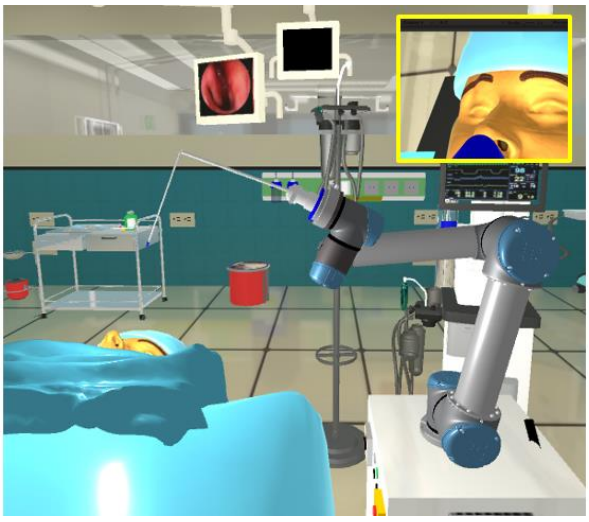


Figura A19. Posición inicial del robot semiautónomo.
Fuente: Propia

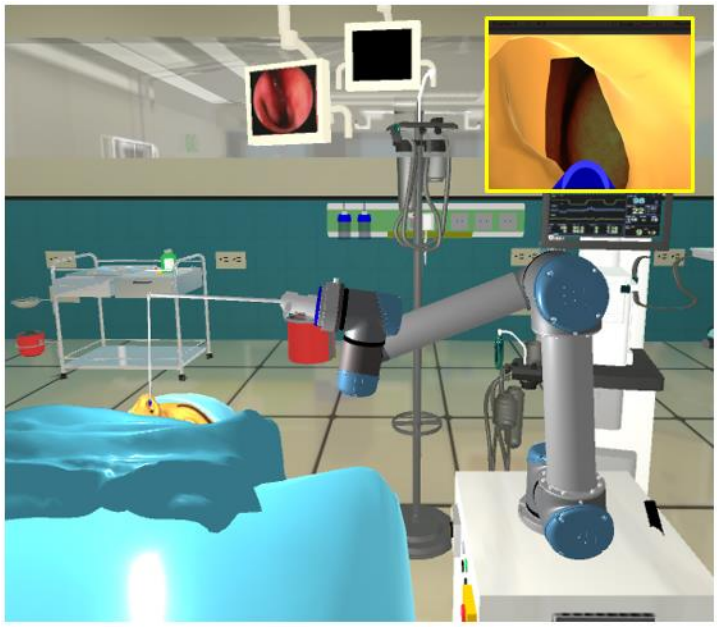


Figura A20: Posición del robot semiautónomo en el cornete inferior.
Fuente: Propia

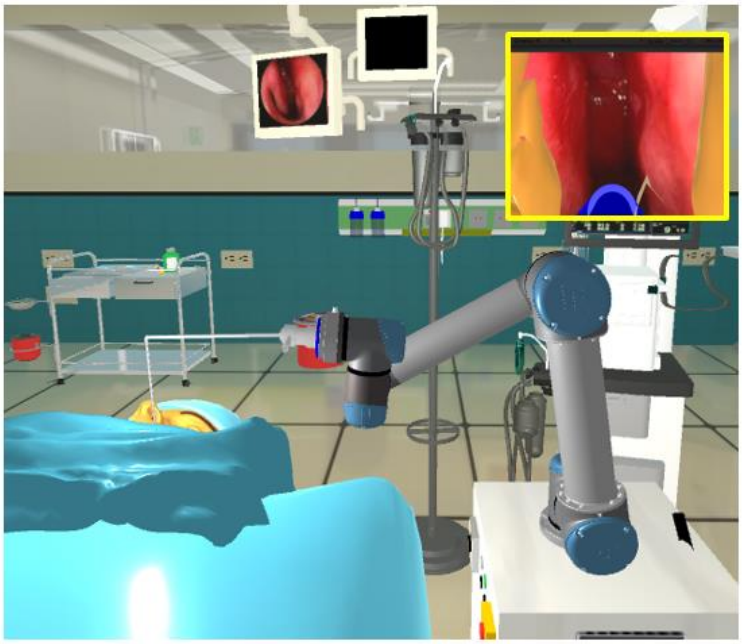


Figura A21. Posición del robot semiautónomo en el cornete medio.
Fuente: Propia

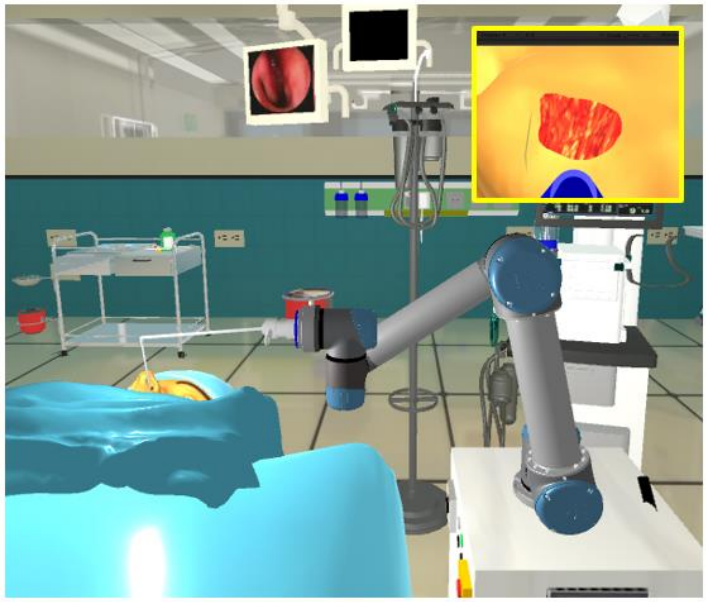


Figura A22: Posición del robot semiautónomo en el hueso esfenoidal.
Fuente: Propia

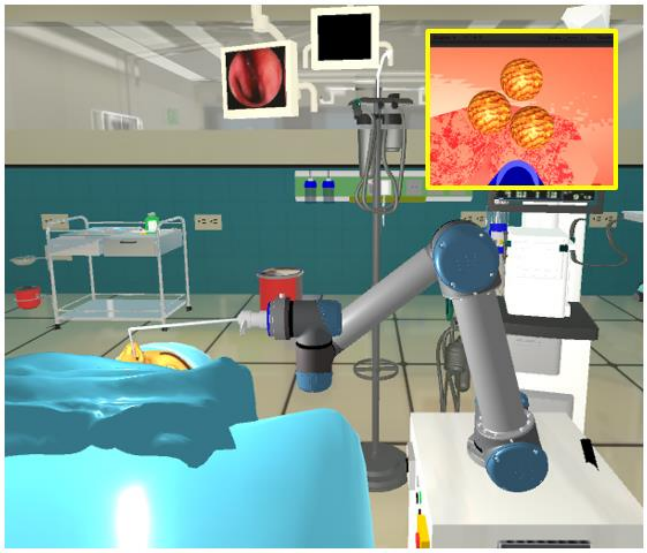


Figura A23. Posición del robot semiautónomo dentro de la hipófisis con vista al tumor.
Fuente: Propia