UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

Simulador laparoscópico de realidad virtual para el entrenamiento de habilidades psicomotrices de los cirujanos

TESIS

Que para obtener el título de

Ingeniero Mecatrónico

P R E S E N T A

Eduardo Valentin Ruiz Rojas

DIRECTOR DE TESIS

Dr. Fernando Pérez Escamirosa

Ciudad Universitaria, Cd. Mx., 2018

A mi familia,
por apoyarme en todo momento.

A mis amigos,
quienes siempre me motivaron.

A la UNAM,
un lugar asombroso lleno de inspiración, conocimiento, alegría y grandes amigos

AGRADECIMIENTOS

Para comenzar quiero brindar un agradecimiento al doctor Fernando Pérez Escamirosa, quien fungió como asesor de este trabajo, por darme la oportunidad de trabajar en el proyecto, también agradezco todo el apoyo que me otorgo y el conocimiento compartido, gracias por su tiempo y paciencia.

En segunda instancia, agradezco a la Facultad de Medicina de la UNAM, más en particular al Departamento de Cirugía por facilitarme los recursos que fueron de gran importancia a lo largo del proyecto. Otra institución con la cual estoy enormemente agradecido es con la Facultad de Ingeniería de la UNAM en donde se impulsaron mis conocimientos y mi desarrollo académico, que durante todo mi tiempo de estudio ha sido un segundo hogar para mí, que me ha brindado tantos momentos de inspiración, así como una formación profesional, ética, social y cultural un lugar que me permitió conocer grandes personas a las que tengo el gusto de llamar amigos.

Como tercer punto, agradezco a mis padres, Guadalupe del Rocío Rojas Salas y Valentin Ruiz Mancera los cuales día tras día me brindaron todo su apoyo, me motivaron, dieron lo mejor de ellos para que yo pudiera dar lo mejor de mí, por todos los consejos, por su conocimiento y experiencia que lograron inspirar e impulsar varios de mis proyectos y también por ayudarme en los trabajos y desvelase conmigo si era necesario con el fin de terminarlos. Del mismo modo, un gran agradecimiento para mis abuelos Ofelia Mancera Zepeda, Alfonso Ruiz Monroy, Felicitas Salas Aguirre y Antonio Rojas Estrada, que cada uno de ellos aporto su ayuda de la mejor forma que pudo a lo largo de mi vida, que cuidaron de mi cuando fue necesario y que siempre han estado ahí para ofrecer su ayuda incondicional. Gracias a todos por su guía.

Otro agradecimiento es para a mis amigos, esas personas que encontré en el camino, las cuales me compartieron su conocimiento, me hicieron pasar buenos momentos, me impulsaron a seguir adelante cuando quería darme por vencido, por compartir tantas experiencias que lograron hacerme crecer de muchas formas, les agradezco por escucharme y realizar el papel de consejeros en el momento que estuve débil tanto académica como emocionalmente del mismo modo gracias por brindarme un espacio en donde los problemas se hacían insignificantes, por lo menos un rato.

Finalmente les comparto que en el pasado sembré una planta cuyas raíces eran amargas, hoy se ha convertido en un árbol y gracias a todos puedo disfrutar del fruto dulce que me dio y compartirlo con ustedes. ¡Gracias Zaira, Yessi, Fernanda, Melissa, Janet, Paola, Gaby, Perlita, Elena, Rafael, Emanuel, Germán, Ricardo y a mi familia por apoyarme hasta los últimos instantes!

Algunas cosas del pasado desaparecen, otras abren una brecha hacia el futuro un futuro que puede dar miedo, sin embargo, es ahí donde se desarrollan nuestros objetivos, nuestros amigos son verdaderos y nuestra felicidad segura, por eso avanzare hacia él con valor y optimismo.

Índice

Índice de figuras.

Resumen

En el presente documento, se muestra la realización de un simulador virtual para la adquisición de habilidades psicomotrices de los cirujanos en laparoscopía; una técnica quirúrgica de mínima invasión la cual ha incrementado su demanda entre los cirujanos expertos en los últimos tiempos.

El desarrollo de *hardware* de este simulador está compuesto por un mecanismo donde se instrumentaron electrónicamente los instrumentos laparoscópicos que controlan la interfaz virtual del programa mediante los movimientos, repartidos en las dos pinzas laparoscópicas convencionales. Asimismo, hablaremos de los dispositivos empleados para capturar los movimientos reales generados por los cirujanos, su procesamiento mediante el microcontrolador Arduino DUE y redirección o codificación a la interfaz gráfica del simulador.

En esta tesis, se explica la creación del simulador el cual está dividido en dos grandes partes. La primera enfocada a las tareas laparoscópicas, donde se cuenta con 5 actividades distintas que varían en dificultad. La segunda parte del simulador laparoscópico está relacionada con la interacción del usuario, su registro para el seguimiento de sus avances y la administración de los datos. El desarrollo del *software* fue realizado en el motor gráfico de *Unity* del cual mencionaremos algunos aspectos esenciales para su uso.

Por último, presentaremos los resultados del desempeño gráfico y mecánico del simulador mediante pruebas realizadas en distintos equipos de cómputo con características diferentes y algunas evaluaciones de los voluntarios. Del mismo modo, se mostrarán las evaluaciones obtenidas por los voluntarios, donde se observará un pequeño análisis de los datos.

1. Introducción.

Para proporcionar experiencia quirúrgica a los alumnos y a los que se encuentran en sus estancias médicas en la cirugía laparoscópica, la Facultad de Medicina de la UNAM cuenta con distintas prácticas que les proporcionan experiencia y les ayuda a familiarizarse con los procedimientos y sus técnicas. Sin embargo, algunas de las experiencias son únicas, esto tratándose de eventos no repetitivos como lo es la operación en cadáveres, que en muchos de estos casos el que realiza la operación es un cirujano superior con mayor experiencia, mientras que los alumnos solo observan y tratan de aprender de la cirugía.

Este método de enseñanza se debe a la gran cantidad de estudiantes e internos que necesitan practicar y debido a que los cadáveres no se regeneran solo se puede realizar la misma operación una vez. Además, las universidades no pueden proporcionar un cadáver distinto para cada practicante. Dada esta situación, se requiere contar con diferentes herramientas para poder satisfacer la demanda de uso. Para lograrlo, existen los simuladores virtuales en los cuales la Facultad de Medicina y otras Universidades tienen un gran interés desde distintos puntos de vista, entre los cuales destaca la facilidad de repetir los eventos y que todos los estudiantes y residentes quirúrgicos puedan realizar la misma tarea bajo las mismas condiciones. Sin embargo, las herramientas que proporcionan este tipo de experiencias no son de uso libre, por lo tanto, mantener la licencia del programa como el equipo en sí resultaría demasiado costoso.

El término de laparoscopia fue empleado por primera vez en 1910 en Estocolmo por H. C. Jacobeus quien retoma el experimento de George Kelling, el cual consistió en emplear el cistoscopio y explorar el contenido intestinal en un perro. Tiempo después, H. C. Jacobeus tras distender la cavidad abdominal en humanos con agua o aire indistintamente, logra inspeccionar su interior y denomina a este método “Laparoscopía”, el cual a través de los años se fue perfeccionando y actualizando hasta alcanzar la precisión con la que hoy lo conocemos [1]. La laparoscopia es una técnica para examinar los órganos abdominales y para tratar quirúrgicamente muchas enfermedades. El instrumento utilizado es llamado laparoscopio, el cual es un tubo flexible que contiene una fibra óptica para propósitos de visualización y un canal a través del cual los médicos pueden pasar instrumentos quirúrgicos especiales dentro de la cavidad abdominal [2].

Por otra parte, derivado de los cambios que ocurren en los sistemas de salud alrededor del mundo, se ha notado una creciente demanda del entrenamiento quirúrgico fuera de la sala de operaciones. Los reglamentos aprobados por la ACGME (*Accreditation Council of Graduate Medical Education*) en 2003 han restringido el número de horas de trabajo quirúrgico que se pueden trabajar, lo que requiere que los nuevos cirujanos llegar a ser competentes en un corto período de tiempo. Además, el alto costo del espacio de la sala de operaciones, junto con la disminución de los reembolsos, han limitado aún más el entrenamiento basado en la sala de operaciones. En consecuencia, ha habido presiones para desarrollar modelos más eficientes de entrenamiento quirúrgico que el diseño tradicional de aprendizaje de los programas de residencia quirúrgica. Cabe añadir, el énfasis creciente en la seguridad del paciente ha limitado aún más la experiencia del entrenamiento de los cirujanos principiantes en el quirófano, lo que requiere el desarrollo de estrategias de capacitación que no involucren a pacientes reales. Por lo anterior, el uso de simuladores en el entrenamiento quirúrgico moderno ha aumentado considerablemente [3].

Lo anterior, no exime a la cirugía laparoscópica ya que, desde su implementación, la demanda de cirujanos expertos en este método ha ido en aumento, por lo que la adquisición de la destreza laparoscópica tiene especial interés. La destreza requerida para la cirugía laparoscópica no es innata, necesita ser adquirida y perfeccionada con el entrenamiento, ya que son pobremente aprendidas al ser mimetizadas con la observación o por el análisis de un texto escrito debido a su dificultad y a su naturaleza no intuitiva [4]. En los últimos tiempos, el desarrollo de nuevas tecnologías es un campo ampliamente explorado. Por este motivo, se hacen investigaciones las cuales validan la adquisición de habilidades por parte de los practicantes, tal es el caso del estudio donde fue tomada una muestra de cirujanos generales, de los cuales ninguno supero los 65 puntos de la prueba de evaluación previa al entrenamiento en un simulador virtual. Una vez completado el entrenamiento basado en once ejercicios con tareas diseñadas para desarrollar distintas capacidades y habilidades, los cirujanos lograron mejorar considerablemente su destreza, donde al final incrementaron su puntaje promedio inicial de 304.9 a 834.2 puntos. Por lo tanto, el entrenamiento sostenido en simuladores virtuales resulta en una mejora significativa en las habilidades laparoscópicas [5].

Hoy en día, sabemos que la simulación es buena para el entrenamiento de los cirujanos. La resección de la vesícula biliar, por ejemplo, es hasta un 30% más rápida con los residentes entrenados por simuladores. Cuando se trabaja con estos dispositivos, los aprendices pueden practicar repetidamente técnicas y manejar complicaciones hasta que logren experiencia en la realización de la operación simulada [6]. Como resultado, las simulaciones quirúrgicas ayudan en el desarrollo de habilidades psicomotoras, técnicas y de juicio crítico [3]. Cuando se compara los residentes entrenados en simuladores contra aquellos sin entrenamiento de simulación, estos últimos tienen hasta 5 veces más probabilidades de lastimar o incluso quemar tejido no objetivo. Como era de esperarse, en 2006, el Comité de Revisión de Residencias para la Cirugía votó por unanimidad el mandato de entrenamiento de simulación en programas de residencia de cirugía [6].

En el presente documento, realizaremos un simulador virtual para el entrenamiento de los estudiantes en formación en cirugía laparoscópica para su uso en la Facultad de Medicina de la UNAM. Con el fin de proporcionar una herramienta valida en un nivel académico y práctico. Este simulador cuenta con actividades normalizadas, de este modo, proporciona un acercamiento al practicante con el instrumental médico para que se familiaricen y entienden los alcances de estos. Para observar sus habilidades, notar sus posibles errores y llevar a cabo un registro del progreso del practicante, el simulador registra los movimientos realizados durante las tareas de entrenamiento, los cuales serán un material de ayuda para el profesor donde podrá mostrar al residente los puntos en los que tiene que mejorar. Finalmente, este simulador es de *hardware* y *software* libre para facilitar su uso en la facultad. Al final de este trabajo se tendrá un simulador virtual con tareas estandarizadas para su uso como una herramienta educativa.

1.1 Planteamiento del problema

Un cirujano requiere aprenden distintas habilidades y destrezas laparoscópicas antes de poder realizar procedimientos directamente en los pacientes de los hospitales y centros de salud. Dicho entrenamiento debe consistir en un mínimo de horas; sin embargo, en México mucho de este tiempo lo realizan directamente en los pacientes, por lo que es indispensable implementar otro tipo de herramientas para la adquisición de esta experiencia. La implementación de simulación quirúrgica ayuda a la adquisición de habilidades laparoscópicas en dicho entrenamiento de una forma segura, repetible y con diversas ventajas como pueden ser educativas en cuanto a evaluación arbitraria o economías al minimizar la compra de insumos médicos requeridos. Por lo tanto, es importante invertir en el desarrollo de tecnología que permita la práctica ya sea de cirujanos, estudiantes o pasantes todo bajo un ambiente controlado. Por lo que, realizaremos un simulador virtual para el entrenamiento de los estudiantes para su uso en la Facultad de Medicina de la UNAM, el cual pueda ser una herramienta valida en un nivel académico y práctico.

1.2 Justificación

En México, los simuladores de este tipo no suelen estar a la mano de todos los cirujanos y residentes, siendo pocas las instituciones, académicas u hospitalarias que cuentan con estos equipos. Del mismo modo, su uso está restringido ya sea por demanda de uso o incluso por averías las cuales pueden tardar tiempo en ser solucionadas. Por esta razón, se presenta este simulador con el fin de tener una herramienta valida con distintas prácticas y ejercicios que les proporcionen experiencia y les ayuda a familiarizarse con el procedimiento quirúrgico y sus técnicas, todo gracias a la implementación de dispositivos comerciales que permiten el desarrollo sin restricciones de licencias como lo son Arduino y *Unity*.

1.3 Objetivo general

Diseño y programación de un simulador laparoscópico basado en realidad virtual y gráficos por computadora, que permita el entrenamiento y la evaluación de las habilidades psicomotrices de los cirujanos, residentes y estudiantes de medicina.

1.3.1 Objetivos particulares

- Diseñar tareas que permitan la adquisición de distintas habilidades quirúrgicas necesarias en la cirugía laparoscópica.
- Implementar una interfaz gráfica de usuario que permita el registro de los usuarios en la aplicación.
- Realizar la evaluación del desempeño de los usuarios y guardar su registro para conocer su progreso.
- Construir un mecanismo para montar los instrumentos laparoscópicos y hacer la comunicación con la interfaz virtual.

1.4 Capitulado

Estado del arte. En este capítulo, se presenta el estado del arte sobre los diferentes tipos de simuladores laparoscópicos que se encuentran en la actualidad, desde los simuladores de baja fidelidad, así como los que incorporan realidad virtual. Se mencionan las características que estos presentan y su forma de interactuar con ellos, así mismo se mencionan algunos de los parámetros que se evalúan en los simuladores.

Metodología. En este capítulo, se muestra el desarrollo del simulador y las partes que lo componen, las cuales van desde el mecanismo físico, que funciona como un mando de control que permitirá la manipulación del *software*. Se hablará sobre los dispositivos electrónicos empleados y su funcionamiento, así mismo, aunaremos con detalle el diseño gráfico del simulador, las tareas virtuales de laparoscópica con las que cuenta y su funcionamiento lógico.

Resultados. En este capítulo, se explica el funcionamiento final del simulador, haciendo un análisis del desempeño tanto de los gráficos como del realismo de los movimientos y las tareas laparoscópicas, así mismo se hace un análisis de los requisitos mínimos requeridos. También, se presentarán las pruebas y resultados de algunos voluntarios que ocuparon este dispositivo y la evaluación que realizaron al simulador laparoscópico. Finalmente, se presentan las conclusiones del trabajo y el trabajo a futuro a desarrollar.

Conclusiones. En esta última parte se presentan las conclusiones del trabajo, el cumplimiento de los objetivos, así como la interpretación de los resultados obtenidos por los participantes. Además, se define el trabajo a futuro con las propuestas de mejora del simulador.

2. Estado del Arte.

En este capítulo, se presentará una descripción a forma de preámbulo sobre los diferentes tipos de simuladores laparoscópicos, desde los simuladores de baja fidelidad, así como los que incorporan realidad virtual. Además, se mencionarán las características que estos presentan y su forma de interactuar con el usuario.

2.1 Tipos de simuladores para la cirugía laparoscópica.

Los simuladores laparoscópicos de pueden clasificar dependiendo de los procedimientos que se vayan a realizar ya sean de tareas básicas hasta procedimientos especializados; sin embargo, puede decirse que esto es una subclasificación de los simuladores ya que podemos encontrar tareas específicas para distintos tipos de entrenadores. A continuación, se presenta una clasificación más general en la que podemos dividir a estos simuladores en:

- Simuladores tradicionales.
- Simuladores con realidad virtual.

En primera instancia los simuladores tradiciones, también conocidos como entrenadores de caja, proveen un contenedor que simula la cavidad abdominal de un paciente (figura 2.1-1). El simulador recrea un ambiente quirúrgico con las condiciones operativas utilizando componentes básicos como una videocámara, una fuente de luz y un monitor. La cámara de los entrenadores de caja se puede conectar a una computadora o un monitor [7]. Este tipo de simuladores presenta ciertas ventajas como pueden ser de portabilidad y sensaciones hápticas,[1] por otro lado, se ven afectados los aspectos como el consumo de insumos, ya que este tipo de entrenadores requieren de material de trabajo el cual termina siendo de desperdicio, otro aspecto negativo se encuentra en la evaluación del practicante pues esta se basa en la observación y supervisión de un cirujano experto, dando lugar a una evaluación subjetiva.

Antes de continuar con la otra categoría tenemos que definir que es la realidad virtual. La realidad virtual, nombrada con frecuencia como “VR”, se puede definir como una simulación de un entrono que coloca al usuario dentro de una experiencia donde se busca que se involucren el mayor número de sensaciones posibles, a través de sonidos, gráficos, e instrumentos mecánicos con el que el usuario se introduce y controla al sistema. De la anterior, es de donde sale la siguiente categoría, en la cual encontramos los simuladores con realidad virtual.

Como breve introducción, podemos destacar que estos simuladores tienen algunas ventajas con respecto a los anteriores, esto dependiendo del tipo de simulador ya que se pueden encontrar simples, sin sistemas de realimentación al usuario como actuadores o complejos los llamados con sensación háptica. Su versatilidad radica en la repetibilidad de procesos, el análisis de los movimientos e incluso la recreación de procedimientos quirúrgicos completos. Aunaremos más del tema en el apartado 2.2.

[1] Háptica: Es el conjunto de sensaciones no ópticas y no acústicas que experimenta un individuo, lo relativo al tacto.

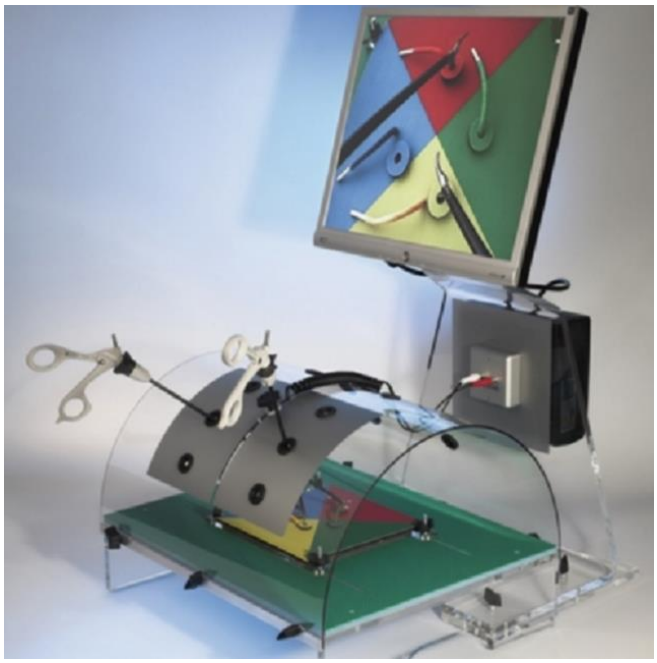

Figura 2.1-1.*Entrenador tradicional. [8]*

2.2 Simuladores laparoscópicos con realidad virtual.

Como ya se mencionó con anterioridad este tipo de simuladores cuentan con una interfaz gráfica donde el usuario interactúa visualmente, en conjunto con esto es necesario tener un manipulador instrumentado mediante sensores el cual permanezca activo durante el proceso. Dichos sensores tienen que ser colocados en el instrumento laparoscópico con el fin de medir todas las combinaciones de entrada que se necesiten representar. En estos dispositivos es necesario hacer el modelado de los distintos elementos con los que se interactúan, desde los más básicos como son las herramientas quirúrgicas hasta los más complejos como pueden ser los tejidos y órganos con los cuales se entrena.

Un aspecto altamente positivo en este tipo de entrenadores es la realimentación al usuario sobre su desempeño mediante una calificación otorgada después de realizar alguna tarea. Este parámetro es analizado por el sistema bajo métricas estandarizadas, lo que provee una estadística de control y avances del cirujano Cabe mencionar que el realismo del simulador se disminuye ya que estos dispositivos no poseen de una sensación táctil real; además de que carecen de una deficiencia en el momento de representar los órganos puesto que a veces no sangran o el material como hilo de sutura, o gasas se queda suspendido en el espacio. Existen diversos tipos de entrenadores ya usados en muchos hospitales del mundo, algunos de ellos se mencionarán a continuación.

2.2.1 Simulador LAP-X VR.

LAP-X VR es un simulador laparoscópico perteneciente a la compañía Medical-X con sede en Rotterdam, Países Bajos. Este entrenador es portátil, de rápido armado, dado que solo se coloca la base y se ponen montan las pinzas quirúrgicas, también proporciona una utilización óptima de espacio según su

fabricante. El simulador LAP-X (figura 2.2-1) cuenta con el soporte de 2.7 kg requerido para la adaptación de las pinzas laparoscópicas, dos pedales empleados para diversas funciones dentro de una operación y una interfaz gráfica con tareas básicas, intermedias, avanzadas y de procedimiento quirúrgico.

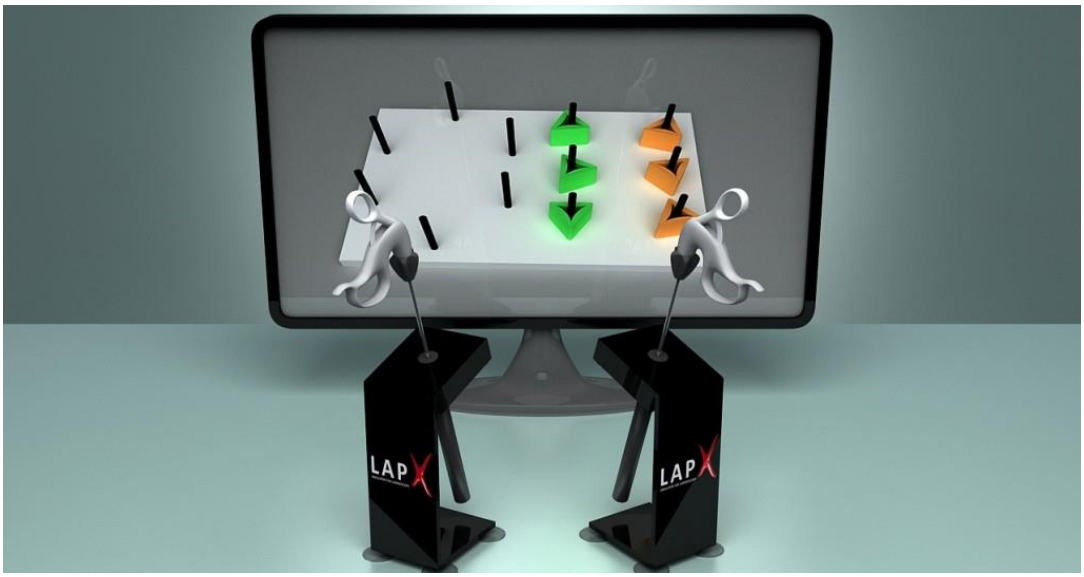

Figura 2.2-1. *Simulador LAP-X VR [9].*

Algunas de las tareas que se pueden realizar en el simulador son:

I. Tareas básicas
   - Control de cámara con diferentes ángulos de visión.
   - Coordinación de la mano en el espacio.
   - Habilidades de control de instrumentos.
   - Coordinación del instrumento en el espacio.
   - Manejo de tijera endoscópica.
   - Movimiento de objetos en el espacio.
   - Captura de objetos en el espacio.
   - Modo de examen y entrenamiento.

II. Tareas de sutura
   - Nudo quirúrgico intracorpóreo con la mano izquierda y con la mano derecha.
   - Técnica de sutura interrumpida.
   - Orientación de la aguja en el porta-agujas.
   - Aguja suturando con la mano izquierda y con la mano derecha.

III. Procedimientos quirúrgicos completos
   - Colecistectomía.
   - Apendectomía.
   - Hernioplastía.
   - Histerectomía.

En cuanto a su programa de *software* con el que cuenta el simulador, este se encuentra en línea con la ventaja de no requerir espacio de almacenamiento sino solo de una conexión a la red, dando como desventaja la necesidad de estar conectados mediante una red de buena velocidad para no ver entorpecido el sistema. Por otro lado, cuentan con un sistema compatible exclusivo con Windows y como requisito mínimo del sistema recomiendan un procesador Intel i5 o similares 2 GB de RAM o superior y una tarjeta gráfica independiente [10].

2.2.2 Simulador Lap Mentor III.

Lap Mentor III es un simulador desarrollado por Simbionix Corp, EU, el cual ofrece una amplia variedad de entrenamiento laparoscópico práctico en múltiples disciplinas. Proporciona un plan de estudios estructurado con diferentes niveles de dificultad para las tareas y habilidades laparoscópicas básicas junto con la capacitación de procedimientos completos básicos y avanzados (figura 2.2-2).

Figura 2.2-2. *Lap Mentor III [11].*

El sistema presenta 17 módulos de capacitación y más de 70 tareas y casos, que incluyen cirugía general, ginecológica, urológica, bariátrica, colorrectal y torácica. Los módulos complementarios se están desarrollando continuamente al ritmo de los avances quirúrgicos en curso, para proporcionar el mayor valor posible del sistema LAP Mentor. A su vez, cuenta con un alto nivel de capacitación práctica que se ofrece a través de dos plataformas LAP Mentor que están disponibles con o sin hápticas avanzados. La

experiencia de herramientas quirúrgicas táctiles más realistas que proporcionan una sensación real de la resistencia de los tejidos durante la simulación quirúrgica [12].

2.2.3 Simulador LapSim.

El simulador LapSim (*Laparoscopic Simulator*), es un entrenador que permite al estudiante practicar diversas tareas que van desde la navegación, tareas intermedias de coordinación ojo-mano, navegación, corte engrapado entre otras. Los desarrolladores de este simulador cuentan con dos versiones, estas se muestran en la figura 2.2-3, la diferencia de estas radica en el sistema háptico. Cabe mencionar que este sistema es embebido, en otras palabras, que incluye la parte de hardware y el sistema de *software* dentro del módulo físico.

Figura 2.2-3. *Simuladores LapSim [13].*

La inclusión de tareas no es la excepción en este producto ya que cuenta con distintos paquetes de ejercicios ya sean para evaluación o a manera de entrenamiento. En la forma de entrenamiento, a diferencia de la evaluación, el programa te va guiando y proporcionando algunos consejos y sugerencias con los que puedas realizar de manera correcta las tareas. También, se incluyen procedimientos médicos completos precargados y permite la opción de adquirir nuevos procedimientos por algún costo adicional. Este simulador trabaja con una resolución de sensores de 22 micrómetros de traslación y 0.26° de rotación; sin embargo, no mide la fuerza aplicada por el usuario, cuenta con cuatro grados de libertad, los cuales son medidos a partir de transductores electromecánicos montados en el mecanismo, un movimiento adicional es la apertura y cierre de la pinza, la cual la consideran por separado de los anteriores grados de libertad [14].

2.2.4 Simulador SIMENDO.

El simulador SIMENDO consiste en una caja de un kilogramo de peso, y sus dimensiones son 10x10x40 cm como se muestra en la figura 2.2-4. El *software* está integrado en el sistema y proporciona la capacidad de

conexión “plug-and-play” a través de un puerto USB. Los requerimientos mínimos de computadora necesarios son un procesador de 722mHz, 128MB de RAM, una tarjeta gráfica estándar y Microsoft Office (con acceso a base de datos) [15].

Figura 2.2-4. *Simulador SIMENDO [15].*

Los ejercicios en el programa de entrenamiento están diseñados para entrenamiento de la coordinación mano-ojo, usando tareas abstractas sin comentarios forzados. El entrenamiento comienza con una breve explicación teórica de las dificultades que enfrenta un cirujano durante los procedimientos endoscópicos. [15].

2.3 Parámetros para el análisis del desempeño del cirujano.

La evaluación de los residentes y cirujanos novatos en esta técnica laparoscópica, mediante el método tradicional, está basado en un modelo en el cual un cirujano experto subjetivamente evalúa el rendimiento del novato. Los cirujanos expertos son capaces de evaluar el rendimiento de un principiante observándolo en el quirófano el control del movimiento de la parte visible de los instrumentos que se han introducido en la cavidad abdominal. Con base en esta información y el resultado de la tarea quirúrgica, el cirujano puede realizar una caracterización cualitativa del rendimiento general del novato en los distintos parámetros clave. [16].

Al momento de realizar dicha evaluar en los simuladores, se emplean diversos parámetros los cuales se calculan a partir de las posiciones en x, y, z de la punta de la pinza laparoscópica. Los cálculos resultan particularmente complicados cuando se trata de un simulador tradicional ya que se tienen que emplear métodos como visión artificial, donde se sigue el movimiento de la punta a través de marcadores de color posteriormente se hace el cálculo con un *software* adicional. Sin embargo, este proceso resulta más sencillo al tener todo inmerso en el mismo programa como es el caso de los simuladores virtuales. Algunos de los parámetros de análisis de movimiento que se emplean para dichas evaluaciones se muestran en la tabla 2.3.1 [17].

Tabla 2.3.1 *Parámetros de análisis de movimiento para evaluar el cirujano en laparoscopía.*

Métrica	Definición	Ecuación
Longitud de la trayectoria	Distancia total seguido de la punta del instrumento mientras se realiza la tarea	$\int_{t=0}^{T} \sqrt{\left(\frac{dx}{dt}\right)^2+\left(\frac{dy}{dt}\right)^2+\left(\frac{dz}{dt}\right)^2}\,dt$
Percepción de profundidad	Distancia total recorrida en el eje de profundidad	$\int_{t=0}^{T} \sqrt{\left(\frac{dy}{dt}\right)^2+\left(\frac{dz}{dt}\right)^2}\,dt$
Suavidad de movimiento	Cambios abruptos en la aceleración que resultan en movimientos bruscos del instrumento	$\sqrt{\frac{T^5}{2\cdot PL^2}\int_{t=0}^{T}\left(\left(\frac{d^3x}{dt^3}\right)^2+\left(\frac{d^3y}{dt^3}\right)^2+\left(\frac{d^3z}{dt^3}\right)^2\right)^2 dt}$
Velocidad	Tasa de cambio de la posición del instrumento	$\frac{1}{T}\int_{t=0}^{T} \sqrt{\left(\frac{dx}{dt}\right)^2+\left(\frac{dy}{dt}\right)^2+\left(\frac{dz}{dt}\right)^2}\,dt$
Aceleración	Tasa de cambio de la velocidad del instrumento	$\frac{1}{T}\int_{t=0}^{T} \sqrt{\left(\frac{d^2x}{dt^2}\right)^2+\left(\frac{d^2y}{dt^2}\right)^2+\left(\frac{d^2z}{dt^2}\right)^2}\,dt$
Economía del área	Relación entre la superficie máxima cubierta por el instrumento y la ruta total	$\frac{\sqrt{[\mathrm{Max}(x)-\mathrm{Min}(x)]\cdot[\mathrm{Max}(y)-\mathrm{Min}(y)]}}{PL}$
Economía del volumen	Relación entre el volumen máximo cubierto por el instrumento y la ruta total	$\frac{\sqrt[3]{[Max(x)-Min(x)]\cdot[\mathrm{Max}(y)-\mathrm{Min}(y)]\cdot[\mathrm{Max}(z)-\mathrm{Min}(z)]}}{PL}$

3. Metodología.

En este capítulo, se muestra el desarrollo del simulador y las partes que lo componen, las cuales van desde el mecanismo físico. Este simulador funcionará como un mando de control que permitirá la manipulación de los instrumentos en el *software*, se describirá sobre los dispositivos electrónicos empleados y su funcionamiento. Asimismo, aunaremos con detalle el diseño gráfico del simulador, las tareas virtuales de laparoscópica con las cuales cuenta y su funcionamiento lógico.

3.1 Diseño del manipulador.

A continuación, se describirá el diseño mecánico del manipulador (figura 3.1-1), al decir manipulador nos referimos al sistema físico que se emplea para hacer el control de los instrumentos laparoscópicos y que estos a su vez transmitirán los datos al entorno virtual con el objetivo de tener un ambiente de inmersión.

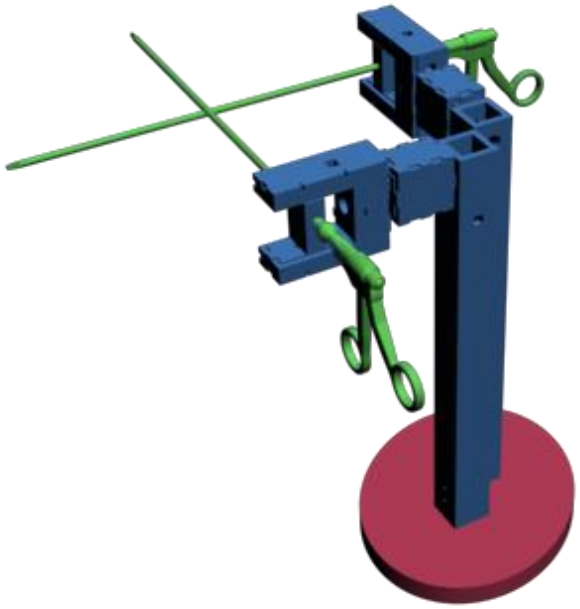

Figura 3.1-1. *Manipulador del simulador.*

La construcción mecánica del manipulador es una expansión sobre el mecanismo que se desarrolló previamente en el Departamento de Cirugía, esto con el fin de mantener la línea de desarrollo y de cierta manera simplificar el trabajo tanto de diseño como de adquisición de nuevo material de trabajo. Este cuenta con dos torres cuadradas de cuarenta centímetros de altura con una base de catorce centímetros de perímetro, estas dimensiones pensadas para que el cirujano pueda manipular el instrumento quirúrgico fácilmente, colocándose enfrente de la estructura [7]. El material de construcción que se ocupó fue acrílico; esta propuesta surge del método de manufactura pensado para el mecanismo el cual se llevó a cabo mediante corte láser. Por otro lado, dentro de la estructura de la torre se consideran dos grados de

libertad (GDL[2]): derecha/izquierda, arriba/abajo, un GDL adicional se encuentra en la caja rectangular del mecanismo, que a diferencia del resto de la torre fue fabricada por manufactura aditiva empleando PLA como material de deposición, donde el movimiento que encontramos aquí es el de profundidad (entrada/salida). Para finalizar, los dos grados de libertad faltantes se encuentran en la pinza laparoscópica, los cuales son el giro de la punta y apertura y cierre de la misma.

3.2 Interfaz electrónica.

Lo primero que necesitamos destacar en la parte electrónica es el medio de comunicación que se emplea entre las partes virtual y física que se logra mediante un microcontrolador. El microcontrolador, comúnmente llamado “micro”, es un dispositivo electrónico que integra en un solo encapsulado un gran número de componentes, en consecuencia, tiene la capacidad de ser programable, es decir, presenta la capacidad de ejecutar de forma autónoma una serie de instrucciones previamente definidas [16].

Con el fin de realizar la comunicación del simulador y el manipulador en este proyecto, se utilizó el microcontrolador Arduino DUE (Figura 3.2-1), dado a su compatibilidad con *Unity*, desde el punto de vista de la comunicación mediante el puerto serial por ambas partes y que presenta un lenguaje de alto nivel para su programación. Arduino es una placa de circuito impreso de hardware libre que incorpora un microcontrolador reprogramable el cual provee cierta facilidad de conectar diferentes componentes electrónicos a sus *Headers* de entrada y salida. Por otro lado, Arduino incorpora un IDE libre en el cual se pueden crear los códigos e implementarlos a la tarjeta mediante un cable USB; este IDE es multiplataforma, en otras palabras, puede ser usado en distintos sistemas operativos de PC [16].

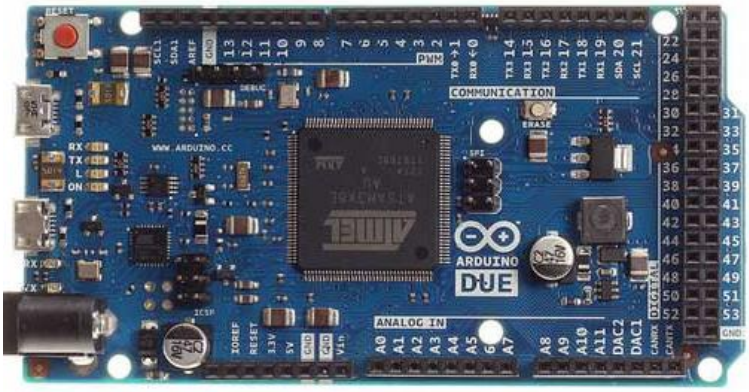

Figura 3.2-1. *Placa de Arduino DUE [17].*

La captura de movimiento en el simulador se llevó a cabo mediante el codificador óptico E4T US Digital de 400 a 2000 pulsos por revolución (PPR) (Figura 3.2-2). Este codificador es un transductor rotativo, que, gracias a una señal eléctrica, por lo regular un pulso, nos indica el ángulo girado. Los codificadores ópticos empleados generan las dos señales de pulsos que se muestran en la figura 3.2-3 [18].

[2] El grado de libertad es el número de entradas independientes requeridas para mover el mecanismo. [22]

Figura 3.2-2. *Codificador óptico EAT.U US DIGITAL [19].*

El código requerido en la interpretación de las señales provenientes de los codificadores se realiza mediante una comparación entre los pulsos A y B ya antes mencionados (figura 3.2-3), en dichas señales generan una máquina de estados de dos bits como se muestra en la tabla 3.2.1, con estos valores se puede conocer el sentido de giro del sensor, así como el número de pasos que da. Este tipo de programación generalmente se realiza paso a paso; sin embargo, el IDE de Arduino incluye la librería de *encoder*.h la cual simplifica la interpretación de dichas señales entregando un conteo creciente o decreciente según el sentido de giro y la posición.

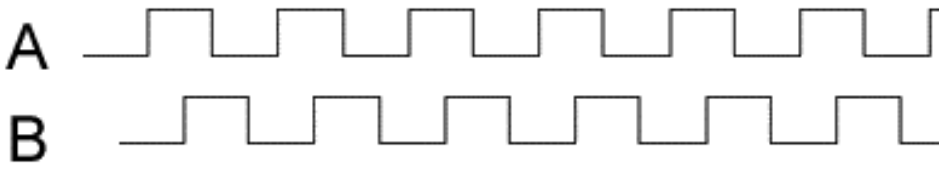

Figura 3.2-3. *Señal de pulsos generados por el codificador.*

Tabla 3.2.1. Máquina de estados y sentido de giro.

Señal A	Señal B	Resultado
0	0	Giro a la izquierda
0	1	Giro a la derecha
1	0	Giro a la derecha
1	1	Giro a la Izquierda

Con la finalidad de obtener, almacenar y procesar estas señales se emplearon las entradas del Arduino que van desde el *header* 22 al 41. Una vez obtenidos los datos se interpretan y se acotan mediante un mapeo, es decir, cada valor de pulsos es convertido en un valor angular o de traslación el cual pueda ser

fácilmente interpretado en la plataforma de *Unity*. Dichos mapeos se envían mediante el puerto serie separados por comas y a una frecuencia de 100 milisegundos.

Finalmente, para ayudar al proceso de conexión de los sensores se realizó una placa de circuito impreso que contemplara todas las entradas requeridas, así como la alimentación y tierra del sistema. Dicha placa se creó basándose en los "*shields*" de Arduino, lo que quiere decir que el circuito se coloca en la parte superior y se conecta mediante el acoplamiento de sus *headers* sin necesidad de algún cable.

3.3 Interfaz gráfica.

En este apartado se expondrá el *software* del simulador. Para un mejor entendimiento del mismo esta sección se dividirá en cuatro, en la primera de ellas se mencionarán aspectos esenciales de la interfaz empleada, en la segunda se explicará sobre la comparación y realización de las tareas con las que se cuentan en el simulador, la tercera sección estará enfocada en la interacción del usuario en aspectos tales como identificación de usuario toma de datos y consulta de avances mediante un administrador. Finalmente, se hablará sobre la captura de movimiento del instrumento laparoscópico y las variables calculadas para la evaluación del usuario.

3.3.1 Unity.

¿Qué es *Unity*?

Este es un *software* desarrollado como motor para videojuegos, en otras palabras, es una herramienta con la cual se permite la integración de gráficos, sonido, diseño, arte y programación de una manera simple. En cuanto a su compatibilidad, *Unity* es multiplataforma, dando la facilidad de crear proyectos en sistemas operativos como Windows, Mac, iOS, Android, Web, entro otros. Por último, es importante aclarar que es de uso libre en su versión persona [20]. En *Unity* cada proyecto se dividirá mediante escenas; la escena hará referencia a cada nivel, ventana o tarea distinta en el simulado. En otras palabras, si tenemos una ventana de registro, otra de menú y dos tareas podemos decir que el proyecto cuenta con cuatro escenas distintas. Las escenas se basan en una arquitectura de componentes, es decir, el elemento primitivo que se llama objeto el cual por sí solo no es capaz de hacer nada, para darle una función al objeto es indispensable añadir componentes los cuales le permitirán realizar alguna acción en específico.

Para modificar las escenas y manipular los objetos, este entorno gráfico cuenta con diferentes ventanas, menús, botones y accesos directos desde el teclado y ratón los cuales permitirán el desplazamiento por el entorno, agregar o quitar elementos, cambiar sus propiedades, ejecutar el programa y mantener el proyecto organizado [20]. A continuación, se presentan los elementos más usados en este trabajo junto con una pequeña descripción:

- **Barra de menús**: Ubicada en la parte superior del entorno esta barra contiene diferentes pestañas que permitirán acciones de administración del entorno y escenas.
- **Barra de herramientas**: Esta barra se encuentra en la parte superior por debajo de la barra de menús, en ella se muestran unos cuantos botones con las acciones que más se utilizaran dentro del entorno. Los primeros botones serán para controlar la vista de la escena, mover un objeto,

rotarlo, cambiar su escala y por último mover y escalar en dos dimensiones, estos botones se pueden ver en la parte izquierda de la figura 3.3-1. Ubicados más al centro están los botones de reproducción con los cuales se puede iniciar la ejecución del escenario, poner pausa y ejecutar paso a paso dicha escena (figura 3.3-1).

Figura 3.3-1. *Barra de herramientas.*

- **Ventana del editor (*Scene*)**: En ella se colocan los objetos que formaran parte de la escena y se configuraran sus características de apariencia, rotación y posición. Dentro de esta ventana se halla el eje de coordenadas con el cual se cambiará la perspectiva para un mejor acomode de los objetos.

Para facilitar el movimiento de la vista en esta ventana existen diferentes combinaciones de comandos:

- Botón derecho del ratón: Rotar el escenario.
- Botón izquierdo del ratón: Seleccionar objetos.
- Alt + Botón izquierdo del ratón: Rotación de la vista (movimiento llamado *ORBIT*).
- Botón central del ratón: Desplazamiento de la vista (movimiento llamado *PAN*).
- Alt + Botón derecho del ratón: Acercamiento (*Zoom*).

- **Ventana de juego (*Game*)**: Es la ventana donde se mostrará la apariencia final de una determinada escena esta se pondrá en ejecución, pausa o se detendría con los botones de reproducción figura 3.3-1.
- **Jerarquía de la escena (*Hierarchy*)**: En esta sección se muestran los elementos que aparecen dentro de una misma escena, dichos elementos presentan una relación tipo padre-hijo en la que el padre es el contenedor de uno o más hijos. Un objeto que siempre aparecerá en esta ventana es la "*Main Camera*", la cual se encarga de mostrar la escena por medio de la ventana "*Game*".
- **Proyecto (*Project*)**: Una parte importante de la interfaz es el lugar donde se organizan todos los elementos que contiene el proyecto, para esto se encuentra esta ventana. En ella se encuentran las diferentes carpetas que contendrán todos los elementos que se ocuparán en el proyecto, tales como audio, imágenes, materiales, escenas, *scripts*[3], *prefabs*[4] entre otros.
- **Inspector**: Esta ventana permitirá ver las características del objeto seleccionado, permitirá distinguir entre los diferentes objetos, así como el acceso a la modificación, actualización e incrementación de componentes.

En *Unity* la implementación de componentes es uno de los aspectos más importantes ya que estos cambian el comportamiento, la apariencia y las propiedades de los objetos con los que se interactúa, el número de estos componentes es exorbitante y no es el propósito de este trabajo describir sobre todos, no obstante, mencionaremos los más importantes y algunos otros que necesarios para el proyecto.

[3] *Script*: Es un código de programación que permite establecer el comportamiento de los objetos.
[4] *Prefab*: Son objetos reutilizables que conservan sus propiedades y con alta facilidad de modificación y repetibilidad.

El primer componente se llama transformación, que se agrega automáticamente a todos los componentes ya que sin él no se puede determinar la posición, rotación ni escala de los objetos. Como dato importante, estos elementos pueden ser locales si el objeto es hijo de otro o globales si el de más alta jerarquía, desde código se pueden editar sus propiedades ya sean locales, globales o una combinación de ambas [21].

Los componentes que normalmente aparecerán son el componente cámara y el componente luz estos son parte de la luz de escena y la cámara. En la figura 3.3-2 se muestran los parámetros que tienen, lo que normalmente se edita en estos es la intensidad de la luz, el color, el tipo de luz y la sombra, en cuanto a la cámara se puede editar la pantalla de fondo el ángulo de visión la profundidad y cercanía que alcanza entre otras propiedades más específicas.

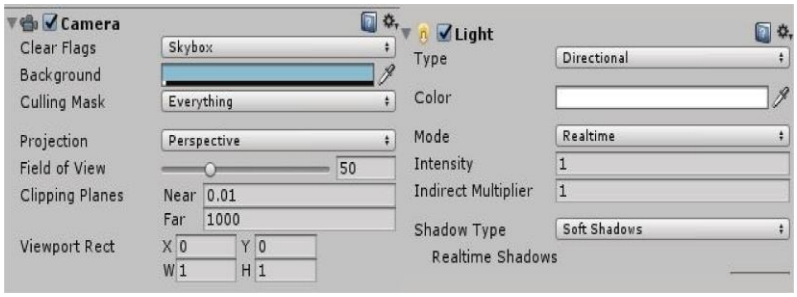

***Figura 3.3-2**. Componentes cámara y luz.*

El siguiente componente se llama malla y es necesario para todos lo que se quiera ver en la escena ya que este construye las partes visuales de los objetos, solo se puede editar la geometría del objeto que se mostrara. Existen dos de estos elementos que funcionan para dar presencia física a los objetos, los cuales son de colisión llamados en inglés *collider*, que existen en tres formas básicas de *Unity*, caja, esfera y capsula tal como se puede ver en la figura 3.3-3. Sus parámetros para editar son de posición y tamaño, además tienen una casilla que los convierte en desencadenadores de eventos, en otras palabras, detecta cualquier otro objeto que entre en su espacio [21]. El otro componente que le da presencia física es el cuerpo rígido, que permite determinar la masa el arrastre, el uso de gravedad y la forma de detectar a otros objetos.

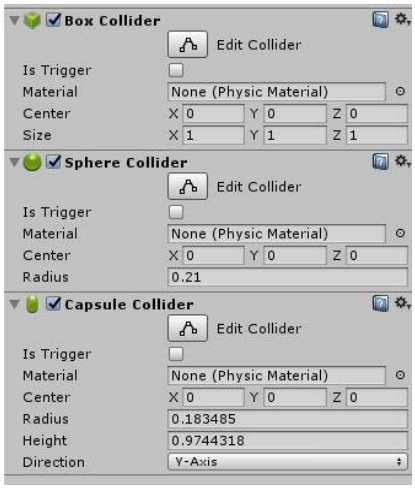

***Figura 3.3-3.** Tipos de componentes de colisión.*

Uno de los componentes que se empleó repetidas veces para dar apariencia personalizada a los elementos es el material (Figura 3.3-4). Este elemento se crea como un archivo que permite personalizarlo, Con el material se puede cambiar el color, la transparencia, el reflejo, propiedades metálicas entre otras propiedades; del mismo modo permite cargar texturas personalizadas mapas normales, mapas de relieve entre otras imágenes que le dan una apariencia realista a los objetos.

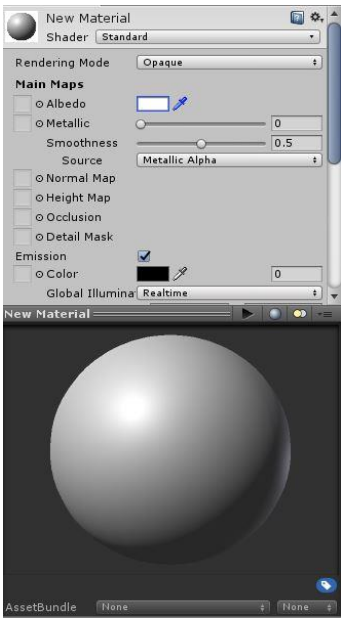

Figura 3.3-4. *Material.*

Por último, existen tres componentes que servirán para la reproducción de audio y video. Los primeros dos son la fuente de audio, en donde se puede cargar el sonido, música o cualquier elemento similar y reproducirlo, pausarlo, cambiar el volumen o bien repetirlo. Este componente requiere de un receptor para el cual se pone en lo general en el jugador, para el caso del simulador este elemento lo encontramos en la cámara principal. El componente reproducción de video es el equivalente al del audio, pero para extensiones mp4, avi, mov entre otras, estos componentes se muestran en la figura 3.3-5.

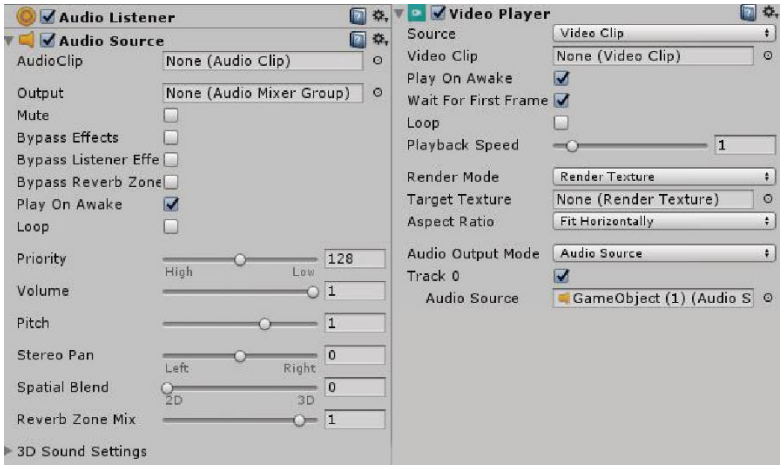

Figura 3.3-5. *Componentes de audio y video.*

Estos son algunos de los componentes básicos que aparecerán a lo largo de este proyecto; sin embargo, más adelante hablaremos de otros componentes que se emplearon en el simulador sin profundizar demasiado en ellos ya que no se repiten de manera significativa como los anteriormente mencionados.

3.3.2 Diseño y programación de tareas en Unity.

El simulador cuenta con cinco escenas dedicadas al entrenamiento y a la adquisición de habilidades de los diferentes usuarios, a este conjunto de escenas las llamaremos tareas. A lo largo de este apartado se irán describiendo tanto el funcionamiento como el diseño de dichas tareas.

Los primeros aspectos que se tienen que destacar de estas tareas son los que comparten en común, estos son la base de las actividades. En este caso, tenemos seis elementos los cuales realizan la misma función en todas las tareas, tres de estos son gráficos como se muestran en la figura 3.3-6, y los otros tres solo se pueden apreciar en la jerarquía de escena (figura 3.3-8).

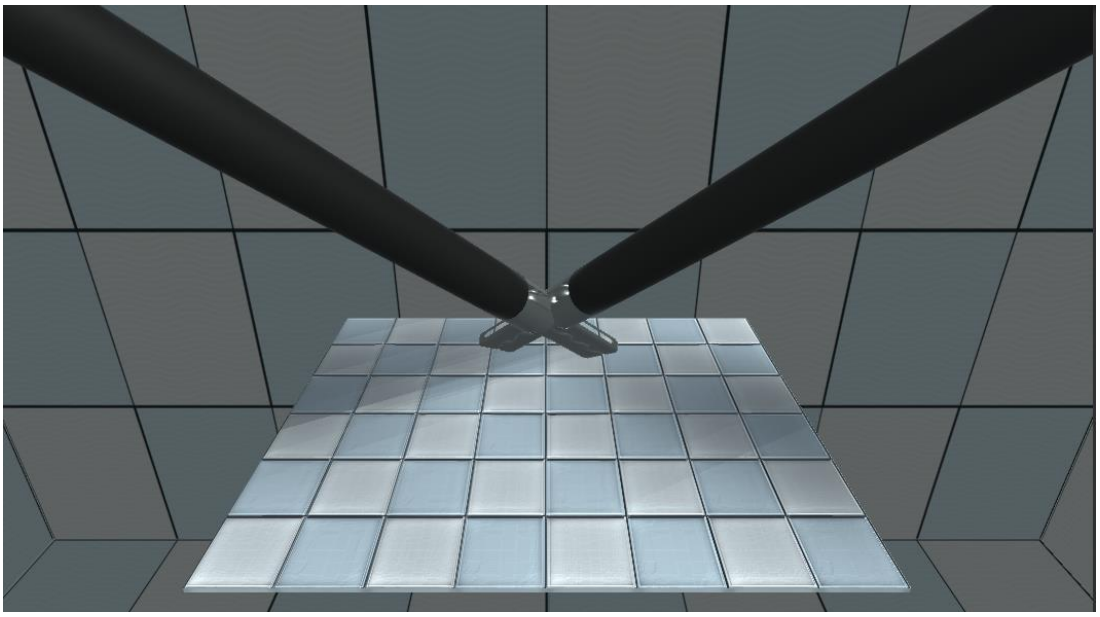

Figura 3.3-6. *Escena básica de las tareas.*

Entre los elementos gráficos de las tareas se tiene: el texturizado del paisaje, conocido como *skybox*, esta textura representa el límite visual de la cámara y fue creada desde cero en un *software* especializado, como se muestra en la figura 3.3-7 a. Lo siguiente es el área de trabajo la cual esta escalada a 10×12 centímetros, estas medidas son un estándar de los simuladores para ayudar con la medición de parámetros. El área está representada por un cubo, geometría primitiva de *Unity*, para darle un atractivo visual se emplearon las texturas como se pueden ver en la figura 3.3-7 a y b, siendo ‘a’ el texturizado de color y ‘b’ el mapa normal que representa la profundidad de la textura.

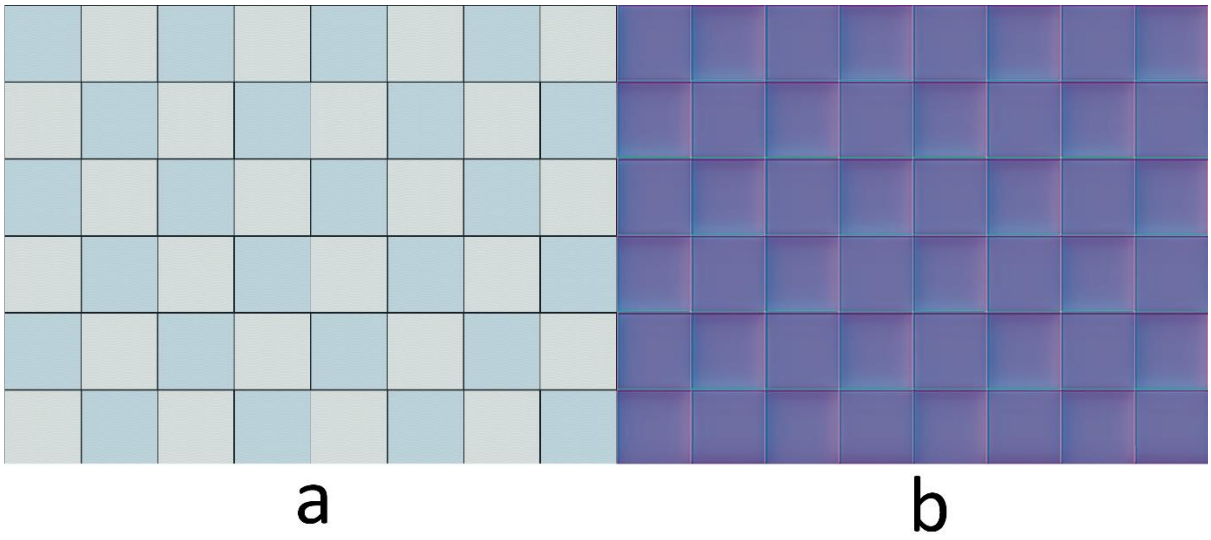

Figura 3.3-7. *Textura y mapa normal.*

Finalmente, el último elemento que se puede ver en la figura 3.3-6 es el modelo de las pinzas laparoscópicas. Este elemento tiene el mismo comportamiento básico en todas las tareas, es decir, está compuesto por dos *scripts* invariables para su funcionamiento; sin embargo, su apariencia y funciones específicas cambian dependiendo de la tarea a realizar, los detalles del comportamiento de las pinzas se irán explicando conforme avancemos en el documento.

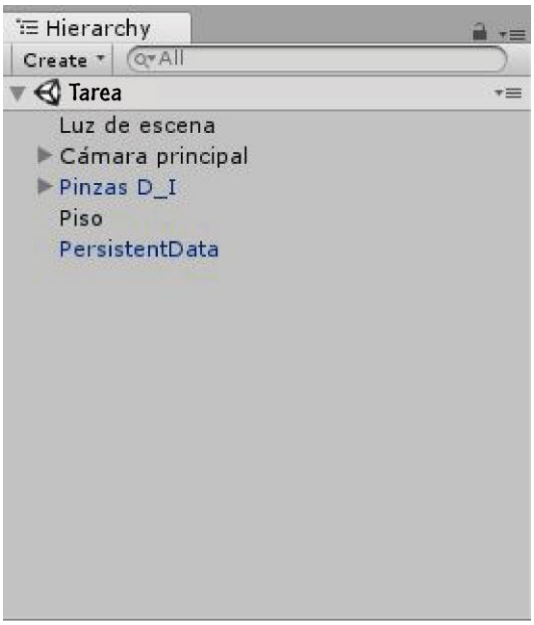

Figura 3.3-8. *Jerarquía elementos básicos.*

Los tres componentes restantes que se encuentran en todas las tareas son: la luz de la escena, la única función de esta es proporcionar una iluminación adecuada para poder ver los diferentes objetos dentro de cualquier escena. El otro objeto básico es la cámara, ya que sin ella no podemos ver ninguno de los

componentes del simulador. Los dos elementos mencionados son propios de *Unity*. Por el otro lado, la última pieza que compone a la tarea básica fue creada específicamente para el simulador con el objetivo de mantener una comunicación entre todas las escenas del simulador incluyendo las tareas. Su función es mantener la información del usuario, el administrador y es el encargado de recibir los datos de evaluación para escribirlos en un documento de texto, dichos parámetros de evaluación se verán en el capítulo 3.3.4.

El simulador cuenta con tareas de reconocimiento de profundidad, que tienen dos objetivos, el primero de ellos es hacer que el usuario se familiarice con la vista en dos dimensiones del entorno y el segundo es lograr y mejorar la coordinación de ambas manos. El primer objetivo se encuentra en la tarea 1, donde se busca que el usuario toque los objetos con la punta de la pinza en el orden indicado. Esta tarea se compone de seis objetos, cada uno con su propio material, que se encuentran acomodados a tres profundidades distintas y a dos alturas como se ve en la figura 3.3-9. El desarrollo de la tarea es utilizando los 3 grados de libertad del instrumento laparoscópico.

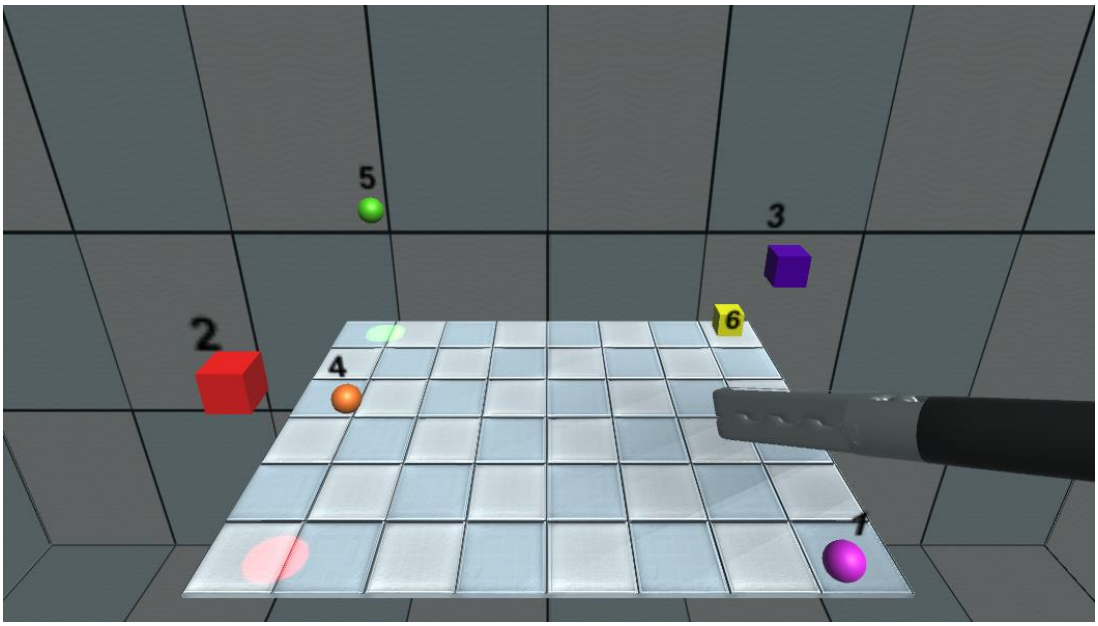

Figura 3.3-9. *Primera tarea de profundidad.*

La programación principal se encuentra en un *script* este se encarga de destruir los objetos con forme son tocados e indica el fin de la tarea, el funcionamiento de este es por medio de un evento el cual detecta las colisiones con cualquier objeto. Para poder detectarlo, este tiene que cumplir con dos condiciones, la primera de ellas es contar con una etiqueta la cual permite hacerle saber al código si es uno de los objetos a desaparecer y de ser así el segundo indicador es una variable de tipo booleana (bandera), para saber si el objeto anterior ha desaparecido de ser así el objeto desaparecerá y de lo contrario no se hará nada. Finalmente, cuando el usuario toque el último objeto el código indicará cuando la tarea esté terminada.

La segunda tarea de coordinación mano-ojo contempla a las dos pinzas tal como se puede ver en la figura 3.3-10. En esta actividad los objetos que el usuario tiene que tocar cambiaron de estar visibles al mismo

tiempo a ir apareciendo conforme son requeridos. Otra particularidad que presenta esta tarea es el color que identifica a las esferas y a la punta de la pinza. Estos colores indican al usuario que pinza es la que se necesita para cada objeto. Esto implica que el usuario no podrá desaparecer un objeto si no es con la pinza correcta. Es importante mencionar que el orden y posición de los objetos que aparecerán está predeterminado, con el fin de tener un control del área de trabajo y así poder medir el avance del usuario.

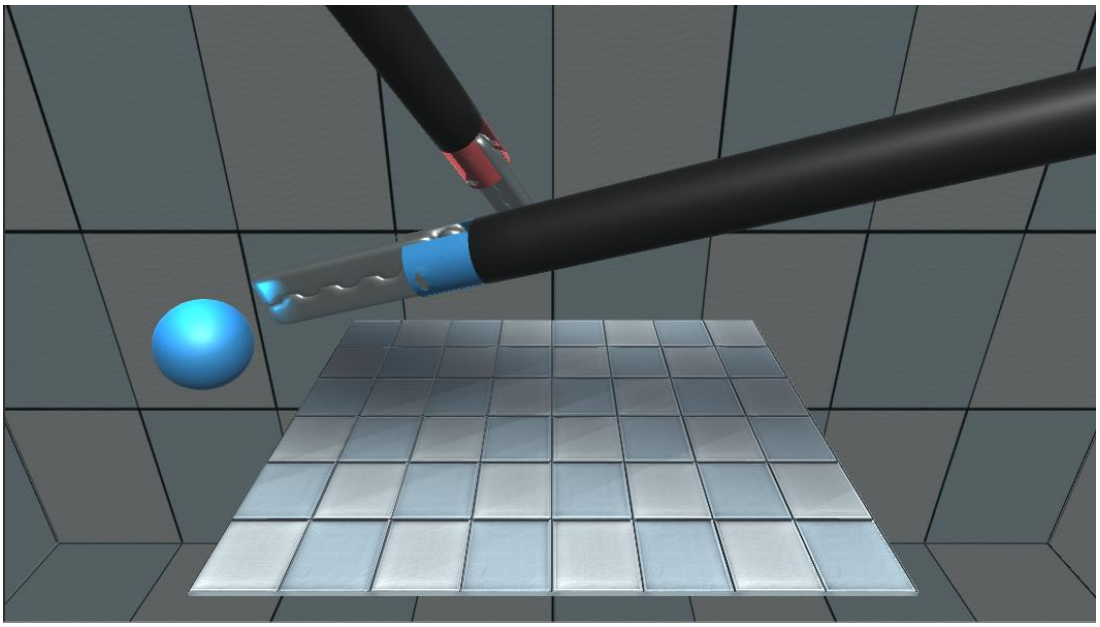

Figura 3.3-10. *Segunda tarea de coordinación mano-ojo.*

Para programar el funcionamiento, se ocuparon los mismos *scripts* que en la tarea anterior más uno que se encarga de comunicar a los objetos que tienen que aparecer y ser destruidos, y como segunda tarea este *script* manda el fin de la tarea. Estos códigos a diferencia de los anteriores no destruyen el objeto que es tocado sino lo desaparecen del campo visual para poder llevar el conteo de los objetos que han sido tocados. Este conteo tiene dos objetivos: el primero es indicar cuál de los objetos es el siguiente en aparecer en la escena y el segundo es saber cuándo todos los objetos han sido tocados ya que con esto se indica el fin de la tarea.

La siguiente tarea es la tarea de transferencia que se realiza únicamente con la mano derecha. En la escena, se muestran seis aros que están dispuestos en seis postes distintos. El objetivo de la actividad es que el usuario traslade todos los aros (uno por uno) a los postes libres. En cuanto a la parte visual de la tarea solo se le agregan los aros y cilindros con su respectivo material como se muestra en lafigura 3.3-11.

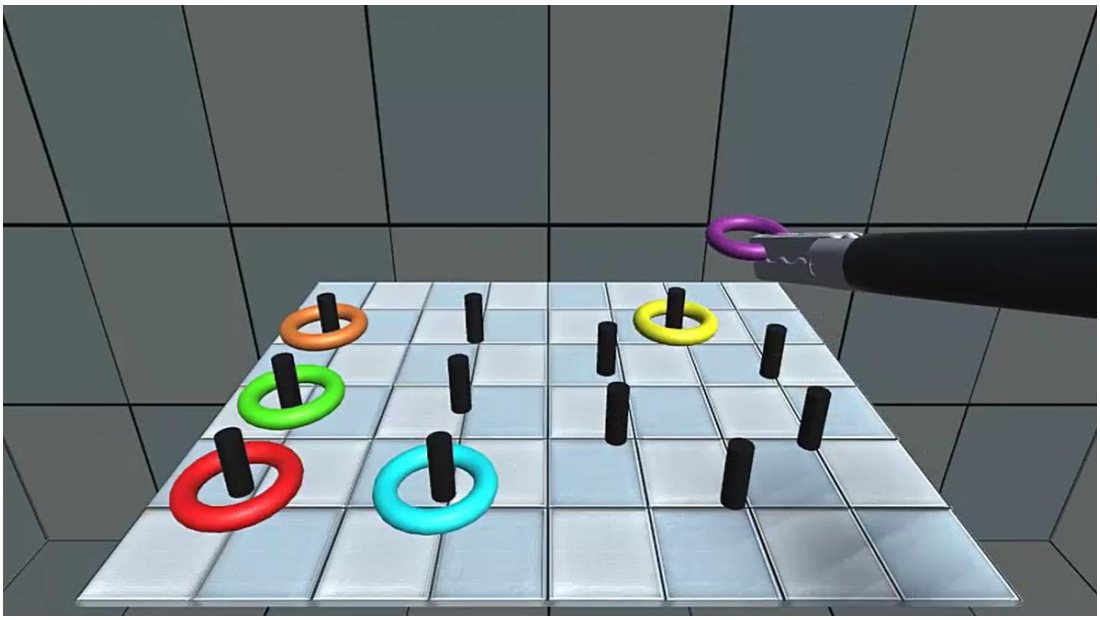

Figura 3.3-11. *Tarea de transferencia.*

En la parte de programación, esta actividad cuenta con cuatro *scripts*. El primer *script* trabaja individualmente y se encarga de regresar al punto de origen los aros en caso de salir del campo visual. Esta acción es posible gracias a que al iniciar el programa se guardan las posiciones iniciales de los objetos en unas variables temporales y debido a la gravedad se puede saber que los aros están por debajo de un límite preestablecido. El siguiente código se encarga de indicar el fin de la tarea, el cual revisa de forma constante que los postes finales se encuentren ocupados cada uno con un aro. Los últimos dos se comunican entre ellos, además sirven para identificar a los objetos que se pueden mover con la pinza. Para esto se necesita etiquetar a los elementos ya que con esta etiqueta la pinza podrá reconocerlos. La comunicación que realizan los códigos es para saber si la pinza está tocando a un aro, gracias a las etiquetas, y por otro lado si las pinzas están cerradas de cumplirse las dos condiciones uno de los *scripts* hace hijo al aro de la pinza lo que permite el movimiento simultáneo de los objetos hasta ser soltado

En la siguiente tarea nombrada tarea tres se presenta una actividad de corte circular. En la parte gráfica se agregan los elementos que representan a la membrana y al área de corte. Estos tienes su propio material con la particularidad de que la membrana presenta una transparencia para permitir al usuario ver toda el área de corte. Por último, la pinza de la mano derecha fue sustituida por una tijera con la cual se tiene que realizar dicho corte, esta escena se puede apreciar en la figura 3.3-12.

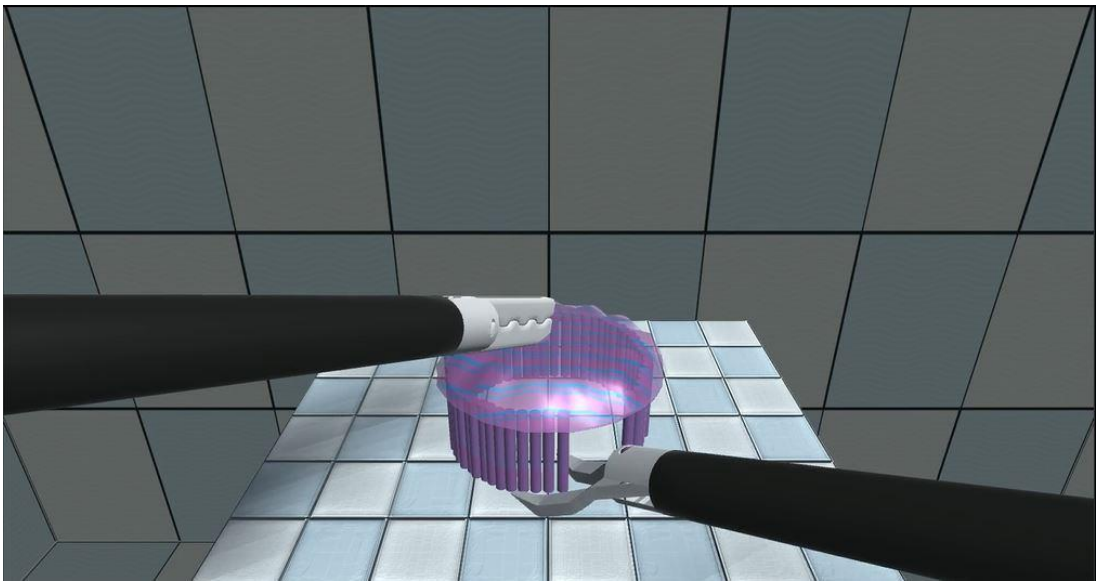

Figura 3.3-12. *Tarea de corte.*

Para lograr el ambiente simulado, se requirió de un componente que permitía dar la apariencia del movimiento real de este procedimiento. El componente de tela representa un papel importante al momento de dar la apariencia de membrana acomodado en forma de círculo que callera ligeramente para formar una cúpula. Otra ventaja que tiene el componente es su zona de viento lo que permite agregar un movimiento aleatorio que de la una apariencia de un elemento gelatinoso y con esto dar una sensación más inmersiva al usuario. El funcionamiento de esta tarea esta dado por un *script*; sin embargo, por si solo este no es capaz de realizar la tarea por completo. Esto se debe a que no reconoce la apertura y cierre de la pinza para esto se tiene que comunicar con la función del código de la pinza y con esto determinar los cambios de la tijera. Una vez obtenida esta información el *script* puede determinar cuando el usuario está cortando. En lo individual, su programación permite determinar el final de la tarea al contar cuantos de los objetos han sido destruidos, este conteo se da cuando el objeto destruido le manda la información al código principal.

Finalmente, tenemos la tarea de engrapado que se muestra en la figura 3.3-13. En este ejercicio el usuario tiene que poner una grapa en cada uno de los elementos que aparecen en la zona indicada de un distinto color. Para poder realizar dicha actividad se tienen que sujetar los aros y jalar los cilindros con esto se tendrá acceso al área a engrapar. El modelo de la pinza cambia nuevamente y ahora se cuenta con una engrapadora.

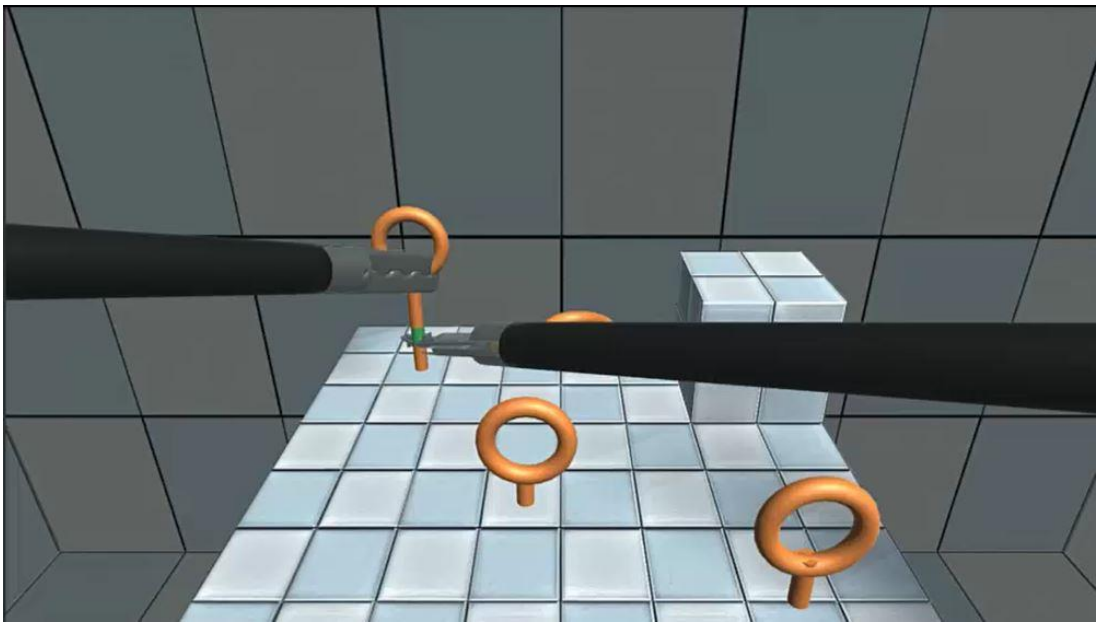

Figura 3.3-13. *Tarea de engrapado.*

Dado el funcionamiento diferente de la engrapadora, en comparación con el resto de las pinzas, se empleó una animación para logra hacer coincidir los movimientos virtuales con el real. Dicha animación se agrega mediante un componente, control de animación, donde se crea un árbol de secuencia que indica el orden en que las animaciones irán apareciendo y cuál será su activador. Esto se puede ver en la figura 3.3-14, la animación de la engrapadora se dividió en tres, el estado inactivo, acción de cerrar y acción de abrir. Estos estados se controlan mediante un código del mismo modo que la tarea anterior se comunica con el *script* de la pinza. Lo que se agrega en esta tarea en comparación con la anterior es la creación de las grapas, estas se generan en el *script* principal y dependen de la animación.

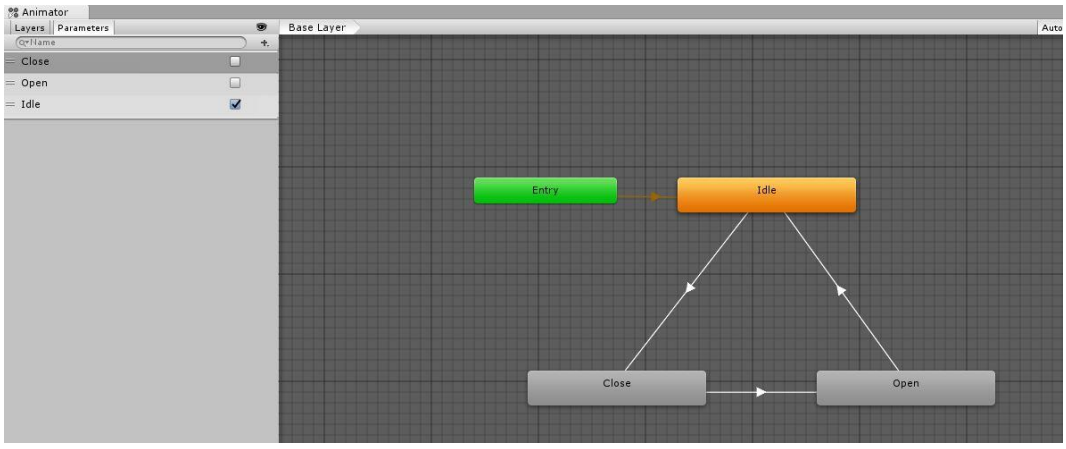

Figura 3.3-14*. Control de animación.*

3.3.3 Diseño y programación de la interfaz gráfica de usuario.

Con la finalidad de tener un control de los usuarios, así como su progreso el simulador cuenta con cuatro escenas, estas forman parte de la interfaz gráfica que se encargan del registro, revisión de datos por parte de un administrador y orientación al usuario en la selección de tareas. Estas escenas están realizadas en una modalidad de dos dimensiones con el objetivo de optimizar la interacción del usuario.

El modelo visual básico de estas tareas no cambia, empleando los mismos elementos de captura de texto, botones, música de fondo, entre otros elementos visuales, diseñados específicos para el simulador. En cuanto a la programación, los diferentes *scripts* realizan cambios de escena mediante rutinas especiales que tienen la particularidad de poder esperar un determinado tiempo y trabajan en paralelo a las actividades normales por lo que no afectan el desempeño del código ni los recursos de la computadora. Para que la interfaz trabaje correctamente en diferentes tamaños de monitor es necesario realizar las configuraciones necesarias. Estas configuraciones se realizan al añadir el *canvas*[5] donde es necesario modificar su escala para que este tome las propiedades del tamaño de la pantalla, llamada "escala con tamaño de pantalla", que se encuentra dentro de la ventana inspector en el componente "*canvas* scaler" como se muestra en la figura 3.3-15.

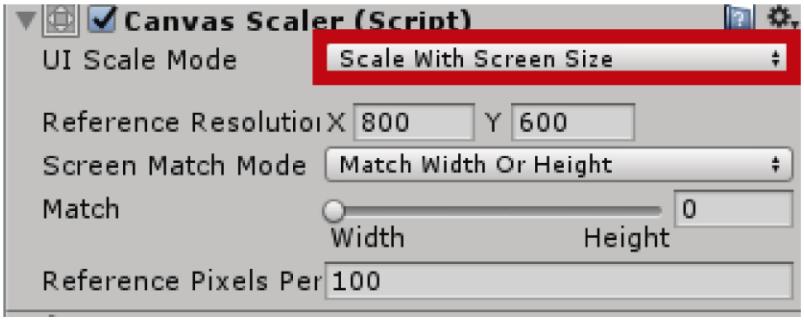

Figura 3.3-15. *Propiedades del canvas.*

De un modo similar se crean las transiciones de los botones, es decir, los botones presentan tres estados que indican al usuario la interacción con el curso. El primer estado es el normal, donde el botón permanece inactivo y se muestra de un color blanco. El segundo se activa cuando el cursor del ratón pasa por encima del botón, pasando a ser de color dorado, junto con este evento se reproduce un sonido, y el tercer estado se da cuando se hace clic sobre el botón, del mismo modo genera un sonido diferente al anterior y su apariencia permanece en dorado pero recibe una transparencia que lo distingue, las transiciones se pueden ver en la figura 3.3-16.

Figura 3.3-16. *Diseño y transición de botones.*

[5] *Canvas*: Objeto de interfaz gráfica que contiene los elementos bidimensionales.

Para el registro de los datos dentro del simulador se utilizan dos bases de datos una dedicada a los supervisores y otra a los practicantes, ambas creadas en extensión xml por la compatibilidad de *Unity*. Es importa crear una carpeta especial donde se guardan las bases de datos y otros elementos que se quieren en el proyecto, donde la carpeta *Resources* garantiza que dichos archivos siempre estén disponibles en el programa y da la certeza de poder encontrar la dirección.

La primera escena con la que se tiene contacto es la de inicio de sesión. En ella se cuentan con los apartados de registro, administrador y la entrada a las tareas, tal como se puede ver en la figura 3.3-17. La programación de esta escena realiza lecturas en la base de datos ya sea del administrador o del usuario y compara los campos necesarios para dar acceso a las siguientes. En las entradas de texto tanto del administrador, contraseña así como en el número de identificación del usuario existe un evento encargado de habilitar los botones correspondientes con la finalidad de que los datos no se encuentren vacíos y no se presente un error al momento de buscar este dato nulo en la base de datos; sin embargo, de proporcionar un usuario, número de identificación o contraseña incorrecta se indicara mediante un cambio de color en el texto de negro a rojo. Una vez que el usuario o administrador inician sección con éxito el dato de la persona se guarda en el *script* de datos persistentes para tener la referencia de quien usa el simulador, la utilidad de esto se verá más adelante.

Figura 3.3-17. *Inicio de sección.*

El inicio de sesión como administrador se encuentra oculto a la vista y se habilita al hacer clic en el botón correspondiente, es necesario proporcionar nombre de usuario y contraseña correcto, de darse algún dato erróneo este se indicará en color rojo. Una vez que se ingresó como administrador pasará a la siguiente escena, donde se podrá consultar el progreso de las personas que estén registradas, actualizar sus datos, borrar usuarios, así como actualizar la contraseña del administrador o registrar a uno nuevo.

Los elementos visuales con los que se cuenta son los botones, la entrada de texto y un panel donde se muestra la información del usuario buscado como se muestra en la figura 3.3-18. El código se encarga de

mantener comunicado a los elementos visuales y realiza la búsqueda del usuario para lograrlo se usa el parámetro de identificación único “ID” por lo que es importante no olvidarlo.

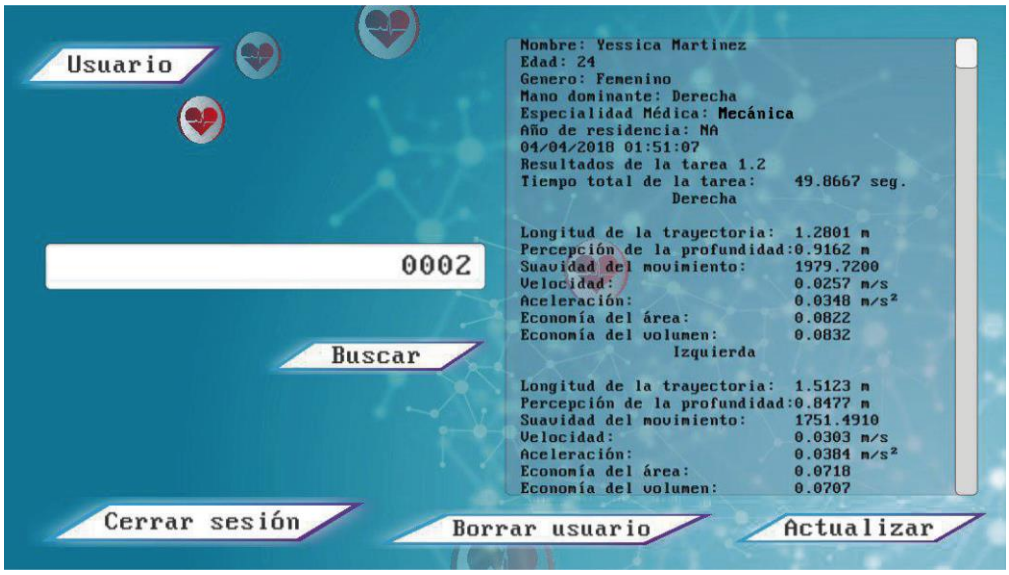

Figura 3.3-18. *Escena del administrador.*

El simulador cuenta con una escena de registro donde se le piden los datos de las personas con el objetivo de conocer su previa experiencia en prácticas de laparoscopia y tener sus datos. En la escena se muestran los campos que se necesitan llenar, cabe destacar que el número de identificación es automático, para lograrlo el *script* lee la base de datos y obtiene el ultimo valor de identificación los cuales son numéricos y comienzan en el “0001”, la organización de la escena se muestra en la figura 3.3-19. Es en esta sección donde se realiza la comunicación completa con la base de datos.

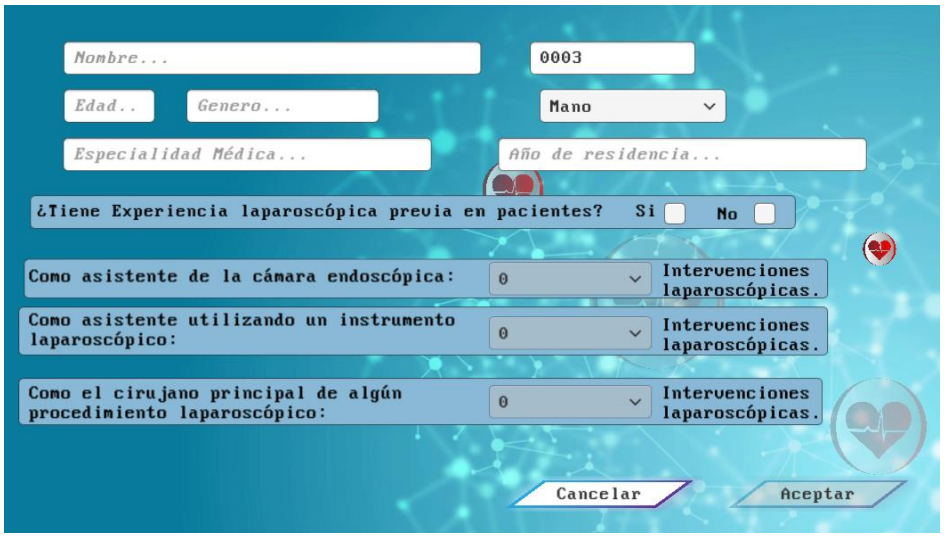

Figura 3.3-19. *Captura de datos del usuario.*

Finalmente, tenemos la escena del menú donde se muestran cinco botones: cerrar sesión, salir anterior, siguiente e inicio, donde los tres últimos permiten navegar para elegir la tarea que se desee realizar. Sumado a estos, se encuentra un cuadro de texto el cual da una breve descripción de la tarea. Por último, aparece un plano donde se proyectan los videos demostrativos de cada tarea, como se muestra en la figura 3.3-20.

Figura 3.3-20. *Menú inicio de las tareas del simulador laparoscópico.*

La programación del menú se divide en casos para mostraran los diferentes videos introductorios a las tareas o ejercicios, gracias a este método se logra facilitar la adición de nuevas tareas y su respectivo video, al programador, para logar el objetivo se tienen dos variables publicas una nombrada “máximo número de escenas” la cual indicara el número total de tareas del simulador, su funcionamiento limita a un contador quien represente el caso al que se tiene que acceder y la otra llamada “videos de tareas” la variable representa el número máximo de elementos de un vector de video en el cual se añaden las vistas previas en formato mp4.

3.3.4 Captura del movimiento del instrumento laparoscópico.

El simulador requiere de distintos parámetros que permitan su comunicación con los instrumentos laparoscópicos, así como el monitoreo en el entorno virtual para su posterior calculo y análisis, por lo que se cuentan con algunos *scripts* que trabajan con una elevada comunicación, el cual se hace posible mediante el *script* de datos persistentes. La comunicación de las pinzas comienza con la interpretación de los datos provenientes del micro controlador el encargado de la tarea es el *script* “*Move*”, aquí se genera la conexión con el puerto serial, de no encontrar la torre conectada inmediatamente el *script* se bloquea dando el control a un código de respaldo, de lo contrario la primer tarea que realiza es encontrar cinco objetos por cada pinza dentro de la interfaz del simulador los cuales recibirán los datos que indican su movimiento, los objetos se pueden ver en la figura 3.3-21. En segunda instancia se toman los diez datos

provenientes del puerto serial para dividirlos, esto se logra mediante la función separado, cada dato se acomoda en una variable, la cual afectará únicamente a un objeto, ya sea en su rotación o traslación. Finalmente, se encarga de activar un indicador para saber si una piza cambio de estado, de abierto a cerrado.

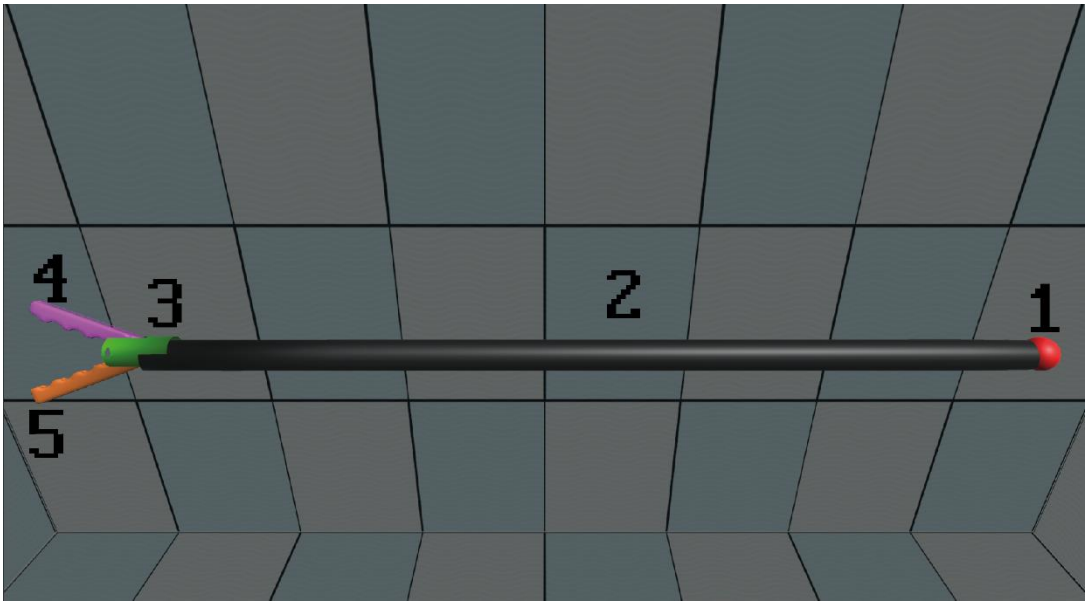

Figura 3.3-21. *Objetos generadores del movimiento.*

Como se mencionó con anterioridad si el simulador no encuentra conectada la torre se activa un segundo *script* en donde se da paso a un control desde el teclado de la computadora o bien con un mando de videojuego. Para que el código trabaje con ambas entradas, es necesario convertir las entradas del mando de videojuego de analógicas y valores enteros a booleanas. Las teclas que se emplean en el código se muestran en la figura 3.3-22.

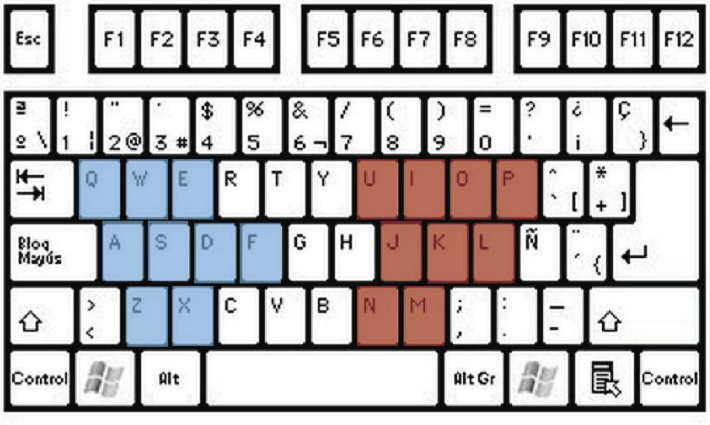

Figura 3.3-22. *Botones de control desde el teclado. Teclas azules controlan la pinza derecha y las teclas rojas a la pinza izquierda en la aplicación del simulador laparoscópico.*

En la tabla 3.3.1 se muestran los movimientos realizados por cada una de las entradas del teclado ya mencionadas.

Tabla 3.3.1. Movimientos realizados por el indumento con el teclado.

Pinza Derecha		**Pinza Izquierda**	
Tecla	**Movimiento**	**Tecla**	**Movimiento**
w	Arriba	**i**	Arriba
s	Abajo	**k**	Abajo
a	Izquierda	**j**	Izquierda
d	Derecha	**l**	Derecha
e	Rotación horaria	**o**	Rotación horaria
q	Rotación anti-horaria	**u**	Rotación anti-horaria
f	Abrir	**p**	Abrir
z	Meter pinza	**n**	Meter pinza
x	Sacar pinza	**m**	Sacar pinza

Ahora bien, para realizar el seguimiento de cada uno de los instrumentos laparoscópicos virtuales, el software cuenta con su propio *script,* el cual es el encargado de tomar las posiciones de las puntas y guardarlas en un arreglo de cinco elementos con los datos obtenidos y por medio de diversas fórmulas se calculan los parámetros a evaluar. Este código utiliza una serie de funciones matemáticas la cuales se pueden encontrar dentro de la librería Mathf incluida en *Unity*. Su principal trabajo es realizar operaciones con los datos de posición a medida que estos son leídos, dado que, para no saturar la memoria de procesamiento de la computadora estos datos son desechados tan pronto como dejan de ser requeridos. Otra de las funciones programadas dentro de este *script* es la comunicación con los datos persistentes con el objetivo que este último *script* reciba los datos necesarios que serán enviados al documento de evaluación del usuario.

4. Resultados.

En este capítulo, se explica el funcionamiento final del simulador, haciendo un análisis del desempeño tanto de los gráficos como del realismo de los movimientos y las tareas laparoscópicas, así mismo se hace un análisis de los requisitos mínimos requeridos. También, se presentarán las pruebas y resultados de algunos voluntarios que ocuparon este dispositivo y la evaluación que realizaron al simulador.

4.1 Simulador laparoscópico virtual.

En primer lugar, para que el simulador pueda ejecutarse en cualquier computadora con los requisitos de hardware necesarios, se tiene que crear un archivo con extensión "exe", desde *Unity* se agregan las escenas y se realizan las configuraciones necesarias dentro de la ventana del constructor tal como se muestra en la figura 4.1-1.

Una vez que se tienen las escenas en la ventana se pueden hacer algunas configuraciones adicionales, entre las cuales encontramos, el nombre del ejecutable, poner una figura como icono, la relación de aspecto entre otras, para el simulador se delimito la relación de aspecto a 16:9 dado que la interfaz gráfica de usuario fue optimizada a esa resolución.

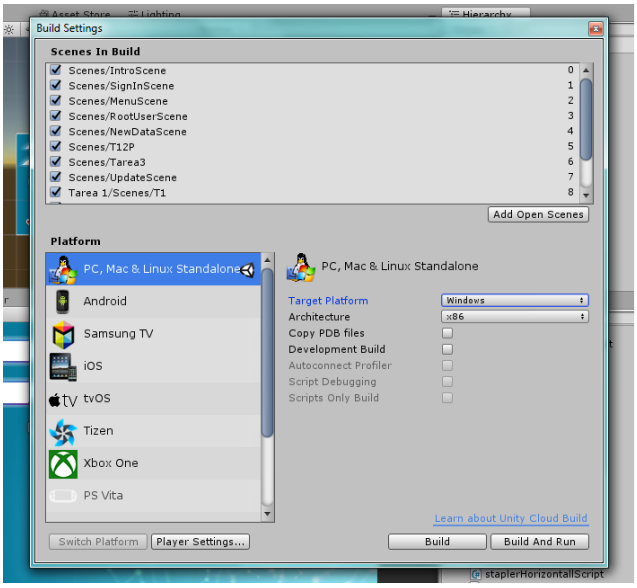

Figura 4.1-1. *Ventana del constructor del proyecto Unity.*

Cuando termina el proceso anterior se genera una carpeta junto con su ejecutable, los dos elementos son necesarios para compartir el simulador a otras computadoras. El simulador se ejecuta como cualquier otro programa de computadora, al iniciarse se abrirá la ventana de la figura 4.1-2, donde se elegirá la resolución del monitor, velocidad y si se ejecutara en pantalla completa o en ventana, las configuraciones se muestran en la figura 4.1-2, en **A** se ejecuta en tamaño de una ventana y en **B** se ejecuta a pantalla completa.

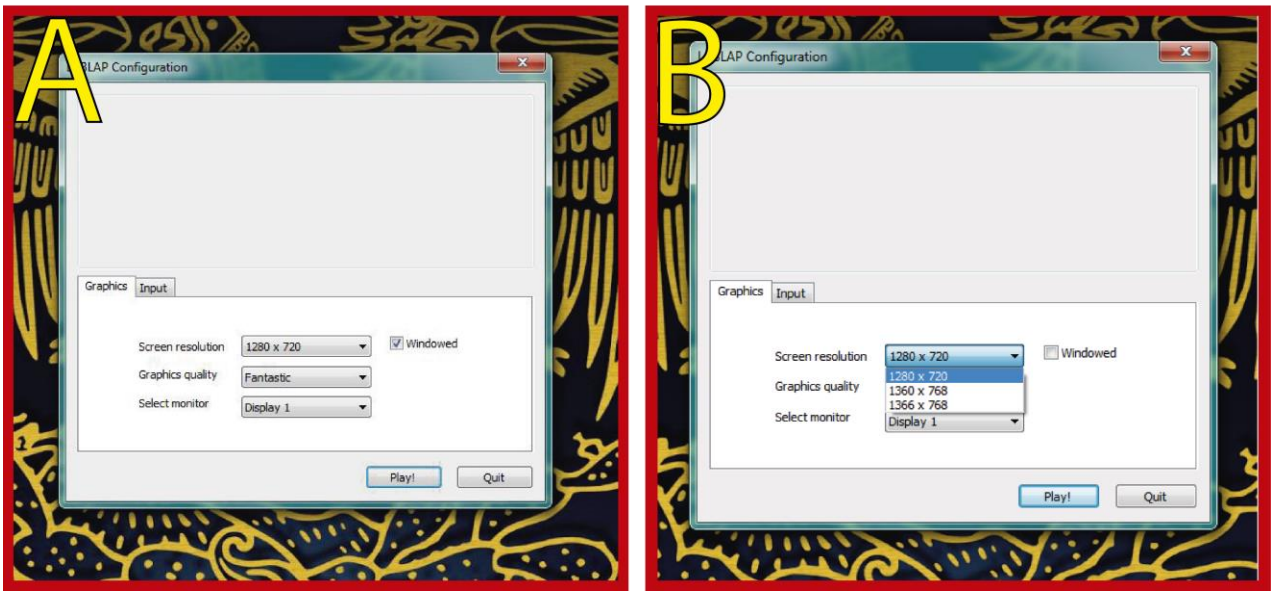

Figura 4.1-2. *Ventanas de ejecución del programa.*

Al iniciar el simulador se verá un pequeño video el cual termina con el diseño del logo propio del simulador, acto inmediato se entrará en la escena de inicio de sección/registro, si el usuario ya está registrado tendrá que indicar su número de identificación para tener acceso a las tareas tal como se ve en la figura 4.1-3, de lo contrario tendrá que crear su registro.

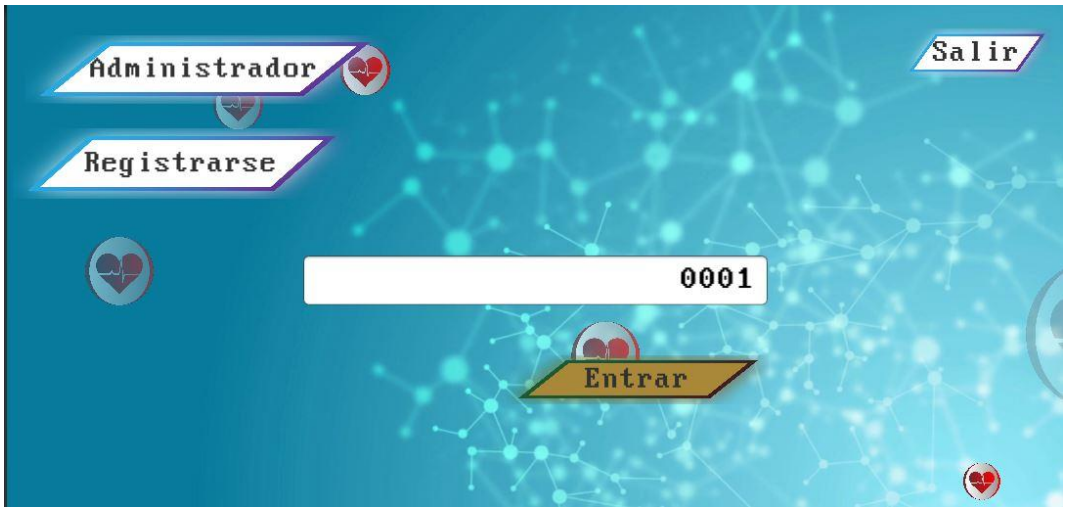

Figura 4.1-3. *Inicio de sección de un usuario.*

Para un nuevo usuario se tiene que hacer clic en el botón de registro, donde se pedirán los datos del nuevo usuario y se indicara cual fue su número de identificación, con el que tendrá su acceso a las tareas, en la figura 4.1-4 se muestra el mensaje para el usuario.

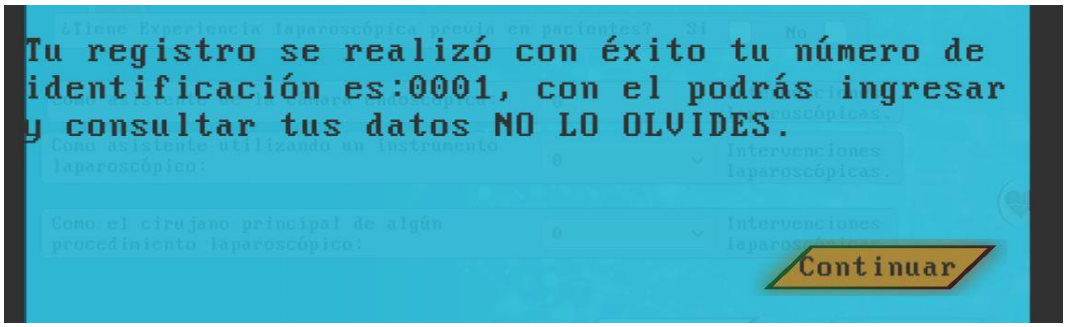

Figura 4.1-4. *Mensaje al registrar un nuevo usuario.*

Al ingresar a las tareas el simulador mantendrá los datos del usuario, por lo que es importante no delegar las tareas a otros usuarios ya que los datos de la persona se verán afectados para su registro y el simulador no cuenta con la opción de borrar individualmente un registro de prueba. En esta parte al elegir una tarea se le dará al usuario 10 segundos en los cuales podrá acomodar los instrumentos laparoscópicos, esto según la figura 4.1-5, que podrá ver el usuario junto con un temporizador.

Figura 4.1-5. *Mensaje de acomodo de las torres y tiempo de espera.*

Al finalizar cualquiera de las tareas, los parámetros de evaluación objetiva del desempeño se escriben en el documento del usuario con el número de identificación como nombre del archivo, después de esto se regresa a la escena del menú y se le muestran los resultados al usuario (Figura 4.1-6).

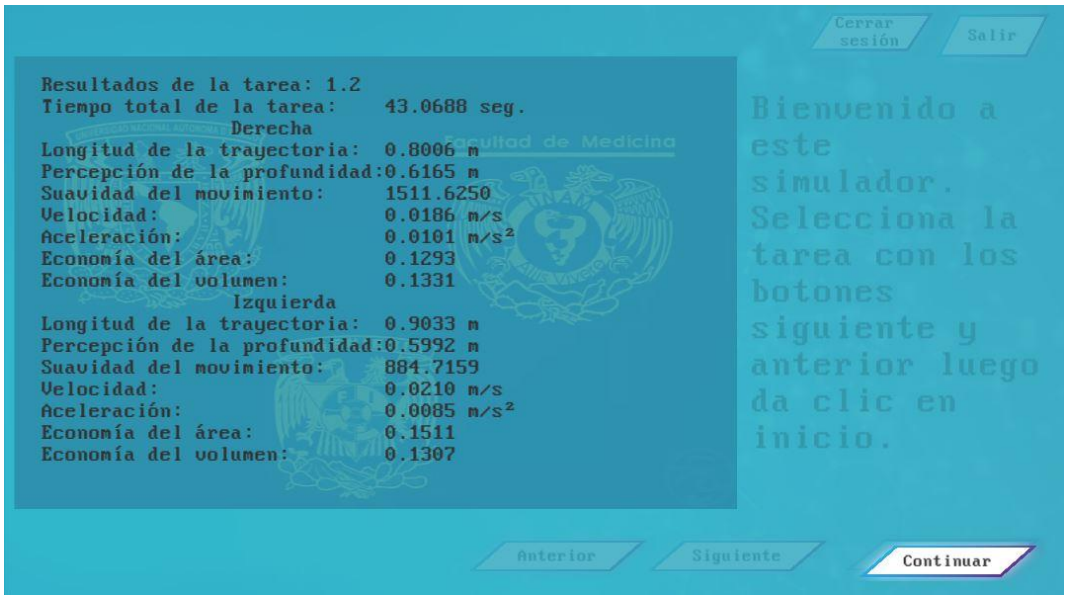

Figura 4.1-6. *Resultados mostrados en pantalla.*

4.2 Evaluación del simulador.

En este apartado se muestran los resultados de las encuestas que se aplicaron a los distintos voluntarios que ocuparon el simulador, estas fueron enfocadas al desempeño de los gráficos y la facilidad de realizar las tareas. En cuanto a la interfaz gráfica del usuario se evaluó el desempeño de las animaciones, así como la interacción que se tenía con esta parte, terminando de realizar sus pruebes se hicieron las siguientes preguntas:

¿Qué tan fácil de entender es la interfaz de usuario?, En respuesta de esto se obtuvo la gráfica de la figura 4.2-1

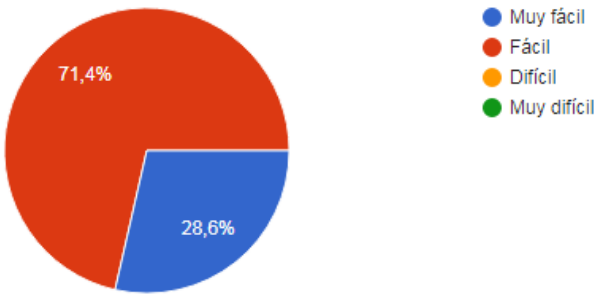

Figura 4.2-1. *Resultados de la interfaz de usuario.*

Aunque la reacción a la interfaz fue muy satisfactoria varios de los encuestados presentaron dificultades al momento de realizar su registro, esto debido a que las instrucciones de llenado del formulario no fueron lo suficientemente claras al momento de elegir la experiencia laparoscópica y el número de

intervenciones. Las siguientes preguntas corresponden al realismo de las tareas, su movimiento y el uso del mecanismo de control. En cuanto al realismo de las tareas la evaluación de los usuarios se ve en la figura 4.2-2.

¿Qué tan realista son los ejercicios implementados?

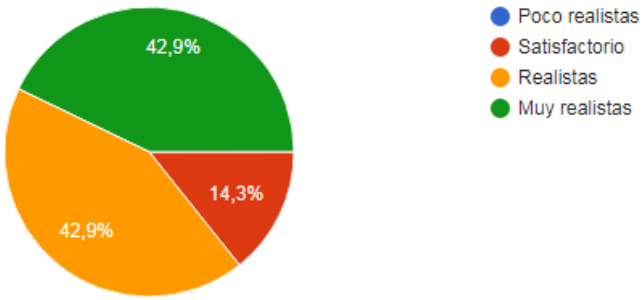

Figura 4.2-2. *Resultados del realismo de las tareas.*

En esta parte la mayoría de los usuarios coincidió que el acercamiento a las tareas a un entrenador tradicional es alto, coincidiendo en el tiempo de realización y similitud de los movimientos, sin embargo, algunos de los encuestados comentaron que sería conveniente tener realimentación por parte del simulador para tener una sensación más real al tocar los objetos virtuales.

Por otra parte, se preguntó sobre el realismo y la velocidad de respuesta del instrumento laparoscópico virtual, en esta parte los participantes notaron que a pesar de que el tiempo de reacción es muy alto y los movimientos realizados por la parte virtual resultaban los mismos, las pinzas virtuales tenían un desfase de ángulo de inclinación, la evaluación de los usuarios se ve en la figura 4.2-3.

¿Cómo calificas el movimiento del instrumento virtual?

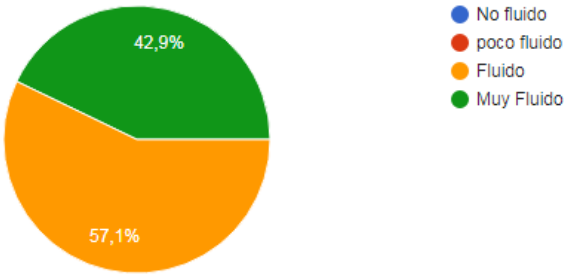

Figura 4.2-3. *Resultados del movimiento del instrumento.*

Finalmente, la evaluación que se dio al montaje de los instrumentos, en cuanto a precisión y movimiento de los instrumentos se muestra en la figura 4.2-4. Los usuarios coincidieron que los movimientos realizados fueron muy similares a los requeridos en los procedimientos y por otra parte comentaron que

es necesario trabajar con el acomodo de las pinzas, en lo que se refiere a distancia y ángulos con el fin de que coincidan mejor con el acomodo real en una intervención, dado que resulto poco más cansado de lo normal.

¿Qué tan realista sientes el movimiento de los instrumentos físicos?

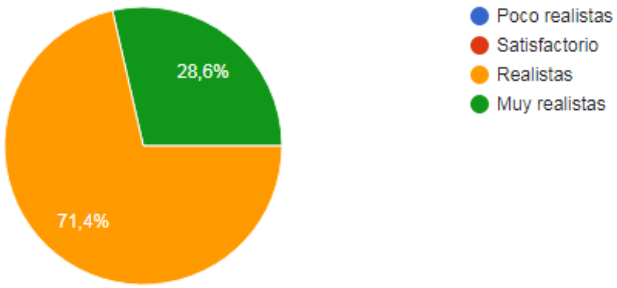

Figura 4.2-4. *Resultados del uso del mecanismo.*

4.3 Pruebas y resultados del simulador laparoscópico.

Para comenzar es necesario dar a conocer los requisitos mínimos que se necesitan para poder ocupar el simulador. Estos parámetros de se obtuvieron al probar el simulador en diferentes equipos de cómputo, entre los cuales se emplea una laptop con procesar *CORE i3*, 4 GB de *RAM* y con 2.1 *GHz* de velocidad, con el equipo mencionado se notó que el simulador solo funcionaba con la calidad más baja, dando prioridad a los movimientos pero no a la parte gráfica, la siguiente computadora en donde se usó el simulador cuenta con un procesador *CORE i5,* 8 GB de *RAM* y una velocidad de 2.3 *GHz*, con estas especificaciones el programa no tiene ningún problema en funcionar con la máxima calidad. Dado las anteriores pruebas se recomiendan equipos con las segundas especificaciones, pero el simulador pide como mínimo los requisitos del primer equipo.

A continuación, se presentan los resultados correspondientes a los voluntarios que usaron el simulador, esto se presenta mediante tablas donde se comparan los tiempos y métricas que se calcularon para cada uno de ellos. Las métricas que se midieron fueron:

- Longitud de la trayectoria, esta representa el recorrido total de la pinza en metros
- Percepción de la profundidad, este parámetro indica la longitud en el plano entrante de las tareas (entrada y salida de la pinza).
- Suavidad del movimiento, con esto se indica el manejo de las pinzas si realiza cambios repentinos o de manera controlada.
- Velocidad y aceleración, estas dos métricas indican los promedios en velocidad y aceleración que realizo el usuario durante la tarea.

- Economía del área y economía del volumen, con esto se indica que tan eficiente se realizó el recorrido de las pinzas laparoscópicas.

La división de las tablas se realizo por las diferentes actividades con las que se cuenta. En la tabla 4.3.1 se presentan los resultados de la tarea de ubicación espacial. Esta tarea se plantea como primer acercamiento al entorno gráfico con el fin de facilitar al participante el cambio de perspectiva tridimensional a bidimensional.

Tabla 4.3.1. Resultados de los usuarios con la tarea de Ubicación espacial.

Métrica	Usuario 1	Usuario 2	Usuario 3	Usuario 4	Usuario 5	Promedio	DE
Tiempo (s)	22.6254	54.9871	22.7992	34.4461	21.5425	31.280	± 14.263
Longitud de la trayectoria (m)	1.3272	3.8126	1.9867	1.1715	1.015	1.863	± 1.151
Percepción de la profundidad (m)	0.9455	3.0244	1.0822	0.8233	0.6814	1.311	± 0.969
Suavidad del movimiento (m/s^3)	281.7912	9926.753	214.8615	735.9832	259.3752	2283.753	± 4277.771
Velocidad (m/s)	0.0587	0.0693	0.0871	0.034	0.0471	0.059	± 0.020
Aceleración (m/s^2)	0.0677	0.1006	0.0798	0.0386	0.05544	0.068	± 0.024
Economía del área (-)	0.0916	0.0709	0.0812	0.0837	0.1123	0.088	± 0.015
Economía del volumen (-)	0.0911	0.0803	0.0695	0.0895	0.1141	0.089	± 0.017

Tabla 4.3.2. Resultados de la tarea de coordinación ojo-mano.

Métrica	Usuario 1	Usuario 2	Usuario 3	Usuario 4	Usuario 5	Promedio	± DE
Tiempo (s)	26.7524	49.8667	34.022	38.9927	26.1761	35.162	± 9.794
Mano Derecha							
Longitud de la trayectoria (m)	1.0399	1.2801	1.6924	1.8357	1.1357	1.397	± 0.350
Percepción de la profundidad (m)	0.7676	0.9162	0.9436	1.636	0.8703	1.027	± 0.347
Suavidad del movimiento (m/s^3)	444.8607	1979.72	617.4817	4927.045	361.5167	1666.125	± 1938.110
Velocidad (m/s)	0.0389	0.0257	0.0497	0.0471	0.0434	0.041	± 0.009
Aceleración (m/s^2)	0.0427	0.0348	0.0499	0.068	0.0424	0.048	± 0.013
Economía del área (-)	0.1087	0.0822	0.0856	0.0352	0.074	0.077	± 0.027
Economía del volumen (-)	0.1108	0.0832	0.0808	0.0295	0.083	0.077	± 0.030

Mano Izquierda							
Longitud de la trayectoria (m)	1.1868	1.5123	1.9335	1.7965	0.9767	1.481	± 0.402
Percepción de la profundidad (m)	0.7623	0.8477	1.2979	1.4505	0.6758	1.007	± 0.345
Suavidad del movimiento (m/s^3)	414.5112	1751.491	560.8372	3823.698	424.6961	1395.047	± 1468.361
Velocidad (m/s)	0.0444	0.0303	0.0568	0.0461	0.0373	0.043	± 0.010
Aceleración (m/s^2)	0.0494	0.0384	0.0557	0.0593	0.0433	0.049	± 0.009
Economía del área (-)	0.1036	0.0718	0.0343	0.0102	0.1342	0.071	± 0.050
Economía del volumen (-)	0.1004	0.0707	0.0309	0.0212	0.1235	0.069	± 0.044

En la siguiente actividad se cuenta con la evaluación de ambas manos. Esta tarea es la coordinación mano-ojo y ayuda al practicante a medir sus movimientos dentro del entorno para poder localizar las profundidades y espacios del ambiente, los resultados de los participantes se muestran en la tabla 4.3.2.

La tercera tarea evaluada fue la transferencia de objetos. En esta el usuario practico la manipulación de objetos y su destreza en los movimientos de una mano, estos resultados se muestran en la tabla 4.3.3

Tabla 4.3.3. *Resultados de la tarea transferencia de objetos.*

Métrica	Usuario 1	Usuario 2	Usuario 3	Usuario 4	Usuario 5	Promedio	DE
Tiempo (s)	71.5641	321.0705	311.8557	98.0682	100.2083	180.553	± 124.623
Longitud de la trayectoria (m)	2.5284	8.0127	5.8202	2.5037	3.0611	4.385	± 2.447
Percepción de la profundidad (m)	1.8321	5.5159	2.5668	1.6406	2.0716	2.725	± 1.598
Suavidad del movimiento (m/s^3)	3800.217	84430.45	59408.65	7067.913	7534.215	32448.289	± 37130.152
Velocidad (m/s)	0.0353	0.025	0.0026	0.0255	0.0305	0.024	± 0.013
Aceleración (m/s^2)	0.0472	0.0354	0.0007	0.0334	0.0418	0.032	± 0.018
Economía del área (-)	0.0466	0.0164	0.138	0.0452	0.0413	0.058	± 0.047
Economía del volumen (-)	0.0458	0.0151	0.141	0.0461	0.0402	0.058	± 0.048

La siguiente tarea es una actividad intermedia, en esta se tiene que manipular las dos pinzas en movimientos finos con el fin de poner grapas en diferentes objetos. Esta tarea es una combinación de la de transferencia y coordinación ojo-mano, los resultados de los participantes se ven en la tabla4.3.4.

Tabla 4.3.4. *Resultados de la tarea engrapado.*

Métrica	Usuario 1	Usuario 2	Usuario 3	Usuario 4	Usuario 5	Promedio	DE
Tiempo (s)	88.4659	255.7527	78.4887	66.716	84.6569	114.816	± 79.215
Mano Derecha							
Longitud de la trayectoria (m)	2.2135	6.2384	2.699	4.5452	1.9404	3.527	± 1.825
Percepción de la profundidad (m)	1.4452	4.5929	1.58	4.2799	1.3707	2.654	± 1.633
Suavidad del movimiento (m/s^3)	5908.319	497704.6	4837.528	23170.55	24585.81	111241.361	± 216238.475
Velocidad (m/s)	0.025	0.0244	0.0344	0.0682	0.0229	0.035	± 0.019
Aceleración (m/s^2)	0.0293	0.0381	0.0385	0.1205	0.0352	0.052	± 0.038
Economía del área (-)	0.0672	0.0102	0.061	0.021	0.0461	0.041	± 0.025
Economía del volumen (-)	0.0601	0.0212	0.0527	0.019	0.0421	0.039	± 0.018
Mano Izquierda							
Longitud de la trayectoria (m)	2.599	5.8201	3.6017	3.8315	2.245	3.619	± 1.398
Percepción de la profundidad (m)	1.5351	3.858	2.2741	2.9931	1.5786	2.448	± 0.988
Suavidad del movimiento (m/s^3)	5928.5	133658.7	12102.09	14716.71	18902.4	37061.680	± 54204.007
Velocidad (m/s)	0.0294	0.0228	0.0459	0.0574	0.0265	0.036	± 0.015
Aceleración (m/s^2)	0.0346	0.027	0.0587	0.0897	0.0385	0.050	± 0.025
Economía del área (-)	0.0349	0.0076	0.0612	0.0301	0.0185	0.030	± 0.020
Economía del volumen (-)	0.0374	0.015	0.0601	0.0256	0.0188	0.031	± 0.018

Finalmente, como última tarea se tiene un corte circular de una membrana. Esta tarea es un ejercicio avanzado debido a que el practicante tiene que controlar de manera muy puntual los movimientos de ambas manos con el objetivo de no exceder los límites de un área dada ya que cuenta con un instrumento de corte y podría resultar peligroso en una cirugía real. Los resultados obtenidos aparecen en la tabla 4.3.5.

Tabla 4.3.5. *Resultados de la tarea corte circular.*

Usuario	Usuario 1	Usuario 2	Usuario 3	Usuario 4	Usuario 5	Promedio	DE
Métrica							
Tiempo (s)	163.1963	107.3648	148.26	76.6437	128.51	124.795	± 34.137
Mano Derecha							
Longitud de la trayectoria (m)	2.7941	2.1875	8.6784	4.8981	2.7005	4.252	± 2.684
Percepción de la profundidad (m)	1.7534	1.4657	5.7798	4.0805	1.8129	2.978	± 1.885
Suavidad del movimiento (m/s^3)	27645.93	9692.713	95439.59	34266.85	21237.12	37656.441	± 33549.054
Velocidad (m/s)	0.0171	0.0204	0.0585	0.0639	0.021	0.036	± 0.023
Aceleración (m/s^2)	0.0281	0.0293	0.0833	0.103	0.0339	0.056	± 0.035
Economía del área (-)	0.0411	0.0394	0.032	0.0402	0.0201	0.035	± 0.009
Economía del volumen (-)	0.0401	0.038	0.0247	0.0484	0.0655	0.043	± 0.015
Mano Izquierda							
Longitud de la trayectoria (m)	1.3343	0.7838	6.7142	2.0331	1.8038	2.534	± 2.385
Percepción de la profundidad (m)	0.8171	0.5433	4.6528	1.2824	1.4366	1.746	± 1.664
Suavidad del movimiento (m/s^3)	34274.86	17208.97	20293.31	7918.606	38417.77	23622.703	± 12561.866
Velocidad (m/s)	0.0082	0.0073	0.0453	0.0265	0.014	0.020	± 0.016
Aceleración (m/s^2)	0.0102	0.0113	0.049	0.0343	0.019	0.025	± 0.017
Economía del área (-)	0.0692	0.1009	0.0046	0.0602	0.0369	0.054	± 0.036
Economía del volumen (-)	0.0718	0.1099	0.0055	0.0507	0.082	0.064	± 0.039

5. Conclusiones.

Los entrenadores para cirugía son herramientas que en la actualidad han ido en aumento rápidamente, esto se debe a la exigencia de los protocolos al requerir horas de práctica en los cirujanos y residentes, es por esto que estas tecnologías tienen que encontrarse en constante desarrollo y mejoras. Con las nuevas tecnologías se busca innovar desde *software* como lo es la implementación de procedimientos complejos e incluso la incorporación de poder recrear la sintomatología de un paciente en específico. Otro punto de innovación es la portabilidad del equipo, buscando con esto simuladores compactos, compatibles con los materiales ya existentes e inclusive adaptables a los hospitales y consultorios médicos.

El trabajo de tesis que se presenta aquí cumplió el objetivo de crear una herramienta válida para el entrenamiento de estudiantes, pasantes, residentes e incluso cirujanos ya formados y que estos pudieran, de alguna manera, adquirir y desarrollar sus habilidades psicomotrices, más en concreto su coordinación mano ojo ya que dicha habilidad es altamente requerían en este tipo de cirugías. El simulador laparoscópico requería tener una sensación visual y sensorial muy cercana a la de una operación en esta técnica quirúrgica, lo cual representó un reto de diseño. Por otro lado, fue necesario incorporar un seguimiento del progreso de los usuarios con el fin de tener una estadística individual de sus mejoras. Para lograrlo se requirió de un trabajo colaborativo multidisciplinario entre las áreas de medicina e ingeniería, donde dicha colaboración se reflejó tanto en el trabajo como en el aprendizaje personal de las partes involucradas. En cuanto a la parte mecánica, se desarrolló un mecanismo de dos torres para montar los instrumentos laparoscópicos y poder instrumentarlos electrónicamente, dicha instrumentación se realizó mediante codificadores digitales logrando obtener los diez movimientos requeridos.

Al momento de hacer la comparación de las métricas obtenidas para los participantes se pudo notar una mejora en sus habilidades y destrezas al manejar los instrumentos, por lo que podemos decir que el simulador se acerca de manera satisfactoria a una nueva herramienta potente en cuanto al entrenamiento laparoscópico; sin embargo, se necesitará de la programación de tareas laparoscópicas de cirugía especializada para alcanzar un nivel competitivo comercialmente. Por el momento, se obtiene una herramienta que ayudara en un nivel académico y de desarrollo permitiendo al Departamento de Cirugía de la Facultad de Medicina de la UNAM su uso para su cuerpo estudiantil, profesores y pasantes en sus estancias médicas.

5.1 Trabajo a futuro.

Como puntos de mejora en el trabajo a futuro, para mejorar la herramienta, se plantea un cambio en la estructura que soporta a los instrumentos quirúrgicos. Algunos de los cambios propuestos son: el primero de ellos sería para la incorporación de un sistema háptico; con la adicción de esto impulsaría el proyecto al siguiente paso de desarrollo, el segundo cambio de la estructura sería en diseño, dejando un mecanismo mejor adaptado en cuestiones estéticas, de ensamble y desensamble y menos propenso a las perturbaciones externas. Otro punto de desarrollo sería complementar el simulador, mediante la

implementación de un nuevo circuito de comunicación, el cual permita la inclusión de pedales para ciertas tareas como es la cauterización. Finalmente, se espera el diseño, desarrollo y programación de nuevas tareas que vayan desde básicas como puede ser la manipulación de objetos, actividades intermedias de cauterización, y otras actividades avanzadas como la sutura e incluso la implementación de procedimientos quirúrgicos completos.

6. Referencias.

[1] M. Pérez Albacete, «Historia de la cirugía laparoscópica y de la terapia mínimamente invasiva,» Desconoido Desconocido 2005. [En línea]. Available: http://historia.aeu.es/Docs/HISTORIA_DE_LA_CIRUGIA_LAPAROSCOPICA.pdf. [Último acceso: 15 Septiembre 2017].

[2] S. S., Laparoscopy, Desconocido: PhD magill's Medical Guide, 2016.

[3] I. Badash, K. Burtt, C. A. Solorzano y J. N. Carey, «Innovations in surgery simulation: a review of past, current and future techniques,» *Annals of Translational Medicine,* vol. desconocido, nº desconocido, p. desconocido, 24 Diciembre 2016.

[4] M. E. Aponte Rueda, R. Saade Cárdenas y S. Navarrete Aulestia , «"Simulador laparoscópico como herramienta de aprendizaje",» *revista de la facultad de medicina Caracas,* vol. Descinocido, nº Desconocido, p. 5, Sin día Diciembre 2009.

[5] H. M. Hasson, N. V. Aruna Kumari y J. Eekhout, «Training Simulator for Developing Laparoscopic Skills,» *Jornal of the Society of Laparoendoscopic Surgeons,* vol. desconocido, nº desconocido, pp. 255-265, Julio-Septiembre 2001.

[6] Flaviu, History of Simulations, Desconocido: No impreso, 2016.

[7] T. D. Moreno Hilarios, Diseño y contrucción de un prototipo de simulador con realidad virtual para cirugía laparoscópica (Tesis), Ciudad Univercitaria Ciudad de México: Tesis, 2017.

[8] A. Rizzuto y P. Nudo, «The twin forceps: A new intrument for SILS,» ResearchGate, Desconocido Junio 2015. [En línea]. Available: https://www.researchgate.net/profile/Antonia_Rizzuto/publication/280841023/figure/fig6/AS:324016201060369@1454262890391/Helago-laparoscopic-box-trainer.png. [Último acceso: 01 Mayo 2018].

[9] Medical-X, «Hralthy simulation,» Desconocido Desconocido Desconocido. [En línea]. Available: http://www.healthysimulation.com/wp-content/uploads/2014/07/LAPX.jpg. [Último acceso: 20 Abril 2018].

[10] K. Kawaguchi, H. Egi, M. Hattori, H. Sawada, T. Suzuki y H. Ohdan, «Validation of a novel basic virtual reality simulator, the LAP-X, for training basic skills,» *Minimally Invasive Therapy & Allied Technologies,* pp. 287-293, 2014.

[11] Simbionix, «3D systems,» [En línea]. Available: http://www.medicalexpo.it/prod/simbionix/product-81276-629619.html. [Último acceso: 17

Mayo 2018].

[12] Simbionix, «3D systems,» 2017. [En línea]. Available: http://simbionix.com/simulators/lap-mentor/. [Último acceso: 17 Mayo 2018].

[13] S. S. S. AB, «Surgical science,» Surgical Science Sweden AB, Desconosido Desconosido Desconosido. [En línea]. Available: https://surgicalscience.com/. [Último acceso: 30 Abril 2018].

[14] M. K. Chmarra y C. A. G. &. J. Dankelman, «Systems for tracking minimally invasive surgical instruments,» *Minimally Invasive Therapy & Allied Technologies,* pp. 328-340, 2009.

[15] 1. L. L. J. E.G.G.Verdaasdonk, «Validation of a new basic virtual reality simulator for training of basic endoscopic skills,» *Surgical endoscopy,* pp. 3-11, 2006.

[16] Ó. Torrente, Arduino curso práctico de formación, Primera ed., Madrid: Alfaomega, 2014, p. 558.

[17] Arduino, «Getting started with the Arduino Due,» Arduino, 10 Enero 2017. [En línea]. Available: https://www.arduino.cc/en/Guide/ArduinoDue. [Último acceso: 11 Mayo 2018].

[18] Desconocido, «mcbtec,» 12 Enero 2008. [En línea]. Available: http://www.mcbtec.com/pdf/Funcionamiento_*Encoder*.pdf. [Último acceso: 05 Mayo 2018].

[19] Usdigital, «Usdigital,» 20 Mayo 2014. [En línea]. Available: https://www.usdigital.com/products/e4t. [Último acceso: 18 Abril 2018].

[20] G. Jiménez Díaz, «Facultad de Informá (UCM),» sin día sin mes 2015. [En línea]. Available: http://gaia.fdi.ucm.es/files/people/guille/taller*Unity*2015/material/guion.pdf. [Último acceso: 13 Abril 2018].

[21] *Unity*, «Manual *Unity*,» *Unity*, 30 Abril 2018. [En línea]. Available: https://docs.*Unity*3d.com/Manual/. [Último acceso: 11 Mayo 2018].

[22] D. H. Myszka, Máquinas y mecanismos, Inglaterra: Person, 2012.

7. Apéndices.

7.1 Códigos del simulador.

7.1.1 Código del microcontrolador.

```
#include <Encoder.h>
Encoder E1 (22,23);
Encoder E2 (24,25);
Encoder E3 (26,27);
Encoder E4 (28,29);
Encoder E5 (30,31);
Encoder E6 (32,33);
Encoder E7 (34,35);
Encoder E8 (36,37);
Encoder E9 (38,39);
Encoder E10 (40,41);
void setup() {
   Serial.begin(115200);
   E1.write(0);
   E2.write(0);
   E3.write(0);
   E4.write(0);
   E5.write(0);
   E6.write(0);
   E7.write(0);
   E8.write(0);
   E9.write(0);
   E10.write(0);
}
void loop() {
   encoder1=E1.read();
   encoder2=E2.read();
   encoder3=E3.read();
   encoder4=E4.read();
   encoder5=E5.read();
   encoder6=E6.read();
   encoder7=E7.read();
   encoder8=E8.read();
   encoder9=E9.read();
   encoder10=E10.read();
if(encoder1>-5) encoder1=0;
if(encoder1<-120) encoder1=-120;
if(encoder2>1000) encoder2=1000;
if(encoder2<-1000) encoder2=-1000;
if(encoder3>10) encoder3=10;
if(encoder3<-5000) encoder3=-5000;
if(encoder4<-210) encoder4=-210;
if(encoder4>0) encoder4=0;
if(encoder5>7120) encoder5=7120;
if(encoder5<-7120) encoder5=-7120;
if(encoder6<10) encoder6 = 10;
```

```
if(encoder6>100) encoder6 =100;
if(encoder7<-210) encoder7 = -200;
if(encoder7>230) encoder7 =230;
if(encoder8<-6000) encoder8 = -6000;
if(encoder8>-1000) encoder8 =-1000;
Serial.print (map(encoder1, 0, -120, 0,  20)); //(map(encoder1, 0, 120, 0, 20)) ; //A/Cierra
inviero el 20 con el 0 en el mapeo  y 0, 120 por 0, -120
Serial.print(“,”);
Serial.print (encoder5); //* Giro
Serial.print(“,”);
Serial.print (map(encoder2, -250, 250, -80, 80)); //* D/I  Modifico 180, -180 a 80, -80 y -360,
360 a 250, 250
Serial.print(“,”);
Serial.print (map(encoder4, 0, -160, -90, -60)); //* A/A d Modifico 50, -50 a -90, 0
Serial.print(“,”);
Serial.print (map(encoder3, 10, -5000, -17, 9)); //zoom (encoder3, -5000, 5000, 100, -150ó-
180)(encoder3, -5000, 5000, 0, 150) Modifico -10, -190 a -5, 4
// Serial.println(encoder4);
Serial.print(“,”);
Serial.print(map(encoder6, 10, 100, 0, 20)); // Abrir/cerrar
Serial.print(“,”);
Serial.print(map(encoder7, -200, 230, 80, -80)); // Derecha/Izquierda
//Serial.print(encoder7);
Serial.print(“,”);
Serial.print(map(encoder8, -1000, -6000, -17, 9)); //zoom 2
//Serial.print(encoder8);
Serial.print(“,”);
Serial.print(map(encoder9, 0, 157, -90, -50)); //arriba/abajo
//Serial.print(encoder9);
Serial.print(“,”);
Serial.println(encoder10); //Guiro
delay(100);//IMPORTANTE
}
```

7.1.2 Código del inicio de sesión.

```
    public void btnRegistro() //Llamado a la rutina del registro
 {
audio1.clip = clipOnSelected;
audio1.Play();
StartCoroutine(NuevoRegistro());
 }
    public void btnEntrada()
 {
audio1.clip = clipOnSelected;
audio1.Play();
comprobarUsuario();
if (exist)
    {
    PersistentDataScript.identificador = Id.text;
    StartCoroutine(MenuTareas());
    }
    else if (!exist)
```

```
        {
        Id.textComponent.color = Color.red;
        }
    }
        public void btnEntrada2() //Comprobación del usuario y la contraseña
    {
    audio1.clip = clipOnSelected;
    audio1.Play();
    bool bandera = false;
    xmldoc.Load(Application.dataPath + "/Resources/databaseadmin.xml");
    string search = User.text;
    XmlElement tabla = xmldoc.DocumentElement;
    XmlNodeList listaUsuarios = xmldoc.SelectNodes("Tabla/usuario");
        foreach (XmlNode item in listaUsuarios)
        {
        if (item.FirstChild.InnerText == search)
            {
    bandera = true;
    User.textComponent.color = Color.black;
    XmlNode foundUser = item;
    passwordTem = foundUser.SelectSingleNode("password").InnerText;
    if (password.text == passwordTem)
                {
    PersistentDataScript.Administrador = User.text;
    StartCoroutine(AdminScene());
    break;
                }
        else
                {
    //contraseña incorrecta
    password.textComponent.color = Color.red;
    break;
                }
            }
        else if (item.FirstChild.InnerText != search && bandera == false)
            {
    //Usuario incorrecto
    User.textComponent.color = Color.red;
            }
        }
    }
        public void btnCerrarApp() //Cerrar el programa
    {
    audio1.clip = clipOnSelected;
    audio1.Play();
    Application.Quit();
    }
        public void backBlackInput()//Cambio de color en la entrada de texto
    {
    password.textComponent.color = Color.black;
    User.textComponent.color = Color.black;
    }
    private void comprobarUsuario() //comprobación del usuario mediante lectura de la base de
datos
    {
    xmldoc.Load(Application.dataPath + "/Resources/databaseusers.xml");
```

```
   XmlNodeList listaUsuarios = xmldoc.SelectNodes(“Tabla/usuario”);
   XmlNode aUser;
   string idex;
   for (int i = 0; i < listaUsuarios.Count; i++)
       {
   aUser = listaUsuarios.Item(i);
   idex = aUser.SelectSingleNode(“id”).InnerText;
   if (idex == Id.text)
           {
      exist = true;
      break;
           }
        else
           {
        exist = false;
           }
       }
    }
        public void OnCursorIn() // función para hacer sonar el botón
    {
   audio1.clip = clipOnCursor;
   audio1.Play();
    }
        IEnumerator NuevoRegistro()
        {
        yield return new WaitForSeconds(0.5f);
    SceneManager.LoadScene (“NewDataScene”, LoadSceneMode.Single); //Entrada al registro de
usuarios
        }
        IEnumerator MenuTareas()
        {
        yield return new WaitForSeconds(0.5f);
    SceneManager.LoadScene (“MenuScene”, LoadSceneMode.Single); //Entrada a la escena del menú
de tareas
        }
        IEnumerator AdminScene()
    {
   yield return new WaitForSeconds(0.5f);
   password.textComponent.color = Color.black;
   User.textComponent.color = Color.black;
    SceneManager.LoadScene(“RootUserScene”, LoadSceneMode.Single); //Entrada a la escena del
administrador
    }
}
```

7.1.3 Código de datos persistentes.

```
        private void Awake() //Mantener el este script y destruir el anterior
    {
        if (persistendata == null)
        {
   persistendata = this;
   DontDestroyOnLoad(this);
        }
        else if (persistendata != this)
        {
```

```
        Destroy(this);
        }
    }
        void Update ()
        {
        if (Enviados1 && Enviados2) //Esperar a resivir los datos
        {
   PersistentDataScript.Enviados1 = false;
   PersistentDataScript.Enviados2 = false;
   deltaT = TiempoFinal / TotalDatos;
   SpeedD = SpeedD / deltaT;
   SpeedI = SpeedI / deltaT;
   AccD = AccD / deltaT;
   AccI = AccI / deltaT;
   try //compribar si el documento existe
           {
    StreamReader reader = new StreamReader(Application.dataPath + "/Resources/" + identificador
+ ".txt"); //abrirlo
   datosleidos = reader.ReadToEnd();
   reader.Close();
           }
        catch //Crearlo
           {
    StreamWriter myfile1 = new StreamWriter(Application.dataPath + "/Resources/" + identificador
+ ".txt");
        myfile1.Close();
           }
    StreamReader reader1 = new StreamReader(Application.dataPath + "/Resources/" + identificador
+ ".txt");
   datosleidos = reader1.ReadToEnd();
   reader1.Close();
    StreamWriter writer1 = new StreamWriter(Application.dataPath + "/Resources/" + identificador
+ ".txt");
   writer1.Write(datosleidos); //escribir en el documento los nuevos datos
   writer1.WriteLine(System.DateTime.Now.ToString("dd/MM/yyyy hh:mm:ss"));
   writer1.WriteLine("Resultados de la tarea " + tarea);
           writer1.WriteLine("Tiempo total de la tarea:    " + TiempoFinal.ToString("F4") + " seg.");
           writer1.WriteLine("               Derecha                                ");
           writer1.WriteLine("Longitud de la trayectoria:  " + PLD.ToString("F4")         + " m");
           writer1.WriteLine("Percepción de la profundidad:" + DPD.ToString("F4")         + " m");
           writer1.WriteLine("Suavidad del movimiento:     " + MSD.ToString("F4")         + "");
           writer1.WriteLine("Velocidad:                   " + SpeedD.ToString("F4")      + " m/s");
           writer1.WriteLine("Aceleración:                 " + AccD.ToString("F4")        + " m/s²");
           writer1.WriteLine("Economía del área:           " + ecoAD.ToString("F4") + "");
           writer1.WriteLine("Economía del volumen:        " + ecoVD.ToString("F4") + "");
           writer1.WriteLine("               Izquierda                              ");
           writer1.WriteLine("Longitud de la trayectoria:  " + PLI.ToString("F4")         + " m");
           writer1.WriteLine("Percepción de la profundidad:" + DPI.ToString("F4")         + " m");
           writer1.WriteLine("Suavidad del movimiento:     " + MSI.ToString("F4")         + "");
           writer1.WriteLine("Velocidad:                   " + SpeedI.ToString("F4")      + " m/s");
           writer1.WriteLine("Aceleración:                 " + AccI.ToString("F4")        + " m/s²");
           writer1.WriteLine("Economía del área:           " + ecoAI.ToString("F4")        + "");
           writer1.WriteLine("Economía del volumen:        " + ecoVI.ToString("F4")     + "");
   writer1.WriteLine("-----------------------------------------------");
   writer1.Close();
   TerninoTarea = true; // indicar que se muestren los resultados en pantalla
        }
```

```
        }
}
```

7.1.4 Código escena del administrador.

```
        public void SearchXML() // buscar usuario y mostrar datos
    {
   audio1.clip = clipOnSelected;
   audio1.Play();
   xmldoc.Load(Application.dataPath + "/Resources/databaseusers.xml");
   string search = id.text;
   XmlElement tabla = xmldoc.DocumentElement;
   XmlNodeList listaUsuarios = xmldoc.SelectNodes("Tabla/usuario");
   foreach (XmlNode item in listaUsuarios)
        {
        if (item.FirstChild.InnerText == search)
           {
        try
               {
    StreamReader reader = new StreamReader(Application.dataPath + "/Resources/" + search +
".txt");
   datosleidos = reader.ReadToEnd();
   reader.Close();
               }
        catch
               {
        datosleidos = "";
               }
   XmlNode foundUser = item;
   datos.text =
        "Nombre: " + foundUser.SelectSingleNode("nombre").InnerText
   • "\nEdad: " + foundUser.SelectSingleNode("edad").InnerText
   • "\nGenero: " + foundUser.SelectSingleNode("genero").InnerText
   • "\nMano dominante: " + foundUser.SelectSingleNode("mano").InnerText
   • "\nEspecialidad Médica: " + foundUser.SelectSingleNode("emedica").InnerText
   • "\nAño de residencia: " + foundUser.SelectSingleNode("residencia").InnerText
   • "\n" + datosleidos;

   actualizar.interactable = true;
   borrar.interactable = true;
   break;
           }

        else
           {
   datos.text = "El usuario no existe";
   actualizar.interactable = false;
   borrar.interactable = false;
           }
        }
    }
    public void Borrar() //Borrar usuario
    {
   audio1.clip = clipOnSelected;
   audio1.Play();
```

```
    xmldoc.Load(Application.dataPath + “/Resources/databaseusers.xml”);
    string search = id.text;
    XmlElement tabla = xmldoc.DocumentElement;
    XmlNodeList listaUsuarios = xmldoc.SelectNodes(“Tabla/usuario”);
    foreach (XmlNode item in listaUsuarios)
        {
        if (item.FirstChild.InnerText == search)
            {
    XmlNode nodeOld = item;
    tabla.RemoveChild(nodeOld);
     StreamWriter killer = new StreamWriter(Application.dataPath + “/Resources/” + search +
“.txt”);
    killer.Flush();
    killer.Close();
            }
        }
    datos.text = “Usuario eliminado”;
    xmldoc.Save(Application.dataPath + “/Resources/databaseusers.xml”);
     }

     public void Actualizar() //entrar para actualizar datos del usuario
     {
    audio1.clip = clipOnSelected;
    audio1.Play();
    PersistentDataScript.identificador = id.text;
    StartCoroutine(GoUpdate());
     }
     public void addNewUser() //agregar nuevo administrator
     {
    audio1.clip = clipOnSelected;
    audio1.Play();
    if (pass1.text != pass2.text)
        {
        pass2.textComponent.color = Color.red;
        }
        else
        {
    AddUser();
    panel1.SetActive(false);
    panel2.SetActive(false);
    panel3.SetActive(false);
        }
     }
     public void newPassBTN() //actualizar la contraseña de un usuario
     {
    audio1.clip = clipOnSelected;
    audio1.Play();
    if (pass3.text != pass4.text)
        {
        pass4.textComponent.color = Color.red;
        }
        else
        {
    DatabaseUpdate();
    panel1.SetActive(false);
```

```
    panel2.SetActive(false);
    panel3.SetActive(false);
        }
     }
     private void AddUser() //agregar al usuario al documento
     {
         xmldoc.Load(Application.dataPath + "/Resources/databaseadmin.xml");
         XmlNode usuario = NewUser(useN.text, pass1.text);
    XmlNode nodoRaiz = xmldoc.DocumentElement;
    nodoRaiz.InsertAfter(usuario, nodoRaiz.LastChild);
    xmldoc.Save(Application.dataPath + "/Resources/databaseadmin.xml");
     }
     private XmlNode NewUser(string idd, string nombred) //agregar al usuario al documento
     {
    XmlNode usuario = xmldoc.CreateElement("usuario");
    XmlElement idElement = xmldoc.CreateElement("nombre");
    idElement.InnerText = idd;
    usuario.AppendChild(idElement);
    XmlElement nombreElement = xmldoc.CreateElement("password");
    nombreElement.InnerText = nombred;
    usuario.AppendChild(nombreElement);
    return usuario;
     }
     public void DatabaseUpdate() //actualizar la contraseña en la base de datos
     {
    xmldoc.Load(Application.dataPath + "/Resources/databaseadmin.xml");
    string update = PersistentDataScript.Administrador;
XmlElement tabla = xmldoc.DocumentElement;
XmlNodeList listaUsuarios = xmldoc.SelectNodes("Tabla/usuario");
XmlNode usuario = NewUser(PersistentDataScript.Administrador, pass3.text);
    foreach (XmlNode item in listaUsuarios)
        {
        if (item.FirstChild.InnerText == update)
            {
    XmlNode nodeOld = item;
    tabla.ReplaceChild(usuario, nodeOld);
            }
        }
        xmldoc.Save(Application.dataPath + "/Resources/databaseadmin.xml");
     }
     public void InvalidUser() //comprobar si el nombre de usuario no está repetido
     {
    xmldoc.Load(Application.dataPath + "/Resources/databaseadmin.xml");
    string update = useN.text;
    XmlElement tabla = xmldoc.DocumentElement;
    XmlNodeList listaUsuarios = xmldoc.SelectNodes("Tabla/usuario");
    foreach (XmlNode item in listaUsuarios)
        {
        if (item.FirstChild.InnerText == update)
            {
       useN.textComponent.color = Color.red;
       StartCoroutine(error());
       break;
            }
        }
```

```
    }
    IEnumerator error() // en caso de que este repetido
    {
   yield return new WaitForSeconds(0.5f);
   useN.textComponent.color = Color.black;
   useN.text = "";
    }

    IEnumerator GoUpdate()//cambiar a la escena actualizar usuario
    {
        yield return new WaitForSeconds(0.5f);
        SceneManager.LoadScene("UpdateScene", LoadSceneMode.Single);
    }

    IEnumerator GoSingin()//Cambiar a la escena inicio de sesión
    {
        yield return new WaitForSeconds(0.5f);
        SceneManager.LoadScene("SignInScene", LoadSceneMode.Single);
    }
}
```

7.1.5 Código del menú.

```
        void Start ()
    {
    if (PersistentDataScript.TerninoTarea) // si termino una tarea mostrar resultados
        {
   PersistentDataScript.TerninoTarea = false;
   panel.SetActive(true);
   resultados.text =
   "Resultados de la tarea: " + PersistentDataScript.tarea.ToString() +
           "\nTiempo total de la tarea:    " + PersistentDataScript.TiempoFinal.ToString("F4") + " seg." +
           "\n               Derecha                                          " +
           "\nLongitud de la trayectoria:  " + PersistentDataScript.PLD.ToString("F4")         + " m" +
           "\nPercepción de la profundidad:" + PersistentDataScript.DPD.ToString("F4")         + " m" +
           "\nSuavidad del movimiento:     " + PersistentDataScript.MSD.ToString("F4")         +
           "\nVelocidad:                   " + PersistentDataScript.SpeedD.ToString("F4")      + " m/s" +
           "\nAceleración:                 " + PersistentDataScript.AccD.ToString("F4")        + " m/s²" +
           "\nEconomía del área:           " + PersistentDataScript.ecoAD.ToString("F4")  +
           "\nEconomía del volumen:        " + PersistentDataScript.ecoVD.ToString("F4")  +
           "\n               Izquierda                                        " +
    "\nLongitud de la trayectoria:  " + PersistentDataScript.PLI.ToString("F4") + " m" +
    "\nPercepción de la profundidad:" + PersistentDataScript.DPI.ToString("F4") + " m" +
           "\nSuavidad del movimiento:     " + PersistentDataScript.MSI.ToString("F4") +
           "\nVelocidad:                   " + PersistentDataScript.SpeedI.ToString("F4") + " m/s" +
           "\nAceleración:                 " + PersistentDataScript.AccI.ToString("F4")        + " m/s²"+
           "\nEconomía del área:           " + PersistentDataScript.ecoAI.ToString("F4")  +
           "\nEconomía del volumen:        " + PersistentDataScript.ecoVI.ToString("F4");
        }
   tiempo = 10.0f;
   contar = false;
    }
        void Update ()
    {
        switch (i)//Mostrar videos y descripciones de las tareas
        {
        case 0:
                            instructions.text = "Bienvenido a este simulador.     Selecciona la tarea
con los botones siguiente y anterior luego da clic en inicio.";
                            instructions.fontSize = 28;
```

```
                            movie.clip = VideoTareas [0];
   inicio.interactable = false;
   break;
        case 1:
    instructions.text = “En esta tarea tiene que tocar con la punta de la pinza los objetos,
estos se encuentran a diferentes distancias y alturas. Ejercicio con la mano derecha.”;
   instructions.fontSize = 25;
   movie.clip = VideoTareas [1];
   inicio.interactable = true;
   break;
    case 2:
    instructions.text = “Esta tarea consiste en trasladar los aros de los seis ejes del lado
izquierdo a los seis ejes del lado derecho. Ejercicio con la mano derecha.”;
        instructions.fontSize = 25;
        movie.clip = VideoTareas [2];
   inicio.interactable = true;
   break;
    case 3:
    instructions.text = “En esta tarea tiene que tocar con la punta de la pinza los objetos,
estos se encuentran a diferentes distancias y alturas.”;
   instructions.fontSize = 25;
   movie.clip = VideoTareas[3];
   inicio.interactable = true;
   break;
    case 4:
    instructions.text = “En esta tarea tiene que cortar alrededor de la membrana, corte con la
mano derecha.”;
   instructions.fontSize = 25;
   movie.clip = VideoTareas[4];
   inicio.interactable = true;
   break;
    case 5:
    instructions.text = “En esta tarea tiene que colocar las grapas en el área indicada de color
verde.”;
   instructions.fontSize = 25;
   movie.clip = VideoTareas[5];
   inicio.interactable = true;
   break;
    default:
    instructions.text = “Bienvenido a este simulador.    Selecciona la tarea con los botones
siguiente y anterior luego da clic en inicio.”;
        instructions.fontSize = 28;
        movie.clip = VideoTareas [0];
   inicio.interactable = false;
   break;
        }
    if(contar) //tiempo de espera para acomodo de torres
        {
   tiempo -= Time.deltaTime;
   tiempoEspera.text = “” + tiempo.ToString(“f0”);
        }
        }
    public void Play()//seleccionar tarea a realizar
```

```
 {
audio1.clip = clipOnSelected;
audio1.Play();
if (i != 0)
    {
contar = true;
espera.SetActive(true);
StartCoroutine(cambiarEscena());
    }
 }
 public void NextBttn()//incrementar el indicador de casos
 {
audio1.clip = clipOnSelected;
audio1.Play();
i++;
if (i > maxScene) i = 0;
 }
 public void BackBttn() //decrementar el indicador de casos
 {
audio1.clip = clipOnSelected;
audio1.Play();
i--;
if (i < 0) i = maxScene;
 }
 public void BttnSalir() //cerrar programa
 {
audio1.clip = clipOnSelected;
audio1.Play();
Application.Quit();
 }
 IEnumerator Entrada()//regresar a la escena inicio de sesión
 {
    yield return new WaitForSeconds(0.5f);
    SceneManager.LoadScene("SignInScene", LoadSceneMode.Single);
 }
 IEnumerator cambiarEscena() //Entrar a la tarea espesifica
 {
yield return new WaitForSeconds(10f);
switch (i)
    {
    case 0:
contar = false;
espera.SetActive(false);
tiempo = 10.0f;
break;
    case 1:
SceneManager.LoadScene("T1", LoadSceneMode.Single);
break;
case 2:
SceneManager.LoadScene("Tarea2", LoadSceneMode.Single);
break;
case 3:
SceneManager.LoadScene("T12P", LoadSceneMode.Single);
break;
case 4:
```

```
    SceneManager.LoadScene(“Tarea3”, LoadSceneMode.Single);
    break;
    case 5:
    SceneManager.LoadScene(“Tarea4”, LoadSceneMode.Single);
    break;
    default:
    break;
        }
     }
}
```

7.1.6 Código para cálculo de parámetros de evaluación.

```
        void Update()
    {
        if (!PersistentDataScript.endTask)
        {
    tiempo = tiempo + 1 * Time.deltaTime;
    ntDatos++;
    posXDf[datoDf] = posX[dato] = transform.position.x;
    posYDf[datoDf] = posY[dato] = transform.position.y;
    posZDf[datoDf] = posZ[dato] = transform.position.z;
    if (posX[dato] < minX) minX = posX[dato];
    if (posY[dato] < minY) minY = posY[dato];
    if (posZ[dato] < minZ) minZ = posZ[dato];
    if (posX[dato] > maxX) maxX = posX[dato];
    if (posY[dato] > maxY) maxY = posY[dato];
    if (posZ[dato] > maxZ) maxZ = posZ[dato];
    dato++;
    if (dato >= 2)
            {
     dis1 = Mathf.Sqrt(Mathf.Pow((posX[dato - 2] - posX[dato - 1]), 2) + Mathf.Pow((posY[dato -
2] - posY[dato - 1]), 2) + Mathf.Pow((posZ[dato - 2] - posZ[dato - 1]), 2));
    dis1 = pld + dis1;
    pld = dis1;

     dis2 = Mathf.Sqrt(Mathf.Pow((posY[dato - 2] - posY[dato - 1]), 2) + Mathf.Pow((posZ[dato -
2] - posZ[dato - 1]), 2));
    dis2 = dpd + dis2;
    dpd = dis2;
    posX[dato - 2] = posX[dato - 1];
    posY[dato - 2] = posY[dato - 1];
    posZ[dato - 2] = posZ[dato - 1];
    dato--;
            }
        diferencial();
        }
        else if (PersistentDataScript.endTask && !flag)
        {
    flag = true;
    totalTime = tiempo;
    PersistentDataScript.TiempoFinal = totalTime;
    PersistentDataScript.TotalDatos = ntDatos;
    PersistentDataScript.PLD = pld;
```

```
   PersistentDataScript.DPD = dpd;
   cte = (Mathf.Pow(totalTime, 5)) / (2 * (Mathf.Pow(pld, 2)));
   ms = Mathf.Sqrt(cte*sumT);
   //enviar dato indicando la pinza
   PersistentDataScript.MSD = ms;
   MSpeed = speed / ntDatos;
   // enviar dato indicando pinza
   PersistentDataScript.SpeedD = MSpeed;
   Macc = acc / ntDatos;
   // enviar dato indicando pinza
   PersistentDataScript.AccD = Macc;
   horizontal = (Mathf.Abs(maxX)) - (Mathf.Abs(minX));
   vertical = (Mathf.Abs(maxZ)) - (Mathf.Abs(minZ));
   altura = (Mathf.Abs(maxY)) - (Mathf.Abs(minY));
   area = vertical * horizontal;
   volumen = area * altura;
   ecoA = (Mathf.Sqrt(area)) / pld;
   ecoV = (Mathf.Pow(volumen, (0.3333333333333333f))) / pld;
   PersistentDataScript.ecoAD = ecoA;
   PersistentDataScript.ecoVD = ecoV;
   PersistentDataScript.Enviados1 = true;
       }
    }
       public void diferencial()
    {
       if (calculo1)
       {
   posXDf1[3] = -posXDf[2] + posXDf[3];
   posXDf2[2] = -posXDf1[2] + posXDf1[3];
   posXDf3[1] = -posXDf2[1] + posXDf2[2];
   posYDf1[3] = -posYDf[2] + posYDf[3];
   posYDf2[2] = -posYDf1[2] + posYDf1[3];
   posYDf3[1] = -posYDf2[1] + posYDf2[2];
   posZDf1[3] = -posZDf[2] + posZDf[3];
   posZDf2[2] = -posZDf1[2] + posZDf1[3];
   posZDf3[1] = -posZDf2[1] + posZDf2[2];
    speed = speed + (Mathf.Sqrt((Mathf.Pow(posXDf1[3], 2)) + (Mathf.Pow(posYDf1[3], 2)) +
(Mathf.Pow(posZDf1[3], 2))));
    acc = acc + (Mathf.Sqrt((Mathf.Pow(posXDf2[2], 2)) + (Mathf.Pow(posYDf2[2], 2)) +
(Mathf.Pow(posZDf2[2], 2))));
   xC = Mathf.Pow(posXDf3[1], 2);
   yC = Mathf.Pow(posYDf3[1], 2);
   zC = Mathf.Pow(posZDf3[1], 2);
   sum = xC + yC + zC;
   sumT = aux3 + sum;
   aux3 = sumT;
   posXDf[2] = posXDf[3];
   posYDf[2] = posYDf[3];
   posZDf[2] = posZDf[3];
   posXDf1[2] = posXDf1[3];
   posYDf1[2] = posYDf1[3];
   posZDf1[2] = posZDf1[3];
```

```
   posXDf2[1] = posXDf2[2];
   posYDf2[1] = posYDf2[2];
   posZDf2[1] = posZDf2[2];
   datoDf--;
       }
       if (datoDf == 3 && !calculo1)
       {
   posXDf1[0] = -posXDf[0] + posXDf[1];
   posXDf1[1] = -posXDf[1] + posXDf[2];
   posXDf1[2] = -posXDf[2] + posXDf[3];
   posYDf1[0] = -posYDf[0] + posYDf[1];
   posYDf1[1] = -posYDf[1] + posYDf[2];
   posYDf1[2] = -posYDf[2] + posYDf[3];
   posZDf1[0] = -posZDf[0] + posZDf[1];
   posZDf1[1] = -posZDf[1] + posZDf[2];
   posZDf1[2] = -posZDf[2] + posZDf[3];
    speed = (Mathf.Sqrt((Mathf.Pow(posXDf1[0], 2)) + (Mathf.Pow(posYDf1[0], 2)) +
(Mathf.Pow(posZDf1[0], 2))))
    + (Mathf.Sqrt((Mathf.Pow(posXDf1[1], 2)) + (Mathf.Pow(posYDf1[1], 2)) +
(Mathf.Pow(posZDf1[1], 2))))
    + (Mathf.Sqrt((Mathf.Pow(posXDf1[2], 2)) + (Mathf.Pow(posYDf1[2], 2)) +
(Mathf.Pow(posZDf1[2], 2))));
   posXDf2[0] = -posXDf1[0] + posXDf1[1];
   posXDf2[1] = -posXDf1[1] + posXDf1[2];
   posYDf2[0] = -posYDf1[0] + posYDf1[1];
   posYDf2[1] = -posYDf1[1] + posYDf1[2];
   posZDf2[0] = -posZDf1[0] + posZDf1[1];
   posZDf2[1] = -posZDf1[1] + posZDf1[2];
    acc = (Mathf.Sqrt((Mathf.Pow(posXDf2[0], 2)) + (Mathf.Pow(posYDf2[0], 2)) +
(Mathf.Pow(posZDf2[0], 2))))
    + (Mathf.Sqrt((Mathf.Pow(posXDf2[1], 2)) + (Mathf.Pow(posYDf2[1], 2)) +
(Mathf.Pow(posZDf2[1], 2))));
   posXDf3[0] = -posXDf2[0] + posXDf2[1];
   posYDf3[0] = -posYDf2[0] + posYDf2[1];
   posZDf3[0] = -posZDf2[0] + posZDf2[1];
   xC = Mathf.Pow(posXDf3[0], 2);
   yC = Mathf.Pow(posYDf3[0], 2);
   zC = Mathf.Pow(posZDf3[0], 2);
   sum = xC + yC + zC;
   sumT = aux3 + sum;
   aux3 = sumT;
   posXDf[2] = posXDf[3];
   posYDf[2] = posYDf[3];
   posZDf[2] = posZDf[3];
   datoDf--;
   calculo1 = true;
          }
       datoDf++;
    }
}
```

7.1.7 Código del movimiento de la pinza desde el mecanismo.

```
        private SerialPort puerto = new SerialPort("COM3", 115200);
        void Start()
    {
        try //Comprobar si esta la torre conectada
          {
   puerto.Close();
   puerto.Open();
   //  posOffset = transform.position;
   towerOk = true;
          }
        catch { towerOk = false; }
    }
    void Update()
    {
        try //comprobar si la torre esta conectada
        {
   puerto.DiscardInBuffer();
   //Split división de cadenas en subcadenas
   datos = puerto.ReadLine().Split(spliters);
   //TryParse convierte la representación de la cadena de números para 32 bits
   //y regrea los valores indicados cuando la conversión haya sucedido
   for (int i = 0; i < datos.Length; i++)
   { // 5
        float.TryParse(datos[i], out entrada[i]);
            }
            pinzaDerecha.transform.eulerAngles = Vector3.down * (1) * (entrada[0]);   //Movimiento de las
mandibulas para abrir y cerrar
            pinzaIzquierda.transform.eulerAngles = Vector3.up * entrada[0];    //El objeto tiene que
referirce desde Unity
        if (entrada[0] == 0) //ver el estado de la pinza derecha
            {
   isClosedD1 = true;
   if (!cambio1)
               {
   cambio1 = true;
   cambioEstadoD1 = true;
   StartCoroutine(MovimientoPD());
               }
            }
        if (entrada[0] >0)
            {
   cambio1 = false;
   isClosedD1 = false;
            }
        pinzaSoporte.transform.eulerAngles = Vector3.forward * entrada[1];
        SphereDerecha.transform.eulerAngles = new Vector3(entrada[3], entrada[2], 0);
    pinza.transform.localPosition = new Vector3(0, entrada[4], 0);  //Entrar y salir de la
piza,el objeto se refiere desde Unity
        pinzaDerechaIZQ.transform.localEulerAngles = Vector3.down * (1) * (entrada[5]);
        pinzaIzquierdaIZQ.transform.localEulerAngles = Vector3.up * entrada[5];
        if (entrada[5] == 0)//ver el estado de la pinza izquierda
            {
   isClosedI1 = true;
```

```
   if (!cambio2)
            {
   cambio2 = true;
   cambioEstadoI1 = true;
   StartCoroutine(MovimientoPI());
            }
        }
        if (entrada[5] > 0)
        {
   cambio2 = false;
   isClosedI1 = false;
        }
        pinzaSoporteIZQ.transform.localEulerAngles = Vector3.forward * entrada[9];
   SphereIzquierda.transform.eulerAngles = new Vector3(entrada[8], entrada[6], 0);
   pinzaIZQ.transform.localPosition = new Vector3(0, entrada[7], 0);
   towerOk = true;
        }
        catch { towerOk = false; }
    }
    IEnumerator MovimientoPD()
    {
   yield return new WaitForSeconds(0.5f);
   cambioEstadoD1 = false;
    }
    IEnumerator MovimientoPI()
    {
   yield return new WaitForSeconds(0.2f);
   cambioEstadoI1 = false;
    }
}
```

7.1.8 Código de movimiento de la pinza desde el teclado y mando de videojuego.

```
void Update()
    {
        if (!Move2PScript.towerOk)
        {
   RotateDerecha();
   TranslateDerecha();
   giroDerecha();
   dientesDerecha();
   XBox();
   if (DosPinzas)
        {
   RotateIzquierda();
   TranslateIzquierda();
   giroIzquierda();
   dientesIzquierda();
        }
        if (open1 == 0)
        {
   isClosedD2 = true;
   if (!cambio1)
            {
   cambio1 = true;
   cambioEstadoD2 = true;
```

```
StartCoroutine(MovimientoPD());
            }
        }
    if (open1 > 1)
        {
cambio1 = false;
isClosedD2 = false;
        }
    if (open2 == 0)
        {
isClosedI2 = true;
if (!cambio2)
            {
cambio2 = true;
cambioEstadoI2 = true;
StartCoroutine(MovimientoPI());
            }
        }
    if (open2 > 1)
        {
cambio2 = false;
isClosedI2 = false;
        }
    }
}

void RotateDerecha()
{
    if (Input.GetKey(KeyCode.A) || RH1)
    {
    SphereDerecha.transform.Rotate(new Vector3(0f, -0f, -speed) * Time.deltaTime);
    }
    if (Input.GetKey(KeyCode.D) || RH2)
    {
    SphereDerecha.transform.Rotate(new Vector3(0f, 0f, speed) * Time.deltaTime);
    }
    if (Input.GetKey(KeyCode.S) || RV1)
    {
    SphereDerecha.transform.Rotate(new Vector3(speed, 0f, 0f) * Time.deltaTime);
    }
    if (Input.GetKey(KeyCode.W) || RV2)
    {
    SphereDerecha.transform.Rotate(new Vector3(-speed, 0f, 0f) * Time.deltaTime);
    }
}
void RotateIzquierda()
{
    if (Input.GetKey(KeyCode.J) || LH1)
    {
SphereIzquierda.transform.Rotate(new Vector3(0f, -0f, -speed) * Time.deltaTime);
    }
    if (Input.GetKey(KeyCode.L) || LH2)
    {
    SphereIzquierda.transform.Rotate(new Vector3(0f, 0f, speed) * Time.deltaTime);
    }
    if (Input.GetKey(KeyCode.K) || LV1)
    {
    SphereIzquierda.transform.Rotate(new Vector3(speed, 0f, 0f) * Time.deltaTime);
```

```
        }
        if (Input.GetKey(KeyCode.I) || LV2)
        {
        SphereIzquierda.transform.Rotate(new Vector3(-speed, 0f, 0f) * Time.deltaTime);
        }
    }
    void TranslateDerecha()
    {
        if (Input.GetKey(KeyCode.Z) || RT)
        {
        pinza.transform.Translate(new Vector3(0f, 0f, zoom));
        }
        if (Input.GetKey(KeyCode.X) || RB1)
        {
        pinza.transform.Translate(new Vector3(0f, 0f, -zoom));
        }
    }
    void TranslateIzquierda()
    {
        if (Input.GetKey(KeyCode.N) || LT)
        {
        pinzaIZQ.transform.Translate(new Vector3(0f, 0f, zoom));
        }
        if (Input.GetKey(KeyCode.M) || LB1)
        {
        pinzaIZQ.transform.Translate(new Vector3(0f, 0f, -zoom));
        }
    }

    void giroDerecha()
    {
        if(Input.GetKey(KeyCode.E) || R3)
        {
        pinzaSoporte.transform.Rotate(new Vector3(0f, 0f, rotacion) * Time.deltaTime);
        }
        if (Input.GetKey(KeyCode.Q))
        {
        pinzaSoporte.transform.Rotate(new Vector3(0f, 0f, -rotacion) * Time.deltaTime);
        }
    }

    void giroIzquierda()
    {
        if (Input.GetKey(KeyCode.O) || L3)
        {
    pinzaSoporteIZQ.transform.Rotate(new Vector3(0f, 0f, rotacion) * Time.deltaTime);
        }
        if (Input.GetKey(KeyCode.U))
        {
    pinzaSoporteIZQ.transform.Rotate(new Vector3(0f, 0f, -rotacion) * Time.deltaTime);
        }
    }

    void dientesDerecha()
    {
    pinzaDerecha.transform.localEulerAngles = Vector3.down *  (open1);   //Movimiento de las
mandibulas para abrir y cerrar
   pinzaIzquierda.transform.localEulerAngles = Vector3.up * open1;
   if (Input.GetKey(KeyCode.F) || StartBnt)
```

```
    {
open1++;
if (open1 >= 21)
        {
    open1 = 20;
        }
    }
    else
    {
open1--;
if (open1 <= 0)
        {
    open1 = 0;
        }
    }

 }
 void dientesIzquierda()
 {
    try
    {
pinzaDerechaIZQ.transform.localEulerAngles = Vector3.down * open2;
pinzaIzquierdaIZQ.transform.localEulerAngles = Vector3.up * open2;
    }
catch { }
if (Input.GetKey(KeyCode.P) || Select)
    {
open2++;
if (open2 >= 21)
        {
    open2 = 20;
        }
    }
    else
    {
open2--;
if (open2 <= 0)
        {
    open2 = 0;
        }
    }
 }
 void XBox()
 {
// Mapeado de botones
if (Input.GetButtonDown(“Button_A”))
    {
    print(“Has pulsado el boton A”);
    }
    if (Input.GetButtonDown(“Button_B”))
    {
    print(“Has pulsado el boton B”);
    }
    if (Input.GetButtonDown(“Button_X”))
    {
    print(“Has pulsado el boton X”);
    }
    if (Input.GetButtonDown(“Button_Y”))
```

```
    {
    print("Has pulsado el boton Y");
    }
if (Input.GetButton("Button_LB")) { LB1 = true; }
else { LB1 = false; }
if (Input.GetButton("Button_RB")) { RB1 = true; }
else { RB1 = false; }
if (Input.GetButton("Button_Select")) { Select = true; }
else { Select = false; }
if (Input.GetButton("Button_Start")) { StartBnt = true; }
else { StartBnt = false; }
if (Input.GetButton("Button_L3")) { L3 = true; }
else { L3 = false; }
if (Input.GetButton("Button_R3")) { R3 = true; }
else{ R3 = false;}
float analogico1 = Input.GetAxis("Left_Horizontal");
float analogico2 = Input.GetAxis("Left_Vertical");
float analogico3 = Input.GetAxis("TriggerLR");
float analogico4 = Input.GetAxis("Right_Horizontal");
float analogico5 = Input.GetAxis("Right_Vertical");
float digitalX = Input.GetAxis("R-Horizontal");
float digitalY = Input.GetAxis("R-Vertical");
if (analogico1 < 0) // J y L
    {
    LH1 = true; //J
    }
    else if (analogico1 > 0)
    {
    LH2 = true; //L
    }
    else if (analogico1 == 0)
    {
    LH1 = LH2 = false;
    }
    if (analogico2 < 0) // I y K
    {
    LV1 = true; //K
    }
    else if (analogico2 > 0)
    {
    LV2 = true; //I
    }
    else if (analogico2 == 0)
    {
    LV1 = LV2 = false;
    }
    if (analogico3 < 0)
    {
    RT = true;
    }
    else if (analogico3 > 0)
    {
    LT = true;
    }
    else if (analogico3 == 0)
    {
```

```
        RT = LT = false;
        }
        if (analogico4 < 0) // A y D
        {
        RH1 = true; //A
        }
        else if (analogico4 > 0)
        {
        RH2 = true; //D
        }
        else if (analogico4 == 0)
        {
        RH1 = RH2 = false;
        }
        if (analogico5 < 0) // W y S
        {
        RV1 = true; //S
        }
        else if (analogico5 > 0)
        {
        RV2 = true; //W
        }
        else if (analogico5 == 0)
        {
        RV1 = RV2 = false;
        }
        if (digitalX != 0)
        {
        print("Valor D-Pad X = " + digitalX);
        }
        if (digitalY != 0)
        {
        print("Valor D-Pad Y = " + digitalY);
        }
    }
    IEnumerator MovimientoPD()
    {
   yield return new WaitForSeconds(0.5f);
   cambioEstadoD2 = false;
    }
    IEnumerator MovimientoPI()
    {
   yield return new WaitForSeconds(0.2f);
   cambioEstadoI2 = false;
    }
}
```