NTNU
Norwegian University of
Science and Technology

Medical Procedural Training in Virtual Reality

Tarald Gåsbakk
Håvard Snarby

Master of Science in Computer Science
Submission date: June 2017
Supervisor: Frank Lindseth, IDI

Norwegian University of Science and Technology
Department of Computer Science

NTNU – Trondheim
Norwegian University of
Science and Technology

TDT4900 MASTER'S THESIS
Spring 2017

Medical Procedural Training in Virtual Reality

Tarald Gåsbakk
Håvard Snarby

Coordinator: Frank Lindseth
Co-coordinator: Ekaterina Prasolova-Førland
Co-coordinator: Aslak Steinsbekk

June 25, 2017

Abstract

Virtual reality tools have seen a large increase in interest over the last few years. Educators have been early adopters of such tools, and research have shown that students enjoy training in a virtual world using virtual reality devices. New tools for interacting with the virtual world, like the controllers offered by virtual reality products like HTC Vive and Oculus Rift opens for more immersive applications. In addition, the usage of real world medical imaging in virtual applications is a good way to further peak the users interest and involvement. This thesis explores the possibilities such hardware and data usage opens for a procedural training application in virtual reality, and the research question was worded as follows: *Which functionality are required to create an application using general purpose virtual reality equipment for medical procedural training supported by imaging data.* In order to answer, an application where users are guided through a neurosurgical pre operation phase was implemented, and tested on two occasions. A video presenting the core functionality may be found here https://goo.gl/5djsSd. The feedback and observations from user tests showed that users were quickly able to interact sufficiently in the virtual environment. Observations also shows that it is indeed possible to utilize real world medical data to support a learning applications. We conclude that it is indeed possible to create a learning application using general purpose virtual reality equipment supported with real world medical imaging.

Sammendrag

Verktøy for virtuell virkelighet har opplevd en stor interesseøkning de siste årene. Pedagoger har vært tidlig ute med å utnytte slike verktøy, og forskning viser at studenter liker å trene i en virtuell verden. Nye verktøy for å interaktere med en virtuell verden, som kontrollerne tilbudt av HTC Vive og Oculus Rift åpner for applikasjoner som gir brukere en bedre følelse av innlevelse. I et tillegg kan integrering av medisinske bildedata fra den virkelige verden gi brukerne en mer virkelighetsnær opplevelse. Denne tesen utforsker mulighetene slik maskinvare og bruk av data tilbyr for prosedyretrening i virtuell virkelighet, og forskingsspørsmålet som ble undersøkt hadde følgende ordlyd: *Hvilke funksjonaliter kreves for å konstruere en applikasjon som bruker generelle verktøy for virtuell virkelighet til å gjennomføre medisinsk prosedyretrening supplementert med medisinske bildedata.* For å svare på dette ble en applikasjon konstruert, hvor brukerne går gjennom en pre-operativ fase av en nevrokirurgisk operasjon. Denne applikasjonen ble testet ved to anledninger. En video som presenter kjernefunksjonaliteten i applikasjonen kan sees her `https://goo.gl/5djsSd`. Tilbakemeldinger og observasjoner tilegnet igjennom brukertesting av applikasjonen viser at brukerne klarer å interaktere med den viritueIle verden med liten opplæringstid. Det ble også observert at medisinske bildedata bygde godt opp under scenariene i applikasjonen. Vi konkluderer derfor at det er mulig å lage en læringsapplikasjon ved å bruke generelle verktøy for virtuell virkelighet som er underbygget med medisinske bildedata.

Glossary

ACS-Panel Panel showing the axial, corronal and sagital view of an MRI(see MRI). 37

HMD Head mounted display. 16

MRI Magnetic resonance imaging. 4

OR Oculus rift. 17

PUN Photon unity network. 72

Vive HTC Vive. 18

VR Virtual reality. 1

Contents

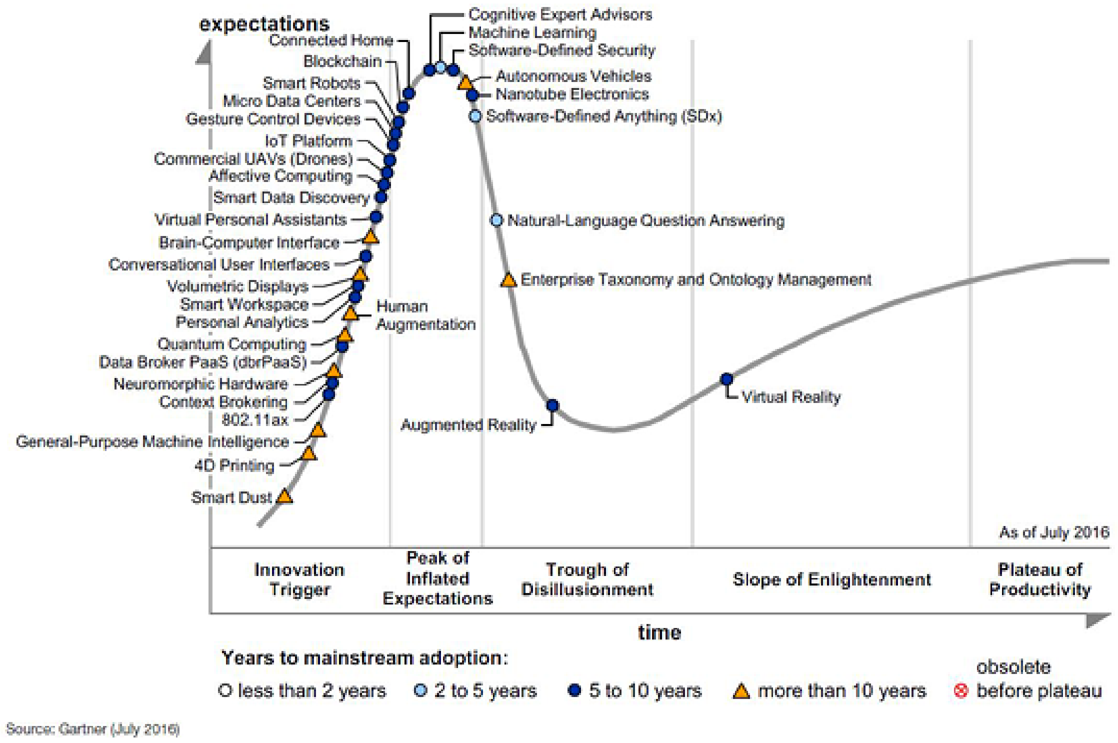
List of Figures

1 Introduction

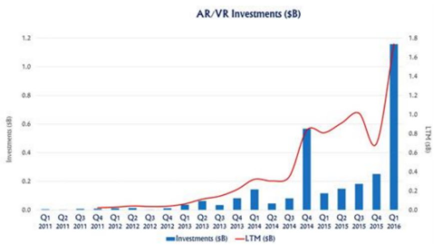
1.1 Motivation

Practical experience is an essential part of medical education on all levels. The need for students to have experience with the hospital environment, patient communication and practical skills with tools and apparatus at the end of their studies are paramount. Traditionally, students have been placed in practice and clinical rotations in a hospital setting, but due to an increased number of students, as well as an increase in the need of expertise in interprofessional communication and collaboration, have set limitations for how much practice a student will get during their education. With the goal of remedying these challenges, the medical world have looked to 3D technology to support its struggles [4] [5] [6].

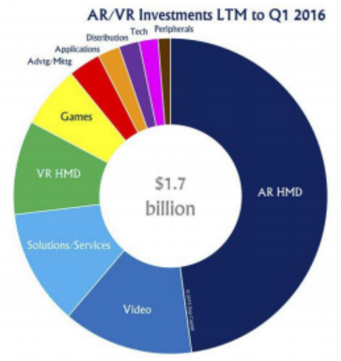
Virtual Reality(VR) tools have traditionally been expensive and largely unpractical for personal use, but in the early 2010s, this took a drastic turn. With the launch of gear like Oculus Rift, HTC Vive, and Samsung Gear VR, the technology got significantly cheaper, more portable, and demanded less computing power than ever before. This have lead to a new dawn for the use of VR in education [7]. Figure 1a shows Gartner's *hype curve* for 2016. As seen on the Figure, VR is in the state of *Slope of Enlightenment*, which means that the interest of VR will steadily increase and stabilize over the next couple of years, according to Gartner. The hype cycle correspond well to Figure 1b, which illustrates the investments in VR technologies over the past six years. In 2016, Augmented Reality (AR) got public notice by Microsoft's announcement of their Hololens. In addition, the company *Magic Leap* have gotten a substantial amount of investors and attention in their coming AR equipment. As Figure 1c shows, almost 50% of the total investment of the first quarter of 2016 was dedicated to AR. Although the biggest investments on the market are on AR, the technology and equipment is still in its infancy. Microsoft has just recently released its product for developers, but have not yet announced a consumer version [8]. Magic Leap's product is not available for developers yet.



(a) Gartner's hype cycle [9]



(b) Investment of AR and VR over the last six years [7]



(c) Overview over investment in VR [7]

Figure 1: VR popularity and investment

While the main reason for the great popularity of VR have been due to the great possibilities offered in personal and business use, educators have found it interesting as well. A multitude of projects have been developed with different combinations of gear and software to make training applications for several different professions. Examples that educational applications are proven to be effective can be found in crisis training [10], general education [11], engineering studies [12] and medical education [13] [14]. Edgar Dale an educationist states that the human brain remembers 10% of what it reads, 20% of what it hears, and 90% of what it does. By using the VR tools, the learning has the potential to be four times as effective as listening or

reading [15]. The present state of research shows that while plentiful of applications have been made utilizing low cost HMDs, there are limited sources on procedural training using general purpose HMDs and controllers like the Oculus Touch.
The motivation for this project is to further increase the knowledge on educational applications by using new general purpose hardware. The interest topics of discussion is how this new hardware can improve the development process and re-usability of training applications, and how to best enhance such applications so that they provide high relevance in the real world.

1.2 Problem description

In order to asses the possibilities offered by HMDs, a VR educational system will be developed. The system will be built around a medical training scenario, which describes a problem that is difficult to prepare for using traditional methods of medical training. The thesis will also explore ways of implementing natural physical interaction using general purpose controllers that may act as interfaces into the virtual world. In addition, the thesis will also explore how the usage of real world medical data may be implemented in an natural way so that they naturally fit into a training scenario.

1.3 Research questions

The main goal of this thesis is to explore new functionality and possibilities that will improve the user's experience and learning potential using general purpose VR equipment. Based on this, research question has been generated for this thesis:

RQ: Which functionality are required to create an application using general purpose virtual reality equipment for medical procedural training supported by imaging data.

The main research question is further divided into two sub questions:

RQ 1: *Can real world medical imaging data be utilized to support training in virtual reality.*
RQ 2: *How do users experience using an application with general purpose virtual reality equipment for Medical Procedural Training*

Data handling
In this master thesis, medical data from real cases and patients have been used. These are limited to the MR and ultrasound images, which have been approved for usage in this thesis and have been made anonymous.

2 Background

In this chapter the necessary background information needed for understanding the rest of the paper will be presented, including previous work conducted in the field of study. In addition the scenarios implemented in the application will be described. Following is essential concepts and terminology for creating virtual reality applications. Lastly, an examination on the different general purpose head mounted displays will be presented.

2.1 Previous work

The usage of VR tools have proven to be beneficial in a number of educational applications. In general education, high school students have shown improvements in their understanding of material that is not easily visualized [11]. In medicine, simulation and VR tools have been gradually more recognized as a valuable asset for improving the skills of surgeons [16] [17]. Even though its usefulness is well documented, the implementation of VR in educational programs have been slow. In the past, such devices and technology have been costly to implement, time consuming to use and limited in their areas of use. After the breakthroughs of cheap and publicly accessible VR gear have lead to an increase in popularity and democratization of VR technology, which in turn leads to more applications made to the platforms. Educators should be aware and utilize this surge of interest to develop immersive learning applications that offers great possibilities of interaction [7]. The educational applications using low cost HMD may be divided into four main categories: communication training, procedural training, visualization, and interactive visualization. The following sections will describe these in more detail, with a focus on medical educational examples.

Procedural training
Procedural training in medical and surgical education have become increasingly difficult to practice in the traditional *See one, do one, teach one* framework, even though its value is still well documented [18]. However, due to patient safety concerns and time constraints on seasoned surgeons, VR simulators have been looked to as an alternative teaching method with advantages like easily available repetetive training, training on unusual cases and skill level maintenance [19]. A prime example of such an simulator is the *NeuroTouch neurosurgical simulator* [20], which have been implemented and tested at multiple hospitals in USA. The simulator consist of two parts, *NeuroTouch Plan* and *NeuroTouch Sim*. The NeuroTouch Plan consists of using Computed Tomography (CT) and Magnetic Resonance Imaging (MRI) data to create 3D model of the a human head, with the ability to examine the brain for preparation of neurosurgery. An example of usage is removal of tumor. Once the tumor is located the users define the surgical corridor. Once the surgical corridor is defined, the user is moved to the NeuroTouch Sim part, where the users perform the surgery based on the planning performed in the first part.

Even though the surgeon simulators have come a long way in preparing surgery students for a multitude of different surgeries, they are still not as widely implemented as one would have believed them to be given their apparent success in literature. Stefanidis et al. argues that simulators today are complex to construct and implement, as well as being perceived expensive. He argues that cost effectiveness and a more effective development process is needed for simulators to achieve greater success. [21]

Multiple alternatives to simulators have been explored after the initial massive success of Oculus Rift and HTC Vive. Some of the early explorations made, used self made devices to simulate hand controllers and give haptic feedback. Buń et al. have experienced with low cost head mounted display accompanied by a haptic device to give procedural training to factory workers, like the specialist that operate glass moulding machines [22]. Other systems have used more general purpose input devices. At Japitur College of Engineering in India, Mathur [23] have developed an surgeon trainer using Oculus Rift with the low cost input device *Razer Hydra* [24]. The users of the systems are using the headset to view the virtual world, and the controllers to interact with it from a stationary sitting position. The findings presented shows that users of the system feels immersed in the scenario and may train independent of supervision. It is also argued that similar applications have great potential.

Virtual Medical Coaching is a company heavy invested in VR procedural training [25]. They have developed an educational VR application called X-Ray Trainer. The goal of the application is to teach radiologist how to perform the complex tasks such as taking X-Ray images and analyze them. They also state that they are working on creating procedural training for taking Magnetic Resonance(MR) and Computed Tomography(CT) scans [26].

Communication training

While most educational applications have been made to train single users, some efforts have been done lately in terms of collaboration and communication in VR [27] [28]. These researches show that communication may be achieved in virtual reality in a manner that allows the users to collaborate efficiently on different tasks. Pan et al. have constructed a VR medical training scenario where a general practitioner is faced with an increasingly aggressive patient. The result of the study shows that immersive VR tools are an effective way of training medical professionals in their communication skills.

VirSam (Virtual communication in education), a local collaboration project between the university and the hospital in Trondheim to create a virtual environment for nurse and medical students to practice on interprofessional communication skills. The target platform for the environment is Second Life (SL). SL has received appreciation for being a suitable platform for creating educational content in a virtual environment [29] [30] [13]. The extendability of SL makes it possible for developers to create customized and closed environment where users can perform the given task. VirSam has developed two different scenarios to be played in SL to improve inter professional collaboration. These scenarios have also been implemented and tested in VR using Oculus Rift [31]. The results shows that users generally like the VR experience. However problems that are tied together with the platform, like no hand interaction and movement that causes most user to experience VR sickness, makes the experience sub optimal.

Interactive visualization
The introduction of HMD have opened for multiple new ways to present complex imagery, and the development of 360 degree videos allows people to immerse themselves into new scenarios and situations. One example of an application using the new technology is the *We are Alfred* system developed by *Embodied labs*. In this application you are watching the world from the point of view of an aging man, who is suffering from visually and hearing impaired. From a test conducted on first year medical students in New England, the application was found to increase empathy, knowledge of macular degeneration and hearing loss as well as shifting keywords associated with *older people* and *aging* from words like *slow* and *frail* to *misunderstood* and *frustration* [32].

In order to take visualization one step further, several actors have used models in VR and made users able to interact with them. In medical education, the most common implementations of such functionality is to create interactive models of the human body. Hamrol et al. [14] and Moro et al. [33] have implemented an interactive virtual experience where students may learn about the body by seeing and manipulate it in an virtual world. Both studies reports benefits for the learning outcomes of the students when using VR tools, as well as reporting that students are greatly enjoying the teaching experience. This intriguing way of implementing learning applications have also caught the eye of commercial actors. Two examples of such applications made are *The Body VR* and *3D Organon anatomy* [34] [35]. In *The Body VR* you are able to load a 3D-model, either from their database of already implemented models or your own, and view them in a fully interactive environment. You might slice open different parts and manipulate the volume in different ways. *3D Organon Anatomy* is an application that ships a human body model with fully detachable parts that the user can interact with. In addition to being able to take apart the body parts and view them independently, the application also provides an informative lexical text on each part.

A more specialized application for virtual interaction with medical data is the Dextroscope [36], which is made to support surgical evaluation and decision making. The system uses preoperative images in combination with segmentation of critical anatomic structures to present a information-fused 3D model on a stereographic display. The user can inspect and manipulate the model using application specific hardware. The hardware consist of an ergonomic handle with used for manipulation of the model and a stylus shaped instrument.

2.2 Terminology and concepts in virtual reality development

When developing applications for regular users, it is of grave importance that the interfaces the users interact with are easy to understand and well constructed. This fact is even more prevalent in virtual reality applications, as the technology is new enough that one should expect the users to have limited, or even no, experience with it. Developing virtual reality applications is a complex task by nature. Stereoscopic rendering and strict performance limitations gives additional challenges to developers compared to developing classically rendered virtual worlds. In this section, the readers will be introduced to terminology and concepts regarding user design and development of virtual reality applications in a game engines.

2.2.1 User interaction and immersion

The design and implementation of computer artifacts with a user interface that is both easy and efficient to use is well researched. In this section, a collection of the most important and relevant theories regarding user interface design will be presented and explained.

Usability

Usability is best defined through ISO 9241-11: the *extent to which a product can be used by specific users to achieve specified goals with effectiveness, efficiency and satisfaction to the context of use.* What is worth noting in this definition is that usability is not a constant, but is derived from variables such as the goals and physical and social context of use. For example, a system made with high usability for computer science students might have low usability for nursing students.

When developing a system, it is important to make the system understandable. The users should be able to learn how the system works with as little effort as possible. If the system is created with a constant focus on usability, it is more likely that the user understands the advantage of using it and is more likely to use it in the future.

Affordance

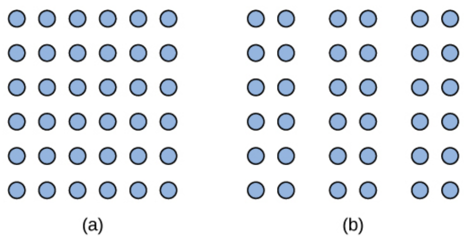
A term that is central in interaction design is affordance. According to Norman [37], affordance describes the relationship between an object and the action that can be performed on the object. The use of affordances reduces the learning curve of the system. An example of affordance is handle on a cup, which provides an obvious affordance for holding.

Gestalt principles

When constructing user interfaces, it is useful to keep Gestalts principles [38] in mind. They explain how the human brain categorizes and creates meaningful assumptions based on what they observe.

The principle of proximity explains how humans organize everything into groups and subgroups. The principle explains that objects close together is often perceived as a group, giving them the same functionality. Figure 2a shows how the bobbles on the figure is perceived as a group.

The principle of similarity states that objects with the same shape or characteristics belong in a group and share the same functionality. Figure 2b illustrates the effect of the principle. The shapes on the figure is perceived as five groups based on the different coloring.



(a) Gestalt principle of proximity

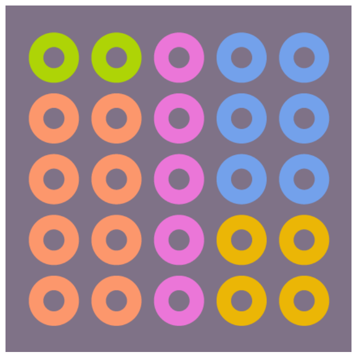

(b) Gestalt principle of similarity

Figure 2: Showing Gestalt principles

Usability testing

When developers work on a system for a long time, they get to know it intimately, and understands exactly how said system works. This often leads to construction of complex user interactions that are obvious and understandable for the developers, but less so for the target users. To ensure the quality in terms of usability and functionality, the system needs to be tested by people not involved in the development. There are multiple different tests that can be performed on a system to assert usability of different artifacts.

The main goal when conducting usability tests is to get feedback from the target users. There are two types of feedback: quantitative and qualitative. A common method of gathering quantitative feedback is through *Likert scale*, which is a questionnaire where the answers to the questions is a rating between 1 and 5. Qualitative feedback is gathered through interviews or recordings.

Feedback

For learning to take place, the user needs to get feedback from their actions. There are different ways of giving feedback that should be used based on the type of learning.

Instant feedback gives the user feedback whenever an action is made. According to Jan Cannon Bowers [39], the use of instant feedback and dynamic difficulty has shown to increase learning. The difficulty on a level is adjusted based on the users performance. If the user makes correct decisions, the game will become more difficult. Likewise, the game becomes easier if the user makes mistakes. The balancing of difficulty makes it possible to challenge the user, but also give them the feeling of accomplishment. If a game becomes too hard or too easy, the user will lose interest.

Even though instant feedback is good in many ways, the feedback may also be a hindrance to the user. The continuation of feedback may disturb the concentration of the user. In addition if the user receives negative feedback, the user might be discouraged or stressed resulting in more mistakes.

Another way of giving feedback is post game feedback. Instead of receiving feedback in real-time, the feedback is presented after a level or game session is completed. The feedback should consist of some positive and negative comments. Screen capturing of the game play is a good way of teaching the user where they made good and bad choices. For the learning to be most efficient, a facilitator should be present to point out these choices for the user while watching the video. If the game learning activity contains collaboration, it is beneficial for the users to meet after the session to discuss the actions performed during session. A facilitator should be present to control the discussion and give objective feedback.

Immersiveness
The term immersiveness indicates how immersed a user is when playing a game. Immersion makes users forget about the real world, and enables them to be fully engulfed.

The more a game resembles reality, the more immersed the user becomes. Attributes that affect immersiveness includes graphics, physics, sounds and user input. Graphic helps the user see similarities between the game and the real world. Physics also plays an important role in the degree of immersion for a user. Actions that breaks with physics often disturbs the feeling of immersiveness. If, for instance, the user walks through a wall, it will ruin the similarity with reality. user movements also helps the feeling of immersiveness. The better control the user has over the movement, the more immersed they become.

2.2.2 Development with game engines

A game engine is a software framework designed specifically for development of games. Developers use game engines to speed up the development process by abstracting core processes such as sound, physics, logic and graphics rendering. Previously, game engines were exorbitantly priced, resulting in most game making companies building their own from scratch. In recent years, the major game engines have been released with payment plans taking royalty fees if the game generate income beyond a certain threshold instead of payment upfront. This have given small companies the opportunity to create low cost games. As game engines matured, they have become more user-friendly, making them more inviting for novice game developers.

It was decided to use *Unity3D* [40] commonly known as Unity as the game engine for development of the application. Compared to other game engines, Unity falls into the category of a young one. First released in 2005, it was mainly aimed at OS X, but have since then extended its support to 21 different platforms. The Unity engine does not possess the graphical capabilities that Unreal Engine [41] and Cryengine [42] boasts, but have had a greater focus on the ability for games to run on a variety of platforms. When it comes to pricing, Unity offers a completely free plan for businesses that makes less yearly revenue than $100 000. The Unity asset store is one of the main features of Unity. The vast amount of community made content, like models or animations, helps new developers get a kick start for their projects.

Unity's editor uses the view illustrated in Figure 3. Unity boasts a solid documentation for developers to ease the learning curve. The code in the documentation is written for C# and UnityScript. In addition, Unity offers video tutorials to further help developers understand the complex view and methods applicable. Once defined, the functionality of the script can be easily added to GameObjects by dragging the script onto the GameObject's inspector view. This easy way of adding functionality to GameObjects makes it easy for the developer to add and edit functionality to multiple objects at once. Another great feature implemented in Unity is the ability to edit the variables of a script in the inspector view in the Unity editor, instead of opening a text editor. This is done by declaring public variables in the script-class.

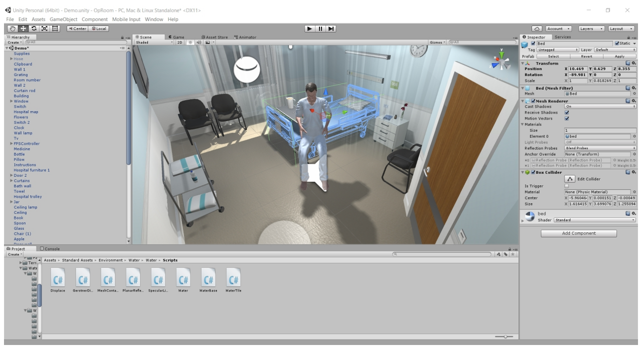
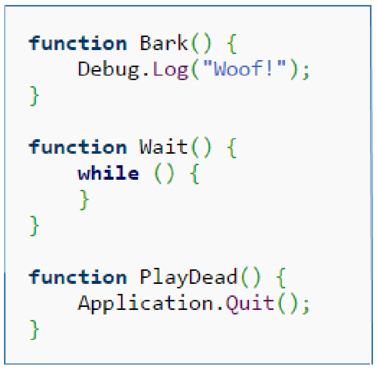


Figure 3: Working with Unity

Scripting in Unity

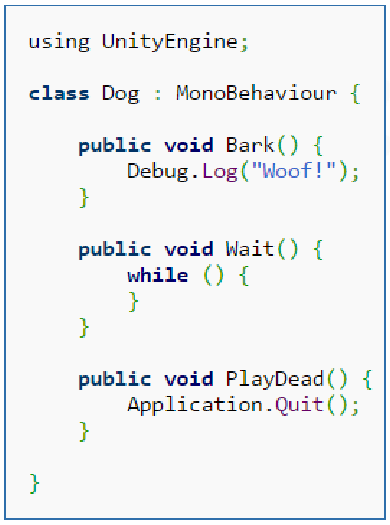
There are mainly two programming languages utilized to create scripts in Unity, namely C# and UnityScript. UnityScript is a scripting language created by Unity. The syntax is similar to the popular interpret language JavaScript, however there are some differences between the languages [43]. The script automatically derives from Monobehavior. It is possible to have several classes inside one script in UnityScript, but it is most common to only have one class per script. The developers have the option to skip the class declaration when the script only contains one class. The filename of the script will act as a class declaration. This is done to save developers from extensive typing. Figure 4 illustrates this example. On the left side the UnityScript file for a dog is shown. The same script is written in C# on the right side. When using C# in Unity, one has to explicitly derive the script from MonoBehaviour, which is the base class all scripts derive from, and contains the core functions needed to make the scripts work. Figure 5 illustrates the lifecycle of the Monobehavior class.



```
function Bark() {
    Debug.Log("Woof!");
}

function Wait() {
    while () {
    }
}

function PlayDead() {
    Application.Quit();
}
```



```
using UnityEngine;

class Dog : MonoBehaviour {

    public void Bark() {
        Debug.Log("Woof!");
    }

    public void Wait() {
        while () {
        }
    }

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

}
```

UnityScript

C#

Figure 4: Difference between UnityScript and C#

In the initialization step of a script's lifecycle consist of three methods, namely *Awake*, *OnEnable* and *Start*. The Awake method is the first method to be executed in the script, and is always called. It is used to initialize variables or game state before the game starts. Awake is executed after the scripts has been loaded into the game, making it safe to instantiate other GameObjects or scripts in this script. OnEnable is called whenever the GameObject is active. If the GameObject starts active, OnEnable will be executed after Awake. Start is similar to Awake in the regard that both initialize variables. The difference between the two methods is that the Start method only executes if the GameObject is active. This makes it possible to delay the initialization of code until it is needed. The Start and Awake method is only executed once, but the OnEnable method is called whenever the GameObject is activated.

At the beginning of each iteration in the lifecycle when the game has started, the physics engine is called. *FixedUpdate* is an update method that is called every physics step. The time between FixedUpdate are consistent, making it independent on frame size. The method is used to update GameObjects influenced by physics, such as rigidbodies. The *OnCollisionXXX* and *OnTriggerXXX* methods are also under the physics part. onCollisionXXX detects if the script's GameObject is colliding with another GameObject. The OnTriggerXXX detects if the GameObject is colliding with a trigger. A trigger is a sensor that sends a message to the GameObject when touching something (triggered).

Before the game logic is handled, the step for detecting and handling user input is performed. This step looks for input from users such as button pressed or mouse clicked. An example of method called during update is *OnMouseDown*.

There are two methods for updating non physics elements, namely *Update* and *LateUpdate*. *Update* is the most common update function. It is called once per frame, making it dependent on frame size. The input events from the previous step is handled in the update method, and it is responsible for moving non-physical GameObjects. The LateUpdate method is called after the update method. Any calculations in the update method will then be completed, making it possible for LateUpdate to use the calculations. An example where LateUpdate is useful is implementation of a third person camera. The user's movements and rotation is calculated during the update method, and the camera adjusts to the users new position and rotation in the LateUpdate method.

When the script has completed it's task, and is no longer needed, it will enter the decommissioning step of the lifecycle. *OnApplicationQuit*, *OnDisable* and *OnDestroy* are three methods used for disabling the script. The OnApplicationQuit method is executed on all GameObjects before the application is quit. OnDisable disables the script, causing the script to stop the update loop and wait until the OnEnabled method is called. OnDestroy is a method used for removing the GameObject and its components from the game.

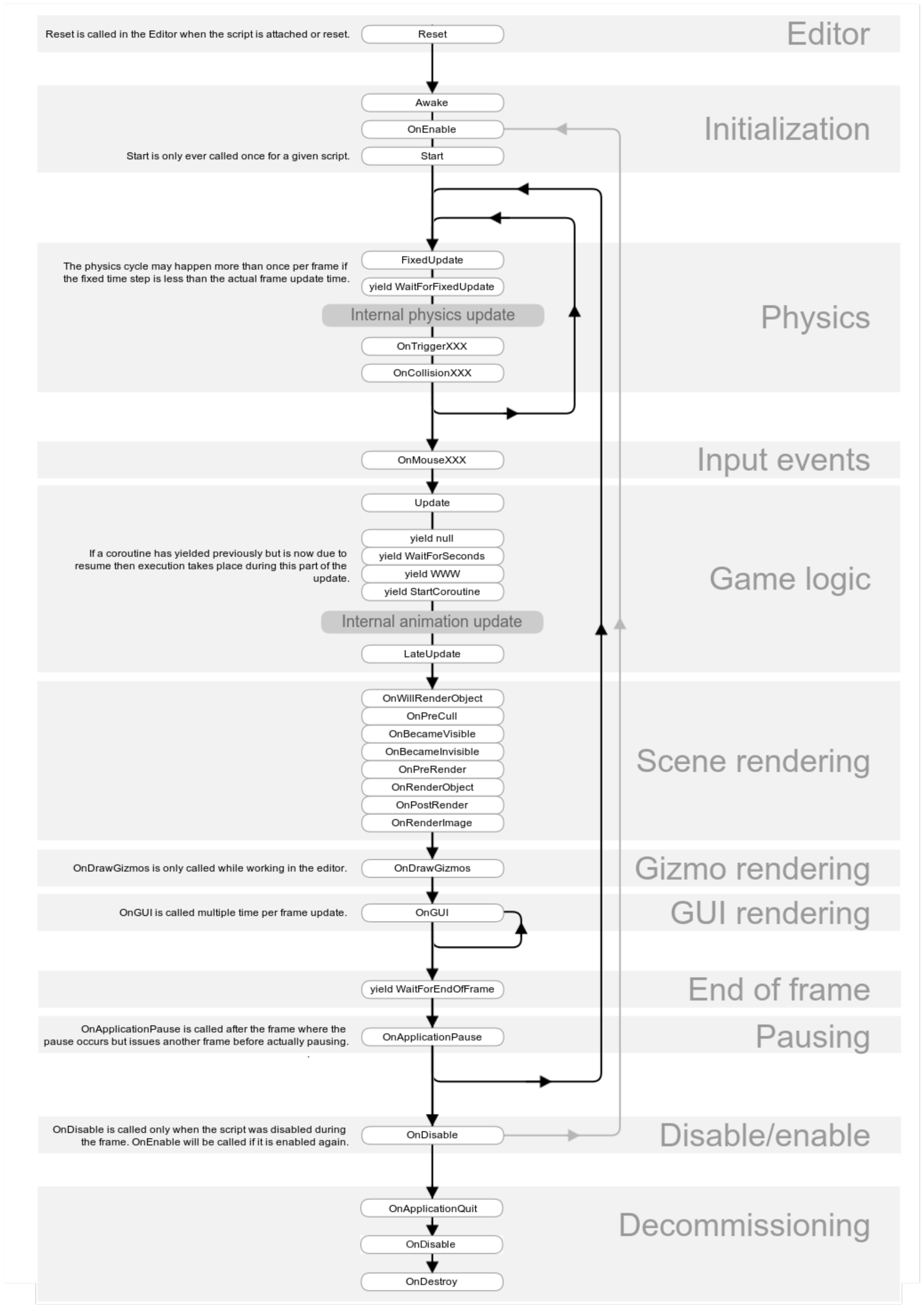

Editor
Reset is called in the Editor when the script is attached or reset.
Reset
Initialization
Awake
OnEnable
Start is only ever called once for a given script.
Start
Physics
The physics cycle may happen more than once per frame if the fixed time step is less than the actual frame update time.
FixedUpdate
yield WaitForFixedUpdate
Internal physics update
OnTriggerXXX
OnCollisionXXX
Input events
OnMouseXXX
Game logic
Update
If a coroutine has yielded previously but is now due to resume then execution takes place during this part of the update.
yield null
yield WaitForSeconds
yield WWW
yield StartCoroutine
Internal animation update
LateUpdate
Scene rendering
OnWillRenderObject
OnPreCull
OnBecameVisible
OnBecameInvisible
OnPreRender
OnRenderObject
OnPostRender
OnRenderImage
Gizmo rendering
OnDrawGizmos is only called while working in the editor.
OnDrawGizmos
GUI rendering
OnGUI is called multiple time per frame update.
OnGUI
End of frame
yield WaitForEndOfFrame
Pausing
OnApplicationPause is called after the frame where the pause occurs but issues another frame before actually pausing.
OnApplicationPause
Disable/enable
OnDisable is called only when the script was disabled during the frame. OnEnable will be called if it is enabled again.
OnDisable
Decommissioning
OnApplicationQuit
OnDisable
OnDestroy

Figure 5: The lifecycle of Monobehavior source: [1]

Prefabs
When developing GameObjects that are used multiple times in a scene, editing these items can be time consuming. Unity presents a solution to the problem with a tool called *prefab*. Prefabs are blueprints for GameObjects and may be spawned as many times as needed in a scene. All changes done to the prefab will be done to it's instances. It is also possible to override components inherited form the prefab in each of its instance.

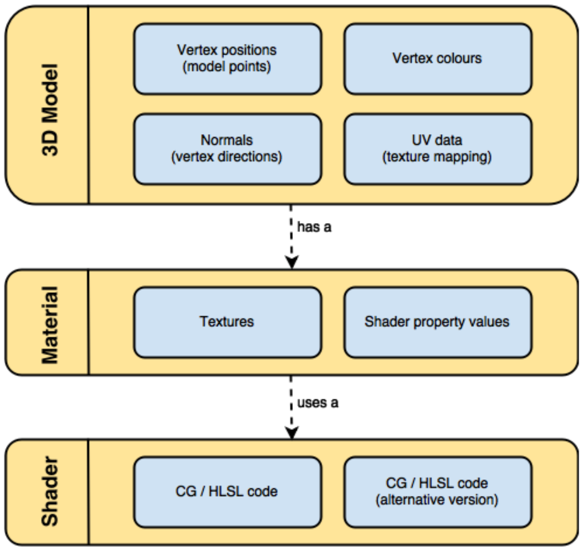

3D Model
Vertex positions (model points)
Vertex colours
Normals (vertex directions)
UV data (texture mapping)
has a
Material
Textures
Shader property values
uses a
Shader
CG / HLSL code
CG / HLSL code (alternative version)

Figure 6: Graphics rendering in Unity, source: [2]

Mesh
Meshes are the main graphics primitive of Unity. Aside from Unity Asset Store plugins, Unity does not include 3D modelling tools. Instead, Unity have made it easy to import and implement 3D models to Unity projects. The imported 3D models are rendered through meshes. Figure 6 shows how the different graphics components are related.

Materials
Materials define the looks of all surfaces in a scene. The material defines everything from the surface structure to its color and how it reflects light.

Shader
Shaders are small computer program used to mathematically calculate the color of each pixel in the scene based on the material configuration and lighting.

2.2.3 Serious games

In an learning appliacation where the user is supposed to play through a scenario, it is useful to learn from serious games what makes this types of applications successful. A game where the main objective is to learn is called serious game. A serious game should contain the common game features making a game fun and addictive in addition to the learning aspect. Bing Gordon argues that traditionally games are defined by story, arts and software [3]. Figure 7 illustrate the requirements for developing a serious game. The development team needs people with expert competence on the field to create the educational part. The design team is responsible for implementing the serious part in collaboration with the field experts.

Thomas W. Malone [44] performed studies on serious games for determining the factors for an educational game to be fun. There are three determining factors for making educational games fun, namely *Challenge*, *Fantasy* and *Curiosity*.

Challenge
In order to make an educational game challenging, it must provide a goal. The more obvious and clear the goal is, the more compelling the game is for the users. The users should also get the feeling of getting closer to achieving the goal.

The game should also have uncertain outcomes. If the outcome of the game is certain beforehand, the users will become tired of playing. Malone lists four ways to make the outcome of the game uncertain:

- Variable difficult level
- Multiple level goals
- Hidden information
- Randomness

Fantasy
In computer games, there are two types of fantasy, *intrinsic* and *extrinsic*. intrinsic fantasy is dependent on the users skill and the skill is also dependent on the fantasy. A dart game where you pop balloons is a good example of intrinsic fantasy. The skill affects the fantasy by throwing the dart towards the balloons and pop them if they hit. The fantasy affect the skill by showing the user how their shot was, if it was either too high or low, and then the user can adjust their next shot based on the feedback from the fantasy. In extrinsic fantasy, the skill is affecting the fantasy, but not the other way round. An example of a game which use extrinsic fantasy is a racing game, where the cars accelerate or decelerate based on the answers on math questions.

Curiosity
Curiosity is a main motivator for learning. Curiosity is divided into two categories, *Sensory curiosity* and *Cognitive curiosity*. Games can affect our sensory curiosity through use of audio and visual effects, and cognitive curiosity can be trigger by limiting the information that is presented to the users in such a way that they will be interested in finding out more.

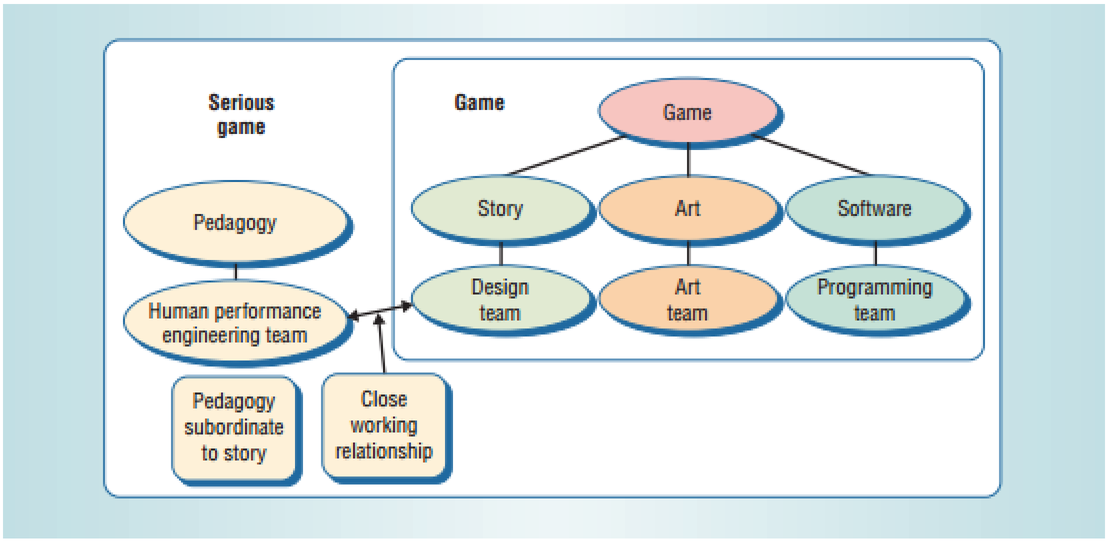

Serious game
Game
Game
Story
Art
Software
Pedagogy
Design team
Art team
Programming team
Human performance engineering team
Pedagogy subordinate to story
Close working relationship

Figure 7: Development structure on serious game, source: [3]

.

Steam VR
To encourage developers to create virtual reality applications, Valve developed a plug-in package called *Steam VR* [45]. The plug-in contains scripts for basic actions such as head tracking and controller input. Valve states that the plug-in is only an example on how the implementation can be done and encourages developers to edit the plug-in to suit their needs. Valve also gives advice on good and bad practices. Disabling head tracking to give a static view and changing the field of view (how far the person can see) are examples of what Valve advice against. They say that these actions will lead to motion sickness due to break with reality.

Virtual Reality Toolkit (VRTK)
Steam VR were criticized for their lack of documentation for their package, and developers raised issues about the usability of it. In response, an open source project called VRTK was initialized by a developer known as *TheStoneFox* [46]. The project contains scripts, prefabs, examples and videos on how to use the package created in the project.

2.3 Virtual Reality Headset/Head Mounted Display (HMD)

This paper explores the possibilities of creating an immersive learning environments. As Klevland discovered, a VR headset is preferable over the usual desktop in order to achieve an immersive and engaging experience [47]. Earlier attempts have been made at constructing a HMD, as the VFX1 Headgear, but the success has limited due to the restrictions on technology at the time [48]. In 2012, a crowdfunding raised enough money for Oculus Rift to become a reality, and from that point on, multiple actors have acquired an interest in the field. Before presenting the different HMDs, some details about HMD needs to be explained.

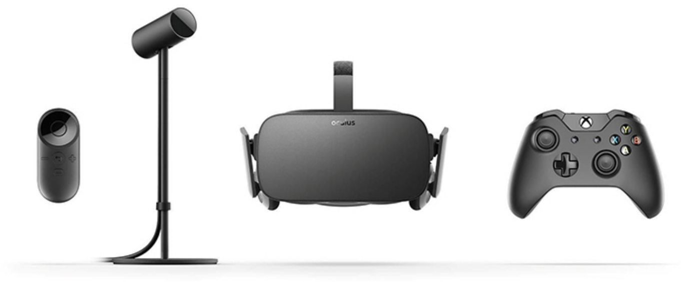
Stereoscopic rendering
Stereoscopy is the art of creating illusion of depth. This illusion is the main strength of HMDs, namely virtual 3D. To achieve this, one has to split the visualization into two different viewports, which represents each eye of the human body.

Render latency
Render latency is the delay between each frame of the visualization. High latency is unpleasant to view, so render latency should be as low as possible. John Carmack studied the effect of render latency and concluded that a latency of 50 milliseconds will feel responsive, but still laggy [49]. He argues that developers should strive to achieve 20 milliseconds or less. High render latency will make the user uncomfortable, which may lead to dizziness or motion sickness. The strain on the eyes with a high latency might also lead to headaches.

Head tracking
A common feature of the HMDs are that they have some way of calculating the position and rotation of the headset. Both Vive and Rift uses accelereometers and a gyroscope to assist the calculation, but their main reference calculation is done quite differently. While the base stations of OR tracks the position of the headset and sends it directly to the computer, Vive uses the basestations for reference points. This means the basestation does not communicate any information to the computer directly, but their placement are used by the headset to calculate its position relative to theirs. One of the biggest drawbacks of cheaper headsets like Google Cardboard and Samsung Gear VR is the lack of the basestations, which makes the position calculation severely limited.

2.3.1 Oculus Rift (OR) with Oculus Touch

Figure 8a illustrates the OR hardware, consisting of a remote controller, a sensor, the HMD and a xbox controller. The OR headset also comes with a mounted music device. The remote controller on the left on the Figure is used to make it easier to navigate in the virtual world.


(a) The OR equipment

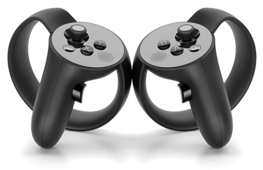

(b) The Oculus Touch

Figure 8: Oculus Rift with Oculus Touch

It is also possible to extend the immersive feeling by buying *Oculus Touch*, Oculus's own controllers. They were developed at a later stage than the headset itself, and can be bought separately. The controllers enables hand tracking, similar to that of the Vive controllers. The controllers are shown on Figure 8b.

OR is easy to transport and setup time is minimal. It is possible to mount it and run within minutes. To run OR applications, a computer with great processing power is needed to support the resolution provided by the headset. The recommended specifications for a computer running OR are:

- NVIDIA GTX 970 / AMD Radeon R9 290 equivalent or greater
- Intel i5-4590 equivalent or greater
- 8GB+ RAM
- Compatible HDMI 1.3 video output
- 2x USB 3.0 ports
- Windows 7 SP1 or newer

2.3.2 HTC Vive (Vive)

After OR received online appreciation and attention, Steam developed and displayed prototypes of a similar headset. They later paired with HTC to produce the headset [50]. There are two main differences between Vive and OR: the room setup and the controllers. Figure 9 shows the Vive equipment. It consist of two cameras to track user movement, a HMD and two game controllers to enable user interaction.
In order to play, one needs to define a space that is at least 1.5 meters times 2 meters. This is called the play area. The user can move physically in the dedicated play area. If the user moves towards an edge on the play area, they will receive a warning in the form of virtual walls. The movement implementation works well for tasks that requires limited movement, such as a shooter game where users shoots and ducks behind cover. Normal practice for movement that exceeds the play area range is teleportation. Teleportation is usually implemented so that users can use the controllers and point and click where they want to travel.

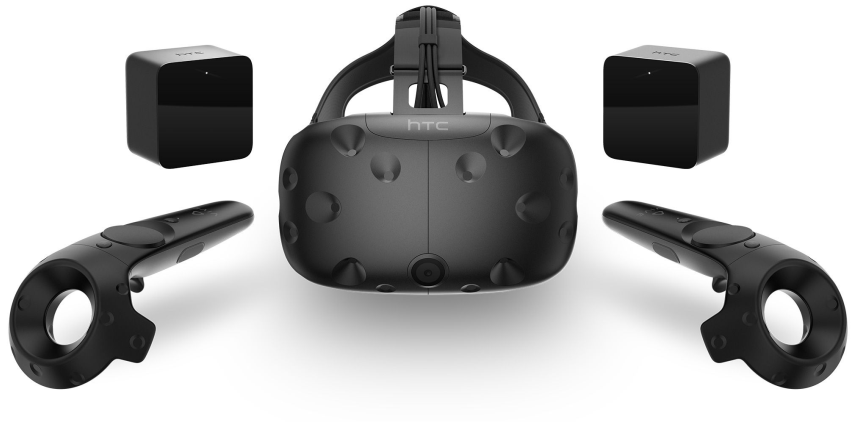


Figure 9: The HTC Vive equipment

Vive significantly differ from OR is the room setup. It takes effort to correctly set up the base stations. As explained in Appendix B, the base stations needs to be placed diagonally and in a high place to track the play area properly. The base stations needs to be synced with each other, and the headset needs to be connected to the computer. The computer requirements to run HTC Vive is considerable. The recommended specifications for Vive are:

- NVIDIA GeForce^{TM} GTX 1060 or AMD Radeon^{TM} RX 480, equivalent or better
- Intel i5-4590 equivalent or greater
- 4 GB+ RAM
- 1x HDMI 1.4 port, or DisplayPort 1.2 or newer
- 1x USB 2.0 port or newer
- Windows 7 SP1 or newer

2.3.3 Samsung Gear VR

The Samsung Gear VR is a mobile virtual reality headset. It works by inserting a compatible Samsung smartphone into the headset and show images through it. The headset is made in collaboration with the founders of OR.

The reason these glasses is worth mentioning is the advantage of great portability and availability these kinds of systems offer compared to OR and Vive. Anyone with a compatible Samsung phone may buy these glasses cheaply when compared to the competing hardware. The portability aspect of the glasses is backed by the fact that users do not need a high-end computer to run applications on the glasses, the smartphone itself is sufficient. This opens for uses not easily achieved by the other systems discussed, such as presenting virtual representations of illnesses to patients or learning games to children in hospitals. The lack of processing power comes with the obvious cost of reduced framerate, resolution, and complexity of the applications.

2.3.4 Google Cardboard

A cheaper version of a VR HMD is Google's cardboard. The physical gear needed to use this system is exactly what it sounds like, a cardboard frame you can put your device in and mount on your head. The rest of the system lies within the application *Cardboard* which may be acquired in the *Google Play Store*. The application takes care of splitting the view and managing headtracking based on the phones accelerometer.

The main reason to buy one of these systems compared to the others mentioned is the price. If you currently own a sufficiently new Android based smartphone, the cost of transforming it to a VR-HMD becomes just short of NOK 100. Compared to the other headsets mentioned, Cardboard falls short in terms of heat management, interaction, build quality and processing capabilities.

2.3.5 VR - sickness

As presented in an earlier study [47], participants of virtual reality experiments often experience physical discomfort when using head mounted displays. The symptoms users experience is parallel to the symptoms of classic motion sickness, but with the key difference of the user being stationary.

A relevant theory relating to cybersickness is the "Sensory Conflict Theory" [51]. The theory is based on the premise that discrepancies between the senses which provides information about the body's orientation and motion cause a perceptual conflict which the body does not know how to handle. A number of other factors that are not directly related to this theory and are more connected to technological flaws. These factors include low resolution, high latency and screen flickering.

There have been performed research into ways to lessen the discomfort while playing in VR. La Viola [51] considers triggering the nerves, either trough platforms or through electrical stimulation, making them believe there is a correlation between sensations experienced by the eyes and the rest of the body. Later research have looked into using a fake nose in the field of view of the application, which have had some effect without disturbing the experience [52]. A later research considers limiting the field of view in the VR-application and have proven the method to be efficient in some applications [53]. In 2015, the inventors of OR published a blogpost with a list of 11 methods they experienced lowered the discomfort when using HMDs. The methods varies from using abrupt 90 degree turns to closing ones eyes when body movement is not user controlled [54].

2.4 Thesis goals and objectives

The use of simulators in medical education has shown potential. NeuroTouch has given the users the ability to use real world medical data to prepare for and perform surgery.

However, according to Stefanidis et al. [21], the success of simulators have been limited due difficulty of complex, customized equipment needed to use the simulators. The equipment is costly and requires a vast amount of time and resources to develop. The application created by Buń et al. [22] uses general purpose VR device such as Vive and OR, but is also reliant on customized hardware such as the haptic device. The first research question in this thesis therefor asks whether or not it is possible to create a procedural training application using only general purpose VR equipment like the Vive and OR with accompanying controllers. An advantage of only relying on general purpose equipment is the ability to create several applications for the same device, resulting in reduced cost for developing applications in addition to space required for the hardware. Furthermore, the users of the application does only need to learn how to use one type of hardware, instead of different types used in simulators.

There are already been developed multiple learning applications to support communication, and interactive visualization training for general puprose VR equipment [32] [34] [35] [31]. However, there have not yet been many learning applications developed for procedural training using general purpose VR equipment. The ability to track hand and head movements as well as rotation, makes it possible to create an immersive, educational application where the user can perform procedure training with the use of hands and give tactile feedback on actions.

The use of medical data in learning applications will further increase the educational benefit gained, by letting the user know that they are working on real data. Rudolf et al. assessed the value of real world medical data in their VR application [55], concluding that although hard to implement, real world medical data could have immense benefit in a VR learning application. The first research question therefore considers how such data could be implemented in virtual reality training, and how users should interact with it.

With the abovementioned state of literature, and given the information gained in the background chapter, the specified research questions were defined as follows:

RQ 1: *Can real world medical imaging data be utilized to support training in virtual reality.*
RQ 2: *How do users experience using an application with general purpose virtual reality equipment for Medical Procedural Training*

3 Equipment, Methods and Implementation

The implementation of well designed educational applications in virtual reality for usage with a HMD requires developers to be able to understand and utilize a plethora of tools, frameworks, and evaluation methods. In this chapter, these tools, frameworks and methods will be presented in detail as well as the implementation of the application.

3.1 Equipment, tools and frameworks

Considering vast amount of readily available equipment, tools and frameworks available for developers in any project, a proper conscious process to select the ones that would fit the thesis was required. In this section, the reader is introduced to the process of selecting equipment, tools and frameworks used in the project.

3.1.1 Head mounted displays

In this project, multiple HMDs have been considered as targeted platforms. To answer the research questions defined in the introduction section, it was decided to work with HMDs supporting devices for hand as well as head tracking. The ability to track the hands in a virtual environments opens the ability for the developers to develop a more immerse and interactive application. In the specialization project, both OR and Vive was examined to determine the best suitable hardware for implementing the application. The Vive was considered to be a superior device due to OR's lack of hand tracking. However, during early phases of the this project, Oculus Touch [56] was released. Oculus Touch is Oculus's own hardware for tracking hand movement. The introduction of the Oculus Touch made it possible to develop an application compatible for both the Vive and the OR. Due to the limitations in interaction compared to Vive and OR, Samsung Gear VR and Google Cardboard have not been targeted as development platforms in this thesis.

3.1.2 Game engines

During the specialization project, research on what game engine would be most beneficial for development of a VR application was conducted. Unity, Unreal and CryEngine was studied, all three being well established software and providing the necessary tools for answering the research questions. In the end, Unity was chosen based on the excellent documentation, the vast amount of user created content and the fact that C# [57] is the main programming language used for development, which is known by both writers of this thesis.

3.1.3 Tools and framework for virtual reality development

In addition to a game engine, further tools were needed for image handling, constructing and editing 3D models, developing scripts and version control tools.

Construction and manipulation of 3D models
Unity does not support creation of 3D models in its editor. To compensate for the

lack of a model editor, Unity have sophisticated software allowing importation of 3D models from external sources in a variety of file formats. Unity supports manipulation of 3D models by changing the rotation and scale values. If more advanced manipulation is needed, such as addition and removal of vertices, nodes and meshes, a more sophisticated tool was required. For this tasks Blender [58] and 3ds Max [59] were used.

Scripting and programming tools
Unity itself does not contain a way to manipulate scripts within its editor, so a 3rd party editor is needed. Most text-editors will have the necessary tools to edit the scripts, but two editors have implemented extensions to work particularly well with Unity, namely *MonoDevelop* [60] and *Visual studio 2017* [61]. Both of these tools ships extensions for code completion, auto compilation and proper error handling for Unity, which made them mandatory for efficient development.

Photo editing software
Many of the meshes on the different panels, buttons and figures needed to be manipulated, or created from scratch. Unity does not offer a photo editing software, so a third party software had to be selected for this task as well. The tool used in this project was a freeware photo editor named *Paint.net [62]*, which offer superior simplicity given its strong coverage of necessary functionality.

3.2 Methods

In this section, the reader is introduced to the details of the scenarios that was implemented and the methods for gathering feedback from the user tests.

3.2.1 Scenarios

In order to ease the development and ensure the quality of educational applications, it is important to have a clear understanding of the high level goals. Scenarios was implemented to supply these goals. One was a gynecology scenario implemented for the earlier iteration of the VUH, while the second was a neurology scenario developed with feedback from the department of neurology at st. Olavs hospital. Both scenarios focuses on giving users procedural training tasks related to pre surgery.

Neurology
In the neurology scenario, the users are playing out some of the first steps performed during neurosurgical removal of a brain tumor. The scenario is split into three tasks:

1: Pre-operative planning
The students are expected to use a panel consisting of the axial, coronal and sagittal view of MRI from a patient to locate the tumor and mark the middle of it (target). When the user have marked the patient, they click the solution button to compare with the correct solution.

2: Patient positioning, shaving and marking the entry point of the craniotomy
The patient is to be positioned correctly on the surgical bed, and the head of the patient needs to be correctly rotated to ensure proper access to the tumor and optimal conditions for ultrasound acquisition. When the patient is laid correctly, the patient needs to be shaved in order to get access to the point of operation. When the patient is shaved, the user marks the point of surgical entry. After marking the point of entry, the user clicks on the solution button to get the correct setup.

3: Navigation and ultrasound acquisition
When the point of entry is marked, the student is expected to confirm its position by using the navigation tools to locate the actual position of the tumor. When the position is correct, The last task is to gather ultrasound imagery in order to get a better view of the tissue above the tumor.

Gynecology
The gynecology scenario focuses on interaction between members in a medical team. It is divided into three tasks.

1: Meet and talk with the patient
The user enters the room where the patient and her husband is located. The user asks a couple of questions to the patient to help diagnosing the patient.

2: Move the patient to the ultrasound machine
The user explains to the patient that an ultrasound is needed to confirm the diagnosis, and asks the patient to follow into a room where the ultrasound should be conducted.

3: Perform ultrasound
An ultrasound examination is performed to further help diagnosing the patient. The user needs to guide the user from the patient's room to the ultrasound room, and then perform the ultrasound examination when they reach the room. The users should be able to detect abnormalities in the ultrasound, which they should conclude that surgery have to take place.

3.2.2 Evaluation

In pursuance of acquiring valuable and relevant feedback on the application made in this project, several tests were constructed. In order to get the most out of the tests, they had to be carefully planned and executed.

Test setup
The setup itself presented two challenges. The first one being the complexity of the VR gear. For Vive, the mounting proved too complex to do efficiently and the testing on it was therefore limited to already assigned rooms with the gear was pre-mounted. OR gave more flexibility, as the base stations did not need to be mounted at height but rather stands in front of the user. Compared to the Vive, the OR hardware offered

movement in a manageable timeframe, and were therefore chosen as the go to device for the test setup outside the lab. Another challenge for testing the application was the need for sufficient high-performing hardware and direct connection between the headset and the computer's graphical processing unit. The requirements excludes most laptops, which practically means a powerful desktop computer is needed to run the tests. The software requirements for running the application is found in Appendix B.2.

Test subjects
To get the most relevant feedback, it is important that the group of testers are potential users of the system. In this thesis, the testers have consisted of one, or both, of two groups; namely trained medicals and technology experts. The trained medicals were consulted for assessing the training value of the application, as well as helping to review the user interface and interactions for non-technological savvy users. The technology experts were consulted to give additional constructive feedback for the application and could give feedback regarding best practices and further improving user interaction.

Test execution
When executing tests in virtual reality, it is imperative that the test subjects get proper time to familiarize themselves with the gear. Explaining how the head-mounted display works, as well as explaining the controllers and the buttons that lies on them are crucial to make the tests efficient. Furthermore, to not get frustrated or distracted by the sensation of VR, a supervisor is present at the test to tell the user where to go and what buttons to press to interact with the system.

Feedback
The feedback acquired in these test were split into two parts, quantitative feedback through questioners, found in Appendix E and qualitative feedback through observations and post testing interviews. The interviews have mostly been used to improve the application while the quantitative feedback have been used to assess the value of the application in relevance to the research questions.

3.3 Implementation

In order to get a better understanding on how the different part of the final scenarios were implemented in VR, a high level description will follow. Note that there will be a minimal amount of source code in this section, but it is readily available for reading at the Github repository [63].

3.3.1 Changes based on specialization project

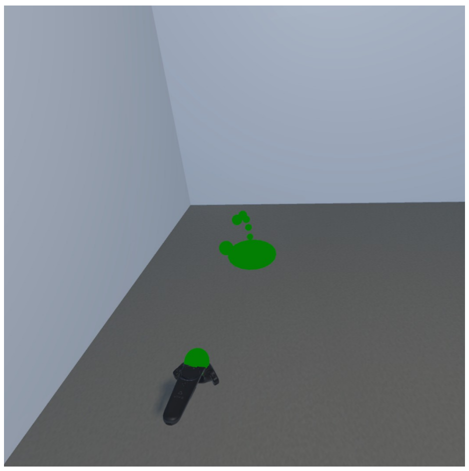
Based on knowledge and feedback gathered from the specialization project, the team decided to make changes for the new project. The changes concerns both the user interaction, visual presentation of the virtual world and architectural decisions. The different choices are explained in the following subsection.

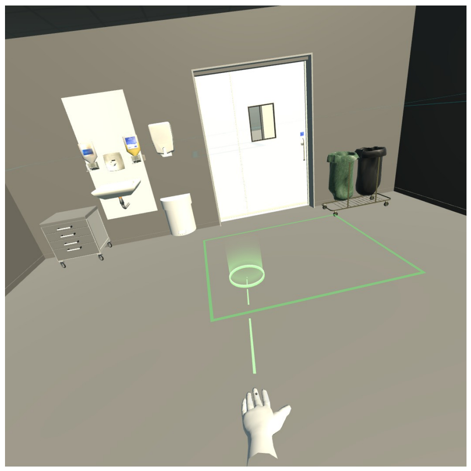
User interaction and virtual world
Insights gathered from the specialization project showed that the more buttons the users could use on the controllers, the more confused they were.

To reduce the confusion, and increase consistency, which again increases the learning curve of new users, the mapping on the controller changed. The sliding movement scheme, interaction laser and room change functions used in the specialization project were completely removed. The controllers' trackpad only have one functionality, namely teleporting. The removal of the other functionalities made it possible to remove the interaction with the menu button, leading to a decrease in number of buttons the user needs to learn and keep track of in game.

To account for the changes in controller mapping, changes also had to be made in the virtual world. Instead of having several scenes with small rooms, the decision to have fewer scenes with more content was made. This made it possible to constrain the user from moving between the scenes by mistake being able to control the sequence of events the user faces.

SteamVR
In addition to providing helpful scripts for handling VR, SteamVR also included materials, shaders and textures used to create The Lab [64], Valve's own VR game. These assets were used to further enhance the VR experience. Figure 10a and Figure 10b illustrates the difference between VRTK teleportation assets and SteamVR. The new teleportation is more pleasing to look at than the old one. In addition, the teleportation beam is an animation going from the controller, to the destination, giving the impression of moving towards the targeted area. Another benefit of the new teleport is the ability to see the play area (the allocated space you are allowed to move in the real world) which makes it easier for the user to keep track of where in the play area they are located. The visualization of the play area also reduce the possibility of teleporting into a wall, which was experienced in the specialization project.


(a) Old teleportation


(b) New teleportation

3.3.2 Constructing the environment

Multiple different designs and scene layouts were explored during development. The final design was inspired by the VirSam project, as it closely simulates the operation hall at St. Olavs, and could lead to greater sence of immersiveness for the users. In this subsection, the implementation of the scene, in addition to the placement of interactable and static objects will be described.

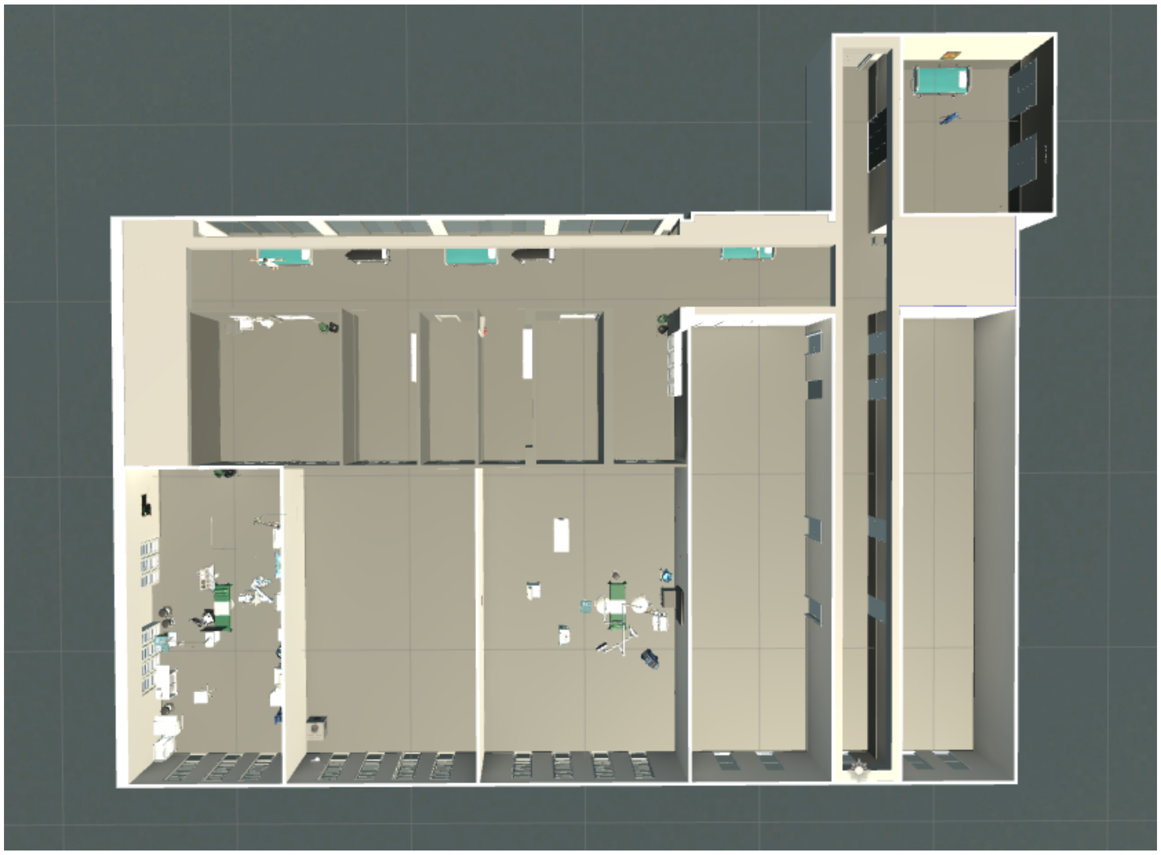


Figure 11: Layout of the hospital

Hallway and rooms
The models for the hallway and rooms were gathered from the designer of the VirSam project. They were modelled from pictures of a operating hall at the Eastern Nevrologial centre at St.Olavs, making it ideal for us to create a familiar environment for neurogical surgeons. The entire model had to be rescaled to an appropriate value for VR in order to work.

The hallways, doors, windows, and equipment had to be placed in accordance with the real room. The hospital modelled in SL was diligently used to inspire the placement of the models.

Operation rooms
Operation room one, the room furthest to the right with equipment in it, viewed

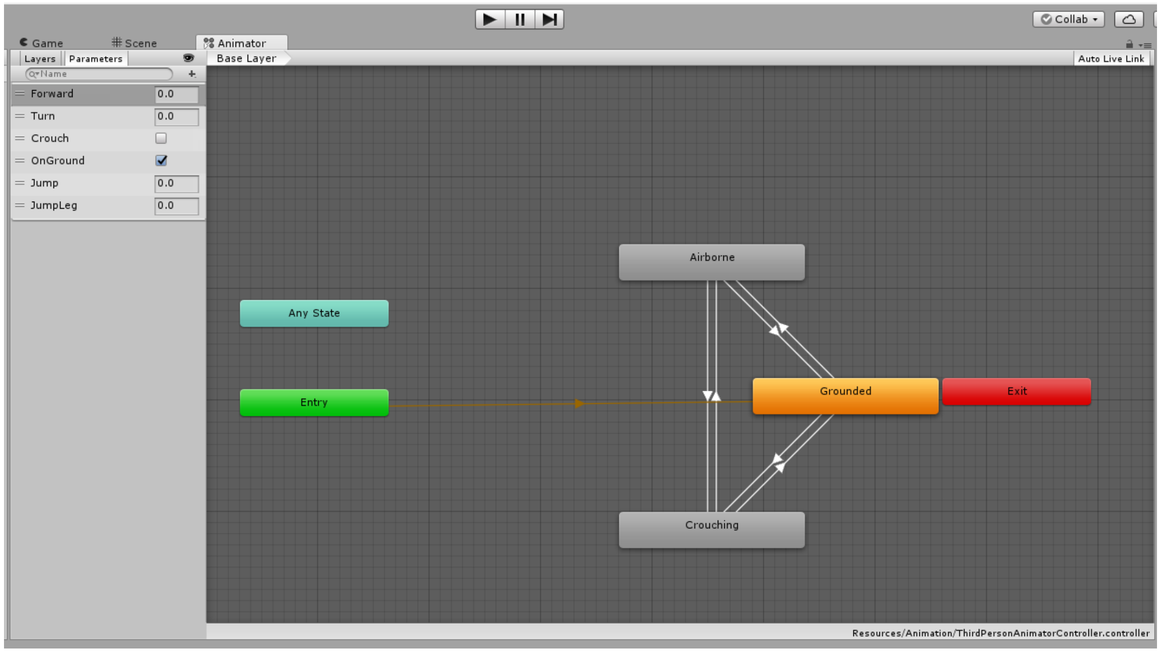
in Figure 11 was designed to be used for the operation part of the gynecology case. Most of the models are from the VirSam project, making it look very similar to the operation room in SL. The room is used to show the ability of creating a operational scenario where the surgical nurse gives the surgeon the correct equipment at the right time.

Operation room two is used to store proof of concept models which is either not relevant to the scenario or too cumbersome for non-technical personnel to use. The GameObjects are not rendering until the user presses the button outside the room. The models contained in the room is resource demanding, so they are not rendering until the button outside the room is pressed to improve in game performance, resulting in less latency which again reduces the potential for VR sickness.

Operation room three is used to perform the neurosurgery scenario. All GameObjects inside the room is interactable, making it possible to move the furniture around the room to create a surgical environment.

Animation
Animating human figures is a common task in game development, and Unity smartly provides developers with a sophisticated system called *Mecanim*, a framework for animating rigged models, which means models containing a certain set of interlinked joints and bones. Developers may create one animation and reuse it on other models that fit the humanoid rigging criteria. Most of the models coming from the asset store were already optimized humanoid animations, and left us with the task of defining the models as optimized for *Mecanim* animations and add the required animation to them.

The task of controlling and displaying an animation can be divided into two separate tasks. The first is choosing what animation is to be played, the second is to actually play the clip. In order to handle this, Unity provides the tools animator controller and animation clip. The animator controller is edited through its own editorial view, where it is possible to graphically set up states and variables that swaps the states. Figure 12 shows the animator controller view. The boxes in the view is a representation of the different states in the controller. The green box is the initial state of the controller. The orange box indicates the default state, in other words the most active state. The arrows between the boxes indicates the possibility of moving from one state to another. By clicking on the arrows, it is possible to edit the transaction time between the two states in addition to what conditions must be met for the transaction to be possible. For instance, for the transaction between the state *Grounded* to *Airborne* to be fulfilled, the variable *OnGround* must be false. The variables are changed in scripts attached on the animating GameObject.

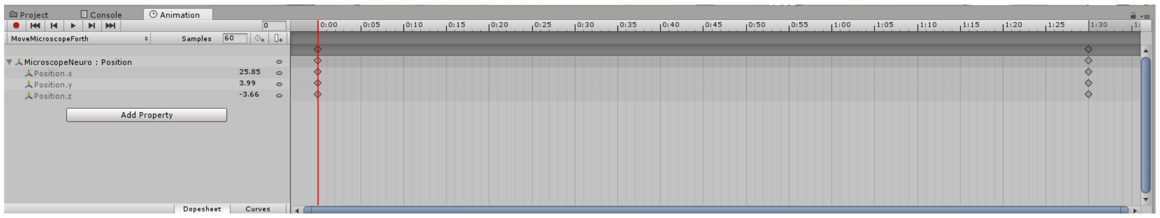


Forward	0.0
Turn	0.0
Crouch	☐
OnGround	✓
Jump	0.0
JumpLeg	0.0

Figure 12: An illustration of the animator view in Unity

Animation clips are the smallest building blocks of animation in Unity. Each animation clip represents isolated pieces of motion, such as run, walk, and jump. These clips are handled by a state machine such as the animator controller to present lively animation at the appropriate times. In Unity, it is possible to both import animation clips from external sources or create them in Unity's editor. Unity has a dedicated view for creating animation clips called *Animation View*. When a GameObject is selected in the scene, its animation clip is shown in the animation view. If the GameObject does not have an animation clip yet, Unity asks if the user wants to create a new animation clip.

Figure 13 shows the animation view when the microscope GameObject is selected. The developer can select which properties to be changed by clicking on the *Add Property* button. In this example, the position of the microscope is added to the property list. The timeline illustrates the duration of the clip. As seen on the Figure, the clip is 1 minute and 30 seconds long. The diamond shaped markings on the timeline represents key frames. The user can edit the values of the properties at the key frames. In this example, there are only two key frames for each property, namely start and end position. The GameObject will move between the start and end position with constant speed during the clip. The red line illustrates where the GameObject is in the animation clip. It is possible to drag the red line between the frames to inspect how the GameObject moves frame by frame. When the animation clip is completed, the play button can be pressed to see the result.



MicroscopeNeuro : Position
Position.x 25.85
Position.y 3.99
Position.z -3.66

Figure 13: Presentation of the animation view in Unity

If the GameObject is suppose to move in a predictable manner, it is beneficial in terms of performance to create animation clips instead of applying translation on GameObjects at runtime. This allows one to set up complex movement by planning their exact movement in a *what you see is what you get* - manner in contrast to have to estimate them mathematically in scripts.

Non playable characters
In order to make the hospital environment more immersive and life like, a multitude of non-playable characters were put into the environment and forced to behave in manners that are expected from their character type.

Patients
In the hallways outside the operation rooms there were placed patients in beds who were animated uniquely. One patient writhes, clearly in significant pain, while another is shifting his laid-down position in timed intervals.

Surgical nurses
Two surgical nurses are also roaming in the hallways. They move randomly over great enough distances to simulate conscious movement. The script *HallwayAiController* was used to implement the movement. The script takes six locations as input and the method *FindRandomTargets* decides which point the nurse should move to next. To calculate the movement paths, Unity's integrated navigation mesh and agent functionality was used. The mesh is a collection of all points in the scene that navigation agents are allowed to move upon, and the agent uses the A* algorithm [65] to calculate the next point in a least cost path towards the target location. In order to make it even more realistic, some points were placed inside rooms that are not used in the case itself.

Models
Most models used in the scenarios was contained in packages bought from the Unity Asset store, namely *Operating Room* [66] and *Hospital Room* [67]. The rest of the models were gathered from the VirSam project and online. If models needed refinements, or were not found, the modelling tools Blender and 3ds Max was used for creation and manipulation of models.

3.3.3 User interaction

The application supports two types of playing modes. The first mode, VR mode, is played by using either Oculus Rift or HTC Vive. The second mode, desktop mode, is played in standard desktop mode. The user selects mode in the main menu of the application.

The camera on the desktop user is set in third person, making it possible for users to see their avatar when playing. The avatar is a surgeon, reused from the surgery room package purchased. The user moves by using the *wasd*- or the arrow keys. It is possible to increase the movement speed of the avatar by holding down the shift key. The avatar also have animation to simulate movement. The hospital room package purchased contained an animation controller, which corresponds to standard Unity humanoid avatars. It is possible to make humanoid 3D models into standard Unity avatars, making it possible to reuse the animation controllers on other models. The doors in the hospital opens whenever the users get close to the door, enabling the users to move freely around the hospital.

The VR user model consists of a head and two hands. The users are only able to see the head of other users, and not their own. The hands are used to increase the user's sense of presence in the world. There are four animation clips attached to the hands that responds on different button clicks. The camera on the head of the VR user is controlled by the *Player* script included in the Steam VR package. The script automatically sets the field of view on the camera to correspond to the field of view that is available in the HMD currently in use (usually 110). Although sufficient for VR, a field of view of 110 is too large for a regular computer monitor. This was a problem when testers and developers wanted to spectate users, as parts of the VR users view was not shown on the monitor used for spectating. The nature of the application made it important to allow spectating, because it makes it easier for the instructors to help the users perform the required actions to complete the tasks in the cases. Additionally, it can be used to give users a quick preview of what is required to complete the tasks. To remedy the field of view issue, an extra camera was added to the application. The camera was mounted with the script *CopyHead* which copies the position and rotation of the head. The camera works independently of the Steam VR application, meaning the field of view could be set manually. Tests conducted revealed that the field of view of 60 is sufficient to give the spectators the same view as the VR user. To make the desktop camera to display on the monitor and the VR camera to display on the VR monitor, the attribute *Depth* had to be edited on the cameras. The VR device displays the camera with the lowest depth number, and the monitor displays the other. The VR camera was given the value -1 and the monitor camera the value 1.

Interactable
The *Interactable* script is responsible for alerting other scripts and other GameObjects whenever an interactable GameObject is within the controllers reach.

Grip button
The grip button was utelized for picking up items in the application. The script *Move Items* is added as a component to all GameObjects that the user should be able to pick up, and contains methods for receiving the notifications from the *Interactable* script.

The methods *OnHoverBegin* and *OnHovedUpdate* is executed when the controller is within range. These two methods checks every update cycle if the user presses the grip button. In addition, it highlights the grip button on the controller touching the GameObject, giving the user a clue of how to interact with the GameObject. If the grip button is pressed, the *OnAttachedToHand* method is executed, which tells the system that this item is picked up by this user with the controller where the button was pressed. A notification is sent to the *Networked Items* script to alert other users that the owner of the GameObject have changed. *OnDetachFromHand* is executed when an item is dropped. *Networked Items* is again notified about the change. An example of a GameObject using this script is the marker pen used for marking the operation corridor.

Trigger button
When the user needs to interact with GameObjects in any other way than picking them up, the trigger button was used. It is difficult to create a generalized script to take care of all usage of the trigger button, since the interaction with the environment differ from GameObject to GameObject. Smaller scripts customized to each GameObject was developed instead. An example of a unique case is the marker pen, which had a script developed called *Draw With Pen*. The script checks whether or not the user is holding the pen, and if the user is holding it, the script checks if the user is pressing the trigger button. When both these conditions are met, the pen spews ink when moved. Another example is the interactions with panel buttons. All buttons on the panel uses the same script, *Game Panel Buttons* which looks for user interaction. The script has the two methods *OnHoverBegin* and *OnHovedUpdate* to receive notifications from the *Interactable* script when a controller is in contact with the GameObject. When the methods is called, they look for the trigger button to be pressed. If the trigger button is pressed, the script will determine which button was pressed, based on the name of the GameObject.

Game Panel
The game panel was created to have all the necessary equipment needed to perform task number three and four in the neurosurgery scenario easily reachable for the user. If the user is not satisfied with its position in the room, the user can move the panel by placing a hand to the base of the panel and press the grip button. When the user is satisfied with the location, they release the grip button. The panel will automatically settle on the floor when the user releases the grip button. Figure 14 gives an illustration of the buttons on the game panel. The buttons are designed with Gestalts principle in mind. Gestalts principle of proximity is used to convince the user that there are four groups of functionality. The spacing between the buttons give an indication of grouping. The principle of similarity is used to further convince the user of the intended grouping. The torus buttons to the top left have the same shape and background color. Similarly, the buttons *Reset Patient* and *Reset Ink* both have the same white background in addition to the black border. The buttons for controlling patient position all have the same shape, which is distinctly different from the other buttons. The *Solution* button, located in the top left also differentiate itself from the other buttons by its shape. It is significant smaller than the other buttons, and have a different background color than its neighbors.

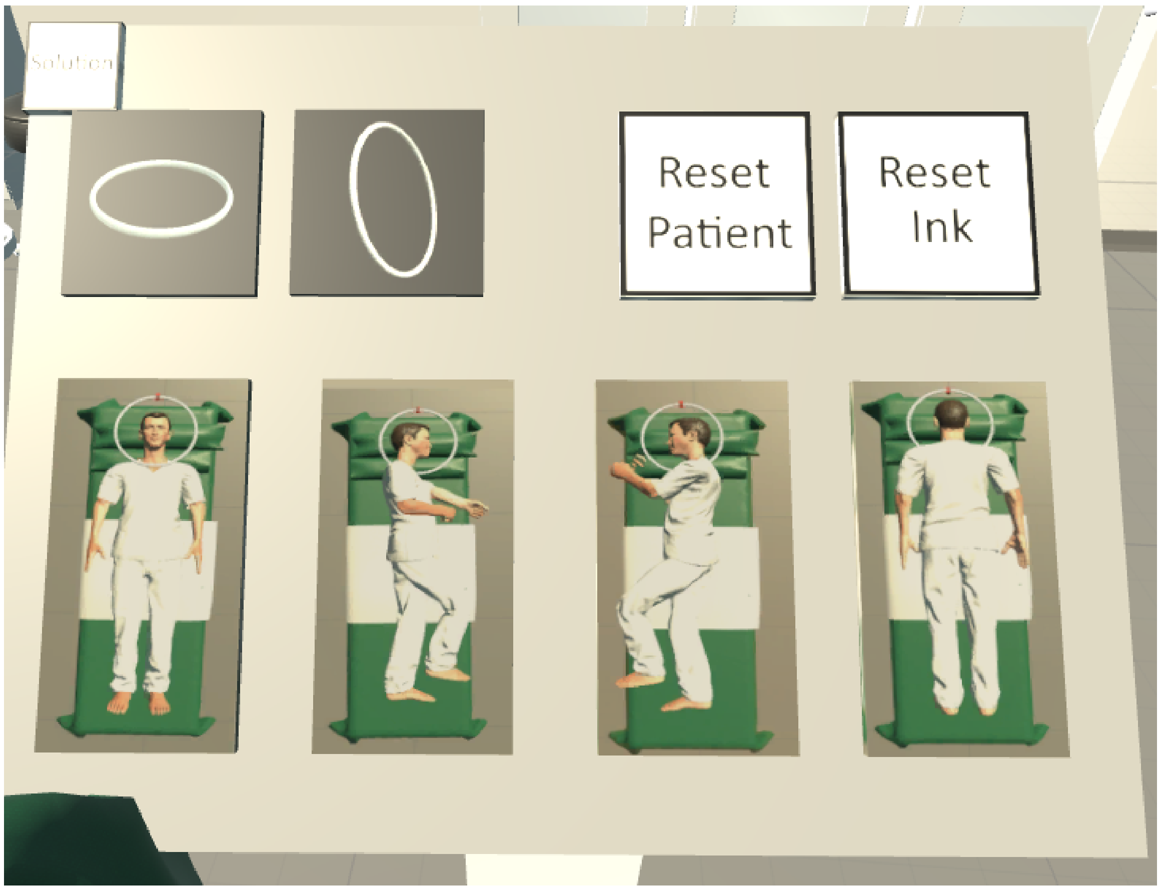


Figure 14: Illustration of the game panel

The buttons on the panel have a script *Game Panel Buttons* for handling the interaction with the user. The user presses the trigger button to interact with the button. The four buttons on the bottom of the panel takes care of patient positioning. Each button represents a position the patient can be placed in. When a button is clicked, the patient will spawn on the operation table. A clicking sound will be played to indicate to the user that a button has been interacted with.

At the top left on the panel, are placed two buttons with torus images. The first one represents a horizontal torus and the latter vertical torus. If the user presses the horizontal torus button, a horizontal torus will be spawned over the patient's head on the operation table. The script *Head Controller* attached on the Torus, gives the user the ability to move the patient's head. To rotate the head, the user moves a hand to the red indicator on the torus and press the trigger button. The head rotator script sends a message to the script *HeadRotator*, located on the patient's head to tell it to move identical to the torus.

3.3.4 Real world medical imaging

The integration of real world medical data have the ability to help developers create a more realistic and immersive application. In the gynecology scenario in the VirSam project, the physician would look at a frozen screen and vocally declare the state of the ultrasound. If the users could perform the ultrasound procedure themselves, the application would feel much more realistic. It was therefore decided to explore the possibility of creating mixed reality. Mixed reality is defined by a mixture of real data combined with virtual ones. In this case, the user is in a virtual world, but the medical data presented to the user is taken from the real world. Two methods were implemented to create the mixed reality environment. The first method was to create an ultrasound machine where the user could perform ultrasound on a patient. When the user moved the probe, the image on the ultrasound machine would update accordingly. The second method was to load and render raw medical data processed through the *Volume Viewer* asset purchased.

Ultrasound

The ultrasound machine works by mapping positions of the ultrasound (US) probe in the application to real ultrasound images. The ultrasound images are frames extracted from an ultrasound examination video. A snapshot of the video clip is illustrated in Figure 15.

FFMPEG was used for extracting frames from the video. FFMPEG [68] is a tool for doing a vast variety of imaging and video manipulations. The program is used from the command line, and the syntax for extracting frames is as follows:

```
ffmpeg -i <input file> -ss <start_time> -t <duration> <file_output_regex>
```

The command used for extracting the necessary frames was as follows:

```
ffmpeg -i vid.mp4 -ss 00:00:11.000 -t 00:00:08.000 img_us%3d.jpg
```

The last part of the file name, *%3d*, is a regular expression that defines the last part of each frame extracted to be enumerated by 3 digits on the tail. The output images will be named, **img_us001**, **img_us002** and so on.

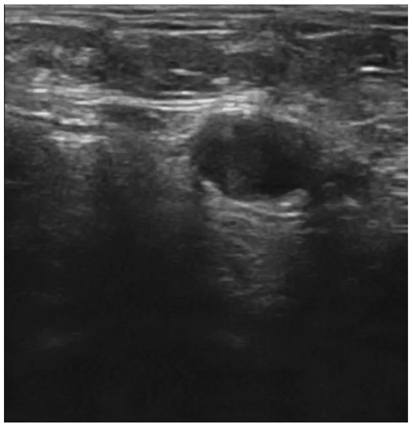


Figure 15: Snapshot from ultrasound video

Using video data
In order to present the correct data, movement of the probe is restricted in one direction and follows a path constructed from a Bezier curve. Once the ultrasound starts, a script attached to the probe makes sure that it follows the Bezier curve every time the Update() function is called. Implementation of the Bezier curve was handled by the external library "BansheeGZ". Their GameObject, "BGCurve", allows for modelling the curve in 3D space.

The script *DragDrop* determines when the ultrasound probe is in the correct position to start the machine. Once it is in the correct place, a message is sent to the *Image-Handler* script to start calculating which image to show. Picking the correct image was done by splitting the Bezier-curve in 80 parts in one direction of a plane orthogonally placed on the curve. The number 80 comes from the fact that it was exactly 80 pictures extraced from the video, but the formula should work for any number of pictures. The formula for determining the number of the current picture can be defined as follows:

$$img\_num = \frac{N_{pics} * (x_p - x_0)}{x_1 - x_0} \tag{1}$$

Where img_num is the current image number, N_{pics} is to total number of pictures, x_p is the current x-position of the probe, x_0 is the start position of the bezier-curve and x_1 is the end position.

Figure 16 shows the ultrasound machine used in the project. The screen provided with the machine was too small to make out subtellties in the image, so a larger screen had to be constructed. A *quad* was chosen as a GameObject for presenting the images, given its format and low performance impact on the world. The components Canvas, RawImage and ImageHandler was added on the GameObject to make the image visible. The RawImage component handles the rendering of the mesh, while Canvas is a support class that simply allows RawImage to be rendered for the cameras in the game world.

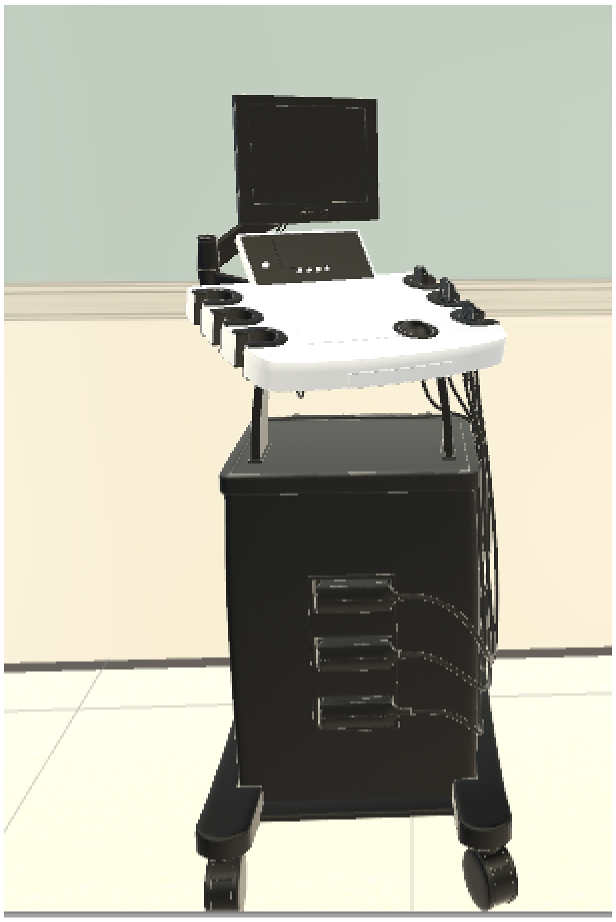


Figure 16: The ultrasound machine used in the application

.

When the correct image number is calculated, the image to the corresponding number is retrieved and placed on the RawImage to be shown on the quad. Figure 17 illustrates the workflow for making the ultrasound machine.

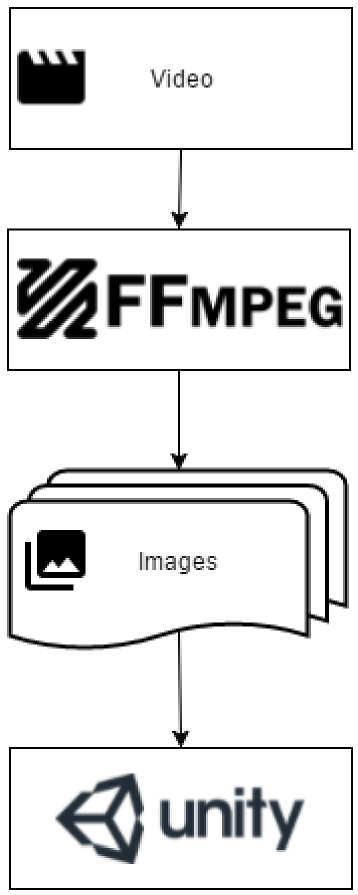


Figure 17: Workflow for ultrasound applications

.

Using raw medical data
The asset Volume Viewer Pro purchased on Unity Asset Store made it possible to load and interact with medical data. The asset provided a prefab containing the necessary scripts for loading and displaying a 3D model of the MR data. The prefab also contained three cameras for recording the model in axial, coronal and sagittal (ACS-Panel) view. For the asset to work, parameters in the Unity editor had to be edited. The procedure for using the asset is further explained in Appendix D.

The loader script included in the asset package was limited to only load data at startup. It was crucial for the application that the models could be changed at runtime to play the different cases. To satisfy this requirement, a customized loader was created called *Load Volume Button.*

Interacting with the data
It was crucial to implement an interface for users to interact with the data in VR. Shaders from the asset, which generated slices from the volume based on a plane normal vector and an offset from the centre were used to create the ACS-Panel. Each axis is displayed on a screen, making it possible for the user to interact with the data. When the user interacts with one screen, the two others will update accordingly.

The two main types of interaction with the data is through use of touch and the use of sliders. The user can interact with the panel by pressing the *grip* button on the controller. When the user presses the button, the hand animation will close all fingers, except the index finger on the hand. When the user touches the screen with the index finger, the panel will update. The script *SlicerInteract* looks for trigger event between the finger and panel. When the event is fired, the script uses the position of the hand, and the position of the panel to calculate the new value to update the views. Equation 2 shows the mathematical calculations for converting positions into vector positions between 0 and 1. The *endPositionX* refer to the end of the screen the hand is touching. The width is the width of the screen and handPosX indicates the hand position at the trigger event. The same calculation for the y vector can be found in Equation 3.

$$x = \frac{endPositionX - handPosX}{width} \tag{2}$$

$$y = \frac{endPositionY - handPosY}{height} \tag{3}$$

The second way of interaction is through the use of the sliders, which are located next to the screens. Each screen have two sliders, corresponding to the x- and y axis. The user can interact with the sliders by moving their hands to the red slider indicator, press the trigger button on the controller, then move the indicator. The script *MRSliders* translate the indicators position to a two dimensional vector between 0 and 1, and send it to the *IntersectionSlicer* script, to update the views.

Each screen have a horizontal and a vertical line. When the user interacts with the screen, the lines will update to illustrate where the user is interacting. In addition to the screens, there are sliders to adjust the brightness and the contrast of the images. They function in the same way as the sliders for the images.

Navigation system

The medical data was also used to create a navigation system, used in the last task of the neurology scenario. The users uses the navigation system by holding the MR-Wand in the patients head. The MR-Wand is used to simulate the navigation pointer used in the real world implementation of the system simulated. The MRI on the ACS-Panel will update based on the MR-Wand's position in the head. It tracks in three dimensions, making it possible to find the depth of the tumor as well as its location.

The Navigation system is case specific, meaning that each case needed a customized solution. The users are not able to use the system until after they have pressed the solution button. Each case has a hidden panel on the patients head. Case number two uses the sagittal view, as seen on Figure 18a. Figure 18b shows what the user can see when playing. The MR-Wand is the white wand with the blue tip, illustrated in both images. The script *MR Scan Helper* is placed on the MR-Wand and detects when the wand collides with the panel. It calculates where on the panel the wand is hitting, and sends the values to the *IntersectionSlicer* script. To give the navigation system three dimensional tracking ability, the depth of the wand also needs to be taken into consideration. The script calculates the depth by taking the distance between the tip of the MR-Wand and the panel. The navigation system displays the images on the ACS-Panel located above the operational table.

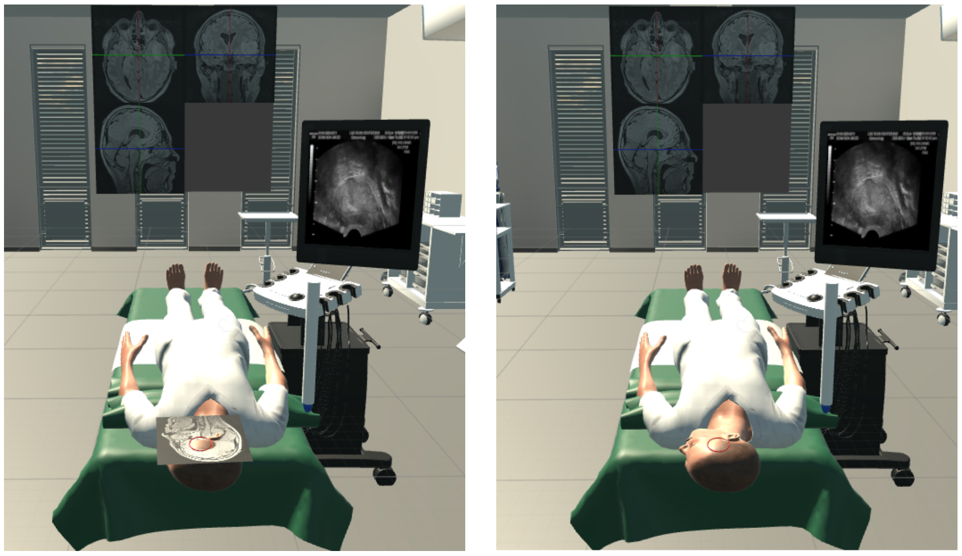

(a) Navigation system panel visible (b) Navigation system panel hidden

Figure 18: An illustration of the implemented navigation system in the application

3.3.5 State and persistence

As the applications complexity increased, the need for a centralized management of the game states increased in kind. Given the number of objects that would either be spawned or destroyed, visible or hidden and the discrete stages of the neurosurgery scenario, a need for an easy access and manipulation of states became necessary. In addition, the need for keeping a state grows when the users are going through the tasks, as it is valuable to asses the progress and degree of accuracy the user have achieved. The users should have a way to do this themselves, without the guidance of a supervisor. To satisfy this requirement, an easy way to access, create and maintain solutions to the scenarios had to be implemented. Given that neither of the developers have medical training, there was a need for an easy way to position the patient correctly and save the solution. There was also a need for editing the solutions at a later time, when a surgical expert was available and could give the correct solution. It was also implemented a way to propagate the saved solutions to other users in case the solutions had to be updated. The implementation made it possible to easily distribute the new solutions without editing the code itself.

State management

Through the lifetime of the application, the different states the application can be in is managed through a singleton script called *GameStateManager*. By using the singleton pattern, the developers can be certain that only one instance of the script is active at all time. This makes it possible for developers to access the script and thereby the active state in other scripts easily. The GameStateManager also handles the state, construction and removal of important GameObject such as patient, torus and ink.

Persisting data
Laying at the core of the saving system was the idea that it should be fast and simple, as well as intuitive for a non technical medical expert to save a state. Such a system should therefore contain as few steps as possible to minimize the extra work needed. The application implemented therefore contains two steps:

1. Positioning the patient and marking the solution in the correct manner for the current scenario.
2. Pressing the *Save-solution*-button

The *GameStateManager* was used to keep track of the important GameObject for easy access when the *Save-solution* button was pressed. The values saved in the GameStateManager was:

1. The case number
2. Patient prefabs name
3. Patient's head global rotation values
4. The solution torus' translation, scale and rotation

These values are collected in a class marked for serialization and saved to a path that will be safe on most systems. In windows, this path will be set to *Appdata*.

Loading data
When the data is loaded from file, the GameStateManager reverse engineer the saving process to create the solution. Due to limitations in the seriazability of classes implemented in Unity, it was decided to save the data as raw values instead of complete GameObjects. Practically, this meant that GameObjects had to be spawned at runtime and set the stored solution values to these objects.

Propagating persisted data
In order to ensure that all users in the same application have the same solution to asses the work done, the application made sure that all users connecting to the server gets the latest version saved. The implemented solution makes use of RPC from Photon to check the timestamp the solution was created. The *OnJoinedLobby* function calls an RPC that is received by all active clients. This RPC goes through every solution and checks if it has outdated or updated solutions. The flow can be observed in 19.

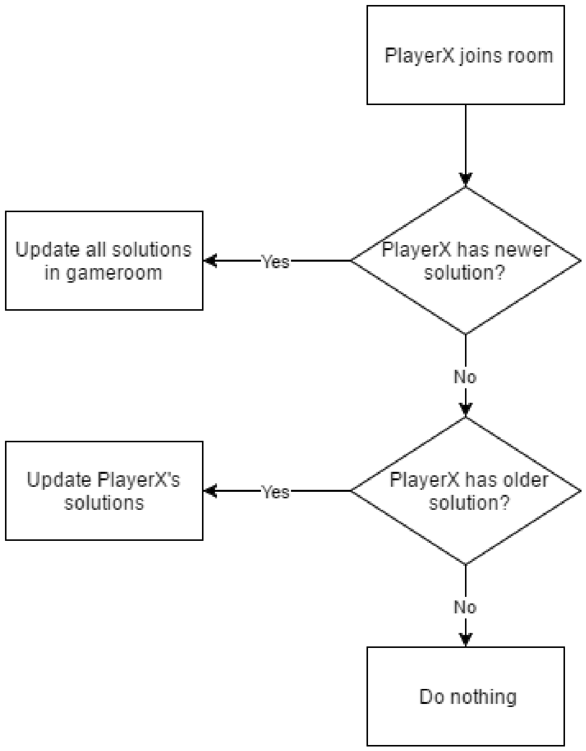

PlayerX joins room
PlayerX has newer solution?
Yes
Update all solutions in gameroom
No
PlayerX has older solution?
Yes
Update PlayerX's solutions
No
Do nothing

Figure 19: Flowchart describing the flow distribution of solutions

.

4 Results

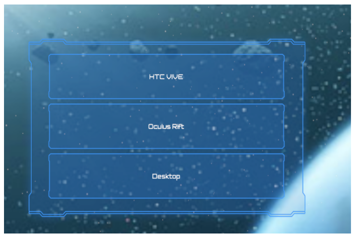
Based on the research questions, with the help of tools and methods described in the previous chapter, an application with two scenarios was developed. In this chapter, the application will be described in detail as well as present the evaluation gathered during the user testing.

4.1 Application and scenarios

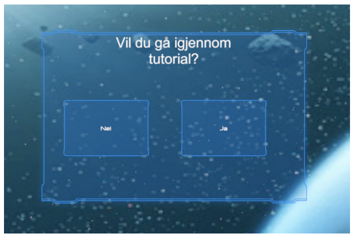
The final application developed became a complex system, composed of multiple different parts. Firstly the menu and tutorials will be presented. Following is the main part of the application, the neurology scenario, which is ready for testing and usage with supervision. Presented thirdly is the implemented parts of the gynecology scenario. In addition, multiple implementations of functionality that was not critical for the scenarios was implemented, and will be presented last in this section as *explored functionality*.

4.1.1 Gear and scenario selection

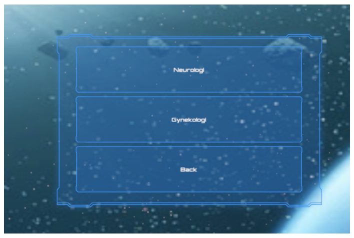
The first thing the user will see when starting the application is the main menu screen. Here, the user first have to select the device they want to play with. The choices are presented in Figure 20a. If the user selects either the *HTC Vive* or *Oculus Rift*, the application will make certain that the device is connected and active. If the user has not yet turned on the controller, they will be prompted to turn them on. Figure 21 shows the message the user receives when the controllers are disconnected. Both controllers needs to be active to continue to the next step. It is important for the scenarios that the controllers are active at start up. If the controllers are not active, key functionality such as hand gestures will be disabled. If the user have selected VR equipment, the application will ask the user if they want to run the tutorial scene prior to playing the scenarios, to get a better understanding on how the system and the buttons work. If the user do not want to go through the tutorial scene, then the last choice the user have to perform before emerging into VR is to select which scenario they want to play. Figure 20c illustrates the different cases the user can select. The first scenario is the neurology scenario, which is the main focus of this application. The second scenario is the partially implemented gynecology scenario.



(a) Select device

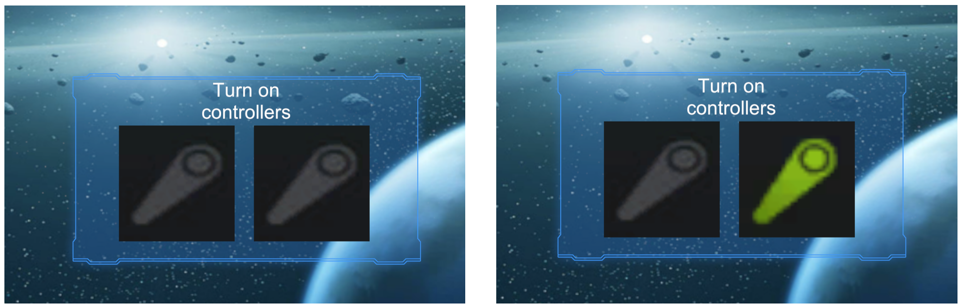


(b) Tutorial prompt



(c) Select scenario

Figure 20: Snapshots of the main menu

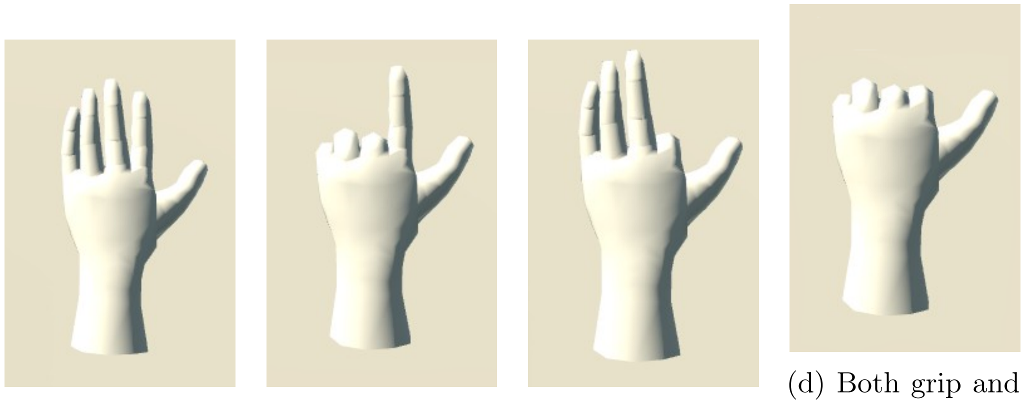


(a) Both controllers turned off (b) One active controller

Figure 21: Illustrating the turn on controller message

Tutorial
First time users of HMD often need to spend some time getting used to the environment. In order to give the users a helping hand in learning how the equipment and software works, a tutorial with vocal instructions was developed. A video presentation of the tutorial may be found in Appendix F. In this tutorial, the users performs basic tasks to learn how to use the controllers and interact in the virtual world. First, the users are prompted to press the grip button. When the user presses the grip button, colorful drawings will appear in the air. When the user moves the controller, the drawing extends. The drawing functionality is developed to encourage users to test out the functionalities the application offers. In addition, by having a fun or engaging feedback from pressing the button, the user will easier distinguish between the different buttons. After pressing the grip button, users are instructed to use the teleport function. After successfully doing so the first time they are told to do it three more times. The next step is to familiarize the user with interacting with objects in the virtual world. It was decided to use a longbow with arrows in this example. The Steam VR package already contained the model and scripts for the equipment. The reason for choosing the bow and arrow was that it forces the users to use both hands with different items simultaneously, as well as using multiple buttons.

Becoming familiar with the hands
Having gone through the tutorial, the users needs to familiarize themselves with the hand gestures. The users have four different hand gestures they can perform. The first is presented in Figure 22a. This is the default, and is used when the user is not interacting with any buttons on the controller. The second gesture is performed when the user press the grip button, as illustrated on Figure 22b. When the user press the trigger button, the index finger will close, as shown in Figure 22c. When both the trigger and the grip button is pressed, the fingers will close to form a fist, illustrated on Figure 22d. These hand gestures are created to make it understandable for the user which button performs which activity.



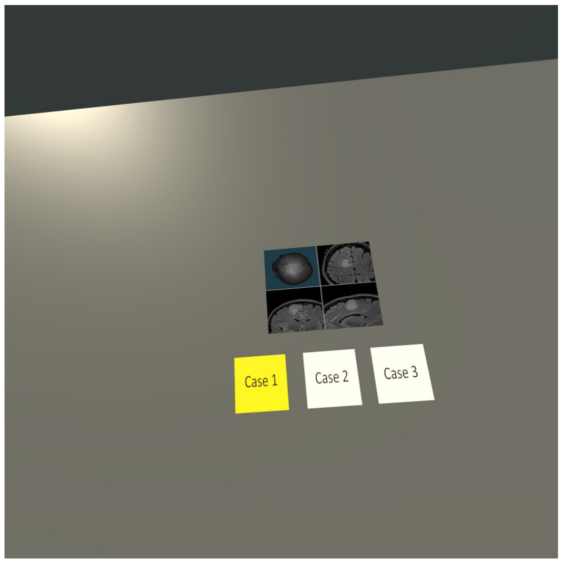
(a) Default hand gesture (b) Grip button pressed (c) Trigger button pressed (d) Both grip and trigger button pressed

Figure 22: Showing the different hand gestures in the application

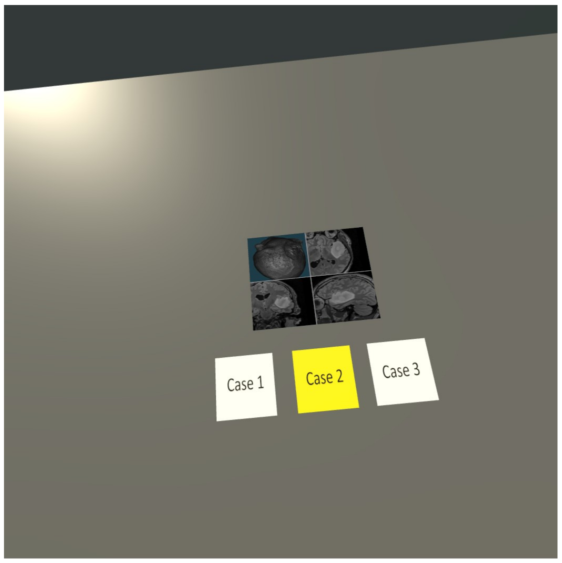
4.1.2 Neurosurgical scenario

The scenario is divided into several tasks. A video presentation may be found in Appendix F.

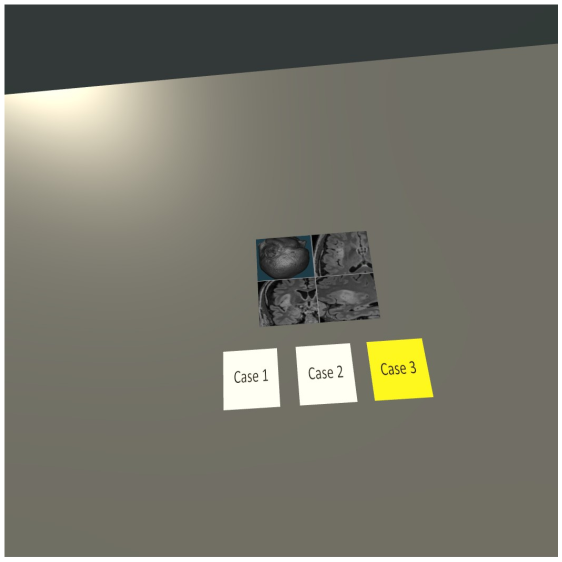
Task 1: Choosing case and becoming familiar with the operation room
The user has three different cases to chose from. The user places his hand over the case button he wants to play, and press the trigger button. The button will turn yellow and a preview of the brain with the tumor will be presented on the panel over the buttons. This makes it possible for the users to view which case they want to play before committing to a case. The three cases are presented in Figure 23. When the user is certain of what case they want to play, they press the button once more. The button will then turn green, indicating that it is the active case. An illustration of a selected case is shown in Figure 23d. If the user presses one of the other buttons, they will turn yellow and show a preview of the case. The green button will remain green until a new case is selected.



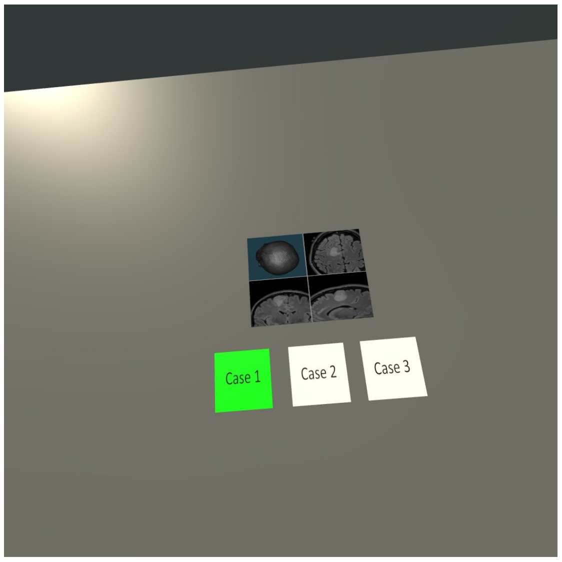
(a) Case 1



(b) Case 2



(c) Case 3



(d) Selected case

Figure 23: Preview of the three cases

After selecting a case the user needs to locate and move to operation room three. The user moves by teleportation, which is done by pressing the trackpad button. When the button is pressed a teleportation beam is created. The beam is used to aim at the location of the user wants to travel, and when the button is released, the user will be teleported to the target location. In addition to the beam, the users will be able see where in their play area they are located, allowing them to plan how far they can walk in the physical space without exiting it.

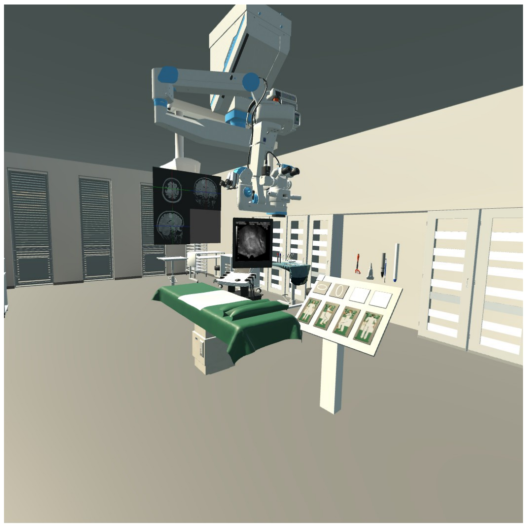
When the user locates operation room three, they should familiarize themselves with the equipment and their usage in the room. The controllers will act as a guide for the users, and indicate when an object is interactable. Whenever the controller touches such an object, the controller on the hand hovering over the object will turn to a translucent yellow. The button the user needs to press to interact with the object will turn a darker shade of yellow to make it distinguishable. In addition, the controller hovering over the object will make a slight vibration, making it easy for the user to comprehend which controller they have to use to interact with the object. The script detecting controller collision with an object can only detect one controller, meaning that if both controllers collides with an object, only one can be used for interaction.

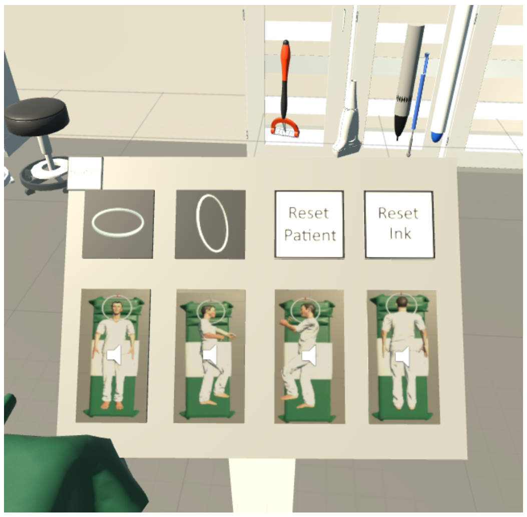
Task 2: Locate the tumor
When the user is well acquainted with operation room three and the equipment within, the user will commence to study the MR scan of the given case. The MR scan of the case can be viewed on three panels in the operation room. The panels shows the axial, the coronal and the sagittal view of a human head generated by the MR data.

The user has two ways of interacting with the panels. When the user interacts with one panel, the two other panels adjust to the changes made in the panel the user is interacting with. The first way of interaction the user can perform on the panel is through the sliders on the sides. Each panel have two sliders for manipulating the view in vertical- and horizontal axis. The user moves a hand to the slider point and press and hold down the trigger button to start sliding. The colored lines on the panel will move to indicate where the user is interacting. The second way the user can interact with the panel is through touch. When the user presses the grip button, all fingers except the index finder will close, making the user able to point. The user can point with the index finger at the panels where they want the focus to be. When the user touches the panels, the images and colored lines will update to accommodate for the changes. The user also have the ability to change the brightness and contrast of the images by using the sliders located under the coronal view.

When the user have located the tumor, they can confirm it by pressing the solution button next to the panels. The solution button toggles between the correct solution and the solution the user has presented. The solution differentiate itself by having a red torus around the tumor. When the user is certain of where the tumor is located, they move on to the next step.

Task 3: Patient positioning
The information acquired from studying the MR scan should be sufficient for the user to position patient correctly for operation. Figure 24a shows the equipment the user have at his disposal. The panels over the table represents the axial, coronal and sagittal view of the MR data. The panels update when the user interacts with the panels in task 2, allowing the user to see the tumor when working at the operation table.


(a) Operation room

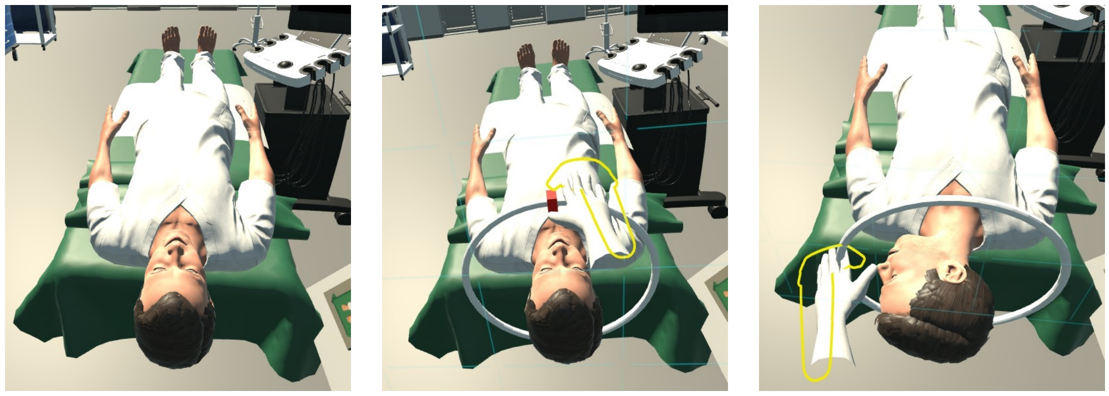


(b) The game panel

Figure 24: An overview of available equipment for task 3

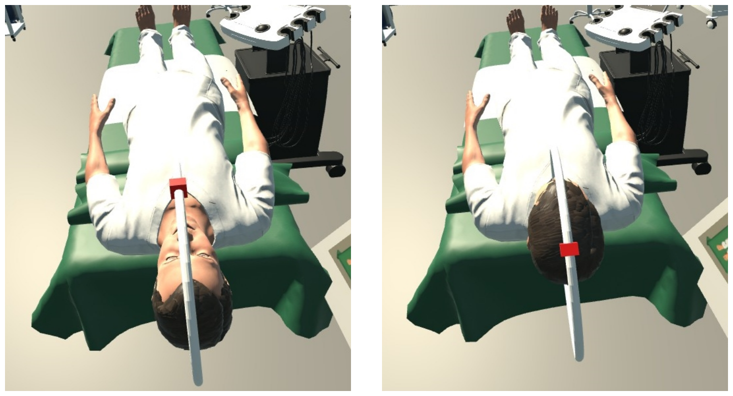
The game panel is used for initial and large scale patient positioning. Figure 24b illustrates the game panel. Four buttons with image of a patient shows the different positions the patient can be placed in. Patient position is chosen by holding a hand over one of the buttons and press the trigger button. When the user presses the button, a sound is played to make the user understand that something has happened. The selected patient will spawn on the table. Figure 25a shows the patient spawned when the user presses the first button. It is possible to change position by clicking on one of the other buttons at any time. When the user is satisfied with the body position, they have to rotate the head into correct position.

To rotate the patient's head the user have to spawn a torus, which is done by clicking on the torus buttons on the game panel. There is a horizontal and vertical torus for rotating the head across the two axis. Figure 25b illustrate the spawning of a horizontal torus around the patient's head, and the vertical torus is presented in Figure 26. To rotate the patient's head, the user places a hand on the red marker on the torus, and press the trigger button. As illustrated on the Figure, the controller turn a translucent yellow when the hand is in contact with the red marker. Figure 25c shows the result of rotating the patient's head 90 degrees.



(a) Patient lying on table (b) Horizontal torus (c) Horizontal torus turned

Figure 25: Illustration on how the horizontal torus is used



(a) Vertical torus (b) Vertical torus turned

Figure 26: Illustration on how the vertical torus is used

When the user is satisfied with the positioning of the patient, the patient's head needs be shaved. To accomplish this, the shaving razor has to be used. The razor is located over the game panel, illustrated in Figure 27. The user picks up the razor by holding a hand over the razor and pressing and holding the grip button on the controller. The shaving procedure is performed by moving the razor back and forth over the patient's head for a couple of seconds. The result of the barbing can be viewed in Figure 28a. If the user wants to make adjustments to the patient's position before proceeding, the user can press the *Reset Patient* button. The button reconstructs the patient's hair, making it possible to rotate the head again.

The last step of the task is marking where the point of entry is. The user marks the patient by picking up the black pen illustrated in Figure 27. The user picks up the pen by pressing the grip button, and can draw with it by pressing the trigger button. Figure 28b shows how the marking of a patient works. If the user is not satisfied with their drawing, they can remove the drawing by pressing the *Reset Ink* button on the game panel.

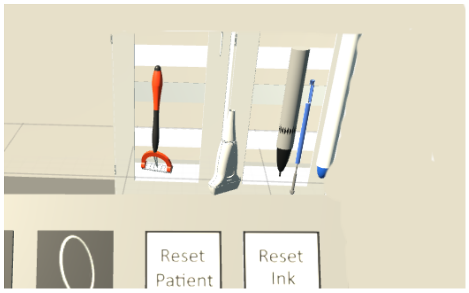
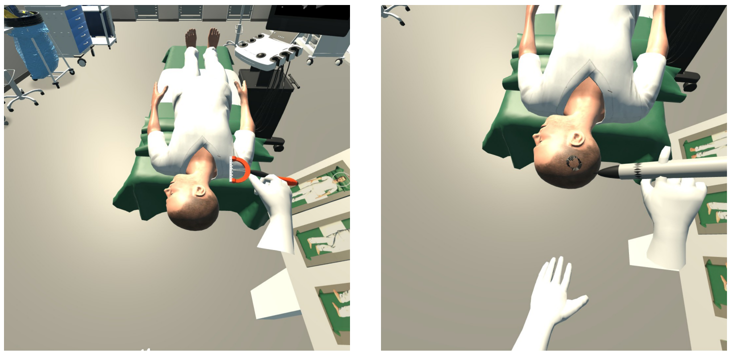


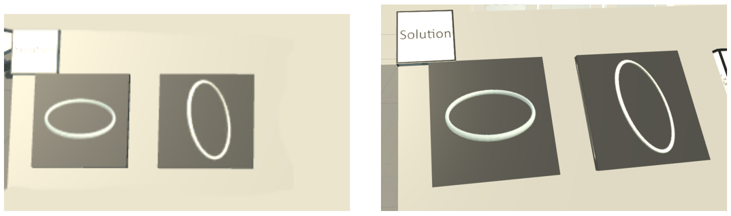
Figure 27: Representation of the equipment located over the game panel



(a) Barbered patient (b) Marked patient

Figure 28: Examples of how to shave and mark the patient

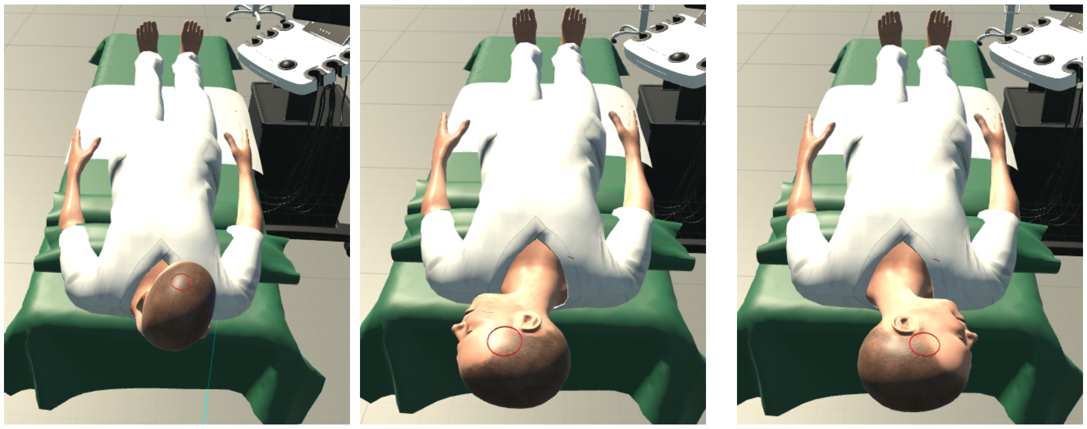
After performing the marking, the task is complete. Before moving on to the next task, the user must check with the solution to see if their assessment were correct. The user can toggle between the solution and their own creation by pressing the solution button on the game panel. The solution button is located at the top left of the panel. The button is significantly smaller than the other buttons, and the text is only visible from a certain angle. It is presented in this manner to prevent users from clicking the button by mistake when performing the task. When the user works with the task, the text on the button is not visible as shown in Figure 29a. The text is however visible when the user leans forward and looks at the game panel from a different angle, as illustrated in Figure 29b.



(a) Solution button text not visible (b) Solution button text visible

Figure 29: Obscuration of solution button

Solutions for the three different tasks can be seen in Figure 30. The red torus on the patient's head indicate where the user should have marked the patient in each case. When continuing to the next task, the user works from the solution on each case.

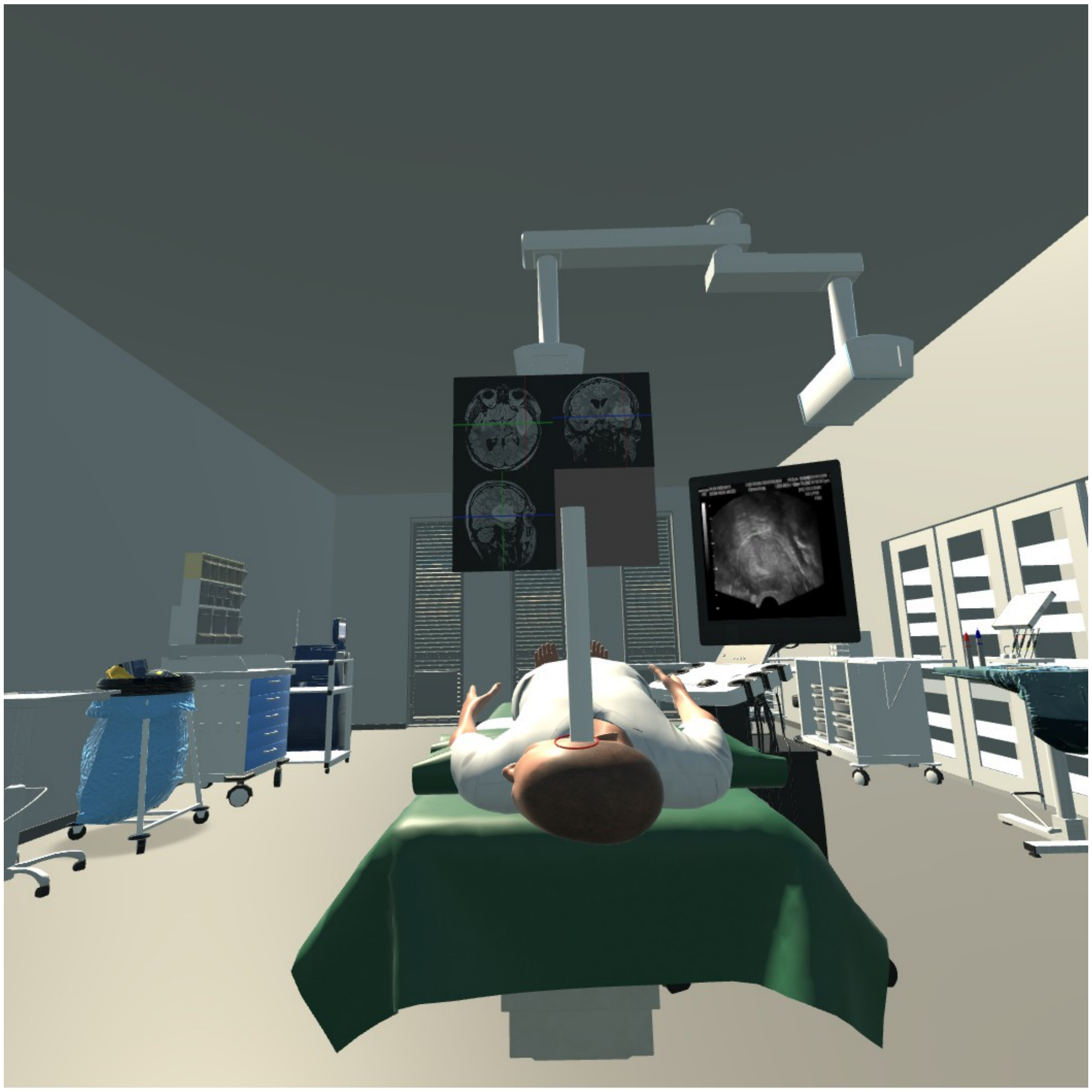


(a) Case 1 (b) Case 2 (c) Case 3

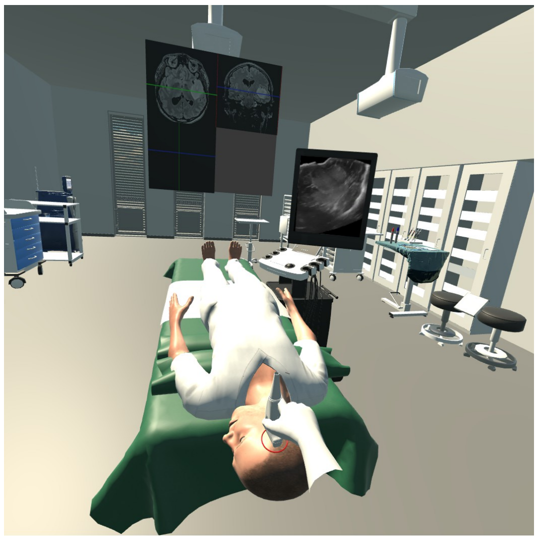
Figure 30: Solutions to the three cases

Task 4: Ultrasound and Navigation system
When the patient is positioned correctly, the next task is to verify the position of the tumor. This is done in two steps. The first step is verification by use of the navigation system. To use the navigation system, the user has to pick up the MR-Wand, the white wand with a blue tip, and place it over the patients head. The ACS-Panel above the operation table updates based on where the pen is located on the head. It tracks in three dimensions, meaning that it is possible to point the marker inside the head to determine how deep inside the head the tumor is located. Figure 31a illustrates how the MR-wand is used to locate the tumor. The ACS-Panel shows that the tumor is in the centre.

The second step is verification by ultrasound. In neurosurgery, it is not sufficient to only rely on the MRI when performing an operation. Firstly, the MRI are not completely accurate. The inaccuracy of the scans can become close to a centimeter. In addition, once the operation has started, the brain might move around, making the tumor move as well. To remedy these issues, ultrasound is used complementary to the MRI during operation. Ultrasound scans are performed in intervals to determine the position of the tumor and critical brain tissue. These scans are compared with each other throughout the surgery. In the last task, the user has to perform the first ultrasound scan. To perform the scan, the user picks up the ultrasound probe and move it to the point of entry on the head. If the tumor in observed at the assumed location the user has completed the scenario. The performing of ultrasound is shown in Figure 31b.



(a) Case 1



(b) Case 2

Figure 31: Use of navigation system and ultrasound

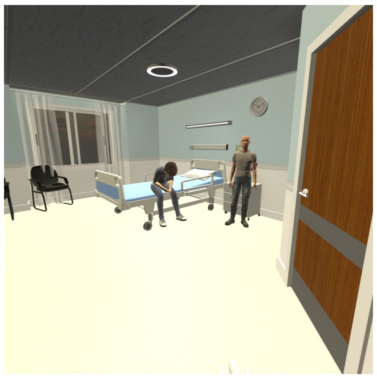
4.1.3 Gynecology scenario

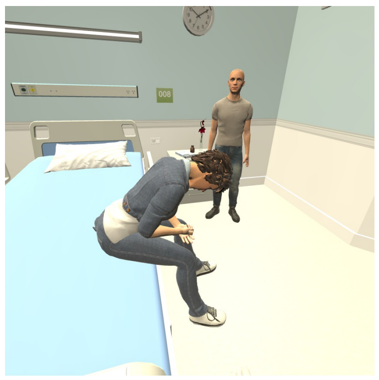
The initial scenario to be implemented for this thesis was the already established gynecology case from VirSam [69]. However, due to lack of medical knowledge on the subject as well as limited use of interaction in the scenario, the focus switched to neurology.

The original scenario is divided into three scenes, *meeting the patient*, *leading the patient to the ultrasound room* and *performing ultrasound examination.*

Task 1: Meeting the patient
In the beginning of the scenario, the nurse enters the room where the patient and her husband is waiting. The patient is not feeling well, and is clutching her stomach in pain. The husband is in distress and looks to the nurse for confirmation. Figure 32a illustrates when the nurse enters the room, and Figure 32b shows that the husband looks at the nurse.

Task 2: Guiding the patient to the ultrasound room
After asking the patient a couple of questions, the nurse tells the patient to follow them to the ultrasound room, for an ultrasound examination. When the patient reaches the ultrasound room, she lies on the bed and awaits the examination. The ultrasound room is presented in Figure 33.


(a) Nurse enters the patient room


(b) Nurse is next to patient

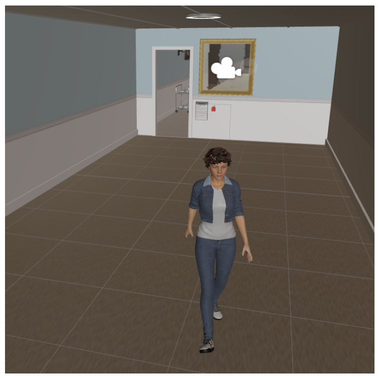

(c) Patient walks to ultrasound room

Figure 32: Guiding the patient to a ultrasound room

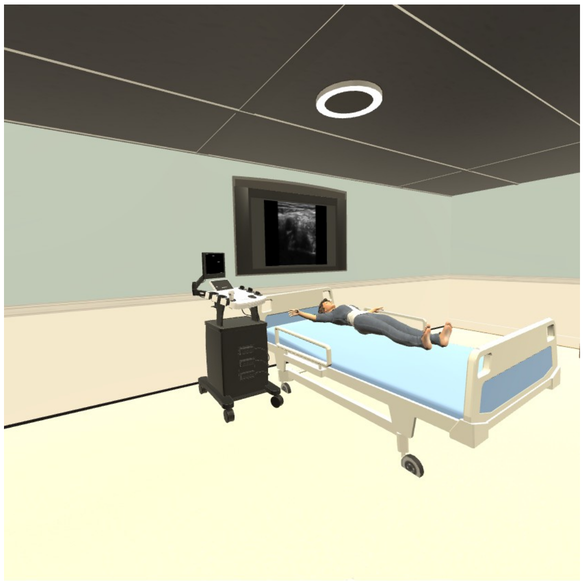
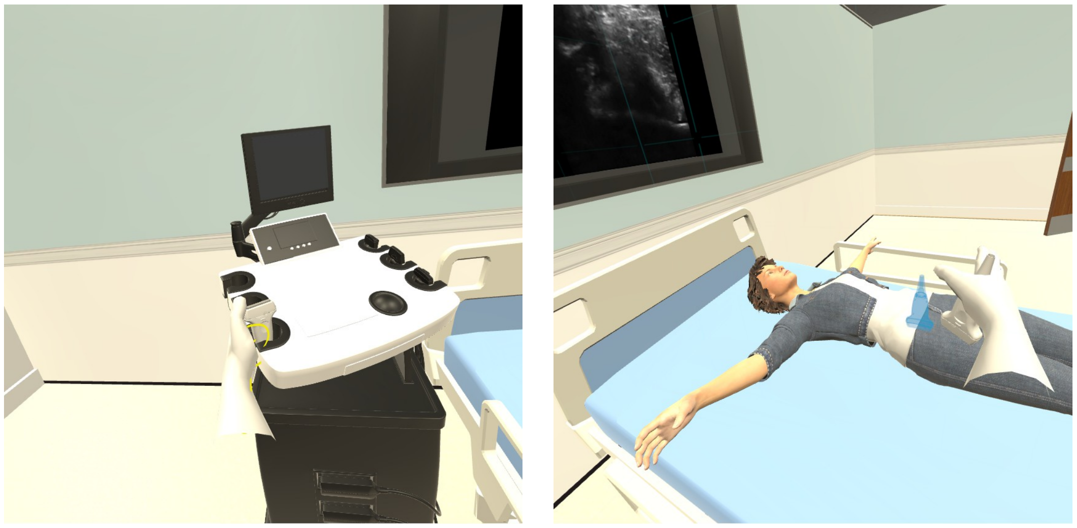


Figure 33: Representation of the ultrasound room in the gynecology scenario



(a) Picking up ultrasound probe (b) Blue silhouette of ultrasound probe

Figure 34: An image of the ultrasound system in the gynecology scenario

Task 3: Performing the ultrasound examination
The last task is to perform the ultrasound examination. The user picks up the up the ultrasound probe by holding a hand over the probe and press the grip button. The controller hovering over the probe will turn yellow, indicating that the GameObject can be interacted with. This is showed in Figure 34a. When the user picks up the probe, a blue silhouette of the probe will illuminate over the patient where the user should place the probe, as illustrated in Figure 34b. When the probe reaches the silhouette, the ultrasound machine begins transmitting images to the television screen behind the bed. The scan is projected on the screen to give the user a sufficient view

of the ultrasound scan. Figure 35 and Figure 36 shows how the probe works. The goal of this tasks is to locate and identify the abnormalities in the patient's uterus. The user should be able to identify the rings in the ultrasound images, which may be observed in Figure 36a. These black rings indicates extrauterine pregnancy, and the patient needs to be taken to the operating room and be prepared for surgery.

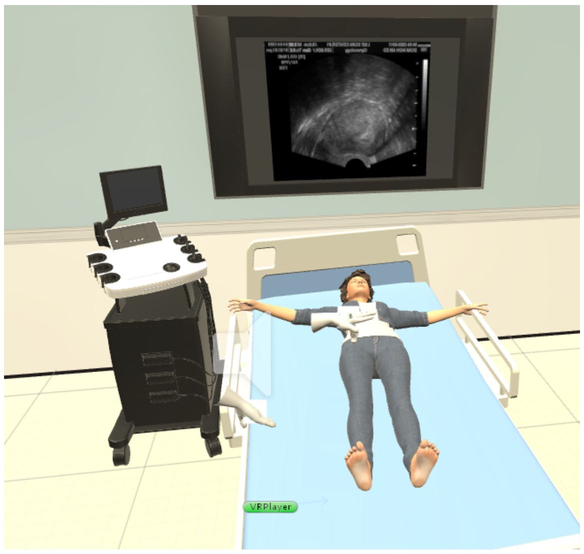
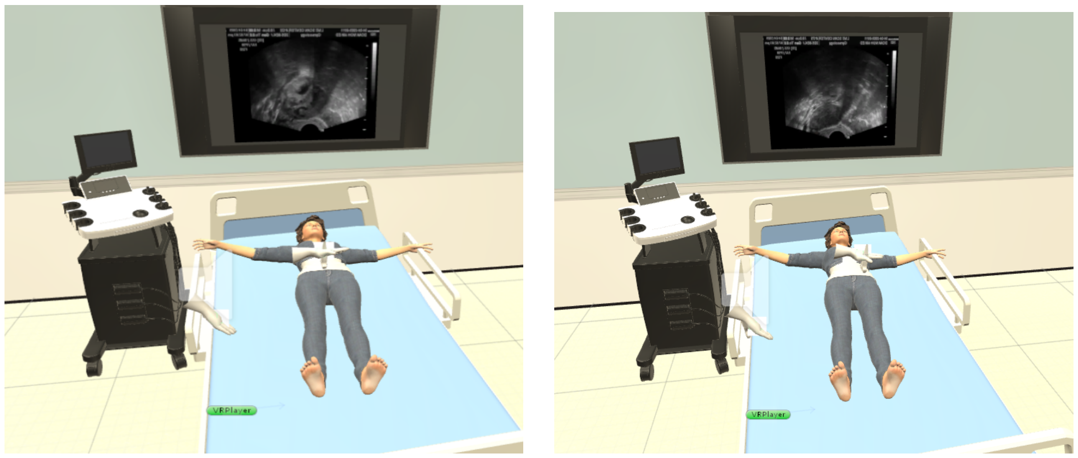


Figure 35: Beginning of ultrasound examination



(a) Black rings illustrated on ultrasound (b) End of ultrasound scan

Figure 36: Illustration of how the ultrasound machine works

4.1.4 Explored functionality

In the process of implementing the scenarios, multiple explorations were conducted to assess the possibilities and functionalities that might be attractive and useful in learning applications using general purpose VR. In order to make the most out of the scenarios however, the functionality implemented into them needed to be strictly streamlined and carefully designed to keep the users from distractions. In this section, we will explain functionality implemented that were either too distracting for the users or no longer useful in the final application. A link to the video presenting this functionality may be found in Appendix F

Incision of scalp and looking through microscope

The next step in the neurosurgery scenario would be to make an incision of the scalp. The surgical equipment located between the MR-Wand and the black marker presented in Figure 37 is used to open the patient's head. The user grabs the item and place it over the patient's head. Ideally, the hole in the patient's head would appear where the item touched, but it was not fully developed. Removing part of the head is a tedious task, requiring external modelling software to manually remove vertices and edges in the model. It was concluded that this step was not essential for the educational application, so focused was changed after creating one hole. A brain was placed inside the head, allowing the users to see the brain when opening the patient. The hole in the patient's head can be viewed in Figure 38b

The microscope above the operating table can be used to get a closer view of the opening created. The user grabs the handles on the side of the microscope and pulls it down. When the height of the microscope is at a satisfactory level, the user can put their head in the ocular to view with the microscope. Figure 38a illustrates how the microscope looks when it is pulled down. The image is a static one, meaning head tracking and rotation is disabled. As mentioned in the background section, Valve advises against disabling head tracking and rotation, because it breaks with immersion and result in VR sickness. Users complained about feeling nauseous after trying the microscope, so the development of the microscope stopped.

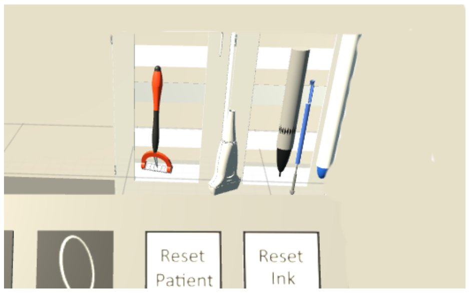
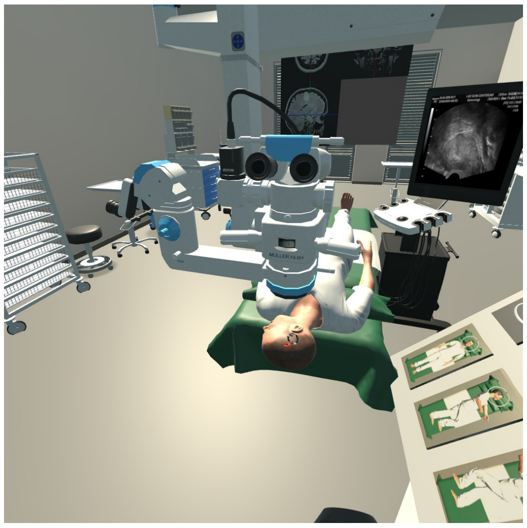


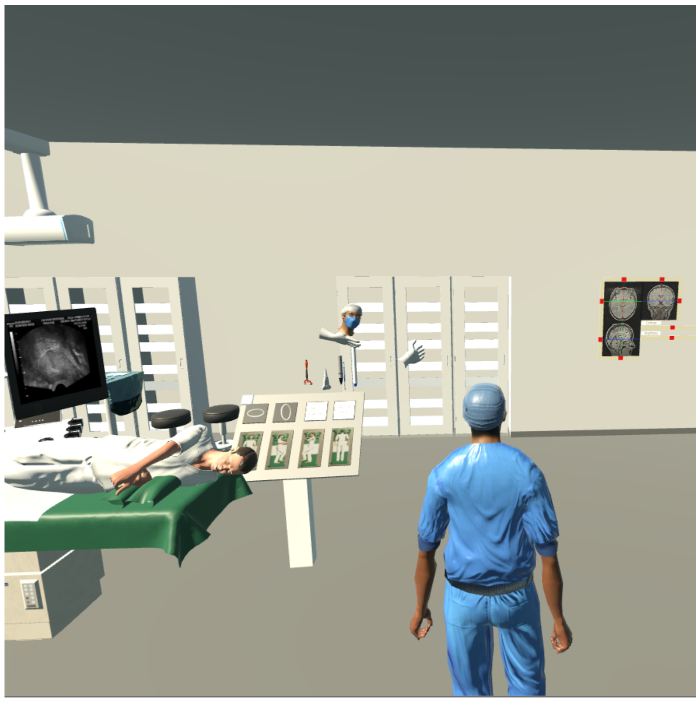
Figure 37: Illustration of the surgical equipment needed for opening head


(a) Illustration of the microscope over the operational table

(b) Presentation of what the user sees in the microscope

Figure 38: Views from different platforms

Collaboration and multi-user possibilities
Giving users the ability to connect to each other was an important functionality to implement. Currently, it is possible to connect users using OR, Vive and desktop together in the same instance of the application. While the users using VR devices will be represented as a head with hands, the desktop user will have a full bodied avatar. Users on a desktop will be able to move around and observe all interactions taking place, but will not be able to interact with objects. This way, the desktop users are considered observers while the VR users are considered actors. The details conserning implementation of such functionality may be found in Appendix A.


(a) VR user viewed on desktop

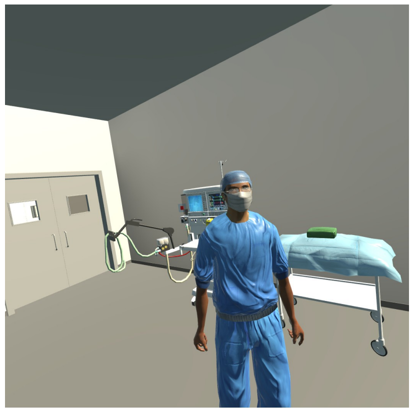

(b) Desktop user viewed in VR

Figure 39: Views from different platforms

Assisting surgeon

A task to be performed during the gynecology scenario was to go through with the surgery. The initial goal was to make one user play as the surgeon and another as the surgical nurse. The main responsibility of the surgical nurse, was to give the surgeon the right tools at the right time. As an early prototype, a non playable character was developed to control the surgeon. The surgeon had animations for requiring and retrieving items from the user. The user picks up the items and place them in the hand of the surgeon. The development of the gynecology scenario stopped, hence no further improvements on this functionality was made.

Interactive body models

Inspired by the multiple visualization applications, offering interactive exploration of the body, the team implemented an interactable body, where body parts can be taken apart in VR. The model used was found on the internet [70], and rescaled to feasible values to use in VR. Then the different pieces was placed correctly according to each other in a VR environment and programmed to be interacted with. The implementation gained no educational benefit, as the model rendered too poorly to represent human anatomy. In addition an application used for education on such a specific level should have a framework developed around it, like the ability of editing the colors of the pieces as well as having some textual information about each piece. The anatomy model is presented in Figure 40

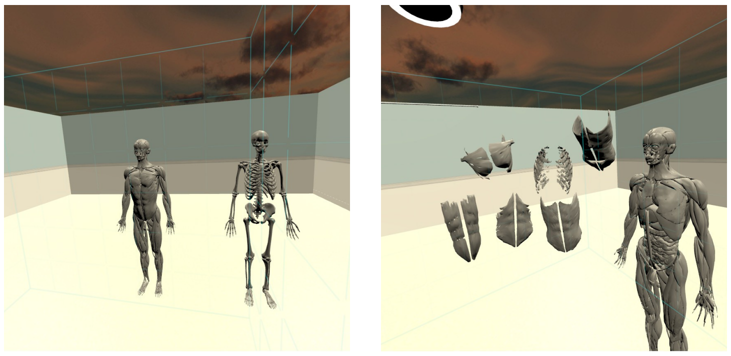

(a) Anatomy (b) Anatomy taken apart

Figure 40: Illustration of interaction with human body

Video tutorials

The development team was encouraged to develop a video system for looking at tutorials. The result is shown in Figure 41. The videos are located on an external server, making it possible to add videos to the application independently of Unity. The user can move between videos by pressing the forward and back button. The preview image shows what the tutorial is about. The user plays a video by holding a hand over the play button and press the trigger button. The system is fully functional, but the tutorial videos were not created. The short amount of time each participant had to go through the cases made the developers chose to instruct during the sessions instead. There are currently two videos on the server. The first video is showing how case number one is played, and the second shows the explored functionality.

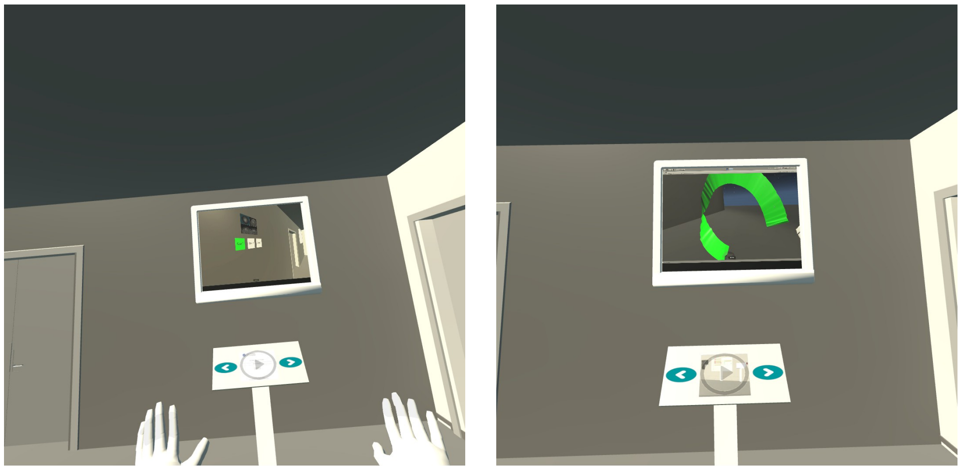
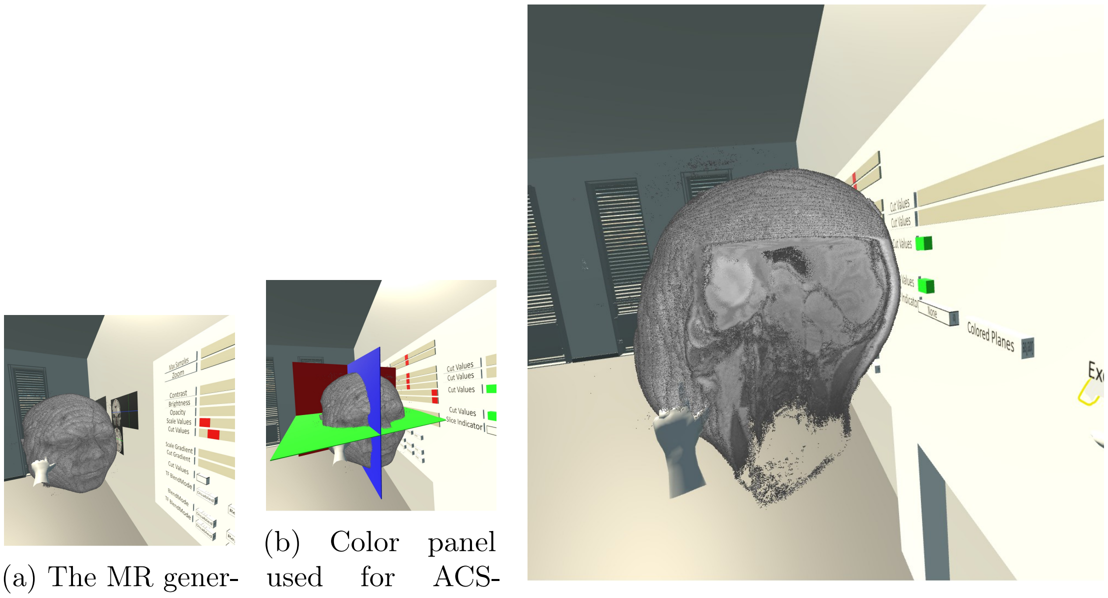

(a) Master thesis video played in the application (b) Tutorial video played in the application

Figure 41: An image showing the video system

Interactive MR model

The volume viewer package purchased contained a prefab for creating a 3D model of the loaded medical data. Advanced tools for manipulating the 3D model was also included. The tools were translated and re-implemented in VR and was experimented with during the user evaluations. A control panel for handling the manipulation was implemented. The panel consists of a number of sliders for changing values such as brightness, contract and scale. In addition the panel made it possible to remove black and white background noise on the model with the use of sliders. The panel supported navigation of the model in two different modes, *color planes* and *exclusion*. The color planes mode made it possible to see where the ACS-Panels slices was on the model. When the user interacted with the ACS-Panel, the color panel on the model changed accordingly. The exclusion mode added the ability to remove parts of the model, making it possible to look inside the head of the person. The removal of the parts are conducted by using the ACS-Panel. The interactions can be seen in Figure 42.



(a) The MR generated human head (b) Color panel used for ACS-Panel (c) Ability to see inside the head

Figure 42: Presentation on different ways of interacting with the MR model

It was concluded, however, that the initial skill required to utilize this functionality was all to high, as well as causing some latency problems. The functionality is something we deem worthy of more exploration, and it is easy to appreciate the value of such functionality if it could assist in getting students to better visualize what they are trying to learn.

4.2 Evaluation

To assess the value of the application and to answer the research questions, the application had to be tested and evaluated. In this chapter, the results from user testing will be presented. The tests are separated into the different times they were acquired and presented in chronological order.

4.2.1 Technoport

In march, 2017 the project was presented on a stand at Technoport [71]. We presented the project basis and technology to the interested participants at the exposition, and used the arena for getting feedback to the project as well as observing user patterns. Following is a short summary of the observations

Input devices
Only the Vive controllers were used during the specialization project, resulting in key mapping based on what felt natural on those controllers. During the exposition, OR was used with its newly released controllers, Oculus Touch due to the ease of transportation and setup. Overall, the controllers are much the same, however one small difference caused some trouble. On the Vive controllers, there is a touchpad. Most users that tested the specialization project had no problems understanding that it was clickable. The Oculus Touch however, has a joystick. The first few users had so much trouble understanding the possibility of pressing down this button, that they eventually gave up.

User interaction difficulties
Most testers at the exposition confirmed the theory that user who test VR for the first time spend much time get used to the sensation of VR and the controllers. This means that they will spend some time being non-productive, and some time spent simply observing should be taken into account for first time users.

4.2.2 Medical simulation centre - network conference

During the 2017 Network Conference [72], held at the Medical Simulation Centre at st. Olavs hospital, we presented the project to volunteering participants on a stand. With instructions from us, the participants got to go through the core parts of the system to try out the functionality. The scenario was mostly implemented, but not yet robust enough to leave the testers on their own. At the end of the test, participants were asked to fill out a short survey asking core questions about the functionality and their experience in the virtual world.

Setup and execution
This test was conducted using one OR, as well as one laptop for testing the network functionality. The original plan was to get two users using OR and two users using laptops, but due to technical difficulties, most users got to test with one OR only. Before and during the tests, the participants were carefully instructed by one of the developers. The tests usually lasted between 5-15 minutes per person.

The test subjects
The individuals participating in this tests was technical and medical experts from different parts of Norway. It was clear from discussion that they were already interested in simulation in medical practice and was excited to try out the application. The total number of participants was 12.

Findings
After a participant had tested the system, he or she would be asked to fill out a survey constructed by the developers. The participants were also encouraged to give feedback in free form.

The results are shown in Figure 43. All questions were posed in a manner that should be answered on a likert scale from 1, strongly disagree, to 5 strongly agree.

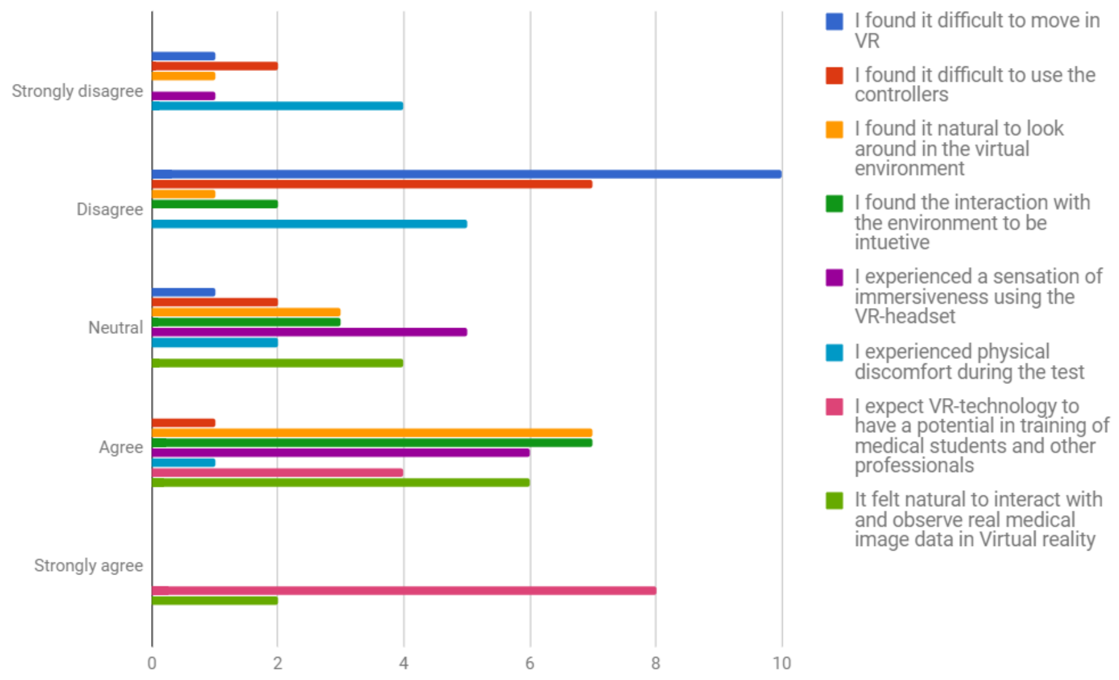


Figure 43: Chart created based on feedback from medsim

.

4.2.3 Neurosurgical ultrasound seminar

In June of 2017, the application presented in this article was invited to present at the 9th annual course in ultrasound in Neurosurgery [73]. The application was used as one of four stations on which the participants should explore the simulation possibilities and get a first hand experience with different equipment and concepts regarding ultrasound in neurosurgery.

The focus on this test was shifted from what was already established in earlier user tests. Instead of focusing on whether the users are able to interact with the VR gear and the virtual environment, observations were made on how the users interacted with the medical equipment and the medical data.

Setup and execution
OR was used as the HMD in this presentation. Moving from earlier experiences, it was decided to make use of three base stations, rather than two. This leads to a less

disruptive experience for the testers, as they will no longer get disconnected from the application if they are rotated away from them.

The execution was a three step process for each test subject. Firstly, in order to give the testers an overview of the application and make them ready for the tasks they were supposed to execute, a quick walk through was given by one who was comfortable with the application and VR. Secondly, they were given a quick introduction to the controller and the different buttons they needed to use in order to manipulate the virtual world. Lastly, they were given the headset and guided through the scenario.

Test subjects
The test subjects in this execution was mainly surgeons in training or surgeons who wanted to expand their knowledge in ultrasound guided neurosurgery. The age of the participant ranged from late twenties to sixties. None of the testers hade used a HMD prior to the tests.

Findings
The data gathered in this test was made by observation, post test talks and a video recording of all subjects in the application. In order to asses the value of the application, four aspects of interest were closely watched:

1. How quickly the testers became familiar with the environment.
2. How quickly were the testers able to locate the tumor in the ACS-Panel.
3. How well they were able to position the patient and marking him correctly.
4. How well they were able to interact with the medical data using the MR-Wand and ultrasound probe.

Due to the different nature of the aspects, they were evaluated differently:

1. Number of seconds from start of test until first teleport and until the case was chosen.
2. Number of seconds between first attempt to interact with the panel and until the tumor was marked correctly.
3. How close to the correct solution the tester managed to position and marked the patient.
4. Did the tester manage to locate the tumor with the MR wand and ultrasound probe.

Results
Aspect one can be viewed in Figure 44, and two can be viewed in Figure 45. In Figure 46 the average and standard deviation is listed.

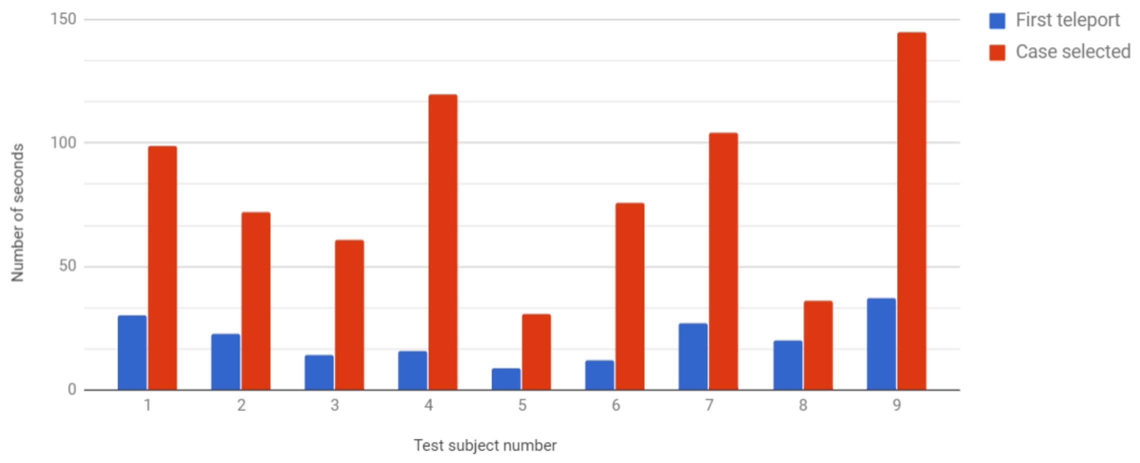


Figure 44: Number of seconds untill first teleport and case selected

.

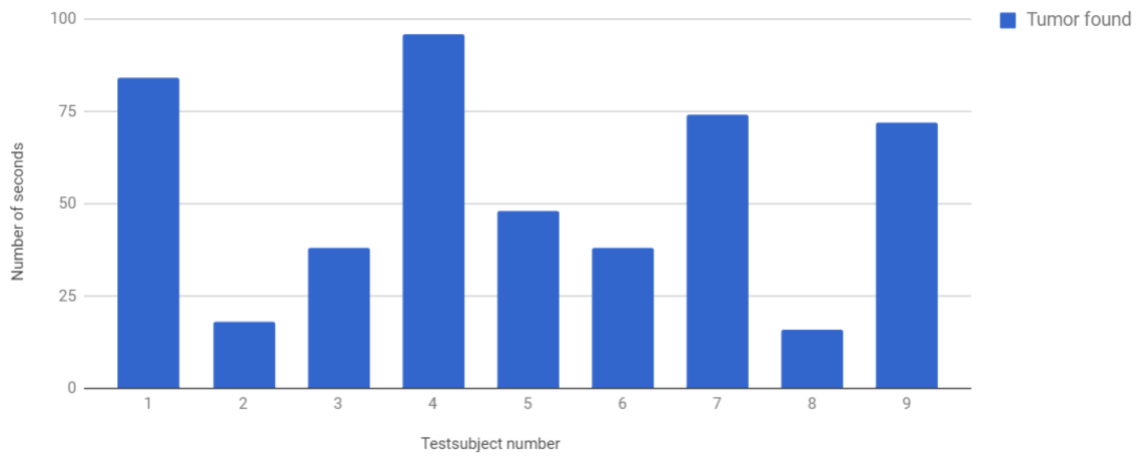
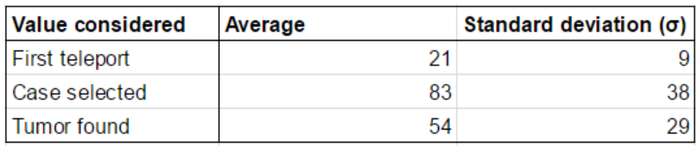


Figure 45: Number of seconds taken to find tumor in the ACS-Panel

.



Value considered	Average	Standard deviation (σ)
First teleport	21	9
Case selected	83	38
Tumor found	54	29

Figure 46: Average and standard deviation for aspect one and two

.

Placing the patients correctly proved mostly to be successful. Only two out of the nine participants had crucial errors in their placements or markings. The rest of the participants had the patient mostly correctly placed. All those who got the time to test the MR wand and ultrasound were able to use those tools and locate the tumor.

5 Discussion

The application presenting the neurosurgical scenario and extended functionality was tested on two occasions, both during and after the development was finished. The feedback gathered presents interesting findings to the usefulness of the application. In this chapter, we will discuss these findings and assess them in regards to the related work and the research questions.

5.1 Interaction and immersiveness with general purpose VR equipment

The second research question for this thesis concerned how the user experienced using the general purpose VR equipment. The feedback from the users that have tested the scenario with such equipment gives an indication of users having little trouble with navigating and interacting with the virtual environment. Multiple implementation choices have been taken when developing the interaction system in order to make the interaction as well functioning as possible. The different choices and reflection around these will be presented in this chapter.

5.1.1 Controller design

Compared to highly specialized hardware to interface user interaction with the virtual environment [20] [22], or to the stationary experience used in others [23], the Vive controllers and Oculus Touch provides quite a different experienced. While it is hard to implement functionality like force feedback, naturally restricted movement and natural haptic feedback, the controllers have proven highly useful in other areas. It is important to notice that the scenario implemented in this thesis had limited need for force feedback and naturally restricted movement. The general purpose controllers are formed to rest naturally in the hands, and have a multitude of buttons easily available for users. Both controllers are wireless, and encourages users to move within a small physical area. This crafts a sensation of room space which further increases the immersed sensation in the applications.

While the controllers have many similarities, some important differences is worth noting. The Vive controller are large, have a touchpad and have buttons that does not stick out of the controller with tactile feedback on press. The Rift controllers are small in comparison to the Vive controllers, and feels more naturally in the hands of the user, as they are tailored for left and right hand. In addition, the buttons on the Rift controller are all standing out slightly in order to make them easier to hit.

The tests conducted have revealed that users take more naturally to using the touchpad on the Vive over the joystick on OR Touch when teleporting, but finds it significantly easier to find and click the buttons on the Touch. Additionally, the touch controllers lay better in the hands and feel more natural, so for novice users it seems more beneficial to use the Oculus Touch system.

5.1.2 User-controller interaction

In order to make new users familiar with the unusual interface, virtual hands were implemented and programmed to animate in a manner simulating physical movements of the fingers. One example is pushing in the *trigger button* that lies under the index finger, will make the index finger bend in the application. An added benefit from the hands was our ability to change the proper button terminology, like *trigger button* and *grip button*, to the more relatable names of the fingers closest to the button when instructing them to new users. In addition, a short and to the point introduction to the proper hand and finger placement on the controllers made users more confident when putting on the headset, further shortening the start up time. As shown in Figure 43 from the test performed at MedSim, there was largely a consensus that the interaction with the controllers were not difficult. It is important to notice however, that all the tests in this thesis have been conducted with the developers precent. Without proper guidance, the result might have been significantly different. A proposition to further ease the learning curve is to keep all interaction with the environment to one button, but this have not yet been tested.

5.1.3 Interaction with VR environment

When the users have acclimated themselves to the virtual environment and the feel and usage of the controllers, they want to explore the virtual world to saturate their curiosity. Feedback gathered from tests conducted in the specialization project made it clear that users had trouble finding out which items could interact with, and which buttons they had to press to interact with it. They would preferred to interact with all objects in one way or another. Remedying this, it was decided to consciously implement design constructed to set apart interactable items from non interactable ones. Corresponding with the teachings of Gestalts, items that could be used by pressing the *trigger button* differentiated themselves from other objects by shape and color, often quadratic and red. In the test conducted on the neurosurgeon course, two measurements for how quickly the users were able to learn the system was used. The first measurement was the time it took users to teleport for the first time, the second being the time it took them to correctly teleport to the case selection panel and select a case. For the most part, users wants to move fast and explore, so a short amount time is spent before they teleport for the first time. The average time users spent before choosing the case was 1:24 minute, with a standard deviation of 38 seconds. The findings that users need some time to become familiar with the VR gear is to be expected. For complete novices in VR, problems with understanding how to aim the teleport beam, how close to button controllers need to be, and what button to click are problems developers need to account for. Even so, the large difference between the users in the time used could indicate that the design and way of presenting the usage of the controllers might only appeal to some users.

A recurring concern with VR applications have been VR sickness, which may leave users unable to further use the application. In this project, a key concern in numerous design decisions have been to minimize the sensation of nausea and strain on the users. The guidelines from the inventors of OR [54] have been followed to reduce VR sickness.

The most important to note here are the continuous focus on performance, as John Carmack [49] argues, poor performance could lead to latency and screen flickering. In addition, the implementation of the teleport functionality removes all discrepancy between a users physical and virtual movement. The result can be seen in the MedSim result section, where only one participant reported discomfort during the test.

5.2 Using real world medical imaging

The first research question asks whether or not real world medical imaging can be utilized to support training in virtual reality. From the literature, the use of real world medical data is perceived as having big potential, but difficult to implement [55]. Evaluations on the application developed shows that it is indeed possible to utilize real world medical data to support a learning applications. One tester announced on the surgical seminar that the navigation system and the ultrasound implementations made it feel like he was working on a real patient. The user did not look at the patient when performing the navigation, instead he was only looking at the screen of the apparatus.

The integration of real world medical data in applications using HMD equipment with controllers have shown considerable potential. Being able to interact with models and imagery of the human body in a fully immersive virtual environment enables attendance of the application to explore the data in a new way. Being set in a more structured educational activity, the usage of real world imagery really shines. One of the major difficulties with earlier applications in VR have been the lack of immersion caused by not having proper procedures to execute, but rather having to imagine the procedures performed. This break with a real scenario not only caused a halt in immersion, but also made participants unsure of how to proceed. Thus, utilizing medical imagery to recreate apparatus and scenes that are known and familiar in the scenario will help users perform already existing scenarios, as well as opening new training scenarios focusing more on interaction and interpretation of medical data.
In the Neurology scenario described in results 4.1.2, the learning experience is focused around maneuvering and interpreting medical data, and applying this interpretation to properly complete a given task. The user should also use their knowledge to confirm the data after the task is completed to their satisfaction.

5.2.1 Imaging extracted from videos

Creating the ultrasound machinery from videos using frames proved to be an efficient manner to integrate the data into the application. The approach removes the need for to correctly place and rotate a volume, and the pipeline for extracting the data in real-time is straight forward and understandable. The application limitations to having a proper volume to read from, like the rotating the plane in any manner, or reading data along a axis that is not predefined.

5.2.2 Raw volume data

The implementation of a volume loader system proved to be more time consuming than using video frames and their position. Multiple factors needs to be in order to use such a system in a learning application. There is a need for making sure that the user of the system viewing the volume has control over the tools in a manner that is familiar to them and is still natural to interact with in virtual reality. The amount of implementation work required to make proper use of these data in a meaningful manner is also considerable compared to using the video data. This is also reported in literature on VR applications with similar functinoality [74]. Even so, the outcome provides value for the time spent implementing. Firstly, the implementation presented in this thesis with the ACS-Panel as the prime example provides the users with some exploration possibilites. Secondly, the developers have a larger degree of freedom to implement interaction with the data that closely resembles the real world interaction. The final positive outcome is the ability to increase visualization of the problem. In the neurology application, a teacher could help students to understand the brain better by illustrating it with the model with the complete model of the head loaded. This would not only increase their understanding of the challenge at hand, but could possibly also increase their understanding of similar visualizations in the future.

5.2.3 Medical imaging interaction

The data used in this application was complex in themselves, and proper use and interpretation of MRI and ultrasound images is a field of study in itself. Therefore, it was paramount that the interaction with the data was as similar to the real world interaction as possible, or manipulated as intuitive as possible where this is not achievable. From the user tests, we can see that the average tester spent an average of just under a minute to properly locate the tumor in the ACS-Panels. Having the ability to use finger pointing and the sliders, the ACS-Panels was made to simulate desktop computer and touch screens on the laboratory gear used in surgery respectively. This enabled the testers to interact in a way that was familiar to them. Interaction with the MR-Wand and the ultrasound probe was mostly a success. Some needed assistance on how to use one, or both, but most understood they were supposed to use the gear as they would do in real life. A main issue with the ultrasound probe was the inability to rotate the probe and get a proper rotated view, but the testers had walked through the proper rotation of the probe in the lecture so most did not notice this flaw.

5.3 Research outcome

The goal of this thesis was to determine key functionality features needed to develop a medical learning application using general purpose virtual reality equipment supported by real world medical imaging. This is defined by the main research question (RQ). The development and implementation of the neurosurgical scenario gave valuable insights into answering the question. Firstly, for the user should be able to move freely in the virtual environment and interact with it by using physical movements. The use of physical movements inside the application immerses the user on a higher level than traditionally application can. The increase of immersiveness has the potential to increase the learning curve by making the application entertaining as well as educational. The use of models to represent the users during playtime further improves the immersive feeling. The hands is a good example of important models to have in the application. The animations on the hands further increased the user's awareness on how to interact with the virtual environment, resulting in reduced time needed to be proficient in the application. Another important factor for making a learning application in VR is reducing the amount of VR sickness experienced when playing. If the users gets uncomfortable or feel nauseous while using the application, they will not benefit from using the system, and would rather not use it at all. Several measures have been taken into reducing the common pitfalls in the field of VR sickness, including focus on low latency and consistent movement scheme. The movement scheme is designed in such a way that all types of movement feels natural. Users can move physically as long as they are in their play area. If they want to travel over longer distances, the teleportation function is used.

Another important factor to take into consideration when creating learning applications is how the flow of the scenario should be. It is critical that the users know at all times what actions needs to be performed in order to proceed to the next task. Objects not directly connected to the task at hand, should be hidden or removed to reduce the amount of distractions in the world. First time VR users can easily be distracted by the sensation of VR. It is also important to give users feedback on performance on the different tasks. The ability to toggle between the solution and their own answer on a task as well as receive comments on the solution can further increase the learning outcome from the application.

The integration of real world medical imaging in the virtual world has shown great potential. The medical data made it possible to develop tasks, forcing the users to use their deduction skills to find solutions. The second task in the neurosurgical scenario, the users needs to use their hands to interact with the ACS-Panels by touching it with the index finger, similar to how it is interacted with in the real world.

6 Conclusion and Future Work

In this thesis, virtual hospital environments designed for general purpose VR equipment have been developed to assess its value in an educational setting. The equipment gives end users the ability to use physical movements to interact with the environment. When users are given the power to interact in such a way, they become more involved and engaged. The initial setup and introduction to VR might be time consuming. However, it have been shown that with a well designed user interface, and with careful guidance, it is achievable to have complete novice users interact with the virtual world with some degree of proficiency after minutes. The design of educational applications is a complex tasks, and one way to solve this is to design such an application after a set of tasks that each bring educational value preferably in collaboration with an expert. This is why the application in this thesis have been centered around scenarios with a clear structure and learning goals.

Integrating real world medical data further increased the sense of immersion the user felt when playing the scenario and gave them an exciting new way to interact with MRI and ultrasound images. The integration itself is not arbitrary, and will give marginal educational benefit if not carefully planned and executed. In this thesis, we have shown that users should interact with the data in a way that closely resembles the actual usage in the real world for the usefulness to fully shine through. In this thesis, a scenario where users were forced to use their knowledge of MRI and ultrasound in order to complete a given task properly through experimentation and analysis with the data was implemented. Construction of these types scenarios needs to be done in collaboration with experts in the given field, as multiple subtle nuances that developers miss or overlook might disrupt or even break the experience for their end users.

Collaboration and communication skills are made possible in application by adding a networking layer. Interaction between users in an virtual world, especially when both are using a HMD, offers great learning possibilities and is unquestionable engaging. However, the networking capabilities were not properly tested in this thesis. Time constraints on the user tests, in addition to the sheer amount of testing areas and gear needed made it difficult. The users that got to experience duo-user play through of the neurology scenario at MedSim did however enjoy it.

Based on experimentation and exploration in this project, multiple avenues of future work have been discovered, discussed and even partly implemented for testing purposes. This section is dedicated to list and discuss recommendations for future work.

Improved volume interaction
Moving forward with the integration of medical data in the application, there are two implementations that are highly desirable. The first one is to create a more realistic ultrasound probe by using raw medical data. The probe should have the ability slice the data at a random angle and offset, simulating free hand ultrasound.

The second way of interacting with the data that could prove useful, and was asked for in tests with surgeons, is the ability to view the navigation markers position on the ACS-Panel that is positioned above the bed. Such functionality will immerse the user further as well as opening for more complex scenarios.

AI-Controlled personnel and patients
Further exploring the goal of a training scenario being low maintenance and easily accessible, it was discussed that a implementation of participants in the scenarios being replaced by artificial intelligence. The core idea is to have implemented patients and personnel that might step in for users and perform a believable act in their roles. Early exploration in this project proved this concept to fall out of the scope, and the exploration is therefore considered future work. In order to make such an application work, one should however have some core concepts implemented.

In order for an AI to simulate conscious decisions, it should respond to voice commands or keywords. There already exists tools making this possible, but there is an overhead setting them up, and in their current state, they are only usable in English. Unity ships itself with a voice recognition tool that is quite simple to set up and get working. The problem with this software is the quality of the recognition, especially when considering any recognition of more than a few commands. The limitations lies in the system it is abstracted from, which is Windows' own personal helper Cortana, which is designed to recognize and respond to simple commands.

However, speech recognition is a field of heavy focus for large companies presently. Both Google [75] and Amazon [76] provides cloud based recognition software, which is the same software they use in their representative interactable home solutions, Google Home and Amazon Echo.

The AI should respond in a natural manner to voice input from the user. In its current state, human simulation is still a hard problem in computer science, and given the scripted nature of the scenarios, it is possible to *hard-code* behaviour and responses to simulate a conscious decision.

Improved feedback system
Moving forward with this application, it should be a natural iteration to implement some kind of automatic feedback and a score system. Today, the application provides an expert-set solution for the scenarios, and the user may asses his or her success in the scenario based on the solution. While this solution is valuable in itself, it is really the minimal valuable product. During development, the team came up with different ideas for more complex feedback systems.

Following the theories about serious games, the users will be more invested in the application if there is some sort of numerical feedback that in a meaningful way grades the performance of the user. Implementing such functionality would require to a consultant with expertise in the given field who can declare what metrics should be assessed and what weights the different metrics should have. A score system could be visible after the user is done setting up the patient, and could be showed in VR or outside. With such a metric, the developers might also set pass or fail values, giving the users further feedback on what is expected of them.

More important in educational applications is the ability to give the users of learning systems constructive and informative feedback. An example in our application is if the user marks the entry point at a point in the brain where there lies a significant amount of critical nerves, he should be informed about why the point would be bad for the patient. Personalized feedback based on different metrics like the placement of the torus or the angle of the neck of the patient could be generalized so that the feedback gives the most amount of value possible.

Informational videos made by instructors could also be implemented to work with different types of similar applications so that instructors can record their own instruction videos and present them to their students before or after the scenarios have been played.

7 Appendix

A Networking functionality

A.1 Introduction and choice of framework

In this appendix, we will discuss the implementation of networking functionality to enable communcation and collaboration possibilities in VR applications, as well as cross platform functionality. The possibilities offered by networking funcitonality is exciting, and is therefor well worth exploring.

Networking framework
Instead of developing the network functionality from ground up, an already established networking framework was chosen. The choice of framework were, however, an important one as it would decide large parts of the software architecture. Following is a discussion of the three most prominent alternatives and the major differences between them.

Unity networking
Unity's own networking module might seem the obvious choice to many. It is well integrated in the Unity core and very well documented. The framework supports the most commonly used functions, like Remote Procedure Calls (RPCs), synchronized variables, and commands. The server technology is based on a client-server pattern, meaning one of the clients needs to be a server or host that the rest of the clients can connect to.

Photon Unity Network(PUN)
PUN is a free tool offered through the asset store by *Exit Games* [77]. The networking engine allows developers to decide whether the game application should be hosted on a cloud server or allows one of the client to act as a host, similarly to Unity Networking. PUN does not handle any game logic, and knows nothing of what is happening to the game other than the variables sent over the network. This forces the programmers to use less network traffic, ultimately leading to better network performance. This might cause differences between the states in which the applications are in, but there are methods to remedy this. PUN, like Unity networking, offers RPCs and synchronized variables.

Forge Remastered
Forge remastered is also found in the Unity asset store [78], but compared to Unity networking and PUN, costs $75. Forge uses a third approach, which strongly emphasizes the need for a dedicated server, which may not be a user, that the developers need to set up and keep running themselves. The implementation however, seems to include little to no setup and the performance is reportedly good. Forge also supports all commands needed for this project mentioned above, but has somewhat limited documentation compared its competitors in this comparison.

Conclusion

For this project, the initial cost for Forge, as well as the need of a dedicated server, made us turn to one of the other solutions. Unity Networking was our initial choice given the generous documentation and integration. However, some problems occurred during testing with VR, mainly considering performance. Given that a VR application needs to perform consistently and with low latency, our choice finally landed on PUN. PUN proved to have good documentation and an active community, and providing a robust performance.

A.2 Implementation

Developing an networking application in Photon Unity Networking (Photon), was a tasks that required plentiful of implentation choices and coding. This will be described in this chapter.

The server used in Photon can either be hosted on Photon's cloud service, or it can be hosted by a client. In this project, it was decided to make the first client connecting the server, known as the *Master client*. Master/slave communication is used in the project, and is a model where one device has unidirectional control over the other clients.

The PUN is constructed in a room like manner, making it possible to create several instances of a game in different rooms. When a user connects to PUN, they first connect to Photon Cloud and thereafter connects to one a room. Once connected to a room, the communication is between the master client (the client acting as the server) and the slave client (the one joining the room).

PhotonView
PhotonView is a script component enabling communication between clients through methods such as Remote Procedure Calls (RPCs) and OnSerializePhotonView. The script needs to be attached to a GameObject to make it communicate through the network. Figure 47 illustrates how the script looks when it is added as a component to a GameObject in the Unity editor. As shown on the figure, parameters can be edited after adding the component to the GameObject. The first parameter, *Owner* signifies who owns the GameObject. The owner of the GameObject is the one communicating to the other users about the GameObject. The parameter has three options: *Fixed*, *Takeover*, *Request*.

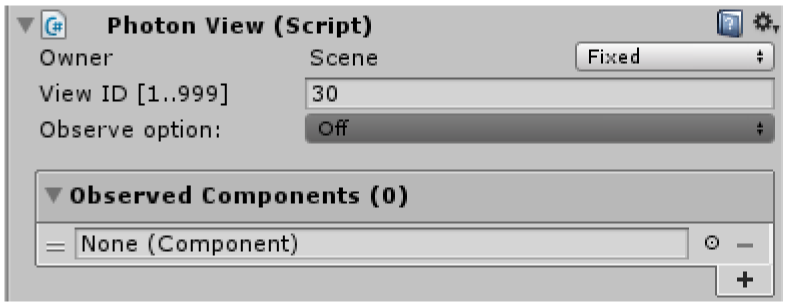


Figure 47: Illustration of the PhotonView script in Unity

If the owner is fixed, that means that the master client is the only one who sends out updates about the GameObject. The other users can only observe what the master client is doing. If *Takeover* is selected, the GameObject can change owner at runtime. This enables all users interacting with the GameObject to send updates, instead of only the master client. The last option, *Request* allows users to ask for permission to take ownership of a GameObject. The master client then either accepts or declines the request.

The parameter *View ID* is a unique identifier for the GameObject. As seen on the Figure, the ID spans from 1 to 999.

The *Observe option* parameter has four options: *Off*, *Reliable Delta Compressed*, *Unreliable* and *Unreliable On Change*. This parameter decides the transmission rate of the GameObject. *Observed Components* tells PhotonView which scripts it should transmit.

Off means that no data will be sent from the GameObject automatically. This is ideal if the GameObject needs to send irregular data transmission. It can send these transmission through RPC. *Reliable Delta Compressed* will compare the data last received by the client to the current state. If the observed data has not changed, then no data will be sent. If for instance the position of the GameObject has changed, but the rotation has not, only the position will be sent over the network. This method reduces the strain on the bandwidth by only transmitting the changed data. To be certain that all packets have been received and thus is reliable, the sender will continue to send packages until confirmation that all packets have been received. The new packets awaiting sending is placed in a waiting buffer until the previous packets have been delivered. The *Unreliable* mode on the other hand, will not receive any confirmation about the packages. Since the sender cannot know what data the receiver has, the sender sends the whole state in each update cycle. *Unreliable On Change* will look for changes in each update cycle. If no changes are found since last update cycle, it will stop transmitting. If changes are detected, it will begin to transmitt updates again.

Photon Transform View

Photon Transform View is a script for synchronizing position, rotation and scale values of a GameObject. The script needs to be placed in the *Observed Components* parameter in the PhotonView script for it to synchronize the data.

Photon Animator View
Photon Animator View is a script for synchronizing animation over the network. It too, needs to be added to the PhotonView's *Observe Components* for it to synchronize the data.

Remote Procedure Calls (RPCs)
Not all GameObjects are required to update all the time, but only on certain events. Instead of transmitting unnecessary data, RPCs can be used. RPC enables developers to create methods which are executed on certain events. An example on how a RPC works is the interaction with a button. When a user presses the button, the method *photonView.RPC("ButtonClicked", PhotonTargets.All, buttonID);* will be executed. The first parameter describes which RPC method should run. Each RPC method is unique and is accessed through it's unique name. The second parameter tells PhotonView who should receive the notification. It is possible to set the receivers to be *others*, which means that the notification should be sent to all but the one who is triggering the event, or *MasterClient* which mean that only the master client will receive the notification. In this case, all users, including the one sending the message executes the method. The last parameter on the RPC is the parameters sent to the RPC. In this example, the buttonID will be sent.

Network Controller
The *Network Controller* script is responsible for network connectivity. First the user connects to the Photon Cloud with the unique AppID of the project. Every Photon project have their own unique AppID. When the user is connected to Photon Cloud, it asks the server how many active rooms there are on the server. If the server has zero active rooms, the user creates a new room with the name *VirSamRoom*. The user creating the room will be the master client i.e. hosting the communication channel. If the number of rooms is higher than zero, the user connects to the *VirSamRoom*. When connecting to an already established room, the client will become a slave client. When a new user connects to the room, their solutions to the scenarios will be compared to the solutions of the other users already connected. This is to be certain that all users have the same solution.

The *OnJoinedRoom* method is called when a user joins the room. The user is given an unique player id which is used to identify the different users over the network. Afterwards the method checks whether or not the user is using VR equipment. If the user is using VR, a prefab called *NetworkedVRPrefab* will be instantiated. The prefab *NetworkedPlayerPC* will be instantiated if the user is playing in desktop mode. The method further sends a RPC to the other users telling them a new user has joined with the given player- and device id. The device id determines whether or not the user is using VR.

The *NetworkedVRPrefab* contains the models for surgeon hands and head. The prefab is used to transfer user movement over the network. Each model contains three essential scripts: *Photon View*, *Photon Transform View* and *Copy VR Controller*

Script. The last scripts mimics the position and rotation of the GameObjects corresponding to hands and head, making it possible for *Photon Transform View* to send the position and rotation of the user over the network to other users. Similarly, the *NetworkedPlayerPC* prefab has a script called *Copy Desktop controller Script*. The prefab also have the *Photon Animator View* to synchronize animation over network.

Networked Items
Networked Items is responsible for synchronizing user events over the network. As mentioned previously, every item in the application needs the PhotonView component to communicate over the network. All interactable items have their owner property set to *Takeover*, meaning any users can take ownership over the item. When a user interacts with an interactable GameObject, the *Networked Items* script first declares that the interacting user is now the owner of the given GameObject. Secondly, it sends a RPC to tell the other users who picked up the item, and in which hand, so it can be synchronized over the network.

A.3 Networking results and discussion

Moving away from Second Life, the network communication between users had to be re-implemented in order to fully utilize the potential presented in the VUH framework. The exploration and implementation of this functionality have presented new and interesting use cases for collaboration that demands further exploration.

Actor-observer communication
With the implementation of cross-platform utilities, connecting to a session using either VR equipment or using the desktop version there are multiple ways to play out the neurology scenario. One way is a student walking through the scenario using VR equipment while a teacher is present in the room from a desktop version, able to offer advice and corrections. Another way is a teacher executing the case with student observers that are there to learn.

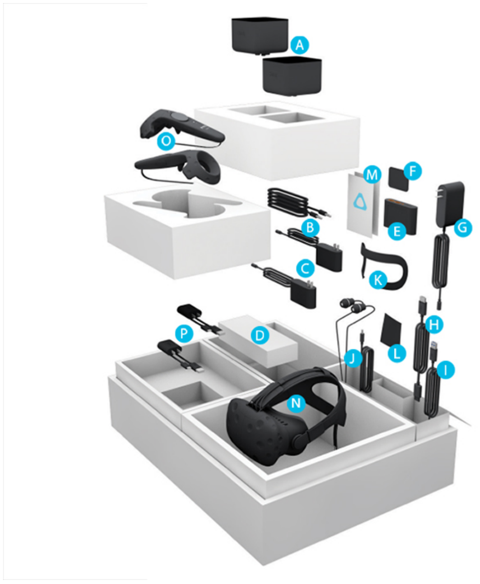
Actor - actor communication
The network functionality that offers the greatest promise is the ability for two VR users to communicate and interact with each other in the virtual world. The VR users are able to pass items with each other, and all actions they perform in the VW will be observed by the other actors. This offers great functionality for the neurology scenario as well as opens for a whole new specter of applications.

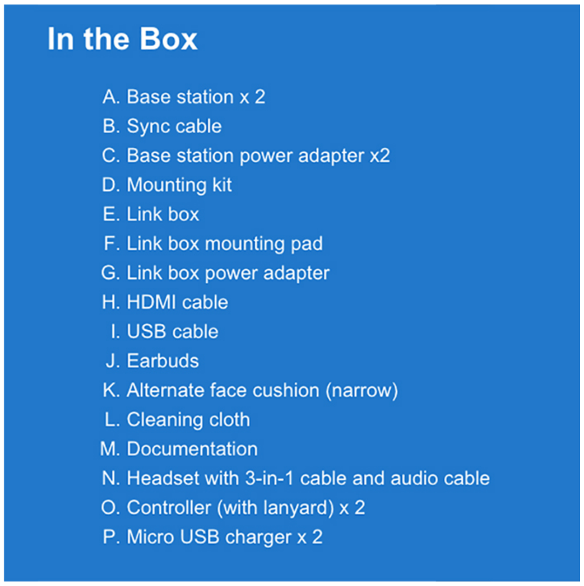
In the neurology scenario, the users might give each other specific items, like the ultrasound probe or the marker. This will allow for a sense of great immersion. Using item transfer functionality in other scenarios opens for usage in applications that simulates nurse-surgeon communication before, during and after a surgery.

B Setup of HMD

B.1 HTC Vive Setup

Figure 48 illustrates the content of HTC Vive consumer edition. The base stations (A) are responsible for tracking user movement. It tracks both the headset and game controllers. For best result, the base stations need to be placed diagonally in a room. They should also be mounted high enough so they can track the whole play area, including the floor.





In the Box

A. Base station x 2
B. Sync cable
C. Base station power adapter x2
D. Mounting kit
E. Link box
F. Link box mounting pad
G. Link box power adapter
H. HDMI cable
I. USB cable
J. Earbuds
K. Alternate face cushion (narrow)
L. Cleaning cloth
M. Documentation
N. Headset with 3-in-1 cable and audio cable
O. Controller (with lanyard) x 2
P. Micro USB charger x 2

Figure 48: Image of the HTC Vive equipment in box

The documentation contains a guide on how to set up the play area. It tells the user to download and install the HTC Vive setup software from `https://www.vive.com/us/setup/`. The software helps the user configure the play area in addition to a tutorial on how the buttons on the controller works. The software also downloads the required software to run HTC Vive: Steam software and Vive software. The user most create an user account for HTC Vive and Steam to run VR applications. The setup software contains gamification features making the setup feel more like playing a game than a chore. An example of gamification used in the setup tool is when the user marks the play area. The user is told to go to a corner in the physical room and press the trigger button on the game controller. While holding the trigger button down, the user has to walk to the remaining three corners in the play area. The progress of the user is illustrated on the software tool. After the user has marked the corners, the software calculates the size of the play area. The minimum size of the play area is 2 x 1.5 meters, and maximum is 5 meters between the base stations.

B.2 Oculus Rift + Touch Setup

Setting up OR are done much in the same way as the Vive. In the OR box, you have the headset and one basestation that needs be connected to the computer. The controllers are sold separately, and are therefore contained in a different box. In order to use the controllers with the headset, you need to connect an additional basestation. The basestations should both be placed besides the computer screen, with at least 1.5m separating them. When everything is connected, the Oculus software needs be downloaded and installed. It should be mentioned that this software has a requirement to be online in order for it to function properly, and updates regularly, rendering the software useless until the it is done. A play area needs to be set up for the OR as well, and has a strictly enforced requirement. Then, the floor needs to be calibrated relative to the position of the base stations.

Now, you are ready to use the headset for applications made specifically for Oculus. In order to use the applications made for this thesis, as well as applications released for *Steam*, you need to install a software called *Steam VR*. This software will require a setup of the room and floor calibration once more.

OR suffers from some issues when the user are facing away from the base stations, but have implemented functionality supporting addition of more base stations. This can be done by simply plugging in another base station and placing it so that it covers the users when they are facing away from the two others.

C How to build and run Unity projects

To build a Unity project means creating an .exe file that is executable independent of Unity. Developers builds Unity projects by clicking on *file* in Unity then *Build Settings*. Then the developer is prompted to select which scenes to include in the build. The scenes can be manually dragged from the project folder. The .exe file plays the scenes in sequential order, meaning the first scene, scene 0 will be the first scene played when running the application. In the build settings, it is possible to set different parameters such as target platform and architecture. In this project, the target platform is Windows. To build the project, the *Build* button is pressed. The developer is then prompted to select where they want the application to be located.

The application created from building a Unity project consists of the .exe file and _*Data* folder. The _Data folder contains the resources required to run the application, such as models and scenes. To run the application, the user clicks on the .exe file.

D Using Volume Viewer

After importing the Volume Viewer into Unity, some project settings needs to be edited for the package to work correctly. Firstly, *VolumeViewer* needs to be added to the User Layers. This is done by clicking on the *Layers* button on the top right in the Editor and click *Edit Layers*. Secondly, the shaders from the package needs to be included in the build settings. The shaders are added to the build settings by clicking Edit -> Project Settings -> Graphics. The section *Always Included Shaders* shows what shaders are included. to include the VolumeViewer shaders, one first has to increase the number of shaders included. There are in total 8 new shaders to be added. Once the array is increased, the user drags the shaders to the open slots. Finally, for the VolumeViewer to work, the script *Volume Renderer* needs to be added to the cameras. In this the use can set how many GameObject the camera should be able to see, usually one. After setting the number of GameObjects, an array will appear. The last thing to do is drag the GameObject inside the open array.

E User feedback

Tilbakemelding Virtuelt sykehus

1. **I found it difficult to move in VR**
Markér bare én oval.

- Strongly disagree
- Disagree
- Neutral
- Agree
- Strongly agree

2. **I found it difficult to use the controllers**
Markér bare én oval.

- Strongly disagree
- Disagree
- Neutral
- Agree
- Strongly agree

3. **I found it natural to look around in the virtual environment**
Markér bare én oval.

- Strongly disagree
- Disagree
- Neutral
- Agree
- Strongly agree

4. **I found the interaction with the environment to be intuetive**
Markér bare én oval.

- Strongly disagree
- Disagree
- Neutral
- Agree
- Strongly agree

5. **I experienced a sensation of immersiveness using the VR-headset**
Markér bare én oval.

- ◯ Strongly disagree
- ◯ Disagree
- ◯ Neutral
- ◯ Agree
- ◯ Strongly agree

6. **I experienced physical discomfort during the test**
Markér bare én oval.

- ◯ Strongly disagree
- ◯ Disagree
- ◯ Neutral
- ◯ Agree
- ◯ Strongly agree

7. **I expect VR-technology to have a potential in training of medical students and other professionals**
Markér bare én oval.

- ◯ Strongly disagree
- ◯ Disagree
- ◯ Neutral
- ◯ Agree
- ◯ Strongly agree

8. **It felt natural to interact with and observe real medical image data in Virtual reality**
Markér bare én oval.

- ◯ Strongly disagree
- ◯ Disagree
- ◯ Neutral
- ◯ Agree
- ◯ Strongly agree

F Video presentations

In the following the table, the URLs to the different videos that shows the functionality developted in this thesis is found:

Number	**Name**	**URL**
1	Core functionality of thesis	`https://goo.gl/5djsSd`
2	Main menu and tutorial	`https://goo.gl/gJh99h`
3	Explored functionality	`https://goo.gl/GmniKQ`

Table 1: Table of URLs

References

[1] Unity. Execution order of event functions. https://docs.unity3d.com/Manual/ExecutionOrder.html, 2016.

[2] Alan Zucconi. A gentle introduction to shaders. https://unity3d.com/learn/tutorials/topics/graphics/gentle-introduction-shaders, 2015.

[3] Michael Zyda. *From Visual Simulation to Virtual Reality to Games*. IEEE Computer Society, 2005.

[4] Heyden R. Sternthal E. Wiecha, J. and M Merialdi. Learning in a virtual world: Experience with using second life for medical education. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2821584/, 2010.

[5] Wilson K. Morra D. Beard, L. and J. Keelan. A survey of health-related activities on second life. https://www.ncbi.nlm.nih.gov/pubmed/19632971, 2009.

[6] Hetherington L. Boulos, M.N.K. and S Wheeler. An overview of the potential of 3-d virtual worlds in medical and health education. https://www.ncbi.nlm.nih.gov/pubmed/18005298, 2007.

[7] Mora C. E. Añorbe-Díaz B. & González-Marrero A. Martín-Gutiérrez, J. Virtual technologies trends in education. *EURASIA Journal of Mathematics, Science and Technology Education*, (13(2)):469–486, 2017.

[8] Sean Hollister. Why microsoft can't say when its incredible hololens will become a reality. https://www.cnet.com/news/microsoft-hololens-no-release-date-kudo-tsunoda-explanation/, 2017.

[9] Gartner. Gartner's 2016 hype cycle for emerging technologies identifies three key trends that organizations must track to gain competitive advantage. http://www.gartner.com/newsroom/id/3412017, 2016.

[10] C. Li, W. Liang, C. Quigley, Y. Zhao, and L. F. Yu. Earthquake safety training through virtual drills. *IEEE Transactions on Visualization and Computer Graphics*, 23(4):1275–1284, April 2017.

[11] A case study - the impact of vr on academic performance. 2016.

[12] Max Hoffmann, Lana Plumanns, Laura Lenz, Katharina Schuster, Tobias Meisen, and Sabina Jeschke. *Enhancing the Learning Success of Engineering Students by Virtual Experiments*, pages 394–405. Springer International Publishing, Cham, 2015.

[13] Fominykh M.-Hansen A. Rasmussen G.-Sagberg L.M. Lindseth F. Kleven N.F., Prasolova-Førland E. Training nurses and educating the public using a virtual operating room with oculus rift. Thwaites H, Kenderdine S, Shaw J (eds.) International Conference on Virtual Systems & Multimedia (VSMM), Hong Kong, December 9–12, pp. 206–213. IEEE, New York, NY, 2014.

[14] Adam Hamrol, Filip Górski, Damian Grajewski, and Przemysław Zawadzki. Virtual 3d atlas of a human body – development of an educational medical software application. *Procedia Computer Science*, 25:302 – 314, 2013. 2013 International Conference on Virtual and Augmented Reality in Education.

[15] Next Galaxy Corp. Next galaxy partners with vr healthnet to bring virtual reality to healthcare. http://www.prnewswire.com/news-releases/next-galaxy-partners-with-vr-healthnet-to-bring-virtual-reality-to-healthcare-30012 html, 2015.

[16] T.M. Lewis, R. Aggarwal, N. Rajaretnam, T.P. Grantcharov, and A. Darzi. Training in surgical oncology – the role of {VR} simulation. *Surgical Oncology*, 20(3):134 – 139, 2011. Special Issue: Education for Cancer Surgeons.

[17] U. Kühnapfel, H.K. Çakmak, and H. Maaß. Endoscopic surgery training using virtual reality and deformable tissue simulation. *Computers Graphics*, 24(5):671 – 682, 2000. Dynamic Medical Visualization (special topic).

[18] Chung KC. Kotsis SV. Application of see one, do one, teach one concept in surgical training. *Plastic and reconstructive surgery.*, 131(5), 2013.

[19] Pelargos et. al. Utilizing virtual and augmented reality for educational and clinical enhancements in neurosurgery. http://ac.els-cdn.com/S0967586816303162/1-s2.0-S0967586816303162-main.pdf?_tid=d5515564-527a-11e7-a96e-00000aacb362&acdnat=1497607497_89518ea873cf10037093d01aa7043374, 2016.

[20] CAE Healthcare. Neurotouch. https://caehealthcare.com/surgical-simulation/neurovr, 2017.

[21] Sevdali . Nick Paige John-Zevin Boris Aggarwal Rajesh MD Grantcharov Teodor Jones Daniel B. Stefanidis, Dimitrios. Simulation in surgery: What's needed next? *Annals of Surgery*, 261(5), 2015.

[22] Paweł Kazimierz Buń, Radosław Wichniarek, Filip Górski, Damian Grajewski, Przemysław Zawadzki, and l Poznan U. Possibilities and determinants of using low-cost devices in virtual education applications. *EURASIA Journal of Mathematics Science and Technology Education*, 13(2):381–394, 2017.

[23] A.S. Mathur. Low cost virtual reality for medical training. *EURASIA Journal of Mathematics Science and Technology Education*, pages 345–346, 2015.

[24] Razer. Razer hydra. https://www2.razerzone.com/au-en/gaming-controllers/razer-hydra-portal-2-bundle, 2017.

[25] Virtual Medical Coaching. Virtualmedicalcoaching - because fantastic education doesn't just happen. http://www.virtualmedicalcoaching.com/, 2016.

[26] Henry Lane. Schools of the future – using vive to provide complete medical imaging education in vr. https://blog.vive.com/us/2017/03/13/schools-of-the-future-using-vive-to-provide-complete-medical-imaging-education-in-v 2017.

[27] Judith Amores, Xavier Benavides, and Pattie Maes. Showme: A remote collaboration system that supports immersive gestural communication. In *Proceedings of the 33rd Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems*, CHI EA '15, pages 1343–1348, New York, NY, USA, 2015. ACM.

[28] Wiley; Maes Pattie Greenwald, Scott W.; Corning. Multi-user framework for collaboration and co-creation in virtual reality. *12th International Conference on Computer Supported Collaborative Learning (CSCL)*, 2017.

[29] Leslie Jarmon, Tomoko Traphagan, Michael Mayrath, and Avani Trivedi. Virtual world teaching, experiential learning, and assessment: An interdisciplinary communication course in second life. *Computers & Education*, 53(1):169 – 182, 2009.

[30] John Wiecha, Robin Heyden, Elliot Sternthal, and Mario Merialdi. Learning in a virtual world: Experience with using second life for medical education. *J Med Internet Res*, 12(1):e1, Jan 2010.

[31] Ekaterina Prasolova-Førland, Aslak Steinsbekk, Mikhail Fominykh, and Frank Lindseth. *Smart Virtual University Hospital: Nursing and Medical Students Practicig InterProfessional Team Communication and Collaboration*. Norwegian University of Science and Technology, 2016.

[32] Allison K. Herrera, F. Zeb Mathews, Marilyn R. Gugliucci, and title = Corina Bustillos.

[33] Christian Moro, Zane Stromberga, Athanasios Raikos, and Allan Stirling. Combining virtual (oculus rift &#38; gear vr) and augmented reality with interactive applications to enhance tertiary medical and biomedical curricula. In *SIGGRAPH ASIA 2016 Symposium on Education: Talks*, SA '16, pages 16:1–16:2, New York, NY, USA, 2016. ACM.

[34] The Body VR. Revolutionizing healthcare through virtual reality. http://thebodyvr.com/, 2017.

[35] Medis Media Pty Ltd. All-in-one solution for learning clinical, topographic and systems-based anatomy. http://www.3dorganon.com/, 2016.

[36] Kimberly Gerweck. Dextroscope® changes the neurosurgical planning paradigm 3d virtual reality system. http://www.kunnskapssenteret.no/verktoy/sjekkliste-for-trygg-kirurgi-who, 2006.

[37] Mads Soegaard. Interaction design foundation. https://www.interaction-design.org/literature/book/the-glossary-of-human-computer-interaction/affordances, 2002.

[38] Mads Soegaard. Interaction design foundation. https://www.interaction-design.org/literature/book/the-glossary-of-human-computer-interaction/gestalt-principles-of-form-perception, 2002.

[39] Jan Cannon-Bowers. *Serious Game Design and Development: Technologies for Training and Learning: Technologies for Training and Learning*. IGI Global, 2010.

[40] Unity. Everything you need to succeed in games and vr/ar. https://unity3d.com/, 2017.

[41] Epic Games. Unreal engine 4.16 released! https://www.unrealengine.com/en-US/blog, 2017.

[42] Crytek GmbH. Why choose cryengine? https://www.cryengine.com/, 2017.

[43] Unify Community. Unityscript versus javascript. http://wiki.unity3d.com/index.php/UnityScript_versus_JavaScript, 2014.

[44] Thomas W. Malone. *What Makes Things Fun To Learn? Heuristics for Designing Instructional Computer Games*. Xerox Palo Alto Research Center, 1980.

[45] Valve. Steamvr plugin. https://www.assetstore.unity3d.com/en/#!/content/32647, 2017.

[46] Thestonefox. Welcome to vrtk. https://vrtoolkit.readme.io/docs, 2017.

[47] Nils Fredrik Kleven, Ekaterina Prasolova-Førland, Mikhail Fominykh, Arne Hansen, Guri Rasmussen, Lisa Millgård Sagberg, and Frank Lindset. *Training Nurses and Educating the Public Using a Virtual Operating Room with Oculus Rift*. NTNU, 2015.

[48] Multiple. Vr-headsets. https://en.wikipedia.org/wiki/Virtual_reality_headset, 2015.

[49] OculusRift-Blog.com. Virtual reality, latency mitigation. http://oculusrift-blog.com/john-carmacks-message-of-latency/682/.

[50] Aaron Souppouris. How htc and valve built the vive. https://www.engadget.com/2016/03/18/htc-vive-an-oral-history/, 2016.

[51] Joseph j. LaViola Jr. A discussion of cybersickness in virtual environments. http://dl.acm.org/citation.cfm?doid=333329.333344, 2000.

[52] David Whittinghill, Bradley Zeigler, Tristan Case, and Brenan Moore. Nasum virtualis: A simple technique for reducing simulator sickness. http://gtp.autm.net/technology/view/70738, 2015.

[53] Ajoy S Fernandes and Steven K Feiner. Combating vr sickness through subtle dynamic field-of-view modification. http://www.cs.columbia.edu/2016/combating-vr-sickness/images/combating-vr-sickness.pdf, 2016.

[54] ALEX. Oculus rift vr motion sickness – 11 ways to prevent it! http://riftinfo.com/oculus-rift-motion-sickness-11-techniques-to-prevent-it, 2015.

[55] Jan Egger, Markus Gall, Jürgen Wallner, Pedro Boechat, Alexander Hann, Xing Li, Xiaojun Chen, and Dieter Schmalstieg. Htc vive mevislab integration via openvr for medical applications. *PLOS ONE*, 12(3):1–14, 03 2017.

[56] XinReality. Oculus touch. https://xinreality.com/wiki/Oculus_Touch, 2017.

[57] Microsoft. C#. https://docs.microsoft.com/en-us/dotnet/csharp/csharp, 2017.

[58] Blender. Open source 3d creation. free to use for any purpose, forever. https://www.blender.org/, 2017.

[59] Autodesk Inc. Programvare for 3d-modellering, animasjon og rendering. https://www.autodesk.no/products/3ds-max/overview, 2017.

[60] MonoDevelop Project. Cross platform ide for c, f and more. http://www.monodevelop.com/, 2017.

[61] Microsoft. Unparalleled productivity for any dev, any app, and any platform. https://www.visualstudio.com/vs/whatsnew/, 2017.

[62] Paint.NET. Paint.net - free software for digital photo editing. https://www.getpaint.net/.

[63] Tarald Gåsbakk and Håvard Snarby. Master's thesis project. https://github.com/Haavarsn/Master-Thesis, 2017.

[64] Valve. The lab. http://store.steampowered.com/app/450390/The_Lab/, 2016.

[65] Wikipedia. A* search algorithm. https://en.wikipedia.org/wiki/A*_search_algorithm, 2017.

[66] Vertigo Games. Operating room. https://www.assetstore.unity3d.com/en/#!/content/18295, 2016.

[67] 3D Everything. Hospital room. https://www.assetstore.unity3d.com/en/#!/content/57399, 2017.

[68] FFmpeg. A complete, cross-platform solution to record, convert and stream audio and video. https://ffmpeg.org/, 2017.

[69] VirSam. Gynekologicase. http://virsam.no/case_gynekologi.html, 2016.

[70] nerveink. Full anatomy model free download. http://www.zbrushcentral.com/showthread.php?169189-Full-anatomy-model-free-download&p=956870&infinite=1#post956870, 2006.

[71] David Nikel. Our story. http://technoport.no/content/306/Our-Story, 2015.

[72] Medisinsk simulatorsenter. Om oss. ://www.simulatorsenteret.no/om-oss/, 2017.

[73] Usgt. Ultrasound imaging guided treatment. http://usigt.org/, 2017.

[74] Jan Egger, Markus Gall, Jürgen Wallner, Pedro de Almeida Germano Boechat, Alexander Hann, Xing Li, Xiaojun Chen, and Dieter Schmalstieg. Integration of the htc vive into the medical platform mevislab, 2017.

[75] Google. Powerful speech recognition. https://cloud.google.com/speech/, 2017.

[76] Amazon. Alexa voice service. https://developer.amazon.com/alexa-voice-service, 2017.

[77] Exit Games. Photon unity networking free. https://www.assetstore.unity3d.com/en/#!/content/1786, 2017.

[78] Inc. Bearded Man Studios. Forge networking remastered. https://www.assetstore.unity3d.com/en/#!/content/38344, 2017.