E. TUNER

ATILIM UNIVERSITY 2019

DEVELOPMENT AND ASSESSMENT OF A SIMULATION SOFTWARE FOR ENDO-NEUROSURGERY NAVIGATION SKILLS

THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
ATILIM UNIVERSITY

EMRE TUNER

A MASTER OF SCIENCE THESIS
IN
THE DEPARTMENT OF SOFTWARE ENGINEERING

JUNE 2019

DEVELOPMENT AND ASSESSMENT OF A SIMULATION SOFTWARE FOR ENDO-NEUROSURGERY NAVIGATION SKILLS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
ATILIM UNIVERSITY

BY

EMRE TUNER

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
THE DEPARTMENT OF SOFTWARE ENGINEERING

JUNE 2019

Approval of the Graduate School of Natural and Applied Sciences, Atilim University.

_____

Prof. Dr. Ali Kara
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of **Master of Science in Software Engineering, Atilim University**.

_____

Prof. Dr. Ali Yazıcı
Head of Department

This is to certify that we have read the thesis DEVELOPMENT AND ASSESSMENT OF A SIMULATION SOFTWARE FOR ENDO-NEUROSURGERY NAVIGATION SKILLS submitted by EMRE TUNER and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

_____

Assoc. Prof. Dr. Nergiz E. Çağıltay
Supervisor

Examining Committee Members:

Assoc. Prof. Dr. Murat Koyuncu
Information Systems Eng. Department, Atilim Uni. _____

Assoc. Prof. Dr. Nergiz E. Çağıltay
Software Eng. Department, Atilim University _____

Assoc. Prof. Dr. Erol Özçelik
Department of Psychology, Çankaya University _____

Asst. Prof. Dr. Bilge Say
Software Eng. Department, Atilim University _____

Asst. Prof. Dr. Damla Topallı
Information Systems Eng. Department, Atilim Uni. _____

Date: June, 28 2019

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Emre, Tuner

ABSTRACT

DEVELOPMENT AND ASSESSMENT OF A SIMULATION SOFTWARE FOR ENDO-NEUROSURGERY NAVIGATION SKILLS

Tuner, Emre

MS., Department of Software Engineering

Supervisor: Assoc. Prof. Dr. Nergiz E. Çağıltay

June 2019, 59 pages

The process of positioning the camera is one of difficult and error-prone skills that need to be developed for endoscopic surgery procedures. In the literature, there are very limited studies conducted to better understand the influencing factors from the instructional design perspective of computer-based surgical simulations by considering the factors such as gender and human computer interface strategies for enhancing these skills of the trainees. Main objective of this thesis is to develop a computer-based simulation software according to the requirements of the endo-neurosurgery education programs in order to better understand the effect of navigation pointer and gender on surgical performance. 64 beginners for the endo-neurosurgery performed cleaning the endoscope tasks on two versions (navigation pointer integrated (NP+) and non-navigation pointer integrated (NP-)) of a computer-based surgical simulation scenario. The effect of NP+ designed to guide the participants is studied by considering the gender effect. NP+ to the endoscope by using a computer-based surgical simulation scenario significantly improved the participants' performance to locate the endoscope. In general, male participants performed better than the females in both NP+ and NP-. For females, task-duration values are significantly lower in the NP+ than that for the NP-. The results of this study show the details of the developed system. According to the evaluation procedure of the developed system, the results show that the training modules developed for endo-neurosurgical procedures, need to consider gender differences for the beginners. NP+ strategies potentially improve beginner females' skills on locating the endoscope and minimize the gender

differences. The effect of NP+ for higher skill leveled surgical residents needs to be further researched.

Keywords: Computer-based Simulation, Surgical Training, Positioning Endoscope, Navigation Pointer Feedback, Gender, Endo-neurosurgery.

ÖZ

ENDO-NÖROŞİRÜRJİ NAVİGASYON BECERİLERİ İÇİN BİR SİMÜLASYON YAZILIMININ GELİŞTİRİLMESİ VE DEĞERLENDİRİLMESİ

Tuner, Emre

Yüksek Lisans, Yazılım Mühendisliği Bölümü

Tez Yöneticisi: Doç. Dr. Nergiz E. Çağıltay

Haziran 2019, 59 sayfa

Kamerayı konumlandırma işlemi, endoskopik cerrahi prosedürleri için geliştirilmesi gereken zor ve hataya açık becerilerden biridir. Literatürde, kursiyerlerin bu becerilerini geliştirmek için cinsiyet ve insan bilgisayar arayüzü stratejileri gibi faktörleri göz önüne alarak, bilgisayar ve cerrahi simülasyonların öğretim tasarımı perspektifinden etkileyici faktörleri daha iyi anlamak için çok sınırlı sayıda çalışma yapılmıştır. Bu tezin asıl amacı, navigasyon işaretçisinin ve cinsiyetin cerrahi performans üzerindeki etkisini daha iyi anlamak için endo-beyin cerrahisi eğitim programlarının gerekliliklerine uygun bir bilgisayar tabanlı simülasyon yazılımı geliştirmektir. Endoneroşirürji için 64 yeni başlayan, endoskop görevlerini bilgisayar tabanlı bir cerrahi simülasyon senaryosunun iki versiyonunda (navigasyon işaretçisinin entegre (NP+) ve navigasyon işaretçisinin entegre (NP-)) temizleme işlemini gerçekleştirdi. Katılımcılara rehberlik etmek için tasarlanan NP+ 'nın etkisi cinsiyet etkisi göz önüne alınarak incelenmiştir. Bilgisayar tabanlı bir cerrahi simülasyon senaryosu kullanarak endoskopa NP+, katılımcıların endoskopun yerini belirleme performansını önemli ölçüde artırdı. Genelde erkek katılımcılar hem NP+ hem de NP- kadınlarda daha iyi performans gösterdiler. Kadınlar için görev süresi değerleri NP+ 'da NP- değerinden anlamlı derecede düşüktür. Bu çalışmanın sonuçları, geliştirilen sistemin ayrıntılarını göstermektedir. Geliştirilen sistemin değerlendirme prosedürüne göre, sonuçlar endo-beyin cerrahisi prosedürleri için geliştirilen eğitim

modüllerinin yeni başlayanlar için cinsiyet farklılıklarını göz önünde bulundurması gerektiğini göstermektedir. NP+ stratejileri, yeni başlayan kadınların endoskopun yerini belirleme becerilerini geliştirir ve cinsiyet farklılıklarını en aza indirir. Daha yüksek beceri seviyeli cerrahi sakinleri için NP+ 'nın etkisinin daha fazla araştırılması gerekir.

Anahtar Kelimeler: Bilgisayara-dayalı Benzetim, Cerrahi Eğitim, Endoskobun Pozisyonlandırılması, Navigasyon, Navigation İşaretçi Geribildirimi, Cinsiyet, Endo-nöroşirürji.

To Mustafa Kemal ATATÜRK

ACKNOWLEDGMENTS

I would like to express my sincere appreciation to my supervisor Assoc. Prof. Dr. Nergiz Ercil Çağıltay for her continuous support, guidance, motivation and insight throughout my thesis study.

I would like to express my appreciation to head of Software Engineering Department, Prof. Dr. Ali Yazıcı.

Additionally, I would like to thank to examination committee members; Assoc. Prof. Dr. Murat Koyuncu, Assoc. Prof. Dr. Erol Özçelik, Asst. Prof. Dr. Bilge Say and Asst. Prof. Dr. Damla Topallı, for their valuable time and contributions.

I would like to express my appreciation to my dearest parents Ayla Tuner and Yunus Tuner for their love and patience. Also, I would like to express my deepest appreciation to my girlfriend Pelin Tamcı for her endless support during this period. They have always believed in me and encouraged me all through my life.

This study is conducted for improving the scenario designs of the educational materials which are developed for endo-neurosurgery education project (ECE: Tubitak 1001, Project No: 112K287) purposes. The authors also would like to thank for this financial support of The Scientific and Technical Research Council of Turkey (TUBITAK) 1001 program for realizing this research.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

JSON	-	JavaScript Object Notation
MIS	-	Minimal Invasive Surgery
NP	-	Navigation Pointer
NP+	-	Navigation Pointer Integrated
NP-	-	Non- Navigation Pointer Integrated
SDK	-	Software Development Kit
M	-	Mean
SE	-	Standard Error

CHAPTER 1

INTRODUCTION

Endoscopic neurosurgery is relatively a new technique. In this surgery, the surgeon operates tool named as endoscope and the other operational tools by using his/her both hands. The endoscope is a tool having a very strong light source and a camera. The surgeon uses the endoscope to enter the part of the brain that need to be operated through the nose holes of the patient. The endoscope helps the surgeon to visualize the inside of the body part of the patient by reflecting the camera view to a monitor that is placed next to the surgeon. In other words, the surgeon performs the operation through the camera view (that is taken through endoscope) reflected to the monitor. Different than the classical surgical procedures that the surgeon directly views the operational area, endoscopic surgeries are performed by watching the camera view of the operational field through a monitor. During this process the surgeon need to clean the camera view by taking the endoscope out of the noise and after cleaning process need to reach the operational field again to continue the surgery.

These types of surgeries are also called as Minimal Invasive Surgery (MIS) as the patient body is not deformed or cut to enter the brain part that required to be operated. Accordingly, these types of operations have several benefits for the patients such as their recovery durations are shorter. However, in order to perform these operations surgeons are required to develop several skills. Positioning the camera is one of the difficult and error-prone skills that need to be developed for endoscopic surgery procedures. Studies are very limited to better understand the influencing factors considering human computer interface strategies to better enhance the trainees' camera positioning skills. This study aims to better understand the impact of navigation guidance on positioning the endoscope in a computer-based simulation environment. In order to reach this aim, in this study, first a simulation software is developed with

haptic user interface by considering and modelling endo-neurosurgery requirements. In this computer-based simulation, cleaning the endoscope task is simulated. In order to understand the effect of navigation support the computer-based simulation software is developed as two versions. One version is developed by providing navigation guidance through a navigation pointer (NP+). The second version of the software is developed without any navigation guidance and navigation pointer (NP-). Afterwards, the developed system is experimentally evaluated by 64 software-engineering students who can be considered as beginner level of surgical residents. This thesis first describes the technical details of the software development stages of the developed surgical simulation software. Afterwards, the result of the experimental study is given.

The thesis is organized to provide a background of the study in Chapter 2. Chapter 3 provides details about the software development process. Chapter 4 provides the research methodology to evaluate the navigation effect in the computer-based simulation environment on the surgical performance. Chapter 5 shows the results and finally, Chapter 6 gives the Discussions and Conclusion from this study.

CHAPTER 2

BACKGROUND OF THE STUDY

As this study is conducted for the endo-neurosurgery procedures, the background of the study first given about these types of surgical techniques. The second part of the background is given by considering the studies about the simulation technology to support the training processes of the endo-neurosurgeons.

2.1. Endo-neurosurgery

Endo-neurosurgery is a minimal invasive surgical (MIS) procedure which is conducted through an endoscope that has a camera and light source. The surgeons perform operations through the camera view in the monitor by controlling both the endoscope and the operational tools. Hence, during the operation they use their both hands. In the case of Endo-neurosurgery that is the scope of this study, the surgeon enters the damaged brain part through the nostrils by using endoscope and the operational tools. These types of operations are preferred as they cause minimal damage for the patients' body and their recovery periods are very short. However, these operations are requiring some extra skills such as navigating through the body parts, eye-hand coordination and depth perception. Hence the navigation skill of the surgeons is one of the important skills that need to be developed for the surgeons of MIS procedures.

2.2.Simulation-Based Training for the Endo-neurosurgery

In order to improve these required skills of the surgeons for the MIS procedures, today hands on training is being provided through simulation environments. Through these simulation environments, as surgeons can get try-and-error type of training, it is possible to practice in an unlimited environment. However, these simulation environments need to be developed by considering the tasks that need to be performed during these operations. For instance, during the endo-neurosurgery, the process of cleaning endoscope is one of the tasks that need to be performed occasionally and very carefully.

2.3.The problem of Cleaning Endoscope in Endo-Neurosurgery Procedures

Today, the advanced surgical techniques require several skills to be developed by the surgeons. For instance, in minimal invasive surgery, the depth perception and field of view of the surgeon is reduced and as the scale of such procedures is quite small, performing the operation with the instruments through the insertion point is a challenge [1]. Additionally, the process of positioning the camera is a difficult, error-prone, and a drain on the user's available resources and attention [2]. Condensation as well as debris on the camera lens is another annoying problem that influences the view and affects the surgeon's mood which is caused either by lens condensation or by contamination due to a dirty port [3]. During endoscopic surgery, in order to clean the camera view regularly, the surgeon must take the endoscope outside the operational field, clean it and reposition to the operational field. Hence to guarantee an optimal field of view, the surgeon has to often frequent reposition the endoscope [4]. This process and re-locating the endoscope to the operational field are challenging tasks for the surgeons. According to Clark et al. (2013), navigation control is the most significant skill for the endoscopic surgery where its training requires greater focus [5].

Studies also report that simulation-based laparoscopic surgery training has several benefits [6] and potentially reduce the learning curve, improve conceptual understanding of complex anatomy as well as enhance visuospatial skills [7] and task efficiency [8]. Supportively, it is reported that, when usability issues such as support and error prevention were reconsidered in more detail, computer-based simulators could be more efficient learning and training tool in the field of surgery [9]. By better analyzing surgeons' behaviors it is possible to gain insights to better guide trainees in surgical education programs [10]. Hence, some strategies can be developed to better train the surgeons in order to gain these skills in a convenient manner. For instance, studies show that providing spatial and positional cues significantly improves way-finding performance [11].

In the literature, research was conducted on evaluating some strategies to better understand their impact on the surgical performance. For instance, the feasibility of

hybrid view which overlays the positions of organs and objects with the path history of the instruments has been evaluated [12]. Lack of effective feedback systems also reported as having an added effect of increasing workload of senior surgeons leading to increased costs and decreased overall efficiency and the effect of real time feedback on movement proficiency shown to be effective in surgical learning [13]. Another study report that a navigation pointer (NP) integrated into a laparoscopic camera and projected onto a surgical display improves efficiency in guiding an instrument to randomly selected targets within a simulated laparoscopic field in a box training [14]. As the number of female surgeons is very limited in the earlier studies, gender differences mostly cannot be performed [15, 16]. However, it is known that there is no gender effect on the learning curve for a fundamental laparoscopic task [17] and there is no significant gender differences for correlations between psychometric scores and performance scores [18]. On the other hand, there are studies reporting that female medical students were less likely to successfully complete the tasks in the allotted time in the proficiency-based training on VR laparoscopy and endoscopy simulators [19].

Hence, there are not many studies conducted to better understand the influencing factors from instructional design and human computer interface perspective of design of surgical computer-based simulation training scenarios and the gender effect on these training programs. In this study, a computer-based surgical simulation scenario is developed to simulate an endo-neurosurgery task. In this scenario, the process of cleaning the endoscope is simulated and the effect of navigation pointer designed to guide the process of re-locating the endoscope through the instrument position is studied by considering the gender effect.

CHAPTER 3

SIMULATION SOFTWARE DEVELOPMENT PROCESS

In order to understand the navigation effect on surgical performance, a simulation software which has a user interface through haptic devices has been developed. In this chapter the details about the developed software has been described.

3.1 Haptic Device Integration Studies

Haptic devices are providing force-feedback feelings in the simulation environments. However, in order to develop a user interface through these devices they are required to be integrated to the Unity environment. This integration details are provided below.

3.1.1 Integration of the Haptic Devices to the Simulation Software

For the integration of haptic devices to the computer simulation software, the manufacturer published the OpenHaptics SDK and Geomagic Touch Driver written in C ++. After the driver and the SDK files are installed, the SDK becomes available. The SDK includes a comprehensive software development environment that includes many sample applications. The main components of haptic applications (See Appendix A) developed within the scope of the project have been developed by using similar solutions in this software environment.

A C ++ project called HapticWrapper was created for the management of Haptic devices. With this application, the haptic device has been transformed into a software interface suitable for the software environment we use in the project by acting as a wrapper on the methods within the SDK. During this transformation, force-force algorithms have been developed for haptic motions. In addition, a third software library called pthread was used to manage the threads that were opened during this development. The methods in HapticWrapper are given in Table 3.1.

Table 3. 1 The methods of the HapticWrapper

Method	Description
`Initialize`	It calls the initial functions of the haptic device's own library to get the device ready. Creates threads through the pthread library for the force to be applied.
`SetConstantForce`	The method used to get the force from the haptic device. The size of the force vector, which is to be given as a parameter, is given by its x, y and z values. The thread that is opened sends these parameters and continues to send the same force to the haptic devices through threads until the next call.
`GetData`	The button states of the Haptic device transform the transform and rotation values into a single object named DeviceData.
`StartScheduler`	Starts importing data from the haptic device.
`StopScheduler`	Stops receiving data from the haptic device.
`DisableDevice`	It is the opposite of the Initialize function. Turns off the status of the device. Release created threads.

The Figure 3.1 shows the Haptic Devices and Unity3D integration model. As seen from this figure, the integration has been conducted through the HapticDevice.cpp (See Appendix A) and by using the haptic controllers.

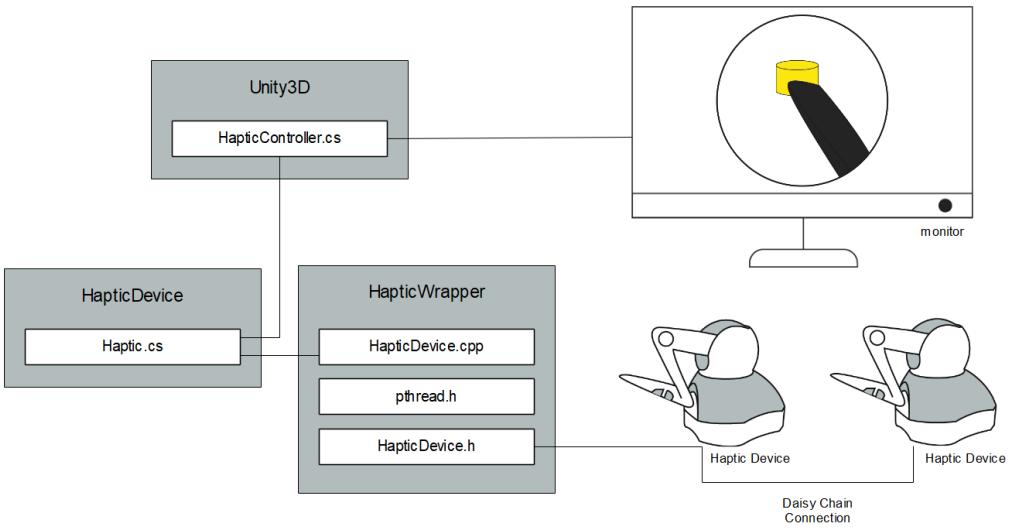

Figure 3. 1 Haptic Device and Unity3D Integration Model

3.1.2 The system Requirements of the Haptic Device

The minimum system requirements of the Haptic SDK are as follows;

- Pentium 4 and above processor
- 512 MB of disk space
- 1 GB RAM

The software has been developed on a computer with an Intel i7 3.50 Hz processor and 48 GB RAM. In terms of hardware, much more is needed than haptic devices need. There are two reasons for this. The first is to create enough resources for more than one haptic (for applications where more than one haptic device is used), and the second is for soft tissue calculations to be used in the simulation application to prevent the healthy operation of haptic devices. The computers' ram and processor are used very intensively to ensure that soft tissue calculations work in real time.

3.1.3 Configuration and Multiple Haptic Device Integration

There are two applications with haptic devices. The first is the setup application, where we connect the devices to which the devices are connected to the devices, and the other is the diagnostic application in which we can test the operability of the devices and perform motor calibrations. To introduce our devices to the computer, we must first select the model of the haptic device. The haptic devices used in this application are the Geomagic Touch model. After selecting the haptic device which port type is connected to the computer, we need to enable the device to be assigned to the selected port by pressing the calibration button on the back of the device.

Since two haptic devices are used in the study, the haptic devices should be connected with each other by the daisy chain method as shown in Figure 3.1. This method allows two devices with serial connection to reach the computer via a single port.

After the setup process is finished, the motor calibrations of the haptic devices are performed with the Diagnostic application. In Geomagic Touch models, motor calibration can be performed by inserting the haptic pen into the slot. Once the calibrations are complete, the haptic devices are ready for use.

3.1.4 Haptic Device and UNITY3D Communication

Haptic devices are managed by a C ++ application called HapticWrapper, which uses the HapticDevice.h library in the haptic SDK. Unity3d can be developed with a substructure compiled with C # code on .net 3.5 framework. To combine these two different platforms, it was necessary to build an architecture that recognizes the C ++ interface from .net C # projects. The methods included in the C ++ application, in which Haptic devices can be managed, have been made accessible to the C # project by a project called HapticDevice and accessed by the C # software language. Thus, haptic devices have become available in the code included in unity3d with this development. This interface software code is given in Appendix B.

Table 3. 2 The methods of the Haptic.cs

Method	Description
`GetHaptic`	The Haptic device has a parameter that requires the name. Returns the information of the device in an object called Haptic.
`GetData`	Calls the GetData function from Wrapper. Returns incoming data by inserting it into the DeviceData class inside the Haptic object.
`ApplyForce`	Calls the SetConstantForce function in Haptic Wrapper.
`DisableDevice`	Calls the DisableDevice function in Haptic Wrapper.
`StartScheduler`	Calls the StartScheduler function in the Haptic Wrapper.
`StopScheduler`	Calls the StopScheduler function in Haptic Wrapper.
`GetHapticConfigs`	Returns the names of the config files of the haptic located in the computer's Documents folder. The purpose of this process is to get the names of the computer-defined haptic devices. The device names returned from this function are then used when calling the GetHaptic function

To access the haptic application with a C # class created from Unity, both the C # and C ++ dlls and the .dll files of the pthread library that we use in the C ++ application should be moved to the plugins folder in Unity. The Plugins folder is an environment created for the package software to be imported from outside. After copying these

three .dll files into the Plugins folder, the class created for the haptic application can be accessed from the C # classes created in unity. The haptic integration on the stage is provided in Appendix C.

The Awake method in Unity is the first running method when the script file is loaded during runtime. Therefore, haptic operations on this method are important for the availability of haptic devices before the scene is created. On this method, a computer-defined list of haptic devices is first drawn into the computer and the objects of haptic devices are created.

The start method is the one-time running method before the update methods. Also, when this method is called, an instance is created from the class it is in. In this method, the StartSchedular method of the haptic devices is called up and the control of the haptic devices starts. Then a different thread is opened for the force information to be sent to the devices. The reason for sending force information through a different thread is to be independent of the main thread and thus a faster data flow. Because when the haptic devices receive 1000 haptic force vector information per second, they can provide a feeling of surface. The intermediate layer created on the Haptic SDK has interpolated this value and the minimum data flow in seconds, which should be sent via unity, has been reduced to 300. In addition, within the force for haptic, the transition between the force vectors is softened and the device gives a smoother surface feel.

3.1.5. The Calibration Process of the Haptic Device with the Simulation

The calibration was previously made for the engines of the device. This time the calibration process is performed to ensure that the position of the object and the haptic device in the game is the same. The haptic devices must be at the top of the grip position in the application. Because the stage passes on the inner surfaces of a horizontal cylindrical object and the beginning of the scene is at one end of the cylindrical object and facing the cylinder.

The calibration process was carried out in an easy to understand manner. It is provided by placing a movable circle controlled by the haptic into a fixed circle on the screen.

In this way, the positions in both the z direction and x, y directions are equalized. Rotation is ignored during the calibration process since there is no need for starting direction in the scene (See Figure 3.2).

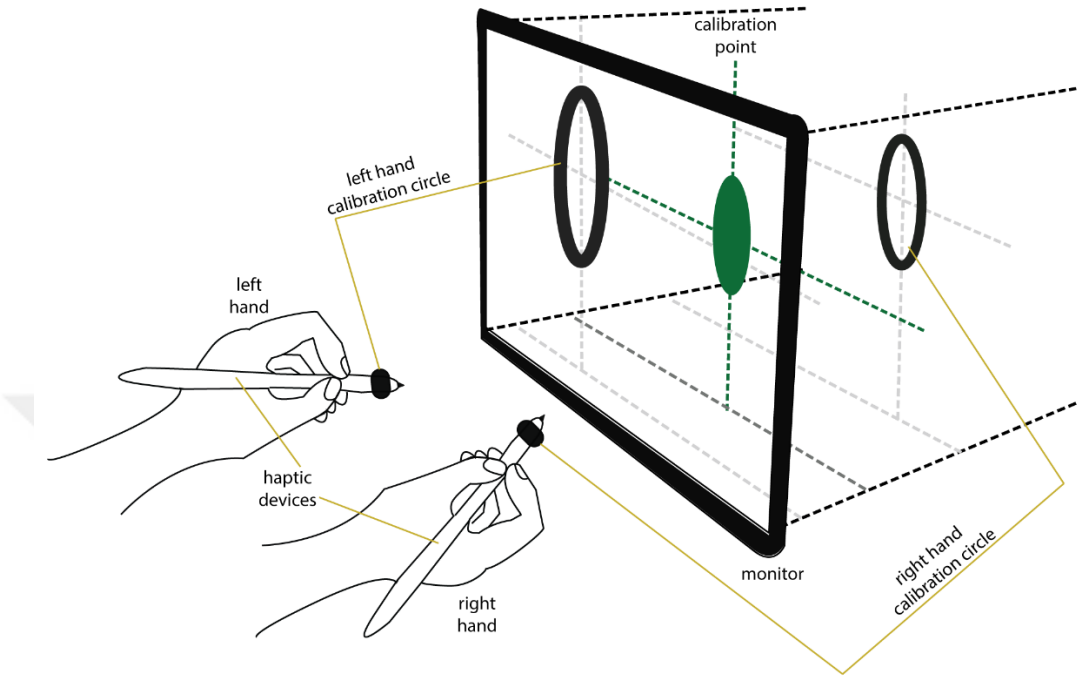

Figure 3. 2 Haptic Device Calibration Model

Since two haptic devices will be used to control both the camera and the tool, a separate calibration circle is shown for each device. The calibration application is integrated to the simulation environment as seen in Figure 3.3.

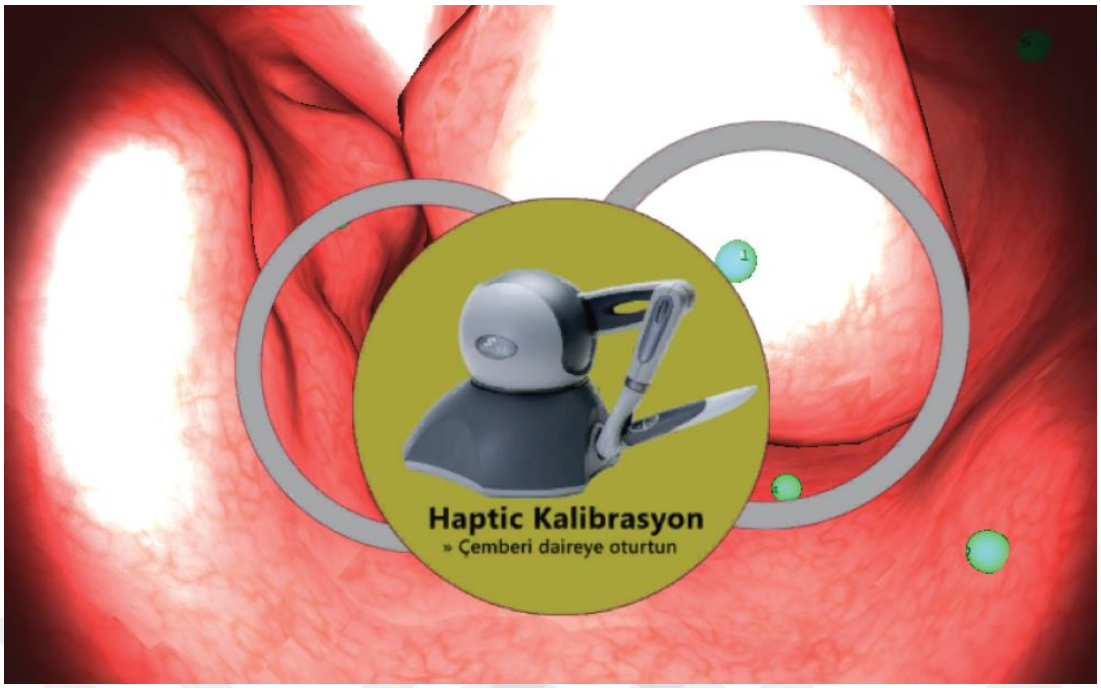

Figure 3. 3 Haptic Device Calibration in the Simulation

When the calibration process is finished, the scene is ready to play.

3.1.6 The Operation Tool and Endoscope Control with the Haptic Devices

Haptic devices are managed from the HapticController class as described previously. The assignment of the HapticController.cs file to the object to be checked is sufficient to be checked with the haptic. Two objects in the scene are controlled by the haptic: as the camera and the tool. Three different types of data are exchanged from haptic devices. The first is the Boolean value that indicates whether the button on the haptic devices is pressed, the second is the vectors that transmit the position and direction information of the haptic devices, and the last is the force vector sent to the haptic devices. Only the force vector information is sent to the devices. Other data comes from haptic devices. Position and direction information comes from two separate vectors. When assigning direction information from objects to objects, the position on the object must be considered. The position of the haptic control will again be at this point as the camera occupies a spot area on the stage. However, the haptic control on the tool must be at the point where it interacts at the very end of the tool. Because the haptic devices are located at the end of the haptic pen where the axis of motion that provides the directions in the x and y axes is located. In this way, the movements of

the haptic pen in the scene become common. As the position information is initially calibrated, the data from the haptic is transferred directly to the object in the scene. In order to equalize the motion values of objects in the scene with the haptic device, a multiplication called precision is multiplied by the number. This coefficient is defined as public and can also be changed from the editor screen. The sensitivity value for this scene was 2. The rotation and transformation for the tool is applied as seen in Figure 3.4.

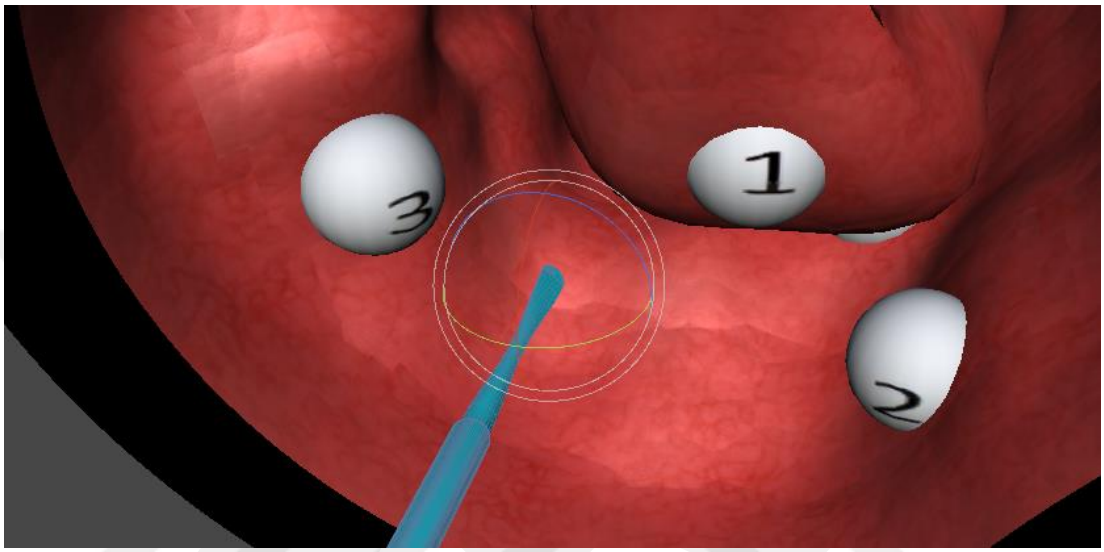

Figure 3. 4 The Rotation and Transformation of the Tool

3.2 The Software Architecture of the Application and Its Main Classes

The application was created in different software classes. The most important of these are the controller classes that control the objects in the game and manage the scenario, and the manager classes where the controllers are managed and the common variables. Apart from these two types of classes, there are auxiliary classes.

As seen from Table 3.3, the HapticController class captures data from the haptic device. The Haptic device transmits an average of 450-500 times per second on the computer we use. To be able to use this data exactly, a reading speed of the same frequency values is required.

Table 3. 3 The Main Classes of the Application

Class	Description
GameManager	Keeps user information and scenario-dependent features.
HapticManager	Manages objects derived from the HapticController class. The haptic devices defined on the computer are detected through this class. The stage works independently. It is not destroyed in transitions between scenes.
LoginManager	Sends the information on the login screen where user information is entered to the GameManager.
LevelManager	Manages the display of balls by level.
FileManager	It ensures that all activities in the scenario are recorded.
HapticContoller	Assigned to objects to be controlled by haptic in the game. All data from the Haptic device is captured through this class and projected onto the object in the scene.
ArrowController	Manages the position of the navigation items on the screen.
BallController	It is the class that manages the dynamics of the ball during clearing the balls in the scenario.
CleaningController	It is the base class that manages the scenario.
DirtController	It is the class that allows the screen to become dirty.
HandTracker	The class that records all interactions with the scene. Enables logging of both objects and data from the haptic device in the text file.
CatchTrigger	A ball must be brought to the correct area to be cleaned. This class understands that the ball is coming to this area.

Unity's "update" function for each frame would be slow for this. The Unity3d game engine runs the update function in sync with the number of frames per second (FPS). It is possible to turn off this conjugate from the settings in unity. However, even when turned off, the data read speed was around 200 per second. Therefore, each haptic device opens a thread to read the data and reads the data coming from the device. This achieved high reading speeds depending on the capacity of the computer.

3.3 The Scenario and Its Algorithm

The scenario is managed by the CleaningController class. This class is assigned to the tool on stage. There is no one-to-one communication with the tool. It is on the tool since there are only one on the stage (See Figure 3.5).

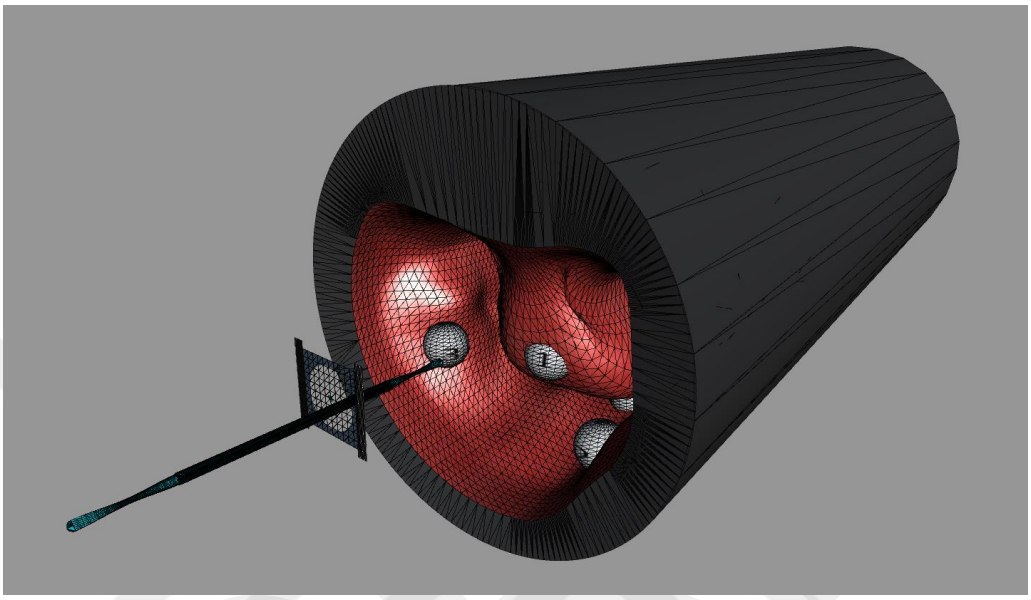

Figure 3. 5 The Stage Objects

As the scene starts, all balls that need to be cleared are collected in a list called ballList in the CleaningController class and sorted by their names (the name of each ball is the same as its own level). Then the function called “SetCurrentBall” is called when the calibration process is finished. This function allows the display of balls in the scene content according to the level. It also triggers audio and recording operations. The capture process of the balls is encoded within the HapticController class. To clean a ball, it must first be immersed with the tool and then removed from the surface by pressing the button. The feedback felt during the capture and removal of the balls is again coded within this class. The captured ball will sweep in the direction it was pulled. This creep is encoded in the BallController. The BallController class also handles the contact of the ball with the surface. As soon as the ball touches the surface, the OnCollisionEnter function is triggered and creeping is performed. When the OnCollisionExit function is triggered, contact with the surface is lost and can be

carried easily on the tool. Every ball that does not come into contact slowly returns to its former sphere form, which is extinguished. If the ball reaches the cleaning point without contact, the CatchTrigger enters the circuit and the corresponding level is completed. At the end of each level, the data related to that level is recorded. To do this, all data is first stored in a class. This class is then serialized and saved as text in json format.

3.4 The Sence of Force-feedback Application for Soft Tissue Study

There are two different force feedback applications in the scenario. The first one is the force feedback felt when cleaning the ball and the other one is felt when the boundary of the cylindrical object resembles the nose. The cleaning of the ball first begins with the tool dipping into the ball. During this process a force from the surface of the ball is applied. Since the balls are spherical, the force is applied in direct proportion to the square of the distance it enters from the surface. Thus, the force increases as the penetration increases. After certain strength, this force value is also set too much lower values. The purpose of this process is to create a sense of piercing the wall on the outer surface of the ball. Because, as you know, the strength applied until an object is drilled decreases after drilling. Another force feedback application of the ball is that it is felt when the ball is removed from the surface. At this stage the user should pull the ball gently from the surface by pressing the button on the haptic device. When pulling, the force of the tool is applied proportionally to the distance of the ball from the center until the tip of the tool comes off the surface of the ball. The other force feedback application results from the walls of the nose-like object. This force feedback is developed to prevent the user from taking the tool off the stage and losing control of the tool and is based on the same principle as the ball. A force proportional to the square of the distance at which the tool extends beyond the tool surface wall passing the nasal walls is applied. These force feedback applications are integrated with very simple methods and meet all the needs of the scenario. The main reason for the simplicity of the application is that the surfaces requiring force are spherical. However, in my more extensive studies with haptic, it is ensured that any desired surface is felt with haptic force feedback. A method is developed which is called the sewing method, which is very different from this application on more complex surfaces. With this method, mesh surfaces of objects can be felt just like in real life. However, this

application requires a very intensive processing load. The sewing method is not added to this study because this load on the computer affected some of the dynamics of our experience.

Also, throughout this work with haptic devices, tissue deformation very close to reality is applied. By combining this deformation with the seam management, the surfaces were made deformable to be felt thanks to the combination of these studies. However, deformation without surface sensation results in poor results during use, because of the force feedback limiting the user. Without feeling the limits, an uncontrolled surface deformation can be observed causing tears in the mesh in some places. Tissue deformation is therefore included only in the application of ball deflation.

Obtaining force feedback through Unity3D has reached the desired point after many improvements. Because for a smooth surface, a force vector of more than 500 per second must be sent to the haptic device. At values below 500, the surface sensation disappears completely, and an uncontrolled vibration is felt. It is impossible to transfer more than 500 data per second without various adjustments via Unity3D. A thread is created to send the same data as the previously opened thread, and it is sent to the haptic devices with an infinite loop in this thread. However, the communication frequency does not exceed 500 at this point. In HapticDevice.cpp, this data can be interpolated (calculated between two values) to send data at a higher frequency.

3.5 Extending the Ball by Pulling

Each ball is managed by a class called BallController. All vertexes of the ball are kept in this class. When the ball is dragged to the surface in one direction by pressing the button after the tool is immersed, all these vertexes are directed in the direction of movement of the tool. Extinction is achieved by the distance of the vertex to the point of impact. The distance of the vertex to the point of action (tool tip) is inversely proportional. As the distance increases, a coefficient obtained becomes smaller and multiplied by the delta motion of this tool to obtain the new motion vector of the corresponding vertex. When the creep process is finished (when the ball is detached from the face or the tool comes out of the ball), the vertexes of the ball are collected into the first form.

3.6 Visualization Studies

For improving the realism and visualization of the application, some camera applications blood effect on the screen has been implemented. These studies are developed as described below in this section.

3.7 Endoscope view of the Camera

Field of View value of the endoscope devices is 60 degrees. For this reason, the Field of View value of the camera is set to 60 degrees as seen in Figure 3.6. In order to provide a circular view of endoscope devices, a plane with a circular cavity in the middle is placed in front of the camera.

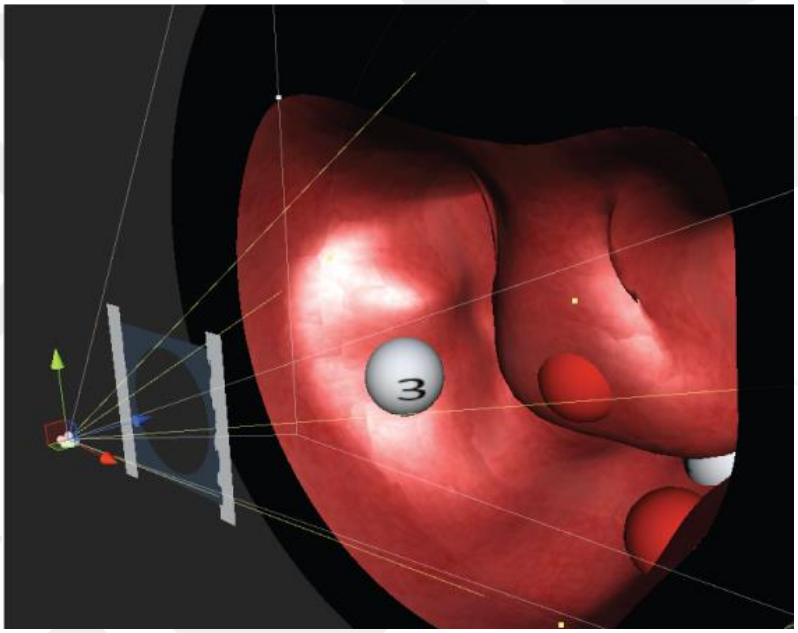

Figure 3. 6 Mask for the Endoscope View

3.8 Blood Contamination of the Screen

DirtController class performs the pollution of the screen. This class includes a sample stain texture. When this tissue contamination process is required, 45 different positions, directions and sizes are displayed on the screen. These dirt textures, which are displayed on the screen, are slowly shifted downwards to improve the authenticity of the screen and the visual appearance of the dirt is completed.

3.9 The Sound Effects

Sound application was used at three different points in the scenario as summarized below:

- A ringtone is played to represent the successful cleaning of the ball, by bringing the ball to the desired point properly.
- A sound is heard when clicking the button on the haptic device to clean the screen.
- When all balls are cleared, a ringtone is played to indicate that the game is over.

3.10 Data Storage

Three different data are kept for each user. These are hand movement data, force data when cleaning the ball, and general performance data. Hand data is held by the Hand Tracker class. This class periodically sends the positions and directions of the tool and the camera to the AddRowToRecord function of the FileManager class. The save interval is held in the class by a variable named HandTrackInterval and can be assigned a value between 0 and 1. The default value of this variable is 0.1 seconds.

The data of the force applied during the cleaning of the ball starts from the moment the key is pressed to remove the tool from the surface. This data is sent from the HapticController class to the AddRowToRecord function of the FileManager class.
The overall performance data is kept at the beginning of each ball level. In it there are ball index, level time, total time, performance, number of touches, total tool's distance and total camera's distance.

Any data that comes to the FileManager class is stored in the Dictionary. At the end of the application, the SaveFile function in the FileManager class is called and three different types of data are saved in separate text files in json format.
See sample data in Appendixes D, E and F.

CHAPTER 4

RESEARCH METHODOLOGY

In this thesis, an experimental study has been conducted to better understand the navigation effect while performing tasks in a computer-based simulation environment on the participants' performance. The details about the experimental setting are as described below.

4.1 Materials and Method

An experimental study is conducted to better understand the effect of a navigation pointer integrated (NP+) to endoscope by using a computer-based surgical simulation scenario. Haptic devices were used to simulate the operational tool (Figure 4.1) and the endoscope (Figure 4.2).

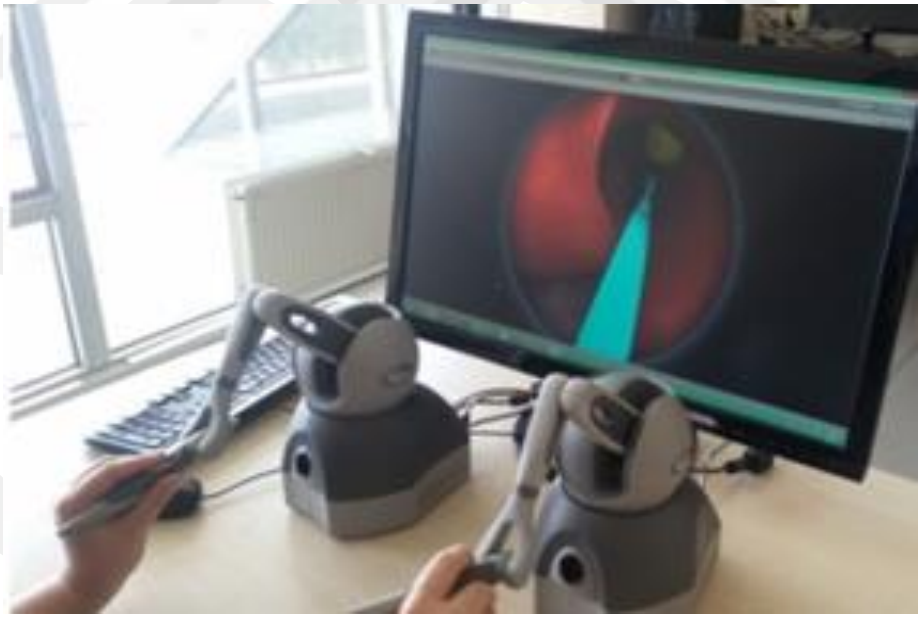

Figure 4. 1 Haptic Device Simulation of Operating Tool

Participants have controlled the endoscope with their non-dominant hand and the operational tool with the dominant one. The haptic devices are in a fixed position that all participants used the system in a similar way.

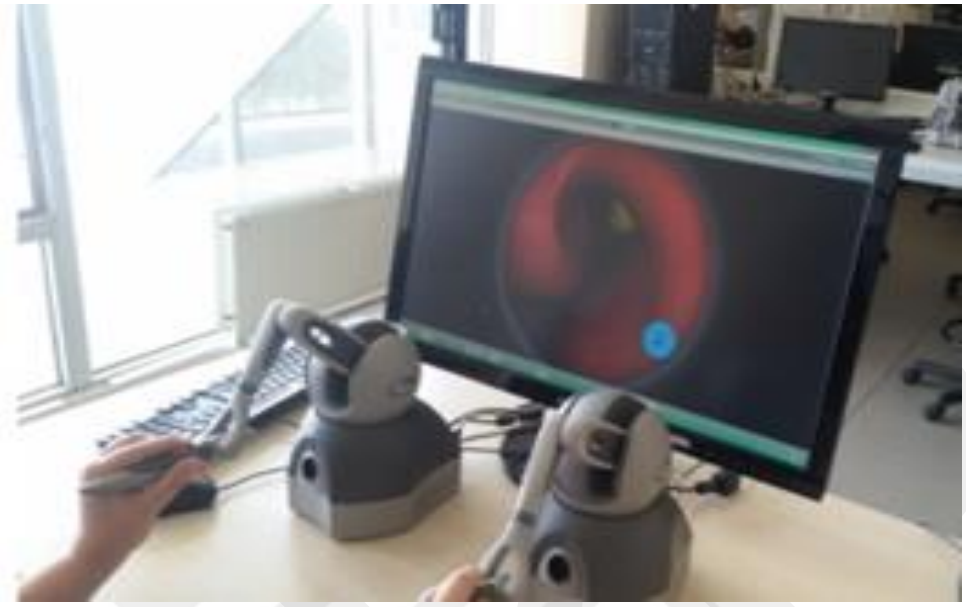

Figure 4. 2 Haptic Device Simulation of Endoscope

In this scenario, each participant asked to remove the harmful substances which are spread through the nose model shown as numbered yellow balls on the scene (Figure 4.3 and 4.4) by using a surgical tool and an endoscope.

Figure 4. 3 Yellow Balls in the Simulation

There were 10 tasks that need to be performed by both-hand where the tool is controlled by the dominant hand and the endoscope by non-dominant hand.

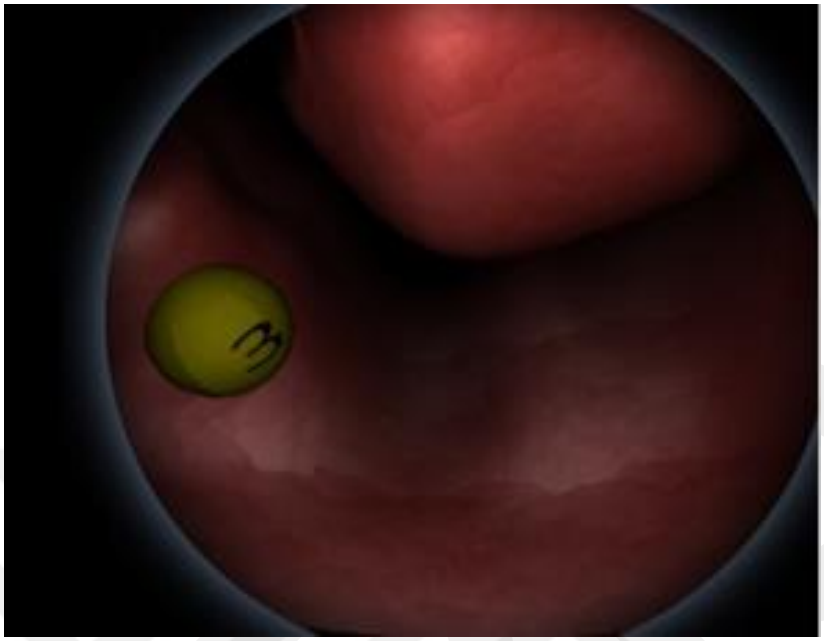

Figure 4. 4 Yellow Balls in the Simulation to be cleaned

In order to perform this task, each participant should first stick into the yellow balls with the surgical tool as seen in Figure 4.5. After sticking into the ball, the ball became green (representing that the ball is caught).

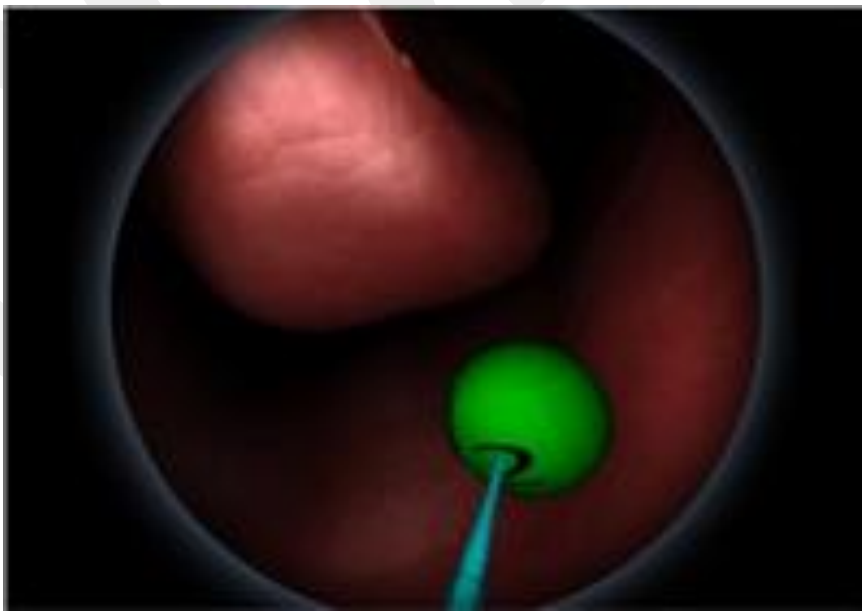

Figure 4. 5 Yellow Balls in the Simulation Cached By Operation tool

Additionally, through the haptic device's force-feedback feature, the participant feels the ball. Afterwards, as seen in Figure 4.6, there occurs a bleeding effect for which the endoscope should be taken out of the noise model and cleaned.

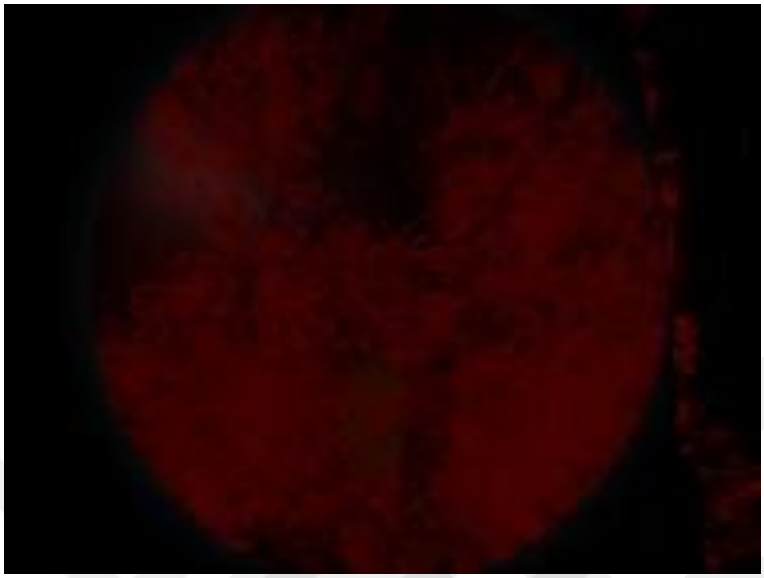

Figure 4. 6 Bleeding Effect in the Simulation

Finally, as seen from Figure 4.7, the green ball hold by the tool should be removed by taking out of the nose model slowly and carefully without touching to the environment.

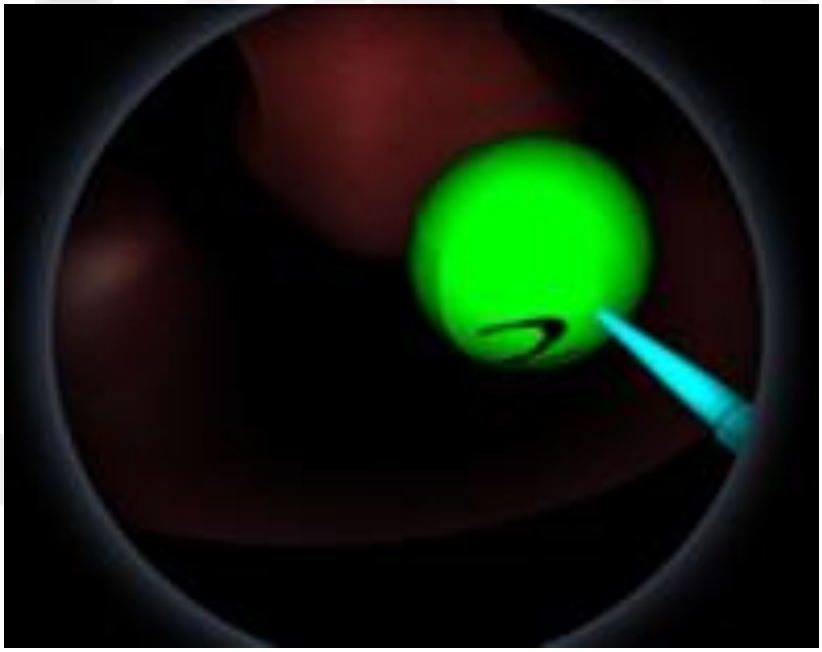

Figure 4. 7 The Green Ball Hold by the Tool in the Simulation

If the ball touches to the environment or fells down, the ‘stick into’ process occurs. In that case, as explained above the endoscope need to be cleaned again by taking it out of the model and should be relocated it and going to the same location where the operational tool and the green ball is located, to take the ball out. At the beginning of the experimental study, an instruction video was prepared for guiding the participants about the simulation. All participants were given 3 minutes to complete each task. The following seven measures are collected for each task in this scenario.

- Task-Duration: the duration to perform each task,
- Total-Tool-Distance: the distance taken with the haptic device that simulates the operation tool and controlled by the dominant hand,
- Total-Camera-Distance: the distance taken with the haptic device that simulates the endoscope and controlled by the non-dominant hand,
- Accuracy: representing if the task is performed successfully in a given time period or not. For the successful task completion, it is set as one, and the value is set as zero for the unsuccessful tasks,
- Touch-number-of-hits: the number of hits to the environment while performing each task,
- Stick-Count: the total number of attempts in order to catch each ball.

The scenario is developed in two versions. The first version of the scenario is designed as the navigation pointer integrated (NP+) to the endoscope. The NP+ version of the scenario provided visual information for the location of the tool regarding to the endoscope. As seen from Figure 4.8 and Figure 4.9, the ‘blue pointer’ set out to navigate the location of the tool.

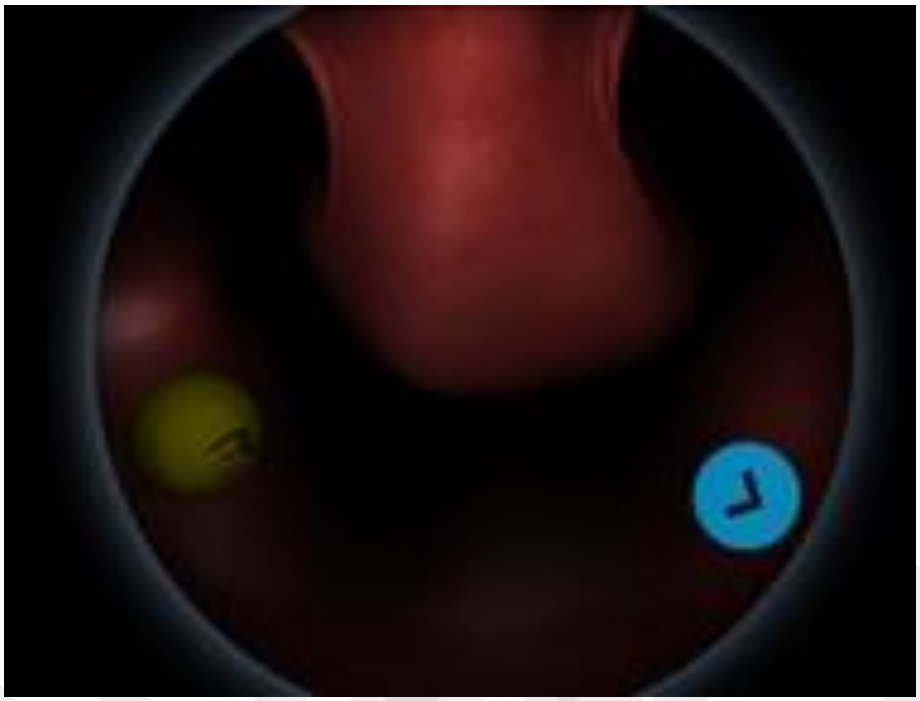

Figure 4. 8 The 'blue pointer' in the NP+ Version of the Simulation

For instance, in Figure 4.8 it is understood from the blue pointer that the tool is located in front of the endoscope. Similarly, as seen from Figure 4.9, the 'red pointer' is implemented to navigate when the tool fell behind the camera.

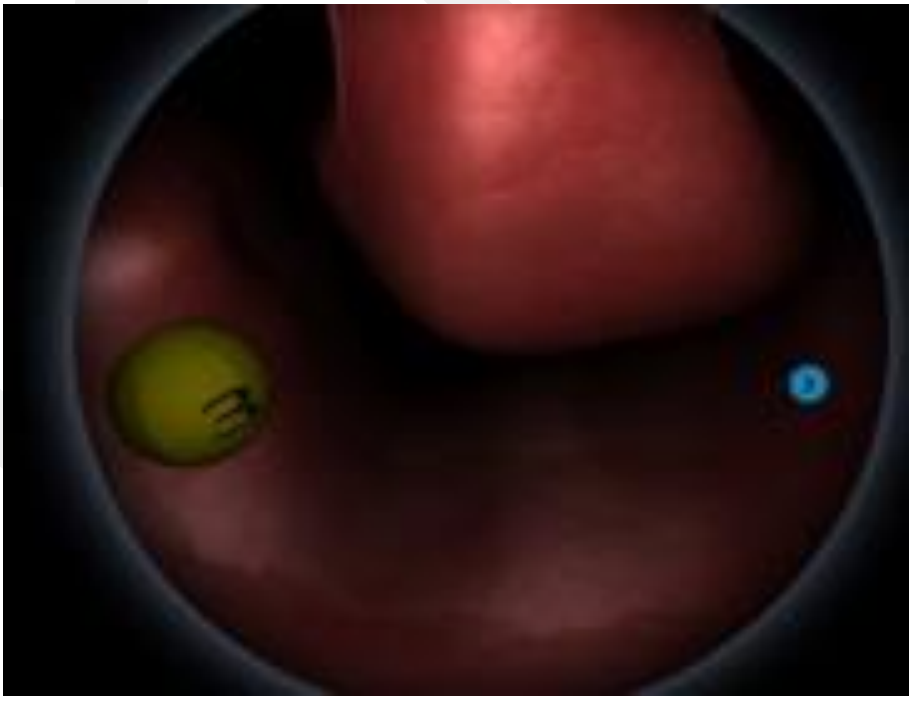

Figure 4. 9 The 'blue pointer' Guidance in the NP+ Version of the Simulation

In that case, the participant should move the tool forward to make the tool visible in the screen. The pointer becomes bigger when the tool moving closer to the nose model. In the second version of the training scenario, as seen from Figure 4.4 and Figure 4.5, there were no navigation pointer integrated (NP-) to the endoscope. Other than this all functionality of the system works in the same way as in NP+ version of the scenario.

4.2 Participants

The experimental study is performed by a total of 64 students from Engineering Faculty, who are voluntarily participated. Most of the participants were studied in Computer, Software and Information Systems engineering departments (53 out of 64 participants, 82.81%). The detailed information about the participants is given in Table 4.1. The participants of this study merely do not have any specific knowledge of the domain and according to earlier definition [19] they can be considered as 'beginner' in the domain. There were 34 male and 30 female students participated in this study, with average age 24.5 and 22.3, respectively.

Table 4. 1 Participants of the Study

	Gender		
Department	Female	Male	**Total**
Computer Engineering	11	10	21
Software Engineering	10	10	20
Information Sys. Engineering	6	6	12
Civil Engineering	0	5	5
Industrial Engineering	2	0	2
Metallurgical and Materials Eng.	1	1	2
Mechanical Engineering	0	2	2
Total	30	34	64

During experimental study, all participants were randomly divided into two groups (NP+ and NP-) according to the order of their attendance. The distribution of

participants among gender and navigation pointer guidance groups is given in Table 4.2.

Table 4. 2 Distribution of Participants among Groups

	Gender		
NP Guidance	**Female**	**Male**	**Total**
NP+	13	19	32
NP-	17	15	32
Total	30	34	64

4.3 Data Analysis

Data analysis was performed by using IBM SPSS Tool. Based on the collected data from the experimental study, a between-subject design is implemented. A two-way analysis of variance is performed to understand the effect of the two independent variables –gender and the navigation guidance- and their interaction on each measure: task duration, total distance covered by the tool and the camera, accuracy, number of hits (touches) and the stick count.

CHAPTER 5

RESULTS

To better understand the effect of gender and the navigation pointer integrated (NP+) to the endoscope by using a computer-based surgical simulation scenario, all measures in this study were subjected to a two-way analysis of variance having two levels of gender (male, female) and two levels of navigation pointer (NP+ and NP-).

A two-way ANOVA was conducted that examined the effect of gender and navigation pointer on 'task duration' measure. All effects were statistically significant at the .05 significance level. As also can be seen from Figure 5.1, there was a significant main effect of gender on task-duration, $F(1,60) = 12.43$, $p < .001$, such that the time taken to perform each task is significantly lower for male (M=77.00, SE=5.23) than for female (M=103.97, SE=5.58). There was a significant main effect of using navigation pointer (NP+) on task-duration, $F(1,60) = 6.33$, $p=.015$, such that the time taken to perform each tasks is significantly lower for the NP+ group (M=80.86, SE=5.45) than for NP- group (M=100.11, SE=5.37).

Additionally, there was a significant interaction between using navigation pointer while performing tasks and the gender on task-duration, $F(1,60) = 5.95$, $p=.018$, indicating that the effect of navigation pointer was different for male than it was for female, shown in Figure 5.1. For male, time taken to perform each task is similar in both NP+ group (M=76.70, SE=6.95) and the NP- group (M=77.30, SE=7.82). However, as also can be seen from Figure 5.2, for female, the time taken to complete all tasks is significantly lower for the NP+ group (M=85.01, SE=8.40) than for NP- group (M=122.92, SE=7.35).

Table 5. 1 Task Duration Significance Values over Gender and NP

	F	Significance
Gender	12.43	0.001
NP	6.34	0.015
Gender*NP	5.95	0.018

Table 5. 2 Task Duration Measurements According to Gender and NP

		Mean	Std. Error
Gender	Male	77.00	5.23
	Female	103.97	5.58
NP	-	100.11	5.37
	+	80.86	5.45

Table 5. 3 Task Duration Measurements According to Gender*NP

Gender	NP	Mean	Std. Deviation
Male	-	77.30	29.39
Male	+	76.70	24.76
Female	-	122.92	31.79
Female	+	85.01	36.27

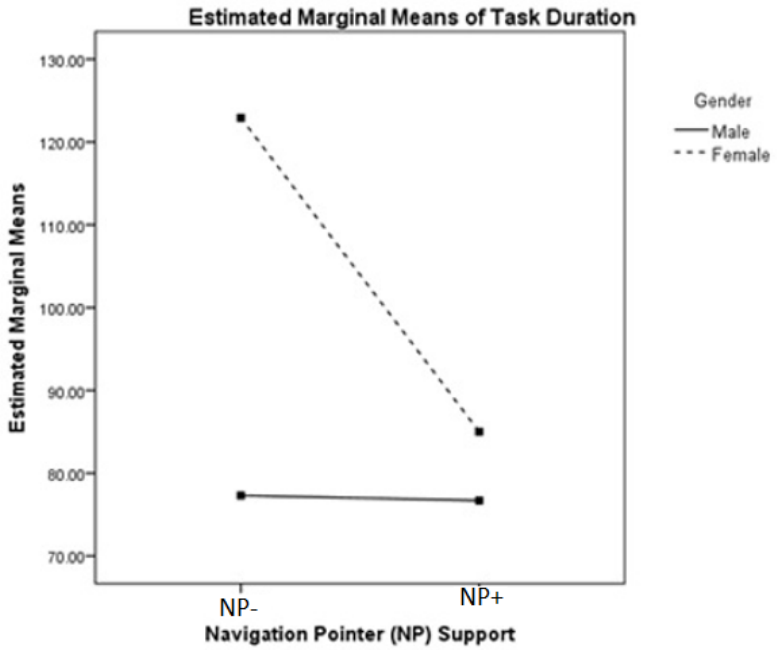

Figure 5.1 Task Duration and Gender in the NP+ and NP-

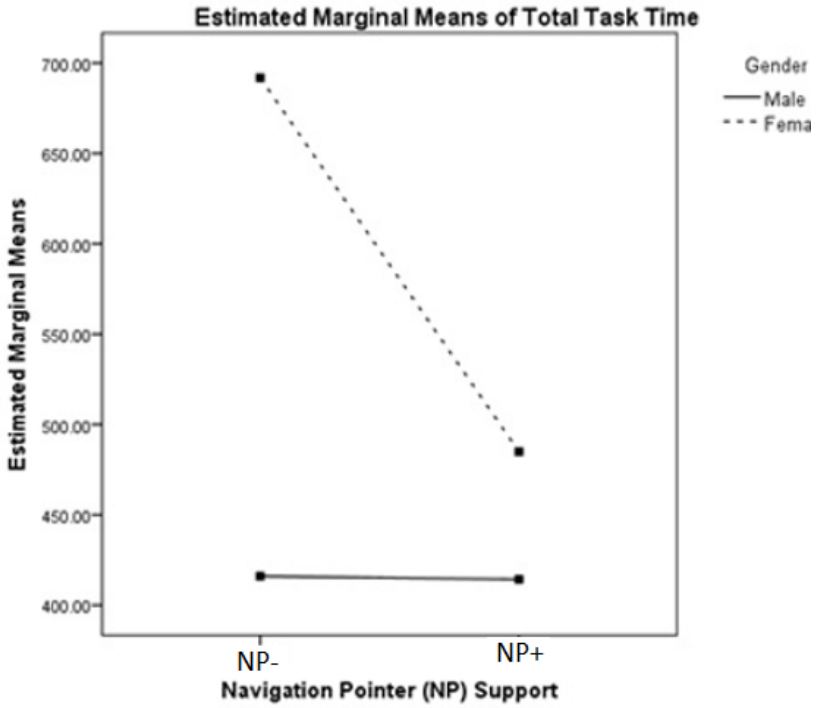

Figure 5. 2 Total Task Time and Gender in the NP+ and NP-

A two-way ANOVA was conducted that examined the effect of gender and navigation pointer on 'total tool distance' measure. There was a significant main effect of gender

on distance covered by the tool, F(1,60) = 6.45, p=.014, such that the covered distance is significantly lower for male (M=1984.60, SE=31.22) than for female (M=3469.51, SE=33.31). There was a significant main effect of using navigation pointer (NP+) on covered total-tool-distance, F(1,60) = 8.50, p=.005, such that the covered distance by the tool is significantly lower for the NP+ group (M=1874.75, SE=416.69) than for NP- group (M=3579.37, SE=410.11) (see Tables 5.4 and 5.5). However, the interaction effect was non-significant, F(1,60) = 3.63, p= .062.

Table 5. 4 Tool Distance Significance Value over Gender and NP

	F	Significance
Gender	6.45	0.014
NP	8.50	0.005
Gender*NP	3.63	0.062

Table 5. 5 Tool Distance Measurements According to Gender and NP

		Mean	Std. Error
Gender	Male	1984.60	399.86
	Female	3469.51	426.54
NP	-	3579.37	410.11
	+	1874.75	416.69

A two-way ANOVA was conducted that examined the effect of gender and navigation pointer on 'total camera distance' measure. There was a significant main effect of gender on distance covered by the camera, F(1,60) = 6.05, p=.017, such that the total-camera-distance is significantly lower for male (M=1805.51, SE=210.63) than for female (M=2562.72, SE=224.68). There was a significant main effect of using navigation pointer (NP+) on covered distance by the camera, F(1,60) = 4.24, p=.044,

such that the covered distance by the camera is significantly lower for the NP+ group (M=1866.92, SE=219.50) than for NP- group (M=2501.31, SE=216.03). However, the interaction effect was non-significant, F(1,60) = 3.74, p= .058 (see Tables 5.6 and 5.7).

Table 5. 6 Camera Distance Significance Value over Gender and NP

	F	Significance
Gender	6.05	0.017
NP	4.24	0.044
Gender*NP	3.74	0.058

Table 5. 7 Camera Distance Measurements According to Gender and NP

		Mean	Std. Error
Gender	Male	1805.51	210.63
	Female	2562.71	224.68
NP	-	2501.31	216.03
	+	1866.92	219.50

A two-way ANOVA yielded a main effect of participants' gender on task accuracy measure, F(1,60) = 9.99, p=.002, such that the average accuracy was significantly higher for male (M=.91, SE=.03) than for female (M=.77, SE=.03). The main effect of using navigation pointer (NP+) on accuracy was non-significant, F(1,60) = 3.78, p=.056. Even the accuracy is higher for NP+ group (M=.89, SE=.03) than for NP- group (M=.80, SE=.03) on average, the interaction effect was also non-significant, F(1,60) = 1.83, p= .182 (see Tables 5.8 and 5.9).

Table 5. 8 Accuracy over Gender and NP

	F	Significance
Gender	9.99	0.002
NP	3.78	0.056
Gender*NP	1.83	0.182

Table 5. 9 Accuracy According to Gender and NP

		Mean	Std. Error
Gender	Male	0.91	0.03
	Female	0.77	0.03
NP	-	0.80	0.03
	+	0.89	0.03

A two-way ANOVA was conducted that examined the effect of gender and navigation pointer on 'touch-number-of-hits' measure. All effects were non-significant at the .05 significance level. The main effect of gender $F(1,60) = 1.28$, $p = .262$ and the main effect of using navigation pointer $F(1,60) = .36$, $p = .55$ were non-significant. The interaction effect was also non-significant, $F(1,60) = 1.52$, $p = .223$. Similarly, a two-way ANOVA was conducted that examined the effect of gender and navigation pointer on the 'stick-count' measure. All effects were also non-significant at the .05 significance level. The main effect of gender $F(1,60) = .969$, $p = .329$, the main effect of using navigation pointer, $F(1,60) = 1.54$, $p = .22$ and the interaction effect $F(1,60) = .02$, $p = .887$ were all non-significant.

CHAPTER 6

DISCUSSIONS AND CONCLUSION

To better understand the effect of gender and the navigation pointer integrated (NP+) to the endoscope by using a computer-based surgical simulation scenario six measures (task-duration, total-tool-distance, total-camera-distance, accuracy, touch-number-of-hits and stick-count) have been collected automatically by the simulation system. Data is collected from 64 participants and analyzed by two-way analysis of variance having two levels of gender (male, female) and two levels of navigation pointer (NP+ and NP-).

Results of this study first show that there are significant differences between NP+ and NP- groups. For instance, task-duration, total-tool-distance, total-covered-distance values are significantly lower for the NP+ group than for NP- group. This study is conducted with the participants who merely have non-specific knowledge of a domain and they can be considered as 'beginner' in the domain [20]. Based on results of this study, it can be concluded that, in general, the navigation support navigation pointer integrated (NP+) to the endoscope by using a computer-based surgical simulation scenario improves the participants' performance to locate the endoscope. This result is supportive one to the earlier studies [14]. Hence, as a conclusion, it can be stated that, NP+ strategies can be used for the beginner surgeons to improve their skills on camera location for the endo-neurosurgery operations.

Additionally, according to the results, significant differences between male and female, while they were performing computer simulated surgical tasks has been found. The results show that, task-duration, total-tool-distance and total-covered-distance values are significantly lower for male than that of for female. Furthermore, the average accuracy was significantly higher for male than that of for female. Similarly, for male, task-duration values were similar in both NP+ and that for the NP- groups. However, for female, task-duration values are significantly lower for the NP+ group than that for the NP- group. These results are supportive to the earlier ones [19]. Based

on these results, it can be concluded that, the feedback from the navigation control potentially improves the participants' performance to locate the endoscope. Additionally, gender differences also need to be considered for developing instructional materials for the endo-neurosurgery procedures.

This study was conducted with 64 participants, who are engineering students and can be considered as a beginner in the domain [20]. On the other hand, these surgical skills need to be also tested for the other skill leveled surgeons such as intermediate, novice, sub-expert and experts. These results may differ depending on these experience levels.

REFERENCES

[1] D. Burschka, J. J. Corso, M. Dewan, W. Lau, M. Li, H. Lin, *et al.*, "Navigating inner space: 3-d assistance for minimally invasive surgery," *Robotics and Autonomous Systems*, vol. 52, pp. 5-26, 2005.

[2] A. Pandya, L. Reisner, B. King, N. Lucas, A. Composto, M. Klein, *et al.*, "A review of camera viewpoint automation in robotic and laparoscopic surgery," *Robotics*, vol. 3, pp. 310-329, 2014.

[3] H. Theeuwes, H. Zengerink, and G. Mannaerts, "Easy cleaning of the camera port during laparoscopic surgery: three practical techniques," *Journal of Laparoendoscopic & Advanced Surgical Techniques*, vol. 21, pp. 821-822, 2011.

[4] C. Staub, S. Can, B. Jensen, A. Knoll, and S. Kohlbecher, "Human-computer interfaces for interaction with surgical tools in robotic surgery," in *2012 4th IEEE RAS & EMBS International Conference on Biomedical Robotics and Biomechatronics (BioRob)*, 2012, pp. 81-86.

[5] J. Clark, F. Orihuela-Espina, M. Sodergren, D. R. James, V. Karimyan, J. Teare, *et al.*, "A quantitative scale to define endoscopic torque control during natural orifice surgery," *Minimally Invasive Therapy & Allied Technologies*, vol. 22, pp. 17-25, 2013.

[6] B. Zendejas, R. Brydges, S. J. Hamstra, and D. A. Cook, "State of the evidence on simulation-based training for laparoscopic surgery: a systematic review," *Annals of surgery*, vol. 257, pp. 586-593, 2013.

[7] A. Bernardo, "Virtual reality and simulation in neurosurgical training," *World neurosurgery*, vol. 106, pp. 1015-1029, 2017.

[8] N. Gélinas-Phaneuf, N. Choudhury, A. R. Al-Habib, A. Cabral, E. Nadeau, V. Mora, *et al.*, "Assessing performance in brain tumor resection using a novel virtual reality simulator," *International journal of computer assisted radiology and surgery*, vol. 9, pp. 1-9, 2014.

[9] M. Silvennoinen and L. Kuparinen, "Usability challenges in surgical simulator training," in *Proceedings of the ITI 2009 31st International Conference on Information Technology Interfaces*, 2009, pp. 455-460.

[10] G. G. Menekse Dalveren and N. E. Cagiltay, "Insights from surgeons' eye-movement data in a virtual simulation surgical training environment: effect of experience level and hand conditions," *Behaviour & Information Technology*, vol. 37, pp. 517-537, 2018.

[11] N. Jäger, H. Schnädelbach, J. Hale, D. Kirk, and K. Glover, "Reciprocal control in adaptive environments," *Interacting with computers*, vol. 29, pp. 512-529, 2017.

[12] C. Feng, J. W. Rozenblit, and A. J. Hamilton, "A hybrid view in a laparoscopic surgery training system," in *14th Annual IEEE International Conference and Workshops on the Engineering of Computer-Based Systems (ECBS'07)*, 2007, pp. 339-348.

[13] K. Kahol, J. French, T. McDaniel, S. Panchanathan, and M. Smith, "Augmented virtual reality for laparoscopic surgical tool training," in *International Conference on Human-Computer Interaction*, 2007, pp. 459-467.

[14] H. Prescher, D. E. Biffar, C. A. Galvani, J. W. Rozenblit, and A. J. Hamilton, "Surgical navigation pointer facilitates identification of targets in a simulated environment," in *Proceedings of the 2014 Summer Simulation Multiconference*, 2014, p. 35.

[15] P. B. Andreatta, D. T. Woodrum, J. D. Birkmeyer, R. K. Yellamanchilli, G. M. Doherty, P. G. Gauger, *et al.*, "Laparoscopic skills are improved with LapMentor™ training: results of a randomized, double-blinded study," *Annals of surgery*, vol. 243, p. 854, 2006.

[16] A. K. Madan, J. L. Harper, C. T. Frantzides, and D. S. Tichansky, "Nonsurgical skills do not predict baseline scores in inanimate box or virtual-reality trainers," *Surgical endoscopy*, vol. 22, pp. 1686-1689, 2008.

[17] N. O. Kolozsvari, A. Andalib, P. Kaneva, J. Cao, M. C. Vassiliou, G. M. Fried, *et al.*, "Sex is not everything: the role of gender in early performance of a fundamental laparoscopic skill," *Surgical endoscopy*, vol. 25, pp. 1037-1042, 2011.

[18] L. Hedman, P. Ström, P. Andersson, A. Kjellin, T. Wredmark, and L. Felländer-Tsai, "High-level visual-spatial ability for novices correlates with performance in a visual-spatial complex surgical simulator task," *Surgical Endoscopy and Other Interventional Techniques*, vol. 20, pp. 1275-1280, 2006.

[19] C. W. Snyder, M. J. Vandromme, S. L. Tyra, J. R. Porterfield, R. H. Clements, and M. T. Hawn, "Effects of virtual reality simulator training method and observational learning on surgical performance," *World journal of surgery*, vol. 35, pp. 245-252, 2011.

[20] M. Silvennoinen, J.-P. Mecklin, P. Saariluoma, and T. Antikainen, "Expertise and skill in minimally invasive surgery," *Scandinavian Journal of Surgery*, vol. 98, pp. 209-213, 2009.

APPENDICES

APPENDIX A

HAPTICDEVICE.CPP CODE

```
#include "HapticDevice.h"

#include <iostream>

using namespace std;

bool HapticDevice::s_schedulerStarted = false;

HapticDevice::HapticDevice(std::string name) : m_hDevice(-1),
m_hConstantForceCallback(-1),
m_hDataCallback(-1), m_strHapticName(name), m_bSmoothing(false),
m_bInitialized(false)
{
	m_forceData.m_dMagnitude = 0;
	m_forceData.m_forceVector[0] = 0;
	m_forceData.m_forceVector[1] = 0;
	m_forceData.m_forceVector[2] = 0;
	m_prevForceData = m_forceData;
	m_currentData = new DeviceData();
}

HapticDevice::~HapticDevice()
{
	delete m_currentData;
	m_currentData = nullptr;
}

void HapticDevice::SetConstantForce(const ForceData *data)
{
	pthread_mutex_lock(&m_forceMutex);

	m_forceData = *data;

	pthread_mutex_unlock(&m_forceMutex);
}

void HapticDevice::SetConstantForce(double x, double y, double z, double mag)
{
	pthread_mutex_lock(&m_forceMutex);

	m_forceData.m_forceVector[0] = x;
	m_forceData.m_forceVector[1] = y;
	m_forceData.m_forceVector[2] = z;
	m_forceData.m_dMagnitude = mag;

	pthread_mutex_unlock(&m_forceMutex);
}

void HapticDevice::GetData(DeviceData* pOutData)
{
```

```
	if (m_bInitialized)
	{
		if (pOutData == nullptr)
			pOutData = new DeviceData();

		pthread_mutex_lock(&m_dataMutex);

		pOutData->m_bBtn1 = m_currentData->m_bBtn1;
		pOutData->m_bBtn2 = m_currentData->m_bBtn2;
		pOutData->m_bInkwell = m_currentData->m_bInkwell;
		pOutData->m_bCalibrated = m_currentData->m_bCalibrated;

		for (int i = 0; i < 3; ++i)
		{
			pOutData->m_rgfPosition[i] = m_currentData-
>m_rgfPosition[i];
			pOutData->m_rgfAngles[i] = m_currentData->m_rgfAngles[i];
			pOutData->m_rgfGimbalAngles[i] = m_currentData-
>m_rgfGimbalAngles[i];
			pOutData->m_rgfVelocity[i] = m_currentData-
>m_rgfVelocity[i];
		}
		pOutData->m_rgfPosition[0] = m_currentData->m_rgfPosition[0];

		pthread_mutex_unlock(&m_dataMutex);
	}
	else
	{
		delete pOutData;
	}
}

HHD HapticDevice::Initialize()
{
	m_hDevice = hdInitDevice(m_strHapticName.c_str());

	HDErrorInfo error;
	if (HD_DEVICE_ERROR(error = hdGetError()))
	{
		// Error handling
		// should exit the program
	}

	hdMakeCurrentDevice(m_hDevice);
	HDint style;

	hdGetIntegerv(HD_CALIBRATION_STYLE, &style);
	if (style & HD_CALIBRATION_ENCODER_RESET)
	{
		m_calibrationStyle = HD_CALIBRATION_ENCODER_RESET;
	}
	else if (style & HD_CALIBRATION_INKWELL)
	{
		m_calibrationStyle = HD_CALIBRATION_INKWELL;
	}
	else if (style & HD_CALIBRATION_AUTO)
	{
		m_calibrationStyle = HD_CALIBRATION_AUTO;
	}
```

```
        m_bInitialized = true;

        // Initialize thread attributes and create.
        // Priorities can be set here via attributes.
(sched_param::sched_priority)
        pthread_mutex_init(&m_dataMutex, NULL);
        pthread_mutex_init(&m_forceMutex, NULL);

        //s_devices.push_back(this);

        m_hDataCallback =
hdScheduleAsynchronous(HapticDevice::DataCallbackHelper, this,
HD_DEFAULT_SCHEDULER_PRIORITY);
        m_hConstantForceCallback =
hdScheduleAsynchronous(HapticDevice::ForceCallbackHelper, this,
HD_MAX_SCHEDULER_PRIORITY);

        return m_hDevice;
}

void HapticDevice::DisableDevice()
{
        if (m_bInitialized)
        {
                // Release threads.
                pthread_mutex_destroy(&m_dataMutex);
                pthread_mutex_destroy(&m_forceMutex);

                m_bInitialized = false;
                hdDisableDevice(m_hDevice);
                if (m_hDataCallback)
                        hdUnschedule(m_hDataCallback);
                if (m_hConstantForceCallback)
                        hdUnschedule(m_hConstantForceCallback);
        }
}

void HapticDevice::StartScheduler()
{
        if (!s_schedulerStarted)
        {
                HDErrorInfo error;
                hdStartScheduler();
                if (HD_DEVICE_ERROR(error = hdGetError()))
                {
                        // Error handling
                }
                s_schedulerStarted = true;
        }
}

void HapticDevice::StopScheduler()
{
        if (s_schedulerStarted)
        {
                s_schedulerStarted = false;
                hdStopScheduler();
        }
}
```

APPENDIX B

HAPTIC.CS CODE

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Runtime.InteropServices;

namespace HapticToUnity
{
  public class Haptic
  {
    private static Dictionary<string, Haptic> initializedHaptics = new
Dictionary<string, Haptic>();
    private string name;
    private Haptic.DeviceData tempData;
    private Haptic.DeviceData currentData;

    public bool Button1
    {
      get
      {
        return this.currentData.btn1;
      }
    }

    public bool Button2
    {
      get
      {
        return this.currentData.btn2;
      }
    }

    public bool Inkwell
    {
      get
      {
        return this.currentData.inkwell;
      }
    }

    public Haptic.Vec3 Position
    {
      get
      {
        return new Haptic.Vec3(this.currentData.position[0],
this.currentData.position[1], this.currentData.position[2]);
      }
    }

    public Haptic.Vec3 Angle
    {
      get
      {
        return new Haptic.Vec3(this.currentData.angles[0],
this.currentData.angles[1], this.currentData.angles[2]);
      }
```

```
    }

    public Haptic.Vec3 GimbalAngle
    {
      get
      {
        return new Haptic.Vec3(this.currentData.gAngles[0],
this.currentData.gAngles[1], this.currentData.gAngles[2]);
      }
    }

    public Haptic.Vec3 Velocity
    {
      get
      {
        return new Haptic.Vec3(this.currentData.velocity[0],
this.currentData.velocity[1], this.currentData.velocity[2]);
      }
    }

    private Haptic(string name)
    {
      this.name = name;
      this.tempData = new Haptic.DeviceData();
      this.currentData = new Haptic.DeviceData();
      Haptic.initialize(name);
    }

    ~Haptic()
    {
      Haptic.initializedHaptics.Remove(this.name);
      Haptic.disableDevice(this.name);
    }

    [DllImport("Assets\\Plugins\\HapticWrapper.dll")]
    private static extern void initialize(string name);

    [DllImport("Assets\\Plugins\\HapticWrapper.dll")]
    private static extern void disableDevice(string name);

    [DllImport("Assets\\Plugins\\HapticWrapper.dll")]
    private static extern void startScheduler();

    [DllImport("Assets\\Plugins\\HapticWrapper.dll")]
    private static extern void stopScheduler();

    [DllImport("Assets\\Plugins\\HapticWrapper.dll")]
    private static extern void setConstantForce(string name, double x, double
y, double z, double mag);

    [DllImport("Assets\\Plugins\\HapticWrapper.dll")]
    private static extern void getData(string name,
[MarshalAs(UnmanagedType.Struct)] ref Haptic.DeviceData data);

    public static Haptic GetHaptic(string name)
    {
      if (Haptic.initializedHaptics.ContainsKey(name))
        return Haptic.initializedHaptics[name];
      Haptic haptic = new Haptic(name);
      Haptic.initializedHaptics.Add(name, haptic);
```

```
      return haptic;
    }

    public Haptic.DeviceData GetData()
    {
      Haptic.getData(this.name, ref this.tempData);
      this.currentData = this.tempData;
      return this.currentData;
    }

    public void ApplyForce(double x, double y, double z, double mag = 1.0)
    {
      Haptic.setConstantForce(this.name, x, y, z, mag);
    }

    public void DisableDevice()
    {
      Haptic.initializedHaptics.Remove(this.name);
      Haptic.disableDevice(this.name);
    }

    public static void StartScheduler()
    {
      Haptic.startScheduler();
    }

    public static void StopScheduler()
    {
      Haptic.stopScheduler();
    }

    public static IEnumerable<string> GetHapticConfigs()
    {
      List<string> stringList = new List<string>();
      string path = Path.Combine(Environment.GetEnvironmentVariable("PUBLIC"),
"Documents\\SensAble");
      if (!Directory.Exists(path))
        throw new IOException("Sensable directory cannot be located. Search
directory : " + path);
      foreach (string file in Directory.GetFiles(path, "*.config"))
      {
        char[] chArray = new char[1]{ '\\' };
        string str = ((IEnumerable<string>)
file.Split(chArray)).Last<string>();
        stringList.Add(((IEnumerable<string>)
str.Split('.')).FirstOrDefault<string>());
      }
      return (IEnumerable<string>) stringList;
    }

    [StructLayout(LayoutKind.Sequential, Pack = 8)]
    public struct DeviceData
    {
      [MarshalAs(UnmanagedType.I1)]
      public bool btn1;
      [MarshalAs(UnmanagedType.I1)]
      public bool btn2;
      [MarshalAs(UnmanagedType.I1)]
      public bool inkwell;
      [MarshalAs(UnmanagedType.ByValArray, SizeConst = 3)]
```

```
        public float[] position;
        [MarshalAs(UnmanagedType.ByValArray, SizeConst = 3)]
        public float[] angles;
        [MarshalAs(UnmanagedType.ByValArray, SizeConst = 3)]
        public float[] gAngles;
        [MarshalAs(UnmanagedType.ByValArray, SizeConst = 3)]
        public float[] velocity;
      }

      public class Vec3
      {
        public float X;
        public float Y;
        public float Z;

        public Vec3()
        {
        }

        public Vec3(float x, float y, float z)
        {
          this.X = x;
          this.Y = y;
          this.Z = z;
        }
      }
    }
  }
```

APPENDIX C

HAPTICCONTROLLER.CS CODE

```
using UnityEngine;
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using HapticToUnity;
using System.Threading;
using ECEFileSystem;

public class HapticController : MonoBehaviour
{

    public Haptic haptic;
    public Haptic StoreHaptic;
    private Haptic.DeviceData currentData = new Haptic.DeviceData();
    private Haptic.DeviceData prevData = new Haptic.DeviceData();
    public ToolType HapticToolType;
    public Thread HapticThread;
    private bool HapticThreadStop = false;
    private static int calibratedHapticCount = 0;
    private bool localHapticCalibrationDone = false;

    public Texture2D CalibrationStartTexture;
    public Texture2D CalibrationEndTexture;
    public Texture2D CalibrationCircleTexture;
    private bool IsForced = false;
    public AudioClip CalibrationSound;
    public static List<HapticController> HapticControllers = new
List<HapticController>();
    [HideInInspector]
    public string HapticName = "null";
    public bool OldHaptic = false;
    public Vector4 ForceVector = Vector4.zero;
[Range(1, 3)]
    public float Sensitivity = 2;


    public static bool CalibrationDone = false;
    private bool IsStarted = false;
    [HideInInspector]
    public Vector3 InitialPosition = Vector3.zero;
    private bool EnableGravity = false;
    private bool Vibration = false;

    public static bool IsReadyToWork = false;
    public bool Button1 = false;
    public bool Button2 = false;
    private static int InitHapticCount = 0;

    private bool LocalHaptic = false;
    [HideInInspector]
    public int HapticIndex = 0;

    public int cam_hit = 0;
```

```
    public HapticController camScript;

    public List<float> forceList = new List<float>();

    void Awake()
    {
        lock (HapticManager.LockHaptic)
        {
            if (IsReadyToWork)
            {
                HapticControllers = new List<HapticController>();
                CalibrationDone = false;
                IsReadyToWork = false;
            }

            HapticControllers.Add(this);
        }

        print("Controller Awake");
        InitHapticCount++;
        if (Camera.main == null)
        {
            GameObject cam = GameObject.Find("Camera");
            if (cam == null)
                cam = GameObject.Find("Main Camera");
            if (cam != null)
            {
                cam.tag = "MainCamera";
                cam.AddComponent<AudioListener>();
            }
        }
        if (!HapticManager.GeneralHapticReady)
        {
            lock (HapticManager.LockHaptic)
            {
                if (HapticManager.HapticNames.Count == 0)
                {
                    HapticManager.HapticNames =
Haptic.GetHapticConfigs().ToList();
                }

                HapticManager.HapticNames.Remove("Default Device");
                HapticIndex = HapticManager.HapticCount;
                haptic =
Haptic.GetHaptic(HapticManager.HapticNames[HapticIndex]);
                StoreHaptic = haptic;
                HapticName = HapticManager.HapticNames[HapticIndex];
                HapticManager.HapticCount++;
            }
        }
    }

void Start()
    {

        try
        {
            print(this.gameObject.name + " " + HapticToolType);

            if (!HapticManager.GeneralHapticReady)
```

```
            {
                LocalHaptic = true;
                HapticManager.HideCross = true;

                lock (HapticManager.LockHaptic)
                {
                    if (!HapticManager.SchedulerStarted)
                    {
                        Haptic.StartScheduler();
                        HapticManager.SchedulerStarted = true;
                        print("Local Scheduler started");
                    }
                }
            }
            else if (!HapticManager.AddHaptic(this))
            {
                Debug.Log("Haptic Error");
            }
        }
        catch (System.Exception ex)
        {
            Debug.LogError(ex);
            OnApplicationQuit();
        }

        InitialPosition = new Vector3(this.transform.position.x,
this.transform.position.y, 0); ;

        HapticThread = new Thread(HapticForceThread);
        HapticThread.Start();
        HapticThread.IsBackground = true;

        AudioSource aSource = this.GetComponent<AudioSource>();
        if (aSource == null)
        {
            aSource = this.gameObject.AddComponent<AudioSource>();
            aSource.playOnAwake = false;
        }
    }

void Update()
    {

        if (HapticManager.SchedulerStarted)
        {
            if (!IsStarted)
            {
                currentData = haptic.GetData();
                if (currentData.position[0] != 0 && currentData.position[1] !=
0 && currentData.position[2] != 0)
                {
                    prevData = currentData;
                    IsStarted = true;
                }
            }
            else if (CalibrationDone)
            {
                if (HapticManager.HideCross)
                {
                    HapticMove();
```

```
                    HapticRotate();
                }

                if (currentData.btn1)
                {
                    Button1 = true;
                }
                else
                    Button1 = false;

                if (currentData.btn2)
                    Button2 = true;
                else
                    Button2 = false;

                var difDir = new Vector3(this.transform.position.x,
this.transform.position.y, 0) - InitialPosition;
                var difMag = difDir.magnitude;
                Vector3 forceVector = Vector3.zero;


                if (difMag > 40)
                    ForceVector = new Vector4(-difDir.normalized.x, -
difDir.normalized.y, forceVector.z, Mathf.Pow(difMag - 40, 2) / 500);

                if (HapticToolType == ToolType.Tool)
                {
                    if (CleaningController.CurrentBall != null)
                    {
                        var catchDir =
CleaningController.CurrentBall.transform.position - this.transform.position;
                        if (catchDir.sqrMagnitude < 100)
                        {
                            var mag = -(100 - catchDir.sqrMagnitude) * 10;
                            if (catchDir.sqrMagnitude <= 50)
                            {
                                if (!CleaningController.IsStick)
                                {
                                    CleaningController.StickCount++;
                                    CleaningController.IsStick = true;
                                    if
(!CleaningController.CurrentBall.GetComponent<BallController>().IsFree &&
!DirtController.isDirty)
                                        DirtController.Dirty();
                                }
                                CleaningController.HoldingBall = true;
                                if (Button1)
                                {
                                    if
(!CleaningController.CurrentBall.GetComponent<BallController>().IsFree)

CleaningController.CurrentBall.transform.position =
Vector3.Lerp(CleaningController.CurrentBall.transform.position,
this.transform.position, Time.deltaTime * Mathf.Pow(catchDir.sqrMagnitude, 2)
/ 2000);
                                    else

CleaningController.CurrentBall.transform.position = this.transform.position;

                                    CleaningController.IsStretch = true;
```

```
                                }
                                else
                                    CleaningController.IsStretch = false;

                                mag = catchDir.sqrMagnitude * 10;
                            }

                            ForceVector = new Vector4(catchDir.normalized.x,
catchDir.normalized.y, -catchDir.normalized.z, mag / 500f);

                            if (CleaningController.IsStretch)
                            {

FileManager.AddRowToRecord(GameManager.ForceTrackerFileName,
                                                                        new
DataPair("DataTime", GameManager.GameTime),
                                                                        new
DataPair("BallIndex", CleaningController.CurrentBallIndex),
                                                                        new
DataPair("StickCount", CleaningController.StickCount),
                                                                        new
DataPair("Force", ForceVector.magnitude));
                            }
                        }
                        else
                        {
                            ForceVector = Vector4.zero;
                            CleaningController.HoldingBall = false;
                            CleaningController.IsStretch = false;
                            CleaningController.IsStick = false;
                        }
                    }
                    else
                    {
                        ForceVector = Vector4.zero;
                        CleaningController.HoldingBall = false;
                        CleaningController.IsStretch = false;
                        CleaningController.IsStick = false;
                    }
                }
                else if (HapticToolType == ToolType.Endoscopy)
                {
                    if (Button1 && this.transform.position.z < -125 &&
DirtController.isDirty)
                    {
                        DirtController.Clean();
                    }
                }
            }
            prevData = currentData;
            currentData = haptic.GetData();

            if (!LocalHaptic && Input.GetKey(KeyCode.Escape))
            {
                HapticManager.HapticsEnd();
            }
        }

        if (Input.GetKeyUp(KeyCode.H))
        {
```

```
            HapticIndex = (HapticIndex + 1) % InitHapticCount;
            haptic = HapticControllers[HapticIndex].StoreHaptic;
            currentData = haptic.GetData();
            prevData = currentData;
        }
    }

void OnGUI()
    {
        if (GameManager.IsLogged)
        {
            GUIStyle style = new GUIStyle();
            style.fontSize = 40;
            if (HapticManager.SchedulerStarted && !localHapticCalibrationDone
&& currentData.position[0] != 0)
            {
                float w = (6 * Screen.width - Screen.width) / 12;
                float h = (6 * Screen.height - Screen.width) / 12;
                Vector3 translate = Vector3.zero;

                translate = new Vector3((float)(currentData.position[0]),
(float)(currentData.position[1]), -(float)(currentData.position[2]));

                float cDif = Screen.width / 48;
                float size = Screen.width / 6 + cDif - translate.z - 70;

                float cW = ((6 * Screen.width - Screen.width) / 12 - cDif / 2)
+ translate.x;
                float cH = ((6 * Screen.height - Screen.width) / 12 - cDif /
2) - translate.y;

                if (translate.z > -70)
                {
                    GUI.DrawTexture(new Rect(cW, cH, size, size),
CalibrationCircleTexture);
                    GUI.DrawTexture(new Rect(w, h, Screen.width / 6,
Screen.width / 6), CalibrationStartTexture);
                }
                else
                {
                    GUI.DrawTexture(new Rect(w, h, Screen.width / 6,
Screen.width / 6), CalibrationStartTexture);
                    GUI.DrawTexture(new Rect(cW, cH, size, size),
CalibrationCircleTexture);
                }

                if (!localHapticCalibrationDone && Mathf.Abs(translate.z + 70)
< 10 && Mathf.Abs(translate.x) < 10 && Mathf.Abs(translate.y) < 10)
                {
                    calibratedHapticCount++;
                    localHapticCalibrationDone = true;
                    print(calibratedHapticCount + ". Haptic Done");
                    if (calibratedHapticCount == InitHapticCount)
                    {
                        print("All Done");
                        CalibrationDone = true;
                        IsReadyToWork = true;

Camera.main.GetComponent<AudioSource>().PlayOneShot(CalibrationSound);
                    }
```

```
                }
            }
        }
    }
void HapticForceThread()
    {
        Vector4 forceTemp = Vector4.zero;
        Vector4 actTemp = Vector4.zero;

        while (!HapticThreadStop)
        {
            if (HapticManager.SchedulerStarted)
            {
                forceTemp = ForceVector;
                if (EnableGravity)
                    forceTemp += new Vector4(0, -1f, 0, 0.1f);

                actTemp = Vector4.Lerp(actTemp, forceTemp, 1f);
                Thread.Sleep(1);
                haptic.ApplyForce(actTemp.x, actTemp.y, actTemp.z, actTemp.w);
            }
        }
    }

    void HapticMove()
    {
        Vector3 translate = new Vector3((float)(currentData.position[0] -
prevData.position[0]), (float)(currentData.position[1] -
prevData.position[1]), -(float)(currentData.position[2] -
prevData.position[2]));
        if (IsForced)
        {
            this.GetComponent<Rigidbody>().AddRelativeForce(translate *
Sensitivity * 5000);
        }
        else
        {
            this.transform.position += translate * Sensitivity;
        }
    }

    private void HapticRotate()
    {
        Vector3 rot = new Vector3(-(float)(currentData.gAngles[0]),
(float)(currentData.gAngles[1]), -(float)(currentData.gAngles[2]));
        this.transform.rotation = Quaternion.Euler(Mathf.Rad2Deg * rot.y,
Mathf.Rad2Deg * rot.x, Mathf.Rad2Deg * rot.z);
    }

    public void Vibrate()
    {
        Vibration = true;
    }

    public void StopGravityForce()
    {
        EnableGravity = false;
    }
```

```
public void StartGravityForce()
{
    EnableGravity = true;
}

public void StopForceThread()
{
    if (HapticThread != null)
    {
        HapticThreadStop = true;
        HapticThread.Abort();
    }
}

void OnApplicationQuit()
{
    StopForceThread();

    if (LocalHaptic)
    {
        lock (HapticManager.LockHaptic)
        {
            if (HapticManager.SchedulerStarted)
            {
                HapticManager.SchedulerStarted = false;
                Haptic.StopScheduler();
            }
            haptic.DisableDevice();
        }
    }

}}
```

APPENDIX D

SAMPLE HAND DATA (JSON)


```
{
   "0":{
         "BallIndex":1,
         "StickCount":0,
         "DataTime":0.109229907,
         "ToolPosition":"(-19.6, 93.1, -91.0)",
         "ToolRotationEuler":"(45.3, 325.0, 10.7)",
         "ToolRotationQuaternion":"(-0.3, 0.3, -0.2, -0.9)",
         "CamPosition":"(-20.7, 102.9, -130.7)",
         "CamRotationQuaternion":"(-0.1, -0.1, -0.4, -0.9)",
         "CamRotationEuler":"(2.3, 17.2, 52.9)"
   },
```

```
"1":{
        "BallIndex":1,
        "StickCount":0,
        "DataTime":0.218399584,
        "ToolPosition":"(-18.4, 93.2, -91.2)",
        "ToolRotationEuler":"(37.5, 338.7, 15.8)",
        "ToolRotationQuaternion":"(-0.3, 0.2, -0.2, -0.9)",
        "CamPosition":"(-20.8, 105.2, -132.2)",
        "CamRotationQuaternion":"(-0.1, -0.1, -0.4, -0.9)",
        "CamRotationEuler":"(2.7, 17.2, 52.9)"
},
"2":{
        "BallIndex":1,
        "StickCount":0,
        "DataTime":0.3120566,
        "ToolPosition":"(-18.5, 93.8, -90.8)",
        "ToolRotationEuler":"(32.0, 343.1, 17.6)",
        "ToolRotationQuaternion":"(-0.2, 0.2, -0.2, -0.9)",
        "CamPosition":"(-20.6, 108.5, -134.0)",
        "CamRotationQuaternion":"(-0.1, -0.1, -0.4, -0.9)",
        "CamRotationEuler":"(2.7, 17.2, 52.9)"
},
"3":{
        "BallIndex":1,
        "StickCount":0,
        "DataTime":0.405642867,
        "ToolPosition":"(-20.1, 92.7, -90.0)",
        "ToolRotationEuler":"(27.7, 340.7, 18.1)",
        "ToolRotationQuaternion":"(-0.2, 0.2, -0.2, -0.9)",
        "CamPosition":"(-20.6, 110.9, -134.9)",
        "CamRotationQuaternion":"(-0.1, -0.1, -0.4, -0.9)",
        "CamRotationEuler":"(2.8, 17.2, 52.9)"
```

```
        }
        …
}
```

APPENDIX E

SAMPLE FORCE DATA (JSON)


```
{
        "0":{
                "DataTime":162.7015,
                "BallIndex":1,
                "StickCount":7,
                "Force":1.26872921
        },
        "1":{
                "DataTime":162.725479,
                "BallIndex":1,
                "StickCount":7,
                "Force":1.2634666
        },
        "2":{
                "DataTime":162.7411,
                "BallIndex":1,
                "StickCount":7,
                "Force":1.24990177
        },
        "3":{
                "DataTime":162.7567,
                "BallIndex":1,
```

```
                "StickCount":7,
                "Force":1.22922707
        },
        "4":{
                "DataTime":162.772263,
                "BallIndex":1,
                "StickCount":7,
                "Force":1.22104156
        },
        "5":{
                "DataTime":162.787888,
                "BallIndex":1,
                "StickCount":7,
                "Force":1.2134459
        },
        "6":{
                "DataTime":162.8035,
                "BallIndex":1,
                "StickCount":7,
                "Force":1.20633674
        },
        "7":{
                "DataTime":162.819046,
                "BallIndex":1,
                "StickCount":7,
                "Force":1.19072461
        },
        "8":{
                "DataTime":162.8347,
                "BallIndex":1,
                "StickCount":7,
                "Force":1.18507946
        },
        "9":{
                "DataTime":162.850586,
                "BallIndex":1,
                "StickCount":7,
                "Force":1.17973268
        }
}
```

APPENDIX F

GENERAL DATA (JSON)


```
{
	"0":{
		"BallIndex":1,
		"LevelTime":176.6253,
		"Duration":176.609726,
		"IsSucceed":true,
		"TouchCount":0,
		"TotalToolDistance":1573.06018,
		"TotalCamDistance":7129.379,
		"StickCount":7
	},
	"1":{
		"BallIndex":2,
		"LevelTime":249.930222,
		"Duration":73.29543,
		"IsSucceed":true,
		"TouchCount":0,
		"TotalToolDistance":1251.787,
		"TotalCamDistance":2169.05322,
		"StickCount":5
	},
	"2":{
		"BallIndex":3,
		"LevelTime":296.464417,
		"Duration":46.54532,
		"IsSucceed":true,
```

```
        "TouchCount":0,
        "TotalToolDistance":419.319153,
        "TotalCamDistance":1006.06311,
        "StickCount":3
},
"3":{
        "BallIndex":4,
        "LevelTime":476.454254,
        "Duration":180.010712,
        "IsSucceed":false,
        "TouchCount":0,
        "TotalToolDistance":4715.32227,
        "TotalCamDistance":6958.48438,
        "StickCount":5
},
"4":{
        "BallIndex":5,
        "LevelTime":656.472,
        "Duration":180.0109,
        "IsSucceed":false,
        "TouchCount":0,
        "TotalToolDistance":6828.358,
        "TotalCamDistance":2841.93262,
        "StickCount":8
},
"5":{
        "BallIndex":6,
        "LevelTime":836.484436,
        "Duration":180.002243,
        "IsSucceed":false,
        "TouchCount":1,
        "TotalToolDistance":4263.57764,
        "TotalCamDistance":4842.308,
        "StickCount":32
},
"6":{
        "BallIndex":7,
        "LevelTime":1013.05518,
        "Duration":176.563873,
        "IsSucceed":true,
        "TouchCount":2,
        "TotalToolDistance":9921.838,
        "TotalCamDistance":3737.325,
        "StickCount":24
},
"7":{
        "BallIndex":8,
        "LevelTime":1125.29431,
```

```
                "Duration":112.227928,
                "IsSucceed":true,
                "TouchCount":1,
                "TotalToolDistance":3029.781,
                "TotalCamDistance":1346.75549,
                "StickCount":22
        },
        "8":{
                "BallIndex":9,
                "LevelTime":1239.39209,
                "Duration":114.068565,
                "IsSucceed":true,
                "TouchCount":0,
                "TotalToolDistance":3675.294,
                "TotalCamDistance":1644.051,
                "StickCount":13
        },
        "9":{
                "BallIndex":10,
                "LevelTime":1419.409,
                "Duration":180.010635,
                "IsSucceed":false,
                "TouchCount":0,
                "TotalToolDistance":4292.53125,
                "TotalCamDistance":3603.67651,
                "StickCount":8
        }
}
```