**CENTRO DE INVESTIGACIÓN Y DE ESTUDIOS AVANZADOS DEL INSTITUTO POLITÉCNICO NACIONAL**

UNIDAD ZACATENCO
DEPARTAMENTO DE INGENIERÍA ELÉCTRICA
SECCIÓN DE BIOELECTRÓNICA

# Plataforma virtual para el desarrollo de habilidades psicomotoras en neurocirugía.

**Tesis que presenta**

Luis Antonio Alvarez Castro

**para obtener el Grado de**

**Maestro en Ciencias**

**en la Especialidad de**

Ingeniería Eléctrica

**Directores de la Tesis**:
Dr. Daniel Lorias Espinoza

Ciudad de México Octubre, 2021

## AGRADECIMIENTOS

Quiero dar las gracias al Consejo Nacional de Ciencia y Tecnología (CONACyT) por brindarme su apoyo y que, sin ello, esto no hubiera sido posible.

A mis padres, por su apoyo incondicional ante las adversidades que se presentaron a lo largo de este camino, por estar siempre ahí cuando los necesite, por darme los mejores consejos para formar a un hombre de carácter y alentarme a cumplir mis sueños.

A Sebastián Flores Espinosa por ser ese amigo, compañero y confidente que ha sido el principal impulsor de que tomara este camino, por sus consejos y por estar ahí siempre que lo necesité.

A Víctor Manuel Chuc Palacios por ser ese gran amigo, compañero y ese gran soporte académico y guía a lo largo de mi camino, por ayudarme cuando más lo necesité y por estar siempre acompañándome en esta etapa.

Me gustaría agradecer en especial al Dr. Daniel Lorias Espinoza por haberme aceptado en su laboratorio y haber creído en mí desde el primer día, por sus grandes consejos y esa gran manera de instruirme, de hacer que creyera en mí mismo y de impulsarme día a día a experimentar cosas nuevas, es algo que le aporta gran valor a mi carrera académica y que aprecio mucho, gracias por ser ese gran equipo y por las experiencias que me llevo a lo largo de este camino.

Le agradezco también al Dr. Fernando Pérez Escamirosa por guiarme e instruirme de la mejor manera siempre aportando ideas nuevas y actuales, gracias a eso este trabajo no hubiera sido posible.

Plataforma virtual para el desarrollo de habilidades psicomotoras en neurocirugía.

CONTENIDO

## Contenido

## RESUMEN

En el presente trabajo se describe detalladamente el diseño y desarrollo de un sistema de evaluación y entrenamiento virtual para que los médicos residentes desarrollen habilidades psicomotoras en neurocirugía basado en el tiempo. El propósito de dicho sistema consiste en la evaluación de las destrezas que el médico realice en un entorno virtual, por medio de diferentes tareas que se le encomiendan y que están relacionadas con habilidades quirúrgicas que ayudarán en gran sentido a una evolución en la práctica y con ello se beneficie al paciente en gran medida.

Este sistema involucra al médico en un ámbito de constante práctica para este tipo de cirugías que requieran una técnica mínimamente invasiva, con el propósito de aumentar el número de especialistas en este tipo de intervenciones. El proyecto se desarrolla en un motor gráfico llamado Unity 3D el cual nos proporciona una visión tridimensional y real, haciendo una evaluación del desempeño de cada uno de los residentes basado en el tiempo.



## ABSTRACT

This written paper describes in detail the design and development of a virtual evaluation and training system for resident physicians to develop psychomotor skills in time-based neurosurgery. The purpose of this system is to evaluate the skills that the doctor performs in a virtual environment, through different tasks that are entrusted to him and that are related to surgical skills that will greatly help an evolution in practice and thus benefit the patient greatly.

This system involves the doctor in an area of constant practice for this type of surgery that requires a minimally invasive technique, in order to increase the number of specialists in this type of interventions. The project is developed in a graphics engine called Unity 3D which provides us with a three-dimensional and real vision, making an evaluation of the performance of each of the residents based on time.

# 1. Introducción

Paralelo al desarrollo de la cirugía en el mundo, la cirugía en México ha tenido avances importantes con las adaptaciones propias del tipo de patología quirúrgica y posibilidades económicas de nuestro país. En general, la evolución de las opciones terapéuticas ha tenido también un desarrollo paralelo a la tecnología aplicada a los procedimientos quirúrgicos [1].

Después de un desarrollo exponencial en la mitad del siglo pasado, se han descubierto diferentes tipos de técnicas y acciones, las cuales pueden ser implementadas para diversos padecimientos en los que métodos convencionales no son de gran ayuda debido a la complejidad de dicho padecimiento, es por esto por lo que la evolución quirúrgica crece a pasos agigantados en cuestión de la práctica.

Un claro ejemplo de esto es la cirugía de invasión mínima en específico la neurocirugía, que se ha desarrollado de manera extraordinaria, aunque de inicio fue aceptada con mucho escepticismo por la comunidad quirúrgica. Casi en todas las disciplinas quirúrgicas existen procedimientos de invasión mínima que son rutinarios hoy en día y que están constituidos en la vía de acceso de primera elección [1].

Muchas de estas técnicas [1] requieren de bastante práctica por parte de los médicos especialistas por lo que es necesario tener como preámbulo una constante preparación a base de entrenamientos especializados para desarrollar habilidades que le ayudarán al cirujano en la toma de decisiones, sin embargo existen técnicas que se encuentran en proceso de consolidación y que aunque todavía no pasan el proceso de aceptación por parte del personal médico, pueden presentar muchas ventajas en intervenciones de alto riesgo.

La cirugía robótica se halla en franco progreso, se ha demostrado que la utilización de estas tecnologías elimina el riesgo del fallo humano y permite practicar intervenciones menos invasivas, que reducen tanto el tiempo de operación como el postoperatorio [1].

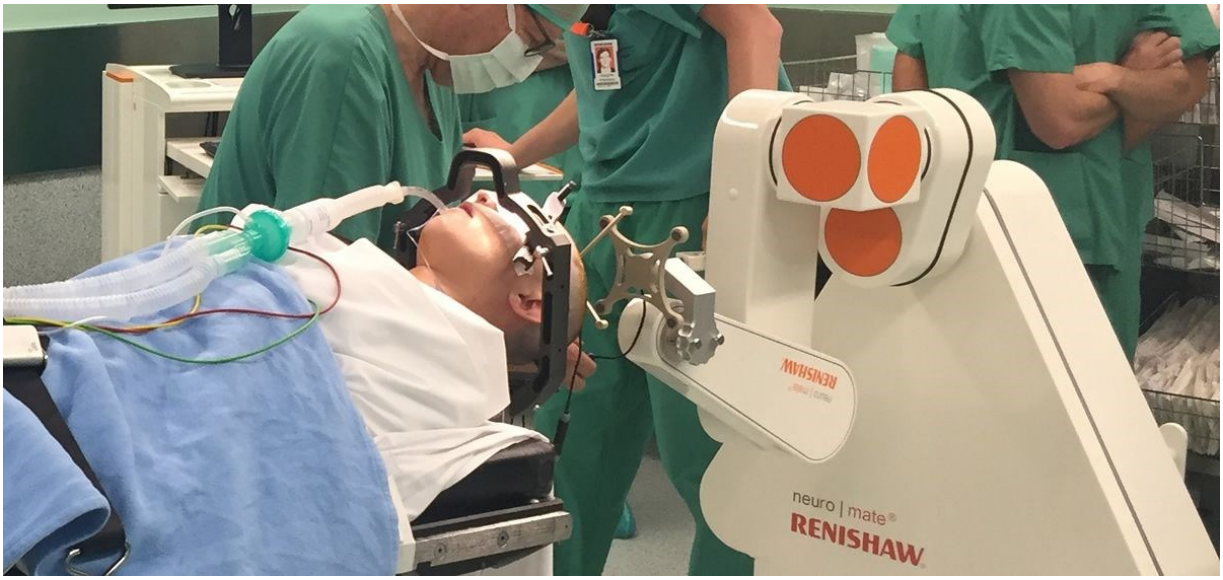



Fig. 1 Imagen del robot Renishaw, utilizado tanto en pacientes con epilepsia como de Parkinson.

## 1.1 ¿Qué es la neurocirugía?

La Neurocirugía es la ciencia médica que estudia las enfermedades que afectan al sistema nervioso que requieren o pueden requerir un tratamiento quirúrgico en algún momento de su evolución. Por tanto, tiene una conexión muy estrecha con la Neurología, puesto que muchas de las enfermedades a estudiar y tratar son comunes para ambas especialidades [2].

El tratamiento quirúrgico se realiza en lugares muy específicos del hospital, requiriendo una infraestructura de quirófanos, anestesia y reanimación, así como unidades de cuidados intensivos, que son también compartidas con otras especialidades quirúrgicas y médicas, además, precisa de otros recursos humanos e infraestructura hospitalaria que los aportan otras especialidades o subespecialidades en relación con el sistema nervioso [2].

En hospitales de alto nivel tecnológico, la neurocirugía es multidisciplinaria por lo que es fundamental que exista un correcto funcionamiento de otras especialidades, ya que, la calidad de asistencia se verá reflejada en el nivel científico de la misma neurocirugía.

## 1.2 ¿Qué hace un neurocirujano?

Un neurocirujano se encarga de los problemas neurológicos que requieren intervención quirúrgica. Su capacitación incluye los conocimientos generales de la neurología y todos los aspectos quirúrgicos de la misma [3].

Un neurocirujano puede actuar en emergencias que requieran el tratamiento del cerebro, cráneo o columna vertebral. También puede remover tumores y coágulos de sangre. Tratamientos con radiación, cirugía abierta, microcirugía y endoscopías son algunos de los procedimientos que puede llevar a cabo. Generalmente, un neurocirujano enfrenta situaciones en las que la vida del paciente depende de su habilidad y experiencia [3].

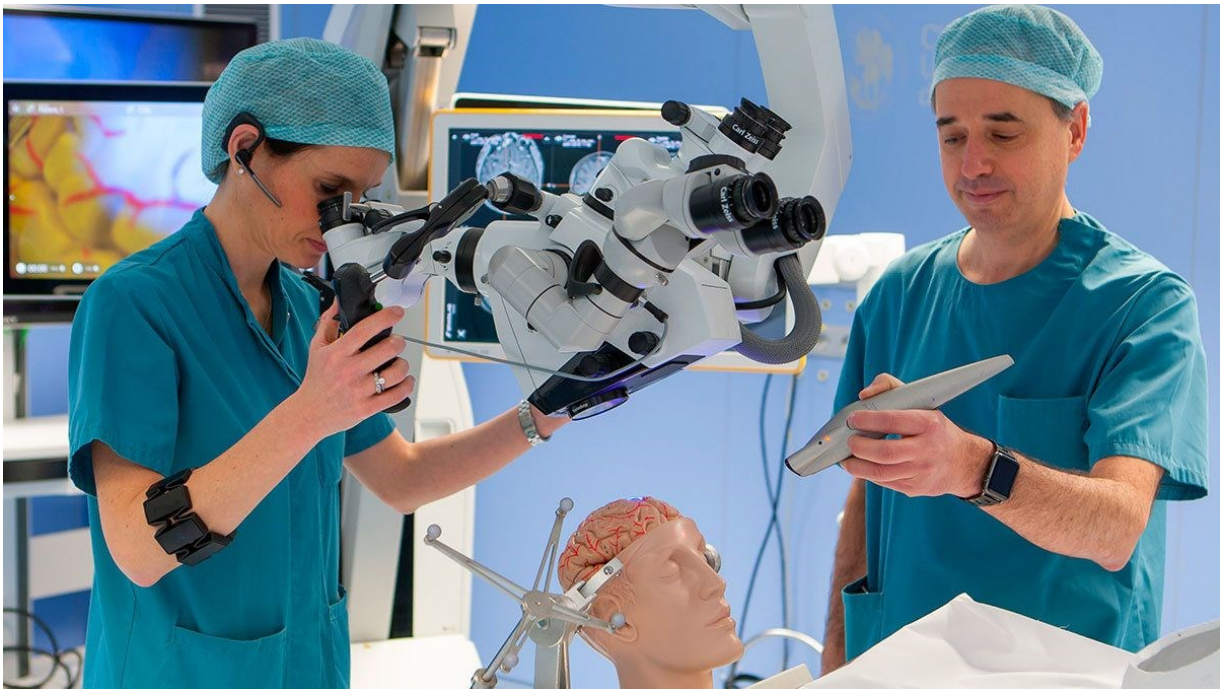


Fig. 2 Aprendizaje de áreas cerebrales para intervenciones en neurocirugía.

Los neurocirujanos requieren de periodos de entrenamiento muy largos y costosos, hasta que los residentes adquieren las destrezas mínimas y necesarias para este tipo de intervenciones. Cabe mencionar que, aunque este tipo de intervenciones requieren de una gran inversión en instrumental, se ve compensado en la reducción de pacientes

y el costo hospitalario que esto implica gracias a las rápidas recuperaciones y las casi nulas complicaciones postoperatorias [4].

En la actualidad, los simuladores quirúrgicos o sistemas de entrenamiento son de gran ayuda en el desarrollo de habilidades del cirujano, de otra forma el utilizar a pacientes reales se convertiría en una situación económica desfavorable y podría decirse que insostenible.

Si bien se sabe que la neurocirugía promete grandes alcances y está en constante desarrollo, debemos tener en cuenta que en México hay pocos sistemas de entrenamiento virtuales por lo que se vuelve punta de lanza la creación de opciones viables para el personal médico en el desarrollo de sus habilidades psicomotoras que permitan tener la experiencia suficiente en cualquier tipo de intervención para la evolución en la práctica.

Es importante entender que el acto quirúrgico representa una agresión al organismo por más mínima que sea, y muchas veces se presentan casos de sangrado excesivo, cortes accidentales y diversas situaciones que hacen que se busquen alternativas para reducir el riesgo en los pacientes, por lo que despertó en la investigación el interés de resolver dicho problema, por medio de soluciones menos traumáticas y agresivas, implementando entrenadores virtuales que desarrollen las habilidades psicomotoras del cirujano.

Dicha tecnología debe ir dirigida a minimizar el trauma sin la pérdida de la eficiencia y la seguridad, que es justamente la forma en la que se entiende y se aclama la inclusión de la neurocirugía en el arsenal del trabajo [5].

Se plantea que la enseñanza de la neurocirugía también se debe incluir en las escuelas de especialidades quirúrgicas y debe formar parte de un programa de actualización obligatoria para médicos de hospitales. Solo de esta manera la enseñanza puede ser

afectiva, y esta disciplina, en la que tanto se confía, alcanzará el desarrollo y los resultados que merece [5].

Las competencias de un cirujano se ven altamente favorecidas si logra incluir en su contexto educacional, laboral y medio social lo adquirido por medio de la práctica, en donde los simuladores virtuales se vuelven una excelente alternativa en el desarrollo de sus capacidades de respuesta en todo tipo de intervenciones quirúrgicas y la evolución de sus habilidades psicomotoras para la resolución de problemas.

# 2. Planteamiento del problema

Debido a la contingencia sanitaria COVID 19 se ha diseñado una plataforma virtual la cual será de gran ayuda en el proceso de formación de neurocirujanos, además contribuirá a la evolución médica y tecnológica ya que es necesario que cirujanos especialistas dominen las técnicas de la neurocirugía, pero ¿Cómo se pueden desarrollar las habilidades psicomotoras necesarias para este tipo de intervenciones médicas?

## 2.1 Objetivos

Proponer el diseño de una plataforma virtual para el desarrollo de habilidades psicomotoras en neurocirugía.

**Objetivos Específicos**

1.- Desarrollar una plataforma virtual para el desarrollo de habilidades en la tarea de Disección.

2.- Desarrollar una plataforma virtual para el desarrollo de habilidades en la tarea de Navegación Espacial.

3.- Desarrollar una plataforma virtual para el desarrollo de habilidades en la tarea de Ubicación Espacial.

4.- Pruebas del sistema de entrenamiento virtual.

5.- Desarrollar una articulación mecánica para la generación de movimientos.

## 2.2 Solución propuesta

El sistema de entrenamiento virtual propuesto contempla un entorno de fácil uso en donde el cirujano desarrollará sus habilidades psicomotoras para intervenciones de mínima invasión. Contiene una metodología de tres tareas a realizar en donde cada resultado será medido de manera objetiva para posteriormente registrarlos cuantitativamente con base en parámetros de evaluación.
Todo esto por medio de un motor gráfico (software) el cual será controlado por una articulación mecánica con la cual el cirujano realizará las tareas correspondientes.

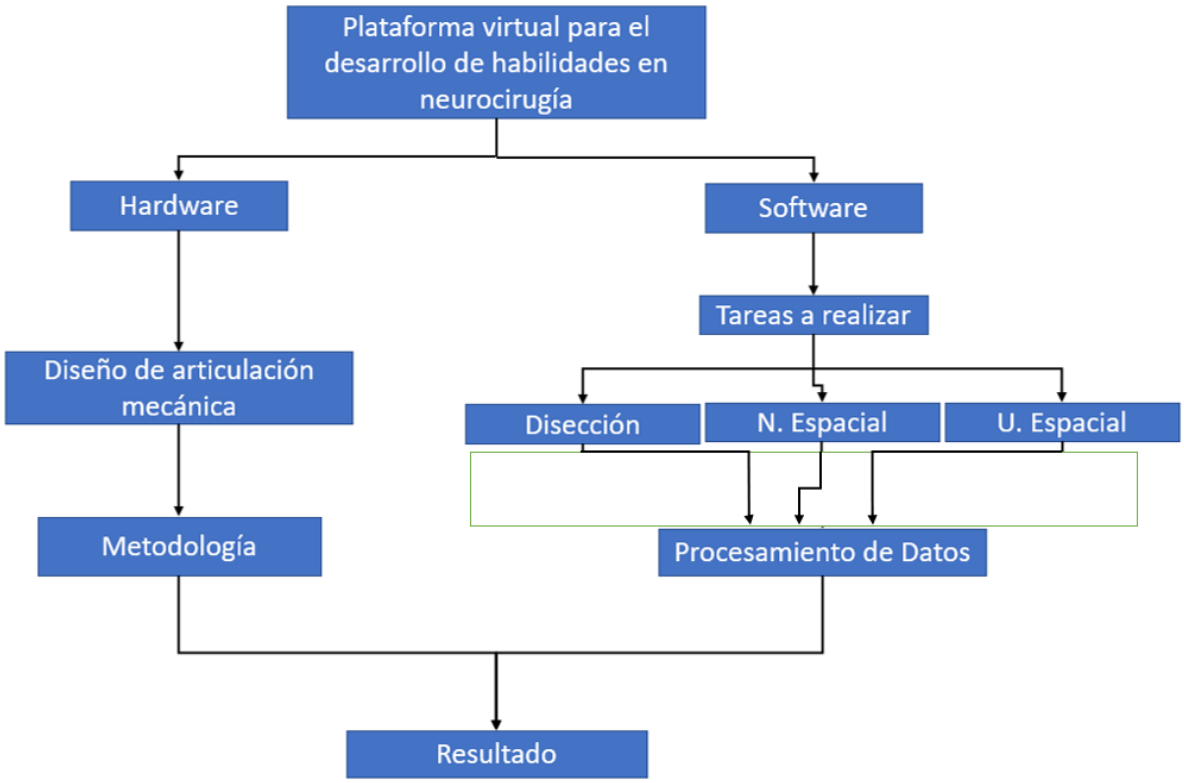



Diagrama a bloques de la solución propuesta

## 3. Antecedentes

Se ha demostrado que los sistemas de entrenamiento clínico o quirúrgico pueden proporcionar datos objetivos de rendimiento y el seguimiento del progreso en la curva de aprendizaje. La experiencia del entrenamiento también puede ser extendida y se mide en términos absolutos, además, los entrenadores con simuladores virtuales se pueden desarrollar para prácticamente todas las condiciones; un médico puede repetir un procedimiento varias veces en un software sin ningún riesgo para la salud profesional [6].


Gráfica 1. Gráfica de pastel mostrando el porcentaje correspondiente de la ventaja de usar sistemas de entrenamiento virtual [7].

Desde la introducción de la simulación en medicina, diversas especialidades médicas se han involucrado cada vez más en el fomento de su desarrollo y validación, sobre todo en especialidades donde se requieren las mejores habilidades quirúrgicas debido a la delicadez de ciertas intervenciones, siendo el caso de la neurocirugía.

Un sistema de entrenamiento quirúrgico debe de ser innovador, pero sobre todo debe ser costeable refiriendo que se puede financiar con los medios que se dispone, puesto

que la obtención de sistemas de entrenamiento quirúrgico comercializados en el exterior es económicamente inalcanzable, en donde los precios de venta aproximados de empresas de primer nivel en el mundo rondan desde los 69,000 dólares hasta los 105,000 dólares, sin mencionar los costos del transporte [8].

Para los estudiantes, residentes y becarios, los simuladores pueden ser utilizados para desarrollar nuevas habilidades, así como para profesores. Los simuladores pueden ser utilizados para actualizar una habilidad utilizada con poca frecuencia o para realizar procedimientos del uso de nuevas tecnologías [9].

Es importante enfatizar que el uso de sistemas de entrenamiento o simuladores en la educación médica no pretende reemplazar en todos sus extremos al método tradicional. Los cirujanos se entrenan a lo largo de su formación, observando y participando en operaciones reales, muy pocas personas pueden ser entrenadas por un experto por lo que se puede decir que es un aprendizaje oportunista y en muchos casos un tanto deficiente [7].

Existen resultados que dejan en evidencia que el tiempo de aprendizaje es menor, el nivel de seguridad al realizar los diferentes tipos de tareas que se obtiene es óptimo como se muestra en la gráfica 1, lo que permitirá contribuir a disminuir el índice de morbilidad para futuros actos quirúrgicos.

Por otro lado, si bien se sabe que el tema de la seguridad del paciente no es nuevo, debe señalarse que en los últimos 20 años se han registrado avances en cuanto a la difusión y la toma de conciencia sobre el error médico y el daño del paciente, al ir abordando de más en más la existencia de los eventos adversos y sus consecuencias sobre los usuarios de los servicios de salud. De aquí surge la importancia del desarrollo de sistemas de entrenamiento tratando de reducir las estadísticas de error médico [10].

Si bien en nuestro país no existen estudios serios y reconocidos sobre este aspecto con representatividad nacional, se plantean datos de mayor reconocimiento internacional.

Esto bajo el supuesto de que, si en países desarrollados el problema tiene las dimensiones que enseguida se presentarán, en países como el nuestro y otros muchos de Latinoamérica la situación podría darse al menos con la misma intensidad o incluso mayor.

Las técnicas de la realidad virtual tienen por objetivo crear de forma artificial un entorno en el que el usuario pueda ver y sentir los objetos, tal y como los vería y sentiría en realidad. El grado de inmersión en un entorno virtual depende en gran medida de la fidelidad con la que este reproduce la realidad que el usuario conoce [11]. Todos los elementos y condiciones que son importantes a la hora de desarrollar un entorno virtual inmersivo dependen de la aplicación objetivo del mismo, es por eso por lo que de ahí parte la idea de incluir de manera obligatoria en la preparación de cirujanos los sistemas de entrenamiento virtual.

Ahora bien, queda aún por cubrir un objetivo fundamental en la investigación y desarrollo de simuladores quirúrgicos. Éste consiste en la validación de la mejora en la calidad del aprendizaje, velocidad y reducción de costes que aportan los simuladores quirúrgicos frente a los métodos tradicionales de aprendizaje. Cuando esto se haya cumplido, los simuladores quirúrgicos pasarán a ser una herramienta extremadamente potente que revolucionará la transferencia de las técnicas quirúrgicas [11].

## 3.1 Simuladores virtuales para neurocirugía en la actualidad

### 3.1.1 Simulador S.I.M.O.N.T craneal

El simulador ofrece un entorno realista para aprender de forma rápida y sencilla las técnicas neurológicas intracraneales, y constituye una opción económica a las preparaciones anatómicas humanas. El sistema de bomba integrado permite imitar la circulación del líquido cefalorraquídeo, de esta forma, se crean unas condiciones quirúrgicas fieles a la realidad. También ofrece la posibilidad de marcar el líquido raquídeo con un color para simular las hemorragias intracraneales.

El modelo es una buena alternativa para países donde está prohibido utilizar preparaciones anatómicas humanas por razones éticas.

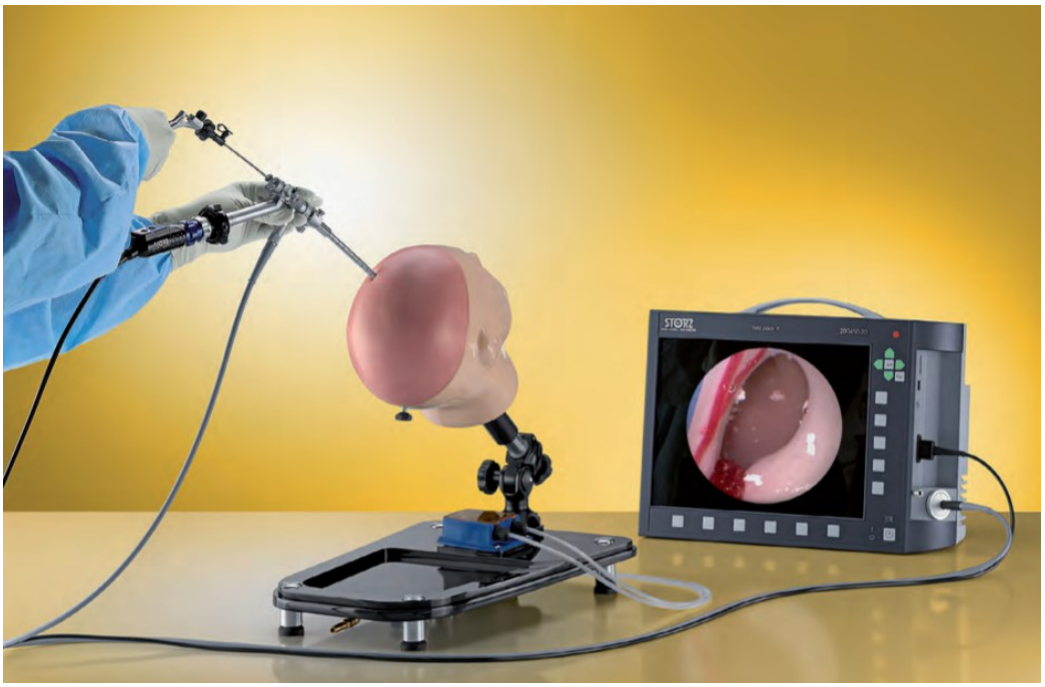



Fig. 3 Simulador S.I.M.O.N.T craneal

### 3.1.1 Simulador en realidad virtual y con tecnología háptica para Neurocirugía por Industria del conocimiento Mendoza

Este simulador, por sus características, es el primero desarrollado en Latinoamérica. La idea de realizar este simulador en Realidad Virtual para Neurocirugía surgió el 3 de octubre de 2015, mientras el Dr. Fabián Cremaschi daba una conferencia organizada por la Asociación Argentina de Neurocirugía, sobre el uso de esta técnica para la toma de biopsia de tumores cerebrales. Con el total apoyo de la Secretaría de Investigación, Internacionales y Posgrado de la UNCuyo, ese sueño se está realizando 5 años después de muchísimo trabajo y sacrificio personal, dedicado a unir la Neurociencia con las Tecnologías, desarrollando algo pionero y único en Argentina [14].

El simulador consta de un kit de Realidad Virtual, en donde el usuario puede llevar a cabo procedimientos con instrumental quirúrgico virtual controlados por 2 palancas tipo joystick y con ayuda de unos lentes de realidad virtual se puede tener la noción real de un entrenamiento y desarrollar las habilidades necesarias para este tipo de intervenciones.

Fig. 4 Simulador “Industria del conocimiento Mendoza” en uso para intervención en neurocirugía.

### 3.1.2 Surgical Navigation Advanced Platform (SNAP)

El George Washington University Hospital ahora utiliza una herramienta avanzada de realidad virtual para neurocirugía y cirugía torácica. La realidad virtual de precisión de Surgical Theatre permite a los cirujanos explorar virtualmente el cerebro y el cuerpo de un paciente antes de realizar un procedimiento. La tecnología se inspiró en simuladores de vuelo diseñados para entrenar a pilotos de combate.
La Plataforma Avanzada de Navegación Quirúrgica, o SNAP, mejora el flujo de trabajo actual del quirófano al integrarse y mejorar el sistema de navegación quirúrgico existente junto con otras herramientas y tecnologías mientras utiliza las capacidades de precisión VR.

Surgical Navigation Advanced Platform está revolucionando la forma en que se realizan los procedimientos quirúrgicos craneales delicados y altamente complejos, y pronto se utilizará en otras aplicaciones quirúrgicas. La plataforma permite a los cirujanos aplicar planes quirúrgicos específicos para el paciente durante cirugías reales con la capacidad de ver y saber lo que sigue en el campo quirúrgico. Cada paso planificado ahora se ve en la pantalla, por lo que el cirujano puede sincronizar sus movimientos para poder realizar los procedimientos más seguros y mínimamente invasivos posibles. Las imágenes detalladas que posibilita el SNAP pueden conducir a procedimientos quirúrgicos mucho más precisos que pueden proteger el cerebro y pueden optimizar los resultados del paciente y los tiempos de recuperación.

Fig. 5 Plataforma Avanzada de Navegación Quirúrgica (SNAP).

| SISTEMA | GRADOS DE LIBERTAD | SISTEMA UTILIZADO | RETROALIMENTACIÓN | INSTRUMENTAL |
|---|---|---|---|---|
| S.I.M.O.N.T | 4 | Instrumental | Visual | Físico |
| Industria del Conocimiento Mendoza | 5 | Joysticks y lentes de Realidad Virtual | Visual y Háptica | Virtual |
| Surgical Navigation Advance Platform (SNAP) | 6 | Instrumental y lentes de Realidad Virtual | Visual y Háptica | Físico |
| Plataforma virtual para el desarrollo de habilidades en neurocirugía | 3 | Virtual y Articulación Mecánica | Visual | Físico |

Tabla 1. Cuadro comparativo de sistemas de entrenamiento para el desarrollo de habilidades psicomotoras.

## 4. Desarrollo

El resumen general de la funcionalidad del proyecto se puede observar en el diagrama 1. El desarrollo de dicho proyecto comienza con el diseño e implementación de una articulación mecánica, que será parte fundamental en el entrenamiento de los neurocirujanos. Dicha articulación es considerada como el hardware del sistema. El sistema cuenta con una metodología de evaluación por medio de un software, en la cual se pueden medir las habilidades de los especialistas por medio de métricas que ayudan a la evolución de las destrezas adquiridas.

Con lo anterior, se pretende obtener datos objetivos gracias a pruebas experimentales a las que es sometido el sistema con el fin de establecer y estandarizar mejores protocolos de entrenamiento en neurocirujanos y evolucionar en el ámbito por medio de la innovación tecnológica.

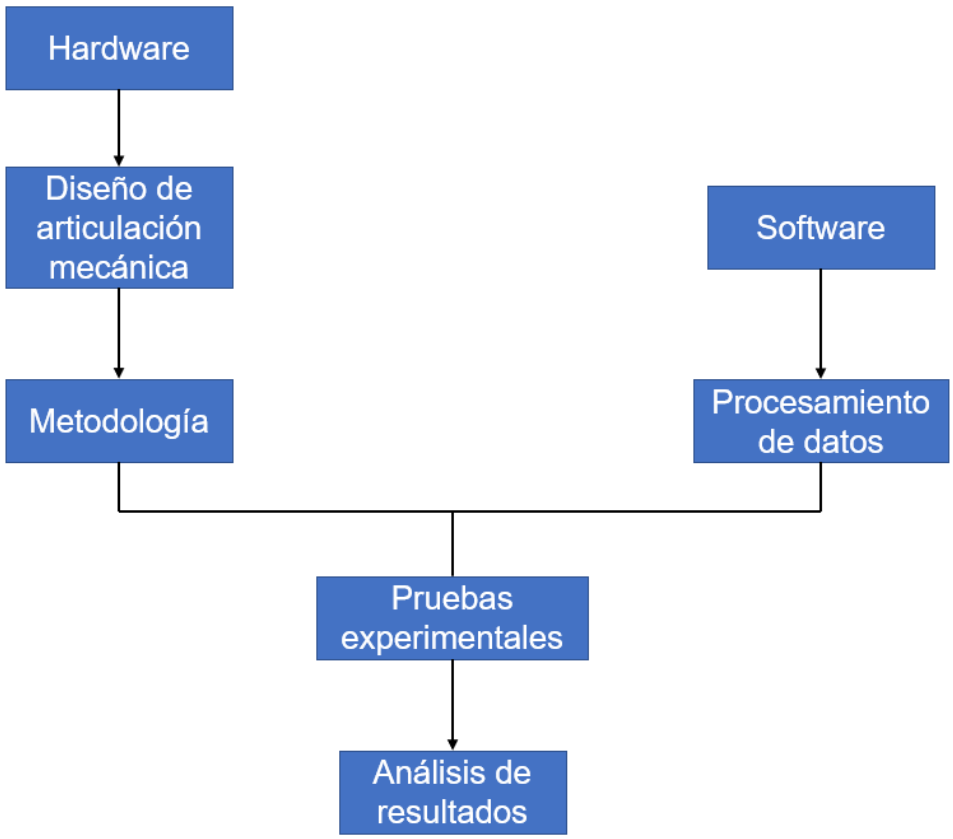



Fig. 6 Estructura de desarrollo de la plataforma virtual para el desarrollo de habilidades.

## 4.1 Hardware

### 4.1.1 Diseño y descripción de articulación mecánica para la generación del movimiento

Se diseñó una articulación mecánica para generar el movimiento físicamente y traducirlo a un movimiento virtual de los instrumentos quirúrgicos (ver Fig. 7). Esta articulación está construida de acrílico debido a que se trata de un material resistente, económico y fácil de encontrar.

La articulación se compone de 3 cubos de diferentes tamaños, el primer cubo que es el cuerpo de la articulación tiene dimensiones de 30 x 30 x 30 cm, el segundo cubo cuenta con unas dimensiones de 16 x 16 x 16 cm y por último el tercer cubo posee unas dimensiones de 10 x 10 x 10 cm. Dicha articulación se diseñó de tal manera que se lograra tener tres grados de libertad, para esto fue necesario centrar los cubos más pequeños en el de mayores dimensiones, quedando el segundo cubo a 7 cm de distancia por lado respecto al primero y el tercero quedando a 3 cm de distancia por lado respecto al segundo.

Tanto el segundo como el tercer cubo cuentan con modificaciones en un par de sus lados, esto con el fin de implementar piezas de rotación tales como rodillos, así como piezas de presión para poder generar un movimiento en tres ejes. Es importante mencionar que el segundo cubo está destinado a moverse en el eje "*y"* y el tercer cubo que es el más pequeño respecto a los demás está destinado a moverse en el eje "*x"*.

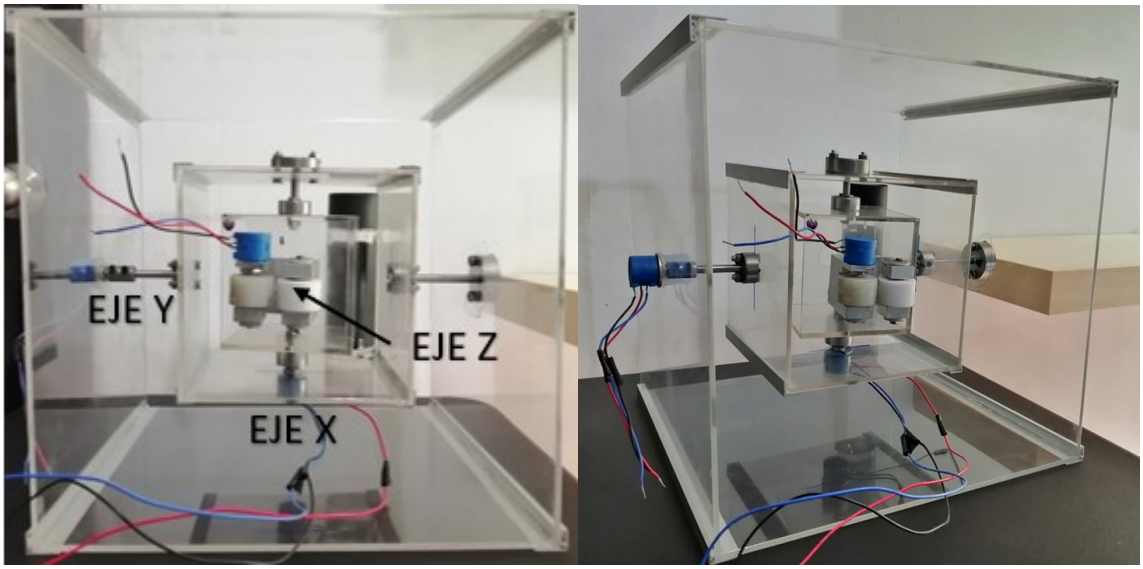



Fig. 7 Articulación mecánica para generar movimiento.

Por otro lado, la traducción del movimiento físico y real al movimiento virtual de los instrumentos quirúrgicos virtuales se lleva a cabo por medio de potenciómetros de precisión que están muy bien ajustados a unas piezas específicas con las que cuentan ambos cubos y que son una especie de tubos huecos en donde se introduce la parte rotatoria de dichos potenciómetros y que gracias a tornillos opresores previamente instalados, hace que al generar movimiento en dichos cubos, también se genere movimiento en los potenciómetros y de esa manera registrar los cambios de posición en cada eje.

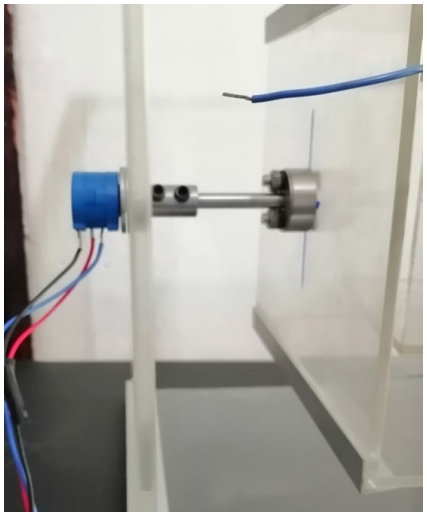


Fig. 8 Implementación de potenciómetros de precisión para registro de cambios de posición.

Por último, para cubrir los cambios de posición en el eje "*z*" se diseñó una pieza compuesta por 2 rodillos (uno de teflón y otro de plástico) y que se implementó en el centro de la cara frontal del tercer cubo (el de menor dimensión), ya que este último cuenta con un orificio de 1 cm para la inserción del instrumental quirúrgico real.

Dicha pieza compuesta por los dos rodillos, al igual que las piezas anteriores contiene una modificación en el rodillo de plástico para que de igual forma con ayuda de tornillos opresores e insertando el potenciómetro pueda registrar cambios en el eje "*z*" simulando la entrada y salida del instrumental quirúrgico para satisfacer necesidades virtuales de profundidad.

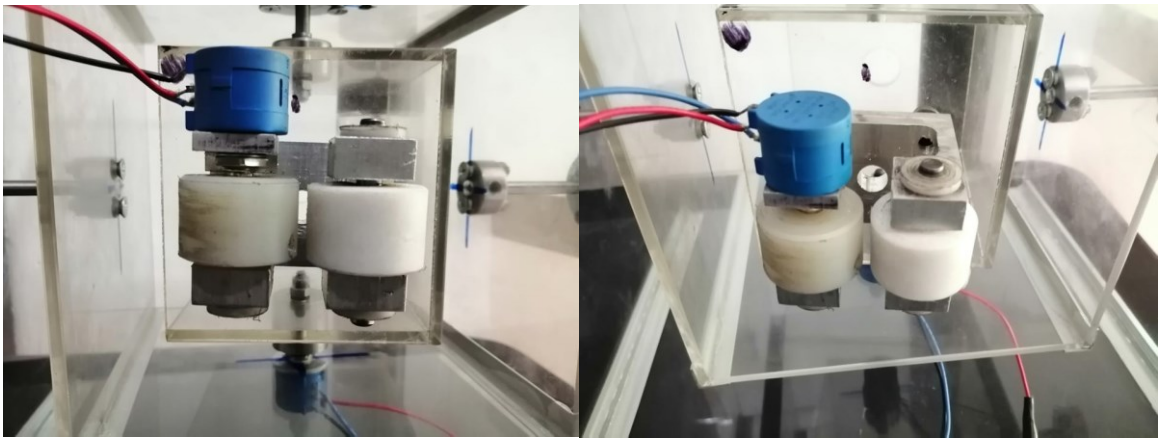


Fig.9 Pieza para cambio de posición en el eje "*z*".

### 4.1.2 Materiales

- **Pinza para laminectomía:** Instrumento quirúrgico utilizado en neurocirugía con una dimensión de 17.8 cm de largo y una abertura de la pinza de 4 x 10 mm (ver Fig. 10). La pinza es utilizada en conjunto con la articulación mecánica antes descrita para generar movimientos y cambios de posición en cada eje.

Fig. 10 Pinza JARIT “Spurling”

- **Potenciómetro de precisión de 50 Kohm +-5% modelo 3590S-2-503L:** Es un potenciómetro rotatorio lineal de precisión de 50kΩ 2W 1 buje con montaje de alambre enrollado con ajuste de ranura de destornillador. El potenciómetro tiene terminales de terminal de soldadura, 10 vueltas y 1° de holgura máxima. El potenciómetro giratorio está diseñado para su uso en aplicaciones HMI. Tiene un rango de temperatura de -40 a 125°C.

Fig. 11 Potenciómetro de precisión modelo 3590S-2-503L

- **KIT Arduino UNO R3 y cable USB:** Incluye el chip microcontrolador oficial ATMEGA328P en su formato extraíble. Esta nueva versión utiliza un chip ATmega16U2 en lugar del ATmega8U2; esto permite velocidades de transferencia más rápidas. No necesita drivers/controladores para Linux o Mac. Voltaje de alimentación: 7-12 V. 6 pines de entrada analógicos. Temperatura de funcionamiento: de -5 a 40°C.


Fig. 12 KIT Arduino UNO R3 con cable USB para la transferencia de datos de los potenciómetros a la PC.

Es importante mencionar que el proyecto ocupa otros materiales mínimos como resistencias de 2.7 Kohm 5% (3 piezas), cable para soldar, Protoboard (1 pieza) y otro tipo de componentes que se describen en los anexos.

## 4.2 Software

Esta etapa es la parte medular del sistema de entrenamiento, ya que, a diferencia de otros sistemas, tanto los módulos de interacción para desarrollar las habilidades psicomotoras que el neurocirujano necesita como el instrumental quirúrgico que normalmente se usaría en dicha intervención, se diseñaron e implementaron de manera virtual, esto con el fin de lograr un apego lo más cercano posible con lo que sería una intervención quirúrgica real. De esta forma se describe cada uno de los

módulos explicando su diseño y funcionamiento para lograr los objetivos deseados con la ayuda e interacción de dicho instrumental quirúrgico virtual. Es importante recordar que con la ayuda de la articulación mecánica antes descrita y con las pinzas para laminectomía (ver Fig. 10) el neurocirujano realiza el movimiento físico de tal manera que el software interpreta y traduce los movimientos reales a virtuales, moviendo de igual forma el instrumental quirúrgico virtual para completar las tareas sugeridas.

### 4.2.1 Módulo de Ubicación Espacial

La perspectiva frontal del módulo virtual desarrollado para Ubicación Espacial puede apreciarse en la Fig. 13. Se trata de un módulo en el que en su interior se encuentran 4 objetivos quirúrgicos (targets), los cuales se encuentran distribuidos espacialmente a diferentes distancias y profundidades como se muestra en la Fig.14, sin embargo, también existen elementos ajenos a dichos objetivos como son tablas de diferentes dimensiones y separadores entre los objetivos.

El funcionamiento de este módulo consta básicamente en poner en contacto el instrumental virtual correspondiente a la tarea con cada uno de los objetivos quirúrgicos de tal manera que el programa principal al registrar que existe una colisión entre ambos elementos elimina el objetivo (target tocado) y da la pauta a seguir con la trayectoria del instrumental en el interior del módulo para continuar con los demás objetivos. Si por alguna razón el neurocirujano toca algún elemento ajeno a los objetivos quirúrgicos (targets), el sistema lo penalizará contabilizando los errores y desplegándolos en la pantalla de puntuaciones al finalizar la tarea, además se toma el tiempo que le tarda al neurocirujano realizar la tarea satisfactoriamente.

El mismo proceso se repite hasta tocar el cuarto objetivo, de esta manera el programa principal sabe que el neurocirujano terminó con dicha tarea y se encuentra listo para pasar a la siguiente.

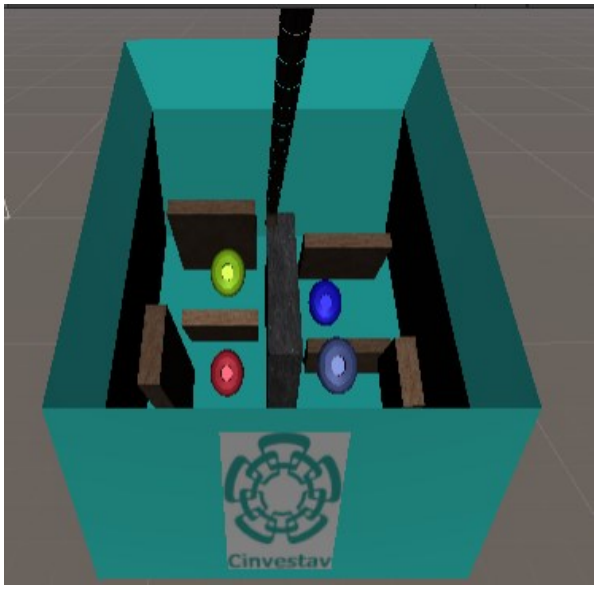



Fig. 13 Vista frontal de módulo de Ubicación Espacial.

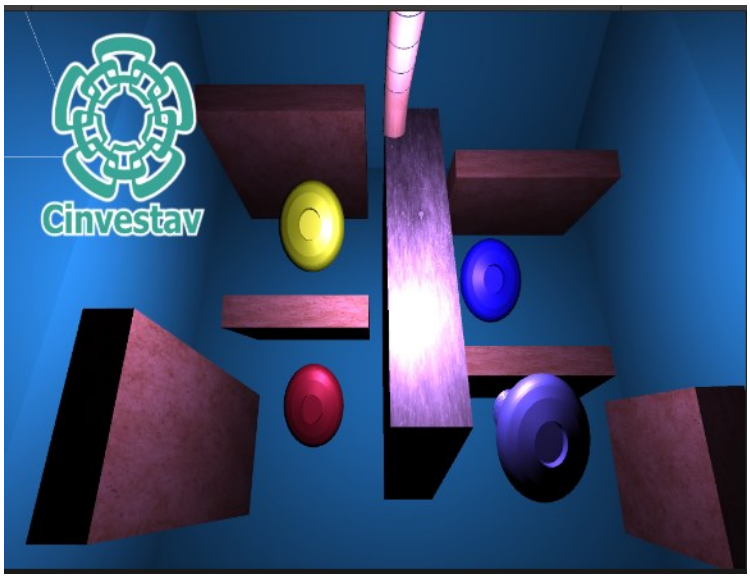



Fig. 14 Vista de usuario al realizar la tarea de Ubicación Espacial, percibiendo distancias y profundidades.

### 4.2.2 Módulo de Navegación Espacial

La perspectiva frontal del módulo virtual desarrollado para Navegación Espacial puede apreciarse en la Fig. 15. Este módulo consta de dos placas separadas una de otra con respecto a la vertical. La placa superior tiene un corte en forma de serpiente y la placa inferior limita en profundidad al instrumental, de tal manera que se pretende que el

neurocirujano haga un desplazamiento del instrumental a través del patrón establecido de la placa superior y se mantenga suspendido durante la ejecución de la tarea. De igual forma que la tarea de Ubicación Espacial si por alguna razón el neurocirujano toca algún borde de dicha placa con el instrumental quirúrgico, el sistema lo penalizará contabilizando las colisiones a lo largo del recorrido y lo desplegará en la pantalla de puntuaciones al finalizar, además se toma el tiempo que le tarda al neurocirujano realizar la tarea.

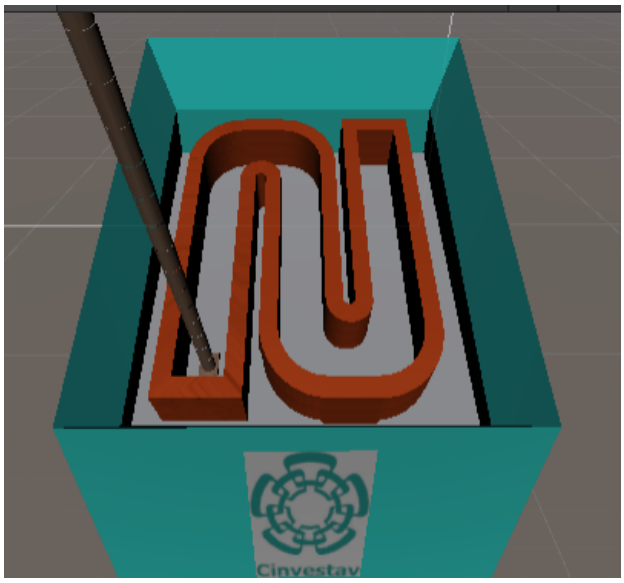



Fig. 15 Vista frontal del módulo de Navegación Espacial.

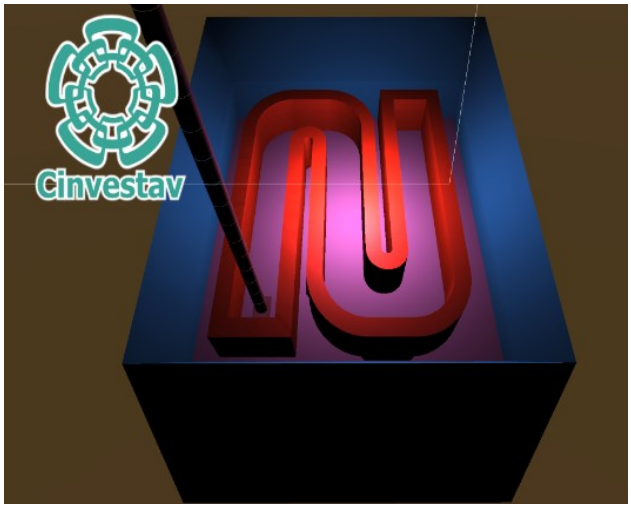



Fig. 16 Vista del usuario al realizar la tarea de Navegación Espacial.

### 4.2.3 Módulo de Disección

La perspectiva frontal del módulo virtual desarrollado para Disección puede apreciarse en la Fig. 17. El módulo contiene objetivos quirúrgicos en forma de cápsula y esferas los cuales se encuentran posicionados al azar en diferentes lugares y como se puede observar en la Fig. 18 se encuentran revueltos, esto con el fin de que el neurocirujano tenga la destreza de tocar los objetivos en forma de esfera con el instrumental virtual sin hacer contacto con los objetivos en forma de cápsula.

Tanto las esferas como las cápsulas tienen un color predeterminado, el color de las cápsulas no efectúa cambios, sin embargo, el color de las esferas cambia si el neurocirujano llega a lograr un buen contacto con ellas, esto simulando lo más cercano a la realidad un corte o coagulación de algún tejido humano. De igual forma que las dos tareas anteriores, si por alguna razón el neurocirujano toca una cápsula, el sistema lo penalizará contabilizando las veces que se tocaron las cápsulas y los cuenta como errores al finalizar, además se toma el tiempo que le tarda al neurocirujano realizar la tarea.

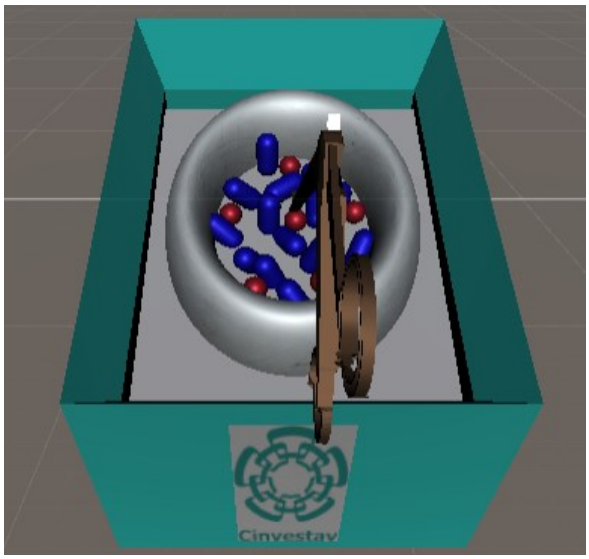



Fig. 17 Vista frontal del módulo de Disección.


Fig. 18 Vista del usuario al realizar la tarea de Disección.

### 4.2.4 Pinza para laminectomía virtual

Instrumento quirúrgico utilizado en neurocirugía con una dimensión de 17.8 cm de largo y una abertura de la pinza de 4 x 10 mm (ver Fig. 19). La pinza es utilizada como instrumental quirúrgico en la tarea de Disección debido a que se considera darle realidad al proceso en la realización de la tarea.


Fig. 19 Pinza para laminectomía virtual.

### 4.2.5 Instrumento para la tarea de Navegación Espacial

Simula el instrumental quirúrgico utilizado en cirugía. Tiene un material virtual de aluminio de 18 cm de largo por 48 mm de ancho y graduado a 1 cm (ver Fig. 20). En su extremo inferior posee un tope rectangular de 1 cm x 1 cm, esto con el fin de que pueda hacer contacto con el patrón de dicho módulo (ver Fig. 16) y elevar el nivel de dificultad de dicha tarea.

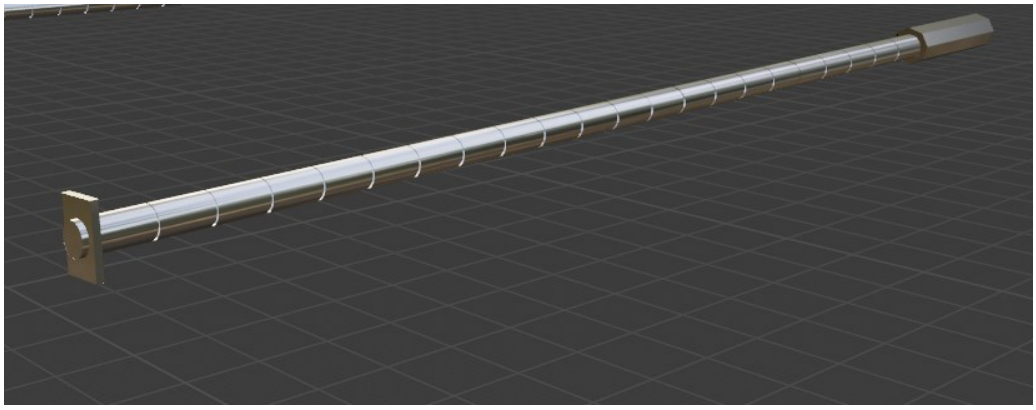


Fig. 20 Instrumento para la tarea de Navegación Espacial.

### 4.2.6 Instrumento para la tarea de Ubicación Espacial

Simula el instrumental quirúrgico utilizado en cirugía. Tiene un material virtual de aluminio de 18 cm de largo por 48 mm de ancho y graduado a 1 cm (ver Fig. 21). En su extremo inferior se encuentra inclinado a 45° aproximadamente para tocar los objetivos quirúrgicos (targets) con mayor facilidad (ver Fig. 14).

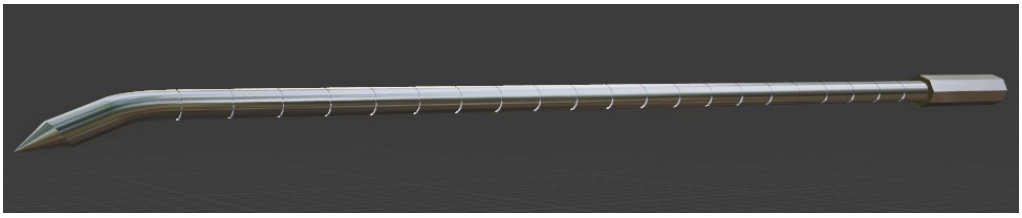


Fig. 21 Instrumento utilizado para la tarea de Ubicación Espacial.

### 4.3 Metodología de Evaluación

El sistema de entrenamiento virtual es una excelente alternativa en donde una de sus ventajas principales es el bajo costo y que ofrece un entorno para que el neurocirujano desarrolle habilidades psicomotoras en práctica. A pesar de lo anterior, es importante mencionar que el sistema no cubre el desarrollo completo de dichas habilidades, sino que es un excelente complemento para el entrenamiento de los especialistas en una intervención.

Con el objetivo de definir cualitativamente los parámetros de desempeño, se tomaron en cuenta los siguientes aspectos: movimiento de precisión del instrumento quirúrgico, percepción de profundidad y distancias [19]. Dichos aspectos se contemplan con base a la tasa de errores y tiempos de ejecución [20].

#### 4.3.1 Tarea de Ubicación Espacial

La tarea está diseñada para obtener una percepción de profundidad y distancia. El propósito de la tarea es tocar cada objetivo quirúrgico (target) utilizando la punta del instrumental, no importa la secuencia, pero teniendo en cuenta el no colisionar con los objetos ù obstáculos ajenos a los objetivos, como se muestra en la Fig. 22.

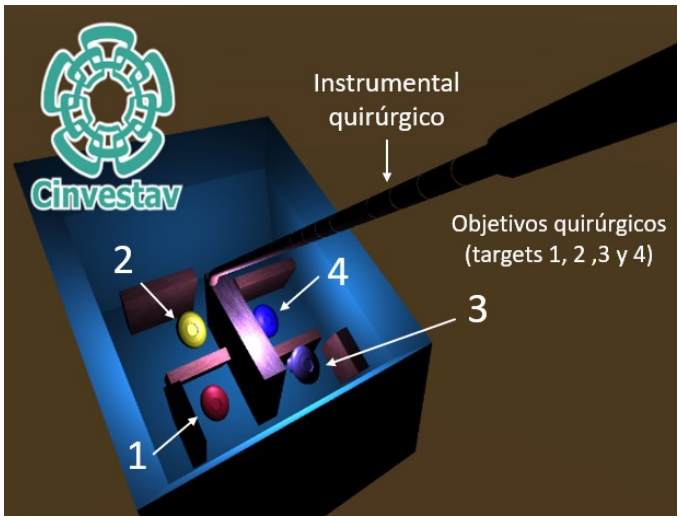



Fig. 22 Esquema para la tarea de Ubicación Espacial.

***Dificultad:*** Lograr percibir las profundidades y distancias entre los objetivos. Como ya se mencionó se considera como penalización si se toca un objeto ajeno a los objetivos.

***Parámetros de medición:***

1. ***Tiempo*** que le toma al neurocirujano realizar la tarea.
2. ***Penalización*** por el número de veces que se toque un objeto ajeno a los objetivos.

### 4.3.2 Tarea de Navegación Espacial

La tarea está diseñada con base a reducir el movimiento de la mano (temblores) y la navegación de forma segura durante la cirugía. La tarea consiste en seguir la trayectoria desde el punto A hasta el punto B con ayuda del instrumental quirúrgico virtual como se muestra en la Fig. 23.

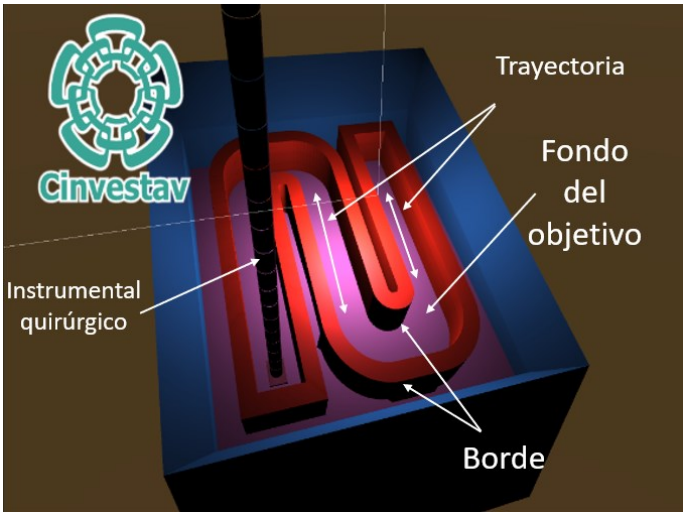



Fig. 23 Esquema para la tarea de Navegación Espacial.

***Dificultad:*** Evitar tocar los bordes laterales y el fondo del patrón. Si a lo largo del recorrido el instrumental toca el fondo o algún borde lateral se considera como penalización.

***Parámetros de medición:***

1. ***Tiempo*** que le toma al neurocirujano realizar la tarea.
2. ***Penalización*** por el número de veces que se toquen los bordes.

### 4.3.3 Tarea de Disección

La tarea está diseñada para evitar el corte o coagulación de tejido sano. La tarea consiste en tocar los objetivos en forma de esfera sin hacer contacto con los objetivos en forma de capsula como se muestra en la Fig. 24.

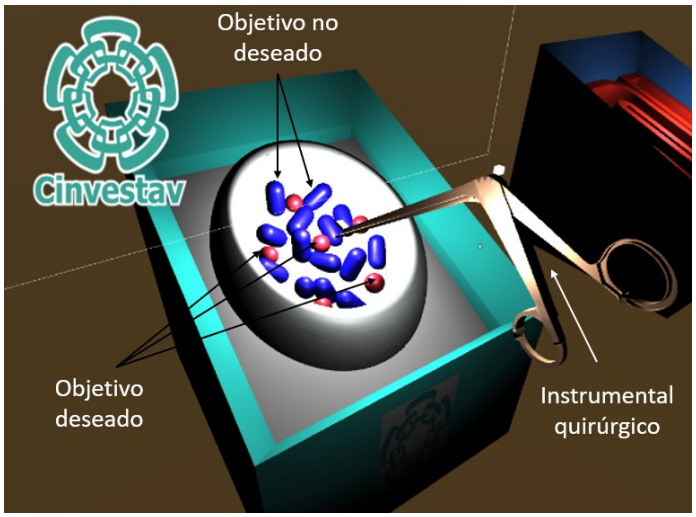



Fig. 24 Esquema para la tarea de Disección.

***Dificultad:*** Hacer contacto con todos los objetivos en forma de esfera con la mayor precisión posible.

***Parámetros de medición:***

1. ***Tiempo*** que le toma al neurocirujano realizar la tarea.
2. ***Penalización*** por el número de veces que se toque un objetivo con forma de capsula.

## 4.4 Diseño de software

El diseño del sistema de entrenamiento virtual se llevó a cabo en 2 plataformas diferentes para satisfacer las necesidades en diferentes áreas. Por un lado, se diseñaron los instrumentos virtuales para las tareas correspondientes en la plataforma Blender 2.91 y para la interacción de la interfaz virtual con el usuario se diseñaron algoritmos orientados a objetos en la plataforma Unity 2019.4.11f1.

Blender 2.91 es la suite de creación 3D gratuita y de código abierto. Además de ser un software libre, es compatible con la totalidad de la canalización 3D: modelado, montaje, animación, simulación, renderizado, composición y seguimiento de movimiento, incluso edición de video y creación de juegos. Los usuarios avanzados emplean la API de Blender para scripts de Python para personalizar la aplicación y escribir herramientas especializadas; a menudo, estos se incluyen en las versiones futuras de Blender. Blender es ideal para individuos y pequeños estudios que se benefician de su canalización unificada y su proceso de desarrollo receptivo [21].

Por lo anterior, Blender es una herramienta de mucha ayuda en la creación de los instrumentos virtuales implementados en el sistema ya que como se mencionó es compatible con plataformas de interacción como es el caso de Unity 2019.4.11f1. Es importante mencionar que la exportación de modelos de una plataforma a otra se tiene que hacer en formato (*FBX.)* de lo contrario no se transferirán todas las propiedades del modelo, por lo que su funcionamiento no será óptimo en el momento de la ejecución del programa.

Si bien es importante la creación de los instrumentos virtuales, la parte fundamental del entorno del sistema aplica en la plataforma Unity 2019.4.11f1. Unity es lo que se conoce en la actualidad como un motor de desarrollo o un motor gráfico, sirve para la creación de videojuegos y se basa en una serie de rutinas de programación C# (C Sharp) que permiten el diseño, la creación y el funcionamiento de un entorno interactivo [12].

Unity es una herramienta muy práctica y versátil ya que permite crear entornos para diversas plataformas (PC, videoconsolas, móviles, etc.), todo esto mediante un editor visual y programación vía scripting obteniendo resultados totalmente profesionales.

### 4.4.1 ¿Unity 3D o Unreal engine?

Normalmente los motores de juegos hacen uso de una serie de librerías propias de cada tipo de programación que permiten el diseño, la creación y la representación de un videojuego o sistema. Los motores brindan herramientas al programador, que le permiten dedicar menos tiempo a aspectos poco importantes para la idea general del videojuego o sistema, pero que son de suma importancia para la experiencia del usuario final.

Teniendo esto en cuenta, la elección del motor es fundamental para definir qué tipo de juego queremos crear.

Existen distintos tipos de motores de juego que brindan diferentes experiencias y por ende se pueden explotar de mejor manera computacionalmente hablando.

*Unity* y *Unreal engine* son alternativas muy populares que destacan sobre otros motores de juego debido a los grandes resultados que se pueden alcanzar con su implementación.

Para elegir uno de ellos es fundamental tener en cuenta que tipo de juego o sistema queremos crear ya que cada uno tiene sus ventajas y desventajas sobre el otro como se muestra en la siguiente tabla comparativa (Tabla 1).

| VENTAJAS Y DESVETAJAS UNITY VS UNREAL ENGINE | |
|---|---|
| **UNITY** | **UNREAL ENGINE** |
| Sencillo y práctico para trabajar con juegos móviles, en 2D y 3D | Funciona mejor para juegos de consola y PC's de última generación |
| Mucho más fácil de utilizar pero menos flexible a la hora de realizar cambios | Más complejo pero gana en flexibilidad |
| Utiliza el lenguaje de programación Javascript o C# | Utiliza lenguaje de programación C++ |
| Unity cuenta con una gran variedad de contenidos de terceros y documentación que nos ayudarán a facilitar la integración de contenido de nuestro juego o sistema | Control completo ya que ofrece de forma gratuita su código abierto para realizar mejoras |

Tabla 2. Cuadro comparativo de ventajas y desventajas *Unity* vs *Unreal engine*

Uno de los objetivos principales es que dicho sistema de entrenamiento virtual sea de bajo costo, ya que los que actualmente existen en el mercado puede llegar a ser muy costosos, además se pretende que el sistema sea sencillo y práctico para el usuario final, en este caso los neurocirujanos sin omitir el grado de complejidad que se requiere en cada tarea y así poder explotar el sistema al máximo.

Otro punto importante es la cuestión del gasto computacional, este debe de contar con un rendimiento óptimo, pero sin ser necesario en ordenadores de última generación.

Por último, como se mencionó *Unity* cuenta con una extensa retroalimentación por parte de programadores a nivel mundial lo que hace que el diseño, la creación y la implementación del sistema vaya más allá de las expectativas, es por eso por lo que el motor gráfico o de juegos seleccionado fue *Unity*, por sus amplias ventajas con base en las necesidades del proyecto.

### 4.4.2 Programa principal

La estructura general de la programación del sistema de entrenamiento virtual se representa en el diagrama 2.

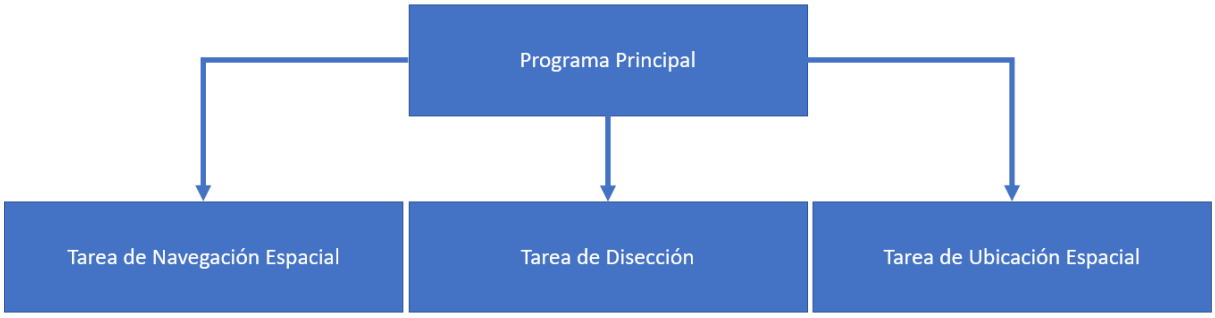



Diagrama 2. Estructura general de programación.

Al iniciar el programa principal se muestra una ventana denominada Menú principal o "Main Menu". Dicho Menú está compuesto por un "Canvas". "Canvas" es un área en donde te permite crear interfaces para que el usuario pueda estar en un ambiente virtual muy práctico y cómodo. En otras palabras, "Canvas" es la parte introductoria al sistema de entrenamiento en donde se puede hacer uso de botones, textos para indicar ciertos datos, imágenes, etc.

En este caso el sistema ocupa diferentes "Canvas" para representar diferentes situaciones y empezaremos por describir el primero, que hace uso en específico de 2 botones principales que son los de *Comenzar* con las tareas del sistema de entrenamiento y el de *Salir* para cerrar el programa.

Es importante mencionar que *Unity* trabaja con un tipo de programación que se denomina *"Padre e hijo"* ó *"Esclavo"*, esto quiere decir que si se crea un objeto cualquiera y se opta por que sea el "*Padre*" de otros objetos, al realizar cambios en él, en consecuencia, todos los objetos *"Esclavos"* a este también se verán afectados.

Otro punto importante para tomar en cuenta es que si se hace uso de un *"Canvas"* generalmente también se hace uso de escenas en la interfaz de Unity. Las escenas son diferentes instancias en donde se pretende que el usuario pase de un lugar a otro en el juego o sistema. Estas se pueden ir creando con base en las necesidades del juego o sistema para que por medio de botones y/o programación el usuario se traslade de una instancia a otra.

Dichas escenas se deben agregar en el apartado de *Build Settings* ya que de otra forma las escenas no serán cargadas en el sistema por lo que los componentes como botones para ir de una instancia a otra en el sistema no reconocen dicha instrucción y la acción será nula. La figura 25 representa el *"Canvas"* que contiene el Menú principal del sistema de entrenamiento virtual.


Fig. 25 Canvas del Menú principal.

El botón o evento de *"Comenzar"* da paso al inicio de las tareas correspondientes (Disección, Navegación Espacial y Ubicación Espacial) las cuales son las que fueron elegidas para el desarrollo de las habilidades psicomotoras en Neurocirugía. El botón o evento de *"Salir"* cierra el programa.

### 4.4.3 Tarea de Ubicación Espacial

Al iniciar el sistema por medio del botón de “Comenzar” se da paso a realizar la tarea de Ubicación Espacial. Como ya se mencionó anteriormente esta tarea consiste en tocar 4 objetivos (targets) que se encuentran a diferentes distancias y profundidades, además de que en el interior del módulo se encuentran obstáculos. La finalidad de la tarea es tocar los 4 targets evadiendo los objetos ajenos a estos.

Para la realización de esta tarea se generó un *Canvas* diferente al del Menú principal, el cual contiene un temporizador para evaluar el tiempo que le tarda al especialista en Neurocirugía llevar a cabo la tarea de manera satisfactoria, además el *Canvas* está diseñado de tal manera que si el usuario llega a cometer una penalización tocando un objeto que no sean los 4 targets, aparece una leyenda con la palabra “PENALIZACION” como se muestra en la Figura 26.

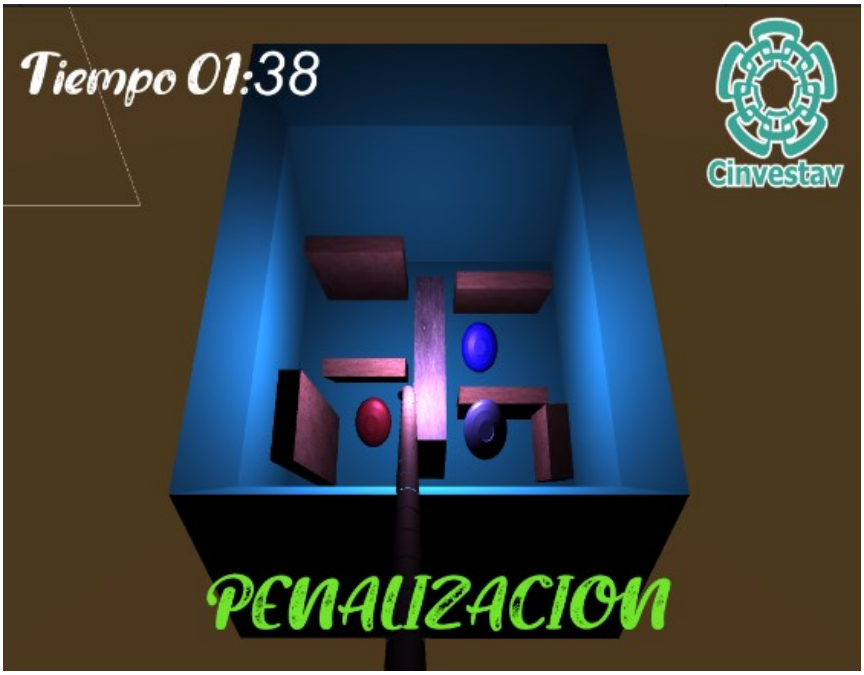



Fig. 26 Penalización en la tarea de Ubicación Espacial.

Unity tiene la facilidad en el diseño de sistemas como este, una de esas facilidades con las que se llevó a cabo esta tarea es que permite que los objetos puedan tener un

“Tag”. Un “Tag” como su nombre lo dice es una etiqueta que se le puede poner a los objetos para identificarlos al momento de la programación si queremos que ocurra algo con ellos.

En este caso se usó un “Tag” denominado “obstáculo” para los bloques que se pueden observar (Fig. 26) y un “Tag” denominado “objetivo” para los targets. Esto permite que la programación haga un comparativo por cada frame e identifique que objeto entra en contacto directo con el instrumento quirúrgico virtual y de esta manera desplegar el mensaje de penalización en caso de hacerlo.

Un punto importante para tener en cuenta es que *Unity* hace uso de físicas en su entorno de programación, es decir, que cada uno de los elementos que se diseñan y se pretende que interactúen deben de llevar un *Collider.* Los componentes llamados Collider son aquellos que definen la forma de un objeto con el propósito de colisiones físicas, en este caso colisiones entre el instrumento virtual con los targets o con los obstáculos. Un *Collider* es invisible ante el usuario y no necesariamente debe tener la forma exacta del objeto, en muchas ocasiones una aproximación es más que suficiente para que el sistema lo registre y pueda llevarse a cabo las colisiones deseadas.

Si bien ya se mencionó en el apartado 4.2.6 del como se encuentra constituido el instrumento virtual para la ejecución de la tarea de Ubicación Espacial, se debe mencionar aspectos respecto al diseño de su movimiento.

En este caso se optó por crear un objeto en forma de cubo el cual debe de ser de dimensiones mucho menores que el instrumento, dicho cubo se coloca en la parte centro superior del instrumento (Fig. 27). Esto se realiza con el fin de que el movimiento que se genera por medio de la articulación mecánica (Fig. 7) se traduzca en movimiento para el cubo. El instrumento virtual de la tarea de Ubicación Espacial contiene un *Script* o código el cual se encarga de que el instrumento siga en todas direcciones el desplazamiento que tiene dicho cubo.


Fig. 27 Diseño de movimiento para instrumento virtual de la tarea Ubicación Espacial.

Esto es necesario ya que si se traduce el movimiento directamente al instrumento este genera demasiados bugs (errores de código en programas informáticos) y la traducción de movimiento real a virtual no sería la óptima. En otras palabras, el cubo es necesario para que el movimiento del instrumento no tenga fallas en programación y se pueda llevar a cabo la tarea. Este cubo puede ir renderizado o no, preferentemente no debería de ir renderizado puesto que genera cierta estética al sistema de entrenamiento.

Por último, los targets irán desapareciendo de escena conforme vayan colisionando con el instrumento virtual y serán identificados por un contador, de modo que cuando desaparezcan los 4 targets se pasará a la siguiente tarea a realizar. El *script* o código de esta tarea se encuentra en los anexos de este documento.

### 4.4.4 Tarea de Navegación Espacial

Al término de la tarea anterior (Ubicación Espacial) se da paso a realizar la tarea de Navegación Espacial. Como ya se mencionó anteriormente esta tarea consiste en realizar un recorrido de un punto A hasta un punto B por medio del instrumental virtual

correspondiente. La finalidad de la tarea es llegar de un punto a otro sin que el instrumento haga contacto con los bordes del patrón establecido en el módulo ni con el fondo del patrón.

Para la realización de esta tarea se generó un *Canvas* diferente al del Menú principal, el cual contiene un temporizador para evaluar el tiempo que le tarda al especialista en Neurocirugía llevar a cabo la tarea de manera satisfactoria, además el *Canvas* está diseñado de tal manera que, si el usuario llega a cometer una penalización tocando los bordes o el fondo del patrón, aparece una leyenda con la palabra “PENALIZACION” como se muestra en la Figura 28.


Fig. 28 Penalización en la tarea de Navegación Espacial.

Un punto importante para tener en cuenta es que *Unity* hace uso de físicas en su entorno de programación, es decir, que cada uno de los elementos que se diseñan y se pretende que interactúen deben de llevar un *Collider*. Los componentes llamados *Collider* son aquellos que definen la forma de un objeto con el propósito de colisiones físicas, sin embargo, es importante entender que existen figuras u objetos que no pueden ser cubiertos por *Colliders* automáticamente, sino que en estos casos se debe

diseñar el *Collider* manualmente tratando de cubrir toda la forma del objeto. Tal es el caso del *Collider* que usa el patrón por donde se desplaza el instrumento virtual.

Al tratarse de un objeto con una figura irregular, los *Colliders* deben implementarse de tal manera que cubra todos los bordes y espacios para que en cualquier zona de dicho patrón exista la posibilidad de hacer una colisión y las penalizaciones sean detectadas de manera correcta como se muestra en la Fig. 29.

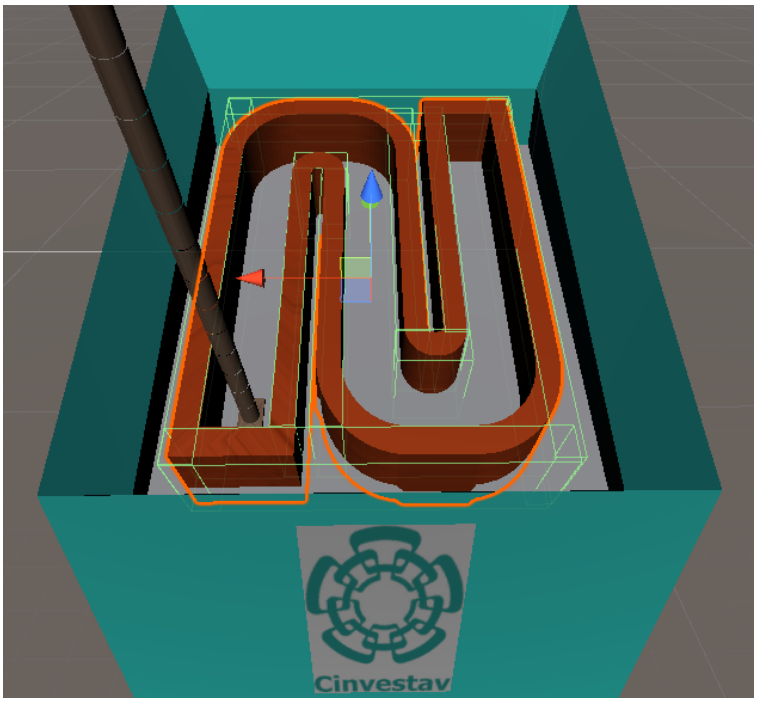



Fig. 29 Collider manual para colisiones en la tarea de Navegación Espacial.

Como se puede observar en la figura anterior, el *Collider* que cubre el patrón de recorrido (líneas delgadas) del punto A al punto B se encuentra formado por un conjunto de *Colliders* de diferentes dimensiones, esto debido a que el *Collider* automático que ofrece la plataforma de *Unity* no es suficiente para cubrir su forma, por lo que con ayuda de varios de estos y aproximando las dimensiones del patrón es posible crear uno nuevo a la medida que cumpla con las expectativas de la física en el sistema de entrenamiento.

Si bien ya se mencionó en el apartado 4.2.5 del cómo se encuentra constituido el instrumento virtual para la ejecución de la tarea de Navegación Espacial, se debe mencionar aspectos respecto al diseño de su movimiento.

En este caso se optó por crear un objeto en forma de cubo el cual debe de ser de dimensiones mucho menores que el instrumento, dicho cubo se coloca en la parte centro superior del instrumento (Fig. 30). Esto se realiza con el fin de que el movimiento que se genera por medio de la articulación mecánica (Fig. 7) se traduzca en movimiento para el cubo. El instrumento virtual de la tarea de Navegación Espacial contiene un *Script* o código el cual se encarga de que el instrumento siga en todas direcciones el desplazamiento que tiene dicho cubo.

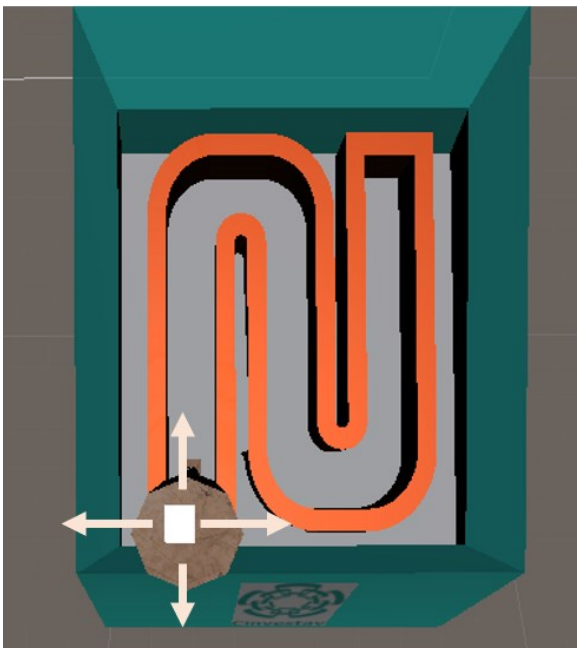


Fig. 30 Diseño de movimiento para instrumento virtual de la tarea Navegación Espacial.

Esto es necesario ya que si se traduce el movimiento directamente al instrumento este genera demasiados bugs (errores de código en programas informáticos) y la traducción de movimiento real a virtual no sería la óptima. En otras palabras, el cubo es necesario para que el movimiento del instrumento no tenga fallas en programación y se pueda llevar a cabo la tarea. Este cubo puede ir renderizado o no, preferentemente

no debería de ir renderizado puesto que genera cierta estética al sistema de entrenamiento.

Para saber el momento en el que el usuario termina de manera satisfactoria la tarea de Navegación Espacial, se colocó un *Gameobject* al final del recorrido (Fig.31). Un *Gameobject* es un objeto vacío en forma cuadrada que sin bien no tiene masa en el sistema de programación de *Unity*, puede detectar colisiones con otros objetos en el entorno con el simple hecho de agregar como un componente un *Box Collider*.

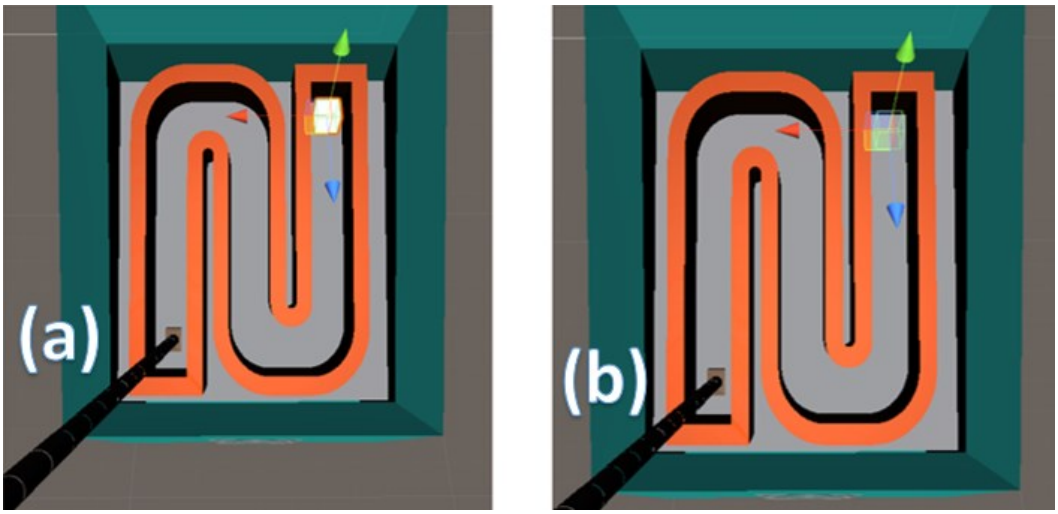



Fig. 31 Gameobject como indicador final en tarea de Navegación Espacial. (a) Renderizado (Hacer visible un objecto) (b) Sin renderizar.

La idea general es que el instrumento virtual colisione con dicho *Gameobject* del final para que el sistema identifique que la tarea ha sido completada y dar paso a la tercera y última tarea del sistema de entrenamiento. El *script* o código de esta tarea se encuentra en los anexos de este documento.

### 4.4.5 Tarea de Disección

Al término de la tarea anterior (Navegación Espacial) se da paso a realizar la tarea de Disección. Como ya se mencionó anteriormente esta tarea consiste en tocar los objetos en forma de esfera sin hacer contacto con los objetos en forma de cápsula. La

finalidad de la tarea es tocar todas las esferas obteniendo el número más bajo de penalizaciones.

Para la realización de esta tarea se generó un *Canvas* diferente al del Menú principal, el cual contiene un temporizador para evaluar el tiempo que le tarda al especialista en Neurocirugía llevar a cabo la tarea de manera satisfactoria, además el *Canvas* está diseñado de tal manera que, si el usuario llega a cometer una penalización tocando los objetos en forma de cápsula, aparece una leyenda con la palabra "PENALIZACION" como se muestra en la Figura 32.


Fig. 32 Penalización en la tarea de Disección.

Un punto importante para tener en cuenta es que Unity hace uso de físicas en su entorno de programación, es decir, que cada uno de los elementos que se diseñan y se pretende que interactúen deben de llevar un *Collider*. Los componentes llamados *Collider* son aquellos que definen la forma de un objeto con el propósito de colisiones físicas. En el caso de la tarea de Disección se puede hacer uso de los *Colliders* predeterminados de *Unity*, si bien una esfera y una cápsula pueden ser formas

irregulares, *Unity* ofrece *Colliders* de estas formas por default por lo que resulta una solución al momento de programar dichas colisiones.

Si bien ya se mencionó en el apartado 4.2.4 del cómo se encuentra constituido el instrumento virtual para la ejecución de la tarea de Disección, se debe mencionar aspectos respecto al diseño de su movimiento.

En este caso se optó por crear un objeto en forma de cubo el cual debe de ser de dimensiones mucho menores que el instrumento, dicho cubo se coloca en la parte centro superior del instrumento (Fig. 33). Esto se realiza con el fin de que el movimiento que se genera por medio de la articulación mecánica (Fig. 7) se traduzca en movimiento para el cubo. El instrumento virtual de la tarea de Disección contiene un *Script* o código el cual se encarga de que el instrumento siga en todas direcciones el desplazamiento que tiene dicho cubo.

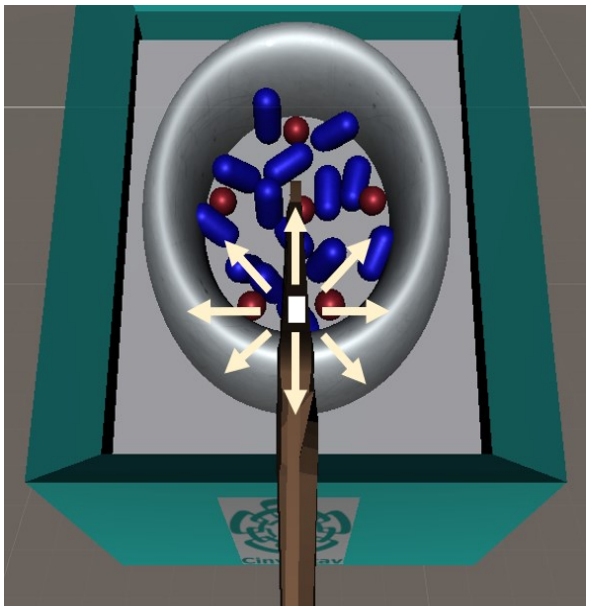


Fig. 33 Diseño de movimiento para instrumento virtual de la tarea de Disección.

Esto es necesario ya que si se traduce el movimiento directamente al instrumento este genera demasiados bugs (errores de código en programas informáticos) y la traducción de movimiento real a virtual no sería la óptima. En otras palabras, el cubo

es necesario para que el movimiento del instrumento no tenga fallas en programación y se pueda llevar a cabo la tarea. Este cubo puede ir renderizado o no, preferentemente no debería de ir renderizado puesto que genera cierta estética al sistema de entrenamiento.

Por último, los objetivos en forma de esfera al ser tocados cambian de su color original a otro, y después de un tiempo (3 segundos) irán desapareciendo de la pantalla, por lo que solo se quedarán visibles los objetos en forma de cápsula (Fig. 34). Los objetivos en forma de esferas serán identificados por medio de un contador que avisa al sistema en el momento que todos los objetivos hayan sido tocados y destruidos, de esta manera el usuario habrá finalizado la tarea. El *script* o código de esta tarea se encuentra en los anexos de este documento.

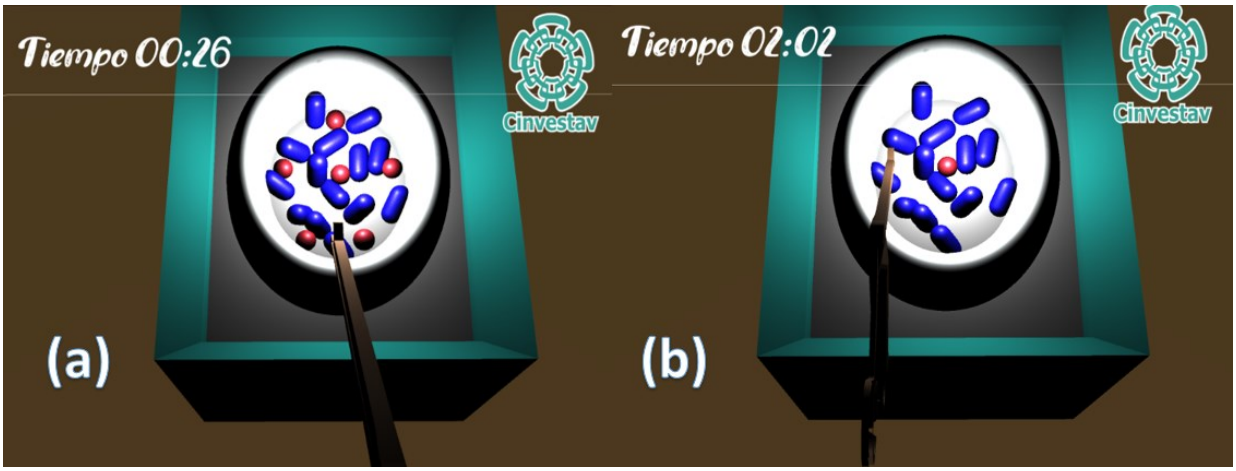



Fig. 34 Tarea de Disección. (a) Inicio de tarea sin haber tocado objetivos forma de esfera (b) 5 objetivos en forma de esfera tocados y desaparecidos.

## 5. Discusión

En la actualidad, los sistemas de entrenamiento o simuladores para la práctica quirúrgica han sido una excelente variante para la formación de médicos especialistas y el desarrollo de habilidades o la mejora de estas, en algunos casos ayudan a la certificación de habilidades, convirtiéndose en una herramienta muy importante de gran alcance en el ámbito médico y tecnológico.

Es importante mencionar que no se pretende que los simuladores sustituyan el método tradicional de aprendizaje de los especialistas, sino que en conjunto a intervenciones reales se pueda generar una retroalimentación adecuada y con ello el tiempo involucrado en dicho entrenamiento sea menor.

Tanto los sistemas de entrenamiento físicos como los simuladores virtuales se han implementado y utilizado obteniendo resultados favorables en el aprendizaje y mejora de habilidades psicomotoras en varios campos de la medicina, sin embargo, el costo de estos sistemas de entrenamiento puede llegar a ser elevado y no todos son costeables para los médicos especialistas y algunos hospitales en nuestro país.

Hoy en día, la tecnología avanza a pasos agigantados y con ello la constante búsqueda de soluciones a problemas para la mejoría de procesos médicos y tecnológicos. Algunos de estos procesos dejan de ser funcionales y se vuelven obsoletos por lo que la innovación es parte fundamental en el desarrollo de dicho ámbito, por lo que sistemas como este, al hacer uso de técnicas como la Realidad Virtual se vuelven una alternativa de aprendizaje muy completa ya que contienen una percepción hacia las técnicas, métodos e instrumentos en la formación a futuro.

Por otro lado, el proyecto fue diseñado para cubrir ciertas necesidades en estos tiempos de contingencia sanitaria. Es sabido que la actual crisis sanitaria afecta a todo el mundo principalmente las economías de cada país, es por eso por lo que dicho

proyecto busca la integración al mercado de plataformas virtuales con el fin de que sea de bajo costo y cumplir con las expectativas del sector médico y tecnológico.

Además, su portabilidad ayuda a ser una herramienta que se puede transportar fácilmente y con ello brindar un entrenamiento a los neurocirujanos en cualquier sitio donde se encuentren ya que solo se necesita de la plataforma (software) y de la articulación mecánica, con ello se evitan aglomeraciones en los centros de entrenamiento de hospitales y por ende contagios de COVID 19.

Uno de los grandes problemas en la práctica quirúrgica, es la desorientación espacial [22]. Se identificaron cuatro mecanismos de motricidad: distribución espacial, movimiento suave, buena percepción de distancias y profundidades y la respuesta en orientación. Las tareas implementadas en el sistema de entrenamiento virtual tienen el propósito de desarrollar la coordinación mano-ojo, destreza de la mano dominante y la percepción de distancias y profundidades. La velocidad del procedimiento se ha utilizado tradicionalmente como medida objetiva de habilidad [22].

Se ha observado que la mayoría de procedimientos usan como métrica principal el tiempo, sin embargo, este por sí solo no es suficiente para valorar las destrezas y habilidades del cirujano, las tasas de error también son fundamentales en la medida del criterio de habilidades específicas, por lo que las métricas de rendimiento cuantitativo que precisa nuestro sistema de entrenamiento virtual es el tiempo que le toma al especialista realizar la tarea de manera satisfactoria y el número de penalizaciones cometidas al ejecutar la tarea.

Esto último, demuestra la importancia de seguir innovando tanto en tecnología como en la ejecución de protocolos que optimicen los procesos de evaluación y entrenamiento para médicos especialistas en Neurocirugía.

# 6. Conclusiones y Perspectivas

El sistema de entrenamiento virtual ha cumplido con los objetivos en la implementación de las tres tareas a realizar (Ubicación Espacial, Navegación Espacial y Disección), permitiendo la evaluación de especialistas en Neurocirugía a través del tiempo. El sistema también permite identificar las penalizaciones cada que se cometa una falta en el proceso de interacción y con ello tener una tasa de error la cual retroalimentará a los especialistas en el desarrollo de sus habilidades psicomotoras.

El sistema de entrenamiento virtual en cuestión favorecerá en la práctica quirúrgica, ya que se buscan constantemente innovaciones que mejoren los protocolos de entrenamiento y evaluación. Además, el especialista podrá ser capaz de practicar y repetir una determinada tarea sin que esto represente un peligro para los pacientes en intervenciones quirúrgicas reales y los tiempos de recuperación postoperatorios sean mínimos.

Una cosa muy importante que mencionar es la optimización del sistema ya que representa la integración de las tres diferentes tareas por medio de la Realidad Virtual. Esto facilita el proceso de aprendizaje en muchos aspectos debido a que se trata de un sistema en el que puede ser ejecutado en cualquier ordenador por lo que no implica complicaciones referentes a el traslado de este.

Los avances médicos y tecnológicos cada día son mayores, con el paso del tiempo ha crecido la necesidad de crear innovación científica de vanguardia, es por esto que el proyecto tiene perspectivas a futuro ya que se pretende explotar en gran medida el campo de la Realidad Virtual, haciendo uso de sistemas propios de este campo como lentes de Realidad Virtual, dispositivos inalámbricos entre otras cosas que ayuden a facilitar los procesos de aprendizaje sin hacer de lado la comodidad y funcionalidad de los sistemas requeridos por parte del personal especialista.

# REFERENCIAS

[1] E. Ruelas, A. Lifshitz, M.A. Mercado, *Estado del Arte de la Medicina: Cirugía.,* Intersistemas, 2014, pp. 24-25.

[2] G. de Sola (2021, March 20). *Networks* [Online].
Available: https://neurorgs.net/informacion-al-paciente/temas-generales/introduccion-a-la-neurocirugia/

[3] The Center David Geffen School of Medicine (2021, February 12). *Networks* [Online]. Available: https://www.newrospine.com.mx/diferencia-entre-un-neurologo-y-un-neurocirujano/

[4] A. Cuevas, “Sistema de registro para la evaluación de las habilidades de mínima invasión en neurocirugía y otorrinolaringología en base a precisión y tiempo,” Ciencias. Maestría. Programa, Centro de Investigación y Estudios Avanzados del Instituto Politécnico Nacional, Ciudad de México, México, 2013.

[5] D. Reyes, A. González, C. Hidalgo, “Las competencias profesionales del cirujano en video cirugía”, vol. 16. No. 1, 2011.

[6] J. F. Nogueira, D. Nogueira, “Real models and virtual simulators in otolaryngology: review of literature”, Braz J Otorhinolaryngology. 2010;76 (1):129 35.

[7] J.R. Dieguez. (2010, Julio). IMPORTANCE OF VIRTUAL SIMULATORS IN TEACHING OF SURGERY MINIMALLY INVASIVE. Volume 10. No. 1 Revista Horizonte Medico.

[8] E. Bertranou, G. Cassanello, “Laboratorio de destrezas quirúrgicas hoy”, unpublished.

[9] R.L. Kneebone, “Practice, rehearsal and performance: anapproach for simulation-based surgical and procedure training”, JAMA. 2009; 302:1336-1338.

[10] Makary M. Daniel M “Medical error, the third leading cause of death in the US” Jons Hopkins University Schools Medicine. BMJ 2016;353: i2139 10.1136/bmj. i2139 (Publicado 3 Mayo del 2016).

[11] C. Moncerrat, et al. “Estado del Arte en Simulación Quirúrgica”, Informática y Salud, No.47. Junio 2004.

[12] Unity Hub, Productos, fecha de consulta [5 de Octubre del 2020 en adelante] se puede encontrar en: https://unity3d.com/es/get-unity/download

[13] *Storz El mundo de la endoscopia, Simuladores,* Storz - Karl Storz - Endoskope, 2018, pp. 1-2–1-3.

[14] F. Cremaschi, “Aportando a la Industria del Conocimiento de Mendoza al Mundo: Simulador en Realidad Virtual y con Tecnología Háptica para Neurocirugía,” in Barcelona Health Hub Summit 2020, Barcelona.

[15] Choy I, Okrainec A: Simulation in surgery: perfecting the practice. Surg Clin North Am 90:457–473, 2010

[16] Malone HR, Syed ON, Downes MS, D’Ambrosio AL, Quest DO, Kaiser MG: Simulation in neurosurgery: a review of computer-based simulation environments and their surgical applications. Neurosurgery 67:1105–1116, 2010

[17] E.S. Deutsch, “Simulation in otolaryngology: smart dummies and more”, Otolaryngology Head Neck Surg. 2011;145:899-903.

[18] M. García Berro y C. Toribio. Ciencias de la Salud El Futuro de la Cirugía Mínimamente Invasiva. Tendencias tecnológicas a medio y largo plazo. Cyan, Proyectos y Producciones Editoriales, S.A. Noviembre 2004.

[19] S. Cotin et al. “Metrics for Laparoscopic Skills Trainers: The Weakest Link! MICCAI 2002, LNCS 2488, pp. 35-43, 2002.

[20] N. Oshima et al. “Development of a Suture/Ligature Training System designed to provide quantitative information of the learning progress of trainees”, 2007 IEEE International Conference on Robotics and Automation Roma, Italy, 10-14 April 2007.

[21] Blender, Productos, fecha de consulta [11 de Enero del 2021 en adelante] se puede encontrar en: https://www.blender.org/about/

[22] H. Egi et al. “Objective Assessment of Endoscopic Surgical Skills by Analyzing Direction-Dependet Dexterity Using the Hiroshima University Endoscopic Surgical Assessment Device (HUESAD)”, Surg Today (2008) 38:705-710.

[23] P. Weinstock, R. Rehder, S.P. Prabhu, P.W. Forbes, C.J. Roussin, A.R. Cohen Creation of a novel simulator for minimally invasive neurosurgery: fusion of 3D printing and special effects J Neurosurg Pediatr, 20 (2017), pp. 1-9

[24] R.J. Garling, X. Jin, J. Yang, A.H. Khasawneh, C.A. Harris Low-cost endoscopic third ventriculostomy simulator with mimetic endoscope J Neurosurg Pediatr, 22 (2018), pp. 137-146

[25] M.C. Dewan, J. Onen, H. Bow, P. Ssenyonga, C. Howard, B.C. Warf Subspecialty pediatric neurosurgery training: a skill-based training model for neurosurgeons in low-resourced health systems Neurosurg Focus, 45 (2018), p. E2

[26] Rudolfson N, Dewan MC, Park KB, Shrime MG, Meara JG, Alkire BC: The economic consequences of neurosurgical disease in low- and middle-income countries. J Neurosurg 54:1–8, 2018

[27] B. Baby, V.K. Srivastav, R. Singh, A. Suri, S. Banerjee Neuro-endo-activity-tracker: An automatic activity detection application for neuro-endo-trainer: Neuro-endo-activity-tracker International Conference on Advances in Computing, Communications and Informatics (ICACCI), IEEE Proceedings, Jaipur, India (2016), pp. 987-993

[28] Kurashima Y, Hirano S. Systematic review of the implementation of simulation training in surgical residency curriculum. Surg Today. 2017;47(7):777–82.

[29] Thawani JP, Ramayya AG, Abdullah KG, Hudgins E, Vaughan K, Piazza M, et al. Resident simulation training in endoscopic endonasal surgery utilizing haptic feedback technology. J Clin Neurosci: official journal of the Neurosurgical Society of Australasia. 2016; 34:112–6.

[30] Fortes B, Balsalobre L, Weber R, Stamm R, Stamm A, Oto F, et al. Endoscopic sinus surgery dissection courses using a real simulator: the benefits of this training. Braz J Otorhinolaryngol. 2016;82(1):26–32.

# ANEXOS

## 7.1 Algoritmo en Visual Studio (Plataforma de programación de Unity)

### 7.1.1 Enviar Datos

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

public class Enviardatos : MonoBehaviour
{

    public SerialPort serialPort = new SerialPort("COM3", 9600);

    // Start is called before the first frame update
    void Start()
    {
        serialPort.Open();
        serialPort.ReadTimeout = 50;
    }

    // Update is called once per frame
    void Update()
    {

        try
        {
            if (serialPort.IsOpen)
            {
                string[] valor = serialPort.ReadLine().Split('-');

                print(valor[0]);
            }
        }

        catch
        {

        }
    }
}
```

### 7.1.2 Mover Gameobject

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

public class movergameobject : MonoBehaviour
{

    public float deltaRotation = 30f;
    public float deltaMovement = 10f;
    public Color color;


    // Start is called before the first frame update
    void Start()
    {
        GetComponent<Renderer>().material.color = color;
    }

    // Update is called once per frame
    void Update()
    {
        Movement();
    }


    void Movement()
    {
        if (Input.GetKey(KeyCode.S))
            transform.Translate(Vector3.forward * deltaMovement * Time.deltaTime);
        else if (Input.GetKey(KeyCode.W))
            transform.Translate(Vector3.back * deltaMovement * Time.deltaTime);
        else if (Input.GetKey(KeyCode.D))
            transform.Translate(Vector3.left * deltaMovement * Time.deltaTime);
        else if (Input.GetKey(KeyCode.A))
            transform.Translate(Vector3.right * deltaMovement * Time.deltaTime);
        else if (Input.GetKey(KeyCode.Q))
            transform.Translate(Vector3.up * deltaMovement * Time.deltaTime);
        else if (Input.GetKey(KeyCode.E))
            transform.Translate(Vector3.down * deltaMovement * Time.deltaTime);
    }


}
```

### 7.1.3 Player Controller

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

public class playerController : MonoBehaviour
{

    public float Movimientox;
    public float Movimientoy;
    public float Movimientoz;
    public float valorinicialx;
    public float valorinicialy;
    public float valorcA;
    public float valorcA2;
    public float movimientoxf;
    public float movimientoyf;
    public float speed = 10f;
    public CharacterController player;
    public SerialPort serialPort = new SerialPort("COM3", 9600);

    // Start is called before the first frame update
    void Start()
    {

        serialPort.Open();
        serialPort.ReadTimeout = 50;
        player = GetComponent<CharacterController>();
        valorinicialx = 971;
        valorinicialy = 578;

    }

    // Update is called once per frame
    void Update()
    {

        try
        {
            if (serialPort.IsOpen)
            {
                string[] movimientos = serialPort.ReadLine().Split('-');

                Movimientox = float.Parse(movimientos[0]);
                Movimientoy = float.Parse(movimientos[1]);
                Movimientoz = float.Parse(movimientos[2]);

                valorcA = Movimientox;
                movimientoxf = (-valorcA + valorinicialx) + transform.position.x;
                valorinicialx = valorcA;

                valorcA2 = Movimientoy;
                movimientoyf = (-valorcA2 + valorinicialy) + transform.position.y;
                valorinicialy = valorcA2;
```

```
                transform.localPosition = new Vector3 (movimientoxf, movimientoyf,
transform.position.z);

            }
        }

        catch
        {

        }

    }
}
```

### 7.1.4 Seguir figuras

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

public class Seguirfiguras : MonoBehaviour
{

    private float speed;
    public GameObject Cubo;
    Vector3 IP;
    public Rigidbody rb;



    // Start is called before the first frame update
    void Start()
    {
        IP = transform.position;
    }

    // Update is called once per frame
    void Update()
    {
    }


    private void FixedUpdate()
    {
        Vector3 target = IP;
        float dist = Vector3.Distance(Cubo.transform.position, transform.position);
        speed = dist * 5;
        target = Cubo.transform.position;
        float fixedspeed = speed * Time.deltaTime;
        rb.MovePosition(transform.position = Vector3.MoveTowards(transform.position,
target, fixedspeed));


    }

}
```

### 7.1.5 Seguir figuras Disección

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

public class Seguirfigurasdiseccion : MonoBehaviour
{

    public float speedD;
    public GameObject CuboD;
    Vector3 IPD;
    public Rigidbody rbD;



    // Start is called before the first frame update
    void Start()
    {
        IPD = transform.position;
    }

    // Update is called once per frame
    void Update()
    {
    }


    private void FixedUpdate()
    {
        Vector3 target = IPD;
        float dist = Vector3.Distance(CuboD.transform.position, transform.position);
        speedD = dist * 5;
        target = CuboD.transform.position;
        float fixedspeed = speedD * Time.deltaTime;
        rbD.MovePosition(transform.position = Vector3.MoveTowards(transform.position,
target, fixedspeed));


    }

}
```

### 7.1.6 Ubicación Espacial

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;
using UnityEngine.UI;
using System;
using System.IO.Ports;

public class Ubicacionespacial : MonoBehaviour
{
    public int contador;

    // Start is called before the first frame update
    void Start()
    {


    }

    // Update is called once per frame
    void Update()
    {

    }

    private void OnCollisionEnter (Collision col)
    {
        if (col.gameObject.name == "tornillo-1")
        {
            Destroy(col.gameObject);
            contador++;
        }
        if (col.gameObject.name == "tornillo-2")
        {
            Destroy(col.gameObject);
            contador++;
        }
        if (col.gameObject.name == "tornillo-3")
        {
            Destroy(col.gameObject);
            contador++;
        }
        if (col.gameObject.name == "tornillo-4")
        {
            Destroy(col.gameObject);
            contador++;

        }


    }


}
```

### 7.1.7 Navegación Espacial

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;
using UnityEngine.UI;
using System;
using System.IO.Ports;

public class Navegacionespacial : MonoBehaviour
{
    public bool final = false;


    // Start is called before the first frame update
    void Start()
    {

    }

    // Update is called once per frame
    void Update()
    {

    }

    private void OnCollisionEnter(Collision collision)
    {

        if (collision.gameObject.name == "Cubofinalnav")
        {
            final = true;
        }



    }


}
```

### 7.1.8 Disección

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;
using UnityEngine.UI;
using System;
using System.IO.Ports;

public class Diseccion : MonoBehaviour
{
    private Collision collision;
    public int contador;
    private void OnCollisionEnter(Collision col)
    {
        if (col.gameObject.name == "esferadiseccion-1")
        {
            col.gameObject.GetComponent<Renderer>().material.SetColor("_Color", new
Color(160, 170, 10, 100));
            StartCoroutine(Esperar());
            contador++;

        }
        if (col.gameObject.name == "esferadiseccion-2")
        {
            col.gameObject.GetComponent<Renderer>().material.SetColor("_Color",
Color.cyan);
            StartCoroutine(Esperar());
            contador++;

        }
        if (col.gameObject.name == "esferadiseccion-3")
        {
            col.gameObject.GetComponent<Renderer>().material.SetColor("_Color",
Color.cyan);
            StartCoroutine(Esperar());
            contador++;

        }
        if (col.gameObject.name == "esferadiseccion-4")
        {
            col.gameObject.GetComponent<Renderer>().material.SetColor("_Color",
Color.cyan);
            StartCoroutine(Esperar());
            contador++;
        }
        if (col.gameObject.name == "esferadiseccion-5")
        {
            col.gameObject.GetComponent<Renderer>().material.SetColor("_Color",
Color.cyan);
            StartCoroutine(Esperar());
            contador++;

        }
        if (col.gameObject.name == "esferadiseccion-6")
        {
```

```
            col.gameObject.GetComponent<Renderer>().material.SetColor("_Color",
Color.cyan);
            StartCoroutine(Esperar());
            contador++;

        }
        {
            Debug.Log(contador);
            collision = col;

        }
    }

    private IEnumerator Esperar()
    {
        yield return new WaitForSeconds(3);
        Debug.Log("3segundos");
        Destroy(collision.gameObject);
    }

}
```

### 7.1.9 Penalización Disección

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

public class Penalizacion : MonoBehaviour
{

    public Text textElement;

    // Start is called before the first frame update
    void Start()
    {
        textElement.enabled = false;
    }

    // Update is called once per frame
    void Update()
    {

    }

    private void OnCollisionStay(Collision collision)
    {
        if (collision.gameObject.CompareTag("capsula"))
        {
            textElement.enabled = true;
        }
    }

    private void OnCollisionExit(Collision collision)
    {
        if (collision.gameObject.CompareTag("capsula"))
        {

            textElement.enabled = false;
        }
    }


}
```

## 7.2 Penalización Navegación Espacial

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

public class PenalizacionNavegacion : MonoBehaviour
{
    public Text textElement;

    // Start is called before the first frame update
    void Start()
    {
        textElement.enabled = false;
    }

    // Update is called once per frame
    void Update()
    {

    }

    private void OnCollisionStay(Collision collision)
    {
        if (collision.gameObject.CompareTag("patron"))
        {
            textElement.enabled = true;
        }
    }

    private void OnCollisionExit(Collision collision)
    {
        if (collision.gameObject.CompareTag("patron"))
        {

            textElement.enabled = false;
        }
    }


}
```

### 7.2.1 Penalización Ubicación Espacial

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

public class PenalizacionUbicacion : MonoBehaviour
{
    public Text textElement;

    // Start is called before the first frame update
    void Start()
    {
        textElement.enabled = false;
    }

    // Update is called once per frame
    void Update()
    {

    }

    private void OnCollisionStay(Collision collision)
    {
        if (collision.gameObject.CompareTag("obstaculo"))
        {
            textElement.enabled = true;
        }
    }

    private void OnCollisionExit(Collision collision)
    {
        if (collision.gameObject.CompareTag("obstaculo"))
        {

            textElement.enabled = false;
        }
    }


}
```

### 7.2.2 Cronómetro

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

public class TiempoAdelante : MonoBehaviour
{
    public static TiempoAdelante instanciar;
    public Text crono;
    private TimeSpan tiempoCrono;
    private bool timerBool;
    public float tiempoTrans;

    private void Awake()
    {
        instanciar = this;
    }

    // Start is called before the first frame update
    void Start()
    {
        crono.text = "Tiempo 00:00";
        timerBool = false;
    }

    public void Iniciartiempo()
    {
        timerBool = true;
        tiempoTrans = 0f;
        StartCoroutine(ActUpdate());
    }

    public void Fintiempo()
    {
        timerBool = false;
    }
    // Update is called once per frame
    private IEnumerator ActUpdate()
    {
        while (timerBool)
        {
            tiempoTrans += Time.deltaTime; //Servira para calculos
            tiempoCrono = TimeSpan.FromSeconds(tiempoTrans);
            string tiempocronostr = "Tiempo " + tiempoCrono.ToString("mm':'ss");
            crono.text = tiempocronostr;
            yield return null;
        }
    }
}
```

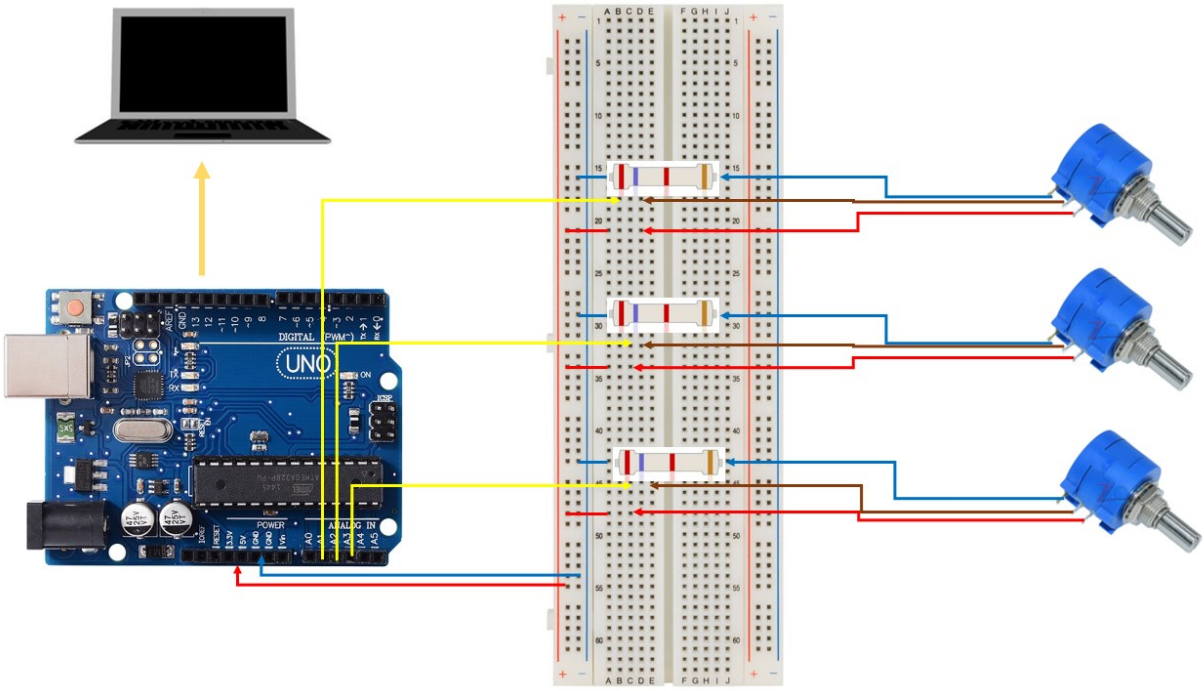
### 7.2.3 Cuenta regresiva

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

public class TimerAtras : MonoBehaviour
{
    public float tiemporestante;
    public bool tiempocorriendo = false;
    public GameObject textotimer;
    public Text texto;
    // Start is called before the first frame update
    void Start()
    {
        tiempocorriendo = true;
    }

    // Update is called once per frame
    void Update()
    {
        if (tiempocorriendo)
        {
            if (tiemporestante > -1)
            {
                texto.text = tiemporestante.ToString().Split('.')[0];
                tiemporestante -= Time.deltaTime;
                if (tiemporestante <= 0)
                {
                    texto.text = "Inicia";
                    texto.color = new Color(255, 245, 00);
                }
            }
            else
            {
                textotimer.SetActive(false);
                tiemporestante = 0;
                tiempocorriendo = false;
                TiempoAdelante.instanciar.Iniciartiempo();
            }
        }
    }
}
```

### 7.2.4 Main Menú

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

public class MainMenu : MonoBehaviour
{

    public void EscenaEntrenador()
    {
        SceneManager.LoadScene("puertoserieUnityArduino");
    }

    public void Salir()
    {
        Application.Quit();
    }


}
```

### 7.2.5 Diagrama electrónico conexión potenciómetros de precisión de 50 Kohm +- 5% modelo 3590S-2-503L



Conexiones de potenciómetros de precisión

| | |
|---|---|
| Azul | Tierra |
| Café | Indicador |
| Rojo | Positivo |

- 3 Resistencias de 2.7 Kohm 5%

Cada potenciómetro es para cada uno de los tres grados de libertad:

- Eje x (izquierda-derecha).
- Eje y (arriba-abajo).
- Eje z (adentro-afuera).