NTNU
Norwegian University of
Science and Technology

Master's thesis

2019

Sindre Windsrygg

Master's thesis

NTNU
Norwegian University of
Science and Technology
Faculty of Information Technology and Electrical
Engineering
Department of Computer Science

Sindre Windsrygg

Learning Algorithms in Virtual Reality as Part of a Virtual University

June 2019

NTNU
Norwegian University of
Science and Technology

NTNU

Norwegian University of
Science and Technology

Learning Algorithms in Virtual Reality as Part of a Virtual University

Sindre Windsrygg

Computer Science
Submission date: June 2019
Supervisor: Frank Lindseth

Norwegian University of Science and Technology
Department of Computer Science

Sindre Windsrygg

Learning Algorithms in Virtual Reality as Part of a Virtual University

Master's thesis in Computer Science
Supervisor: Frank Lindseth
June 2019

Sindre Windsrygg

Learning Algorithms in Virtual Reality as Part of a Virtual University

Master's thesis in Computer Science
Supervisor: Frank Lindseth
June 2019

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

NTNU
Norwegian University of
Science and Technology

NTNU
Norwegian University of
Science and Technology

Learning Algorithms in Virtual Reality as Part of a Virtual University

27. June 2019

Course code
TDT4900

Course name
Master Thesis

Faculty of Information Technology and Electrical Engineering
The Department of Computer Science
NTNU Campus Gløshaugen
Trondheim

Author
Sindre Windsrygg
sindrw@stud.ntnu.no

Supervisor
Frank Lindseth
frankl@ntnu.no

Abstract

In education, new technology can provide better conditions for learning. Virtual reality has over the last few years become very popular, mainly by gamers, but has also started to become a useful tool for professionals in the industry. What this thesis wants to explore, is whether virtual reality can be utilized to learn complex concepts taught at a University level, and whether it can help to improve the learning process of such materials. This was done by implementing a virtual reality application, which can introduce the teaching material to the user through various ways of visualization and interaction, such that the user learns through putting their hands to work in a virtual environment. The application has multiple difficulty levels which provides appropriate support in order to challenge the user. In total, the application contains six algorithms that can be explored. Multiple videos were created in order to present all the functionality of the application, which can be found by clicking this link.
The feedback and observations from the user tests showed big interest for virtual reality in the future, and having such an application as a supplementary tool, but some mixed feelings about replacing traditional teaching methods with virtual reality. It was concluded that virtual reality shows positive signs for improving the learning process of complex concepts, but more testing is required in order to prove this as a fact.

Sammendrag

I utdanning kan ny teknologi gi bedre forhold for læring. Virtuell virkelighet (Virtual Reality) har de siste årene blitt veldig populært hovedsakelig i spillindustrien, men har også begynt å bli brukt som et verktøy av fagfolk i bransjen. Denne tesen ønsker å utforske om virtuell virkelighet kan benyttes for å lære komplekse begreper som er undervist på universitetsnivå kan bidra til å forbedre læringsprosessen av slikt materiale. Dette ble gjort ved å implementere en applikasjon for virtuell virkelighet som kan introdusere undervisningsmaterialet til brukeren gjennom ulike måter å visualisere og interagere med, slik at brukeren lærer seg dette ved å bruke hendene sine i større grad i et virtuelt miljø. Applikasjonen har flere vanskelighetsgrader som gir passende støtte til brukeren slik at de kan utfordre seg selv. Det ble utviklet seks algoritmer for applikasjonen, som kan utforskes av brukeren. Flere videoer ble laget for å presentere innholdet av applikasjonen, og disse kan bli funnet ved å trykke denne linken.
Tilbakemeldingene og observasjonene av brukertestene viser at det er stor interesse for virtuell virkelighet i fremtiden, og for å ha et slikt verktøy som ekstra hjelp i studietiden. Det var litt blandede følelser angående å erstatte de mer tradisjonelle undervisningsmetodene med virtuell virkelighet. Det ble konkludert med at virtuell virkelighet viser positive tegn for å forbedre læringsprosessen for komplekse begreper, men at mer testing er nødvendig for å bevise dette som et faktum.

Acknowledgements

I would first like to thank my supervisor, professor Frank Lindseth of the Faculty of Information Technology and Electrical Engineering at NTNU. He provided me with necessary hardware and gave me support for steering both the specialization project and master thesis in the right direction.

I would also like to thank my laboratory mates at IDI's XR-lab. I would also give a thanks to all those who set off their time and came to the lab in order to test and evaluate my VR application.

Finally, I must express my very profound gratitude to my family - Tormod, Bodil, and Aleksander, for providing me with continuous encouragement and support throughout all these years. I would not have been able to achieve this accomplishment without them. Thank you.

Contents

List of Tables

List of Figures

Abbreviations

XR	=	X Reality
VR	=	Virtual Reality
AR	=	Augmented Reality
HMD	=	Head-Mounted Display
FPS	=	First-Player Shooter
SUS	=	System Usability Scale
CS	=	Computer Science
RQ	=	Research Question
UI	=	User Interface
OLED	=	Organic Light Emitting Diode
BFS	=	Breadth-First Search
DFS	=	Depth-First Search
FIFO	=	First-In-First-Out
LIFO	=	Last-In-First-Out

Chapter 1

Introduction

Virtual reality (VR) has the last few years brought attention to the immersive experience it provides the user. The mainstream for VR has been gaming since the announcement of Oculus Rift, but the potential for education, training, and industry has also started to gain attention.

1.1 Motivation

Current Situation

The traditional teaching model used by schools and other educational institutions is structured by the lecturer giving a lecture, while the students take notes and has the opportunity to ask questions. In additional to the lectures, the students learn by solving assignments, doing experiments in labs, working in groups, reading books, watching videos, and so on.
Students at NTNU have access to the teaching materials of the courses they take through Blackboard and/or other web pages. Video recordings of lectures are also available for many courses, which give the students the opportunity to watch the lecture again, or even study remotely. Supportive software programs and apps are provided to provide better learning conditions for the students.
Different people understand different things on a different level, which means some students finds books useful to read in order to learn, while others do not. Therefore,

looking for new ways to present the teaching materials become important.

Adaptation of New Technology in Education

Albert Einstein once said, "I never teach my pupils, I only attempt to provide the conditions in which they can learn"[17]. Adapting technology into the education system is a way to improve the quality and efficiently of education.

The art of printing made it easier to distribute knowledge, and books are one of the main sources of knowledge in the educational system today. The digital age gave birth to computers and software, smartphones and apps, in additional to the internet where all sort of information has become even more accessible.
The usage of XR has started see the light through its ability to present information on top of the real world, or making the user become immersed inside a virtual world. With its growth in popularity, could VR technology take a more active role in the students' life?

X Reality

VR has become popular the last few years due to the increased popularity and development of XR technology. A VR application can provide a more immersive experience for the user, compared with 3D models on a 2D screen. VR can provide an immersed experience of being inside a virtual world where you can interact with your hands, which opens new paths for exploring things, such as events that you normally would not be able to feel present in. For instance, you can travel back in time and experience a virtual world that recreates a historical event.
VR technology has already started to be used in the industry as well. An example if this is the bridge project in Sørum which ÅF is currently working on. They want the project to be completely free of paper and having all the details within a VR application. This takes time to adapt, but they see the potential of VR becoming more efficient than 3D models on a screen and drawings on papers, due to the benefits of being able experience a full-scale virtual model through VR[37].

Augmented reality (AR) on the other hand augments what we see in the real world by adding a layer of virtual elements on top of the real-world environment. Mixed reality (MR) merges the real world with a virtual environment which both can be interacted with. For instance, Equinor has started to use mixed reality with the Hololens to look for building errors at their oil rigs[42], which has successfully detected hundreds of wrongly assembled pipes and valves[43]. Finding one specific pipe in a module usually took them about one hour to find, while using the Hololens only took about three minutes, saving them lots of time[18].

Virtual Reality and Education

The applications developed for VR is not limited to games only. As already mentioned, the industry has started to make use of XR technology in order to become more efficient.

Study shows that memorization of letters using your hand with a pencil enhances learning in comparison with a keyboard. This is due to the brain receiving feedback from the hand through the movement of using the pencil, which activates parts of the brain, where motoric memory is inserted into the sensory motoric part of the brain. The same thing applies to watching or listening to an activity, and the study also show that the amount of time spent doing an activity improves the learning process. In other words, when writing with a pencil you create more motoric memory compared with using a keyboard[20].
This thesis wants to take the advantage of the immersive experience of VR and combine it with the principle of "learn by doing", in order to let the user, become "one" with the algorithm. This might help the user to learn more efficient through creating even more motoric memory compared with using a pen and paper.

1.2 Problem Description

Some students may find themselves in a situation where they struggle to learn new concepts through traditional teaching methods. This thesis wants to explore the possibilities of utilizing virtual reality as a supporting tool for students who are facing issues of understanding these concepts through traditional learning methods. An educational VR application will be developed, tested, and analyzed in order to find the benefits and drawbacks. The focus will be for computer science concepts; hence the target population will be students with computer science- or similar background, and people who find technology and algorithms interesting.

1.3 Research Questions

The main goal of this thesis is to look for an efficient way of learning new concepts through VR. The following research question will be used to explore the problem description:

RQ: Can virtual reality be utilized for educational purposes in order to improve the learning process, compared with traditional teaching methods?

The main research question will be answered using the following sub questions:

RQ1: What are the important characteristics for an educational virtual reality application, that separates virtual reality from traditional teaching methods?

RQ2: What are the benefits and drawbacks of an educational virtual reality application, in comparison with traditional teaching methods?

RQ3: How do the users feel about replacing traditional teaching methods with virtual reality?

RQ4: Can such an application be implemented in order to learn about algorithm and other concepts?

1.4 Contributions

Educational applications have been developed for virtual reality but has not become adapted to be used on a daily basis within schools. This is due to virtual reality in educational context still is a relatively young field.
Similar research have been conducted earlier at NTNU, such as SortVR[3] by Marius Bang and VirtSort[16] by Tom Xiang-Kun Kong and Anders Marstein Kruke. This thesis wants to further explore the possibilities of virtual reality for educational purposes. This will be done by creating a virtual reality application that can be used to learn concepts at a University level using known theory for creating educational games.

1.5 Thesis Outline

Chapter 1 - Introduction
The first chapter has introduced the motivation and aim for this project.

Chapter 2 - Previous Work and Theory
The second chapter will present previous works related to this project. Some of the key points of virtual reality history, information about hardware and software, and the theory used during the development of the application will also be presented.

Chapter 3 - Methodology
The third chapter will go into details of the implementation, and present what hardware, software, and other tools that were used, and how it connects to the theory from chapter 2. Feedback from SortingVR, the application developed for the specialization project will be presented. The name of the application will be changed into *CourseVR*, which will also be explained in the following chapters.

Chapter 4 - Results
The fourth chapter will present the implementation results, and the results from the user tests that were conducted in order to evaluate the usability and learning potential of the application.

Chapter 5 - Discussion
The fifth chapter will discuss the questionnaire feedback. Each individual statement and the overall result of the system usability scale (SUS) part will be investigated and try to answer the research questions.

Chapter 6 - Conclusion and Future Work
The last chapter will draw a conclusion to this thesis, as well as discuss some of the possible future paths of this research under the future work section.

Chapter 2

Background

This chapter will go into details of the background for this thesis, by first exploring previous works related to VR conducted by the author. Leading on to the next sections, which will present some historical events of virtual reality, look at the variety of VR head-mounted displays (HMD) in the marked, and software that can be used to develop VR applications. The last section will present theory of creating educational games, design principles, and algorithms.

2.1 Previous Works

2.1.1 Feasibility Study

A feasibility study about virtual study halls were conducted in February 2017 by the author of this paper, and five other students who took the course TIØ4258 Technology Management. An online questionnaire with 87 participants, as well as two interviews with a professor and an associate professor at NTNU, were conducted to explore the potential of creating virtual study halls with VR technology. The results from the questionnaire showed that it was an interesting idea, but that the expenses for a student to buy the required equipment would be too expensive. There were some concerns regarding the VR technology being in a growing phase, and that motion sickness could cause issues.
The interviews gave useful information on how to address these problems. For

instance, one could implement an application that uses technology already in possession of most students, namely the smartphone. The smartphone can be used together with some of the cheaper VR headsets in the marked, such as Google Cardboard, and maybe use a Bluetooth device as controller.
Another idea was to narrow down the targeted audience, such as only focusing on students who have problems being present at the University due to sickness, depression, or other reasons. See Appendix A for all the answers from the online questionnaire.

2.1.2 SortingVR

A prototype called SortingVR was implemented in the occasion of the specialization project in the previous semester. This project was conducted in order to learn more about VR, in preparation of writing the master thesis. The prototype was implemented using the game engine Unity, where the user could learn about bubble-, insertion- and bucket sort, through different teaching modes and difficulties. Usability tests were conducted by six students, who tested SortingVR and answered a SUS questionnaire[44]. These tests gave some insights in what could be improved in order to give the user a better VR experience. The main problem was the lack of a good tutorial to build up the player skills[35], due to some participants not having that much experience using VR prior to the test. Section 3.3 will go into more details of the feedback from SortingVR.

2.2 History of Virtual Reality

The history of virtual reality is considered to have started in the 1950's, but some elements can be traced back to the 1860's, long before the development of digital technology[49]. If we take into consideration that virtual reality tries to create an illusion of the user being present somewhere they are not, then the 360° panoramic paintings from the 19th century falls under the scope of virtual reality. Since these paintings covered the viewer's entire field of vision, thus resulting in the feeling of being present in some sort of historical event or scene[52]. Here are some historical events which have contributed to the development of VR technology over the years[52][4]:

1838: Research about how the brain processes the differences of two 2D images from each eye into a single 3D object, was conducted by Charles Wheatstone this year. By viewing two side by side stereoscopic photos through a stereoscope, resulted in the user getting a sense of depth and immersion. The design principles of the stereoscope are some of the components which is used to create virtual reality HMDs, and separates VR from the majority of non-VR systems[6].

1939: The popular View-Master stereoscope was patented this year, which was developed using the knowledge of stereoscopic photos from the 1838. It was often produced with images from popular tourism spots, thus providing a way for the user to go on a virtual trip.

1956: Cinematographer Morton Heilig created a multi-sensory simulator called "Sensorama", which was later patented in 1962, and was the first VR machine. The machine used various technologies in order to stimulate all the senses of the users, combining a full color 3D video, audio, smells, atmospheric effects (e.g. wind), and vibrations. To accomplish this, the machine had a vibrating chair, stereo speakers, a stereoscopic 3D screen, and scent producers. According to Heilig, Sensorama was the "cinema of the future" where the users could experience a fully immersive movie. Six short films were created for it.

1960: The year when the "Telesphere mask", the first head-mounted display, was patented by Morton Heilig. This provided wide vision stereoscopic 3D images and stereo sound, but had no motion tracking system.

1961: Two Philco Corporation engineers, Comeau and Bryan, created the first motion tracking head-mounted display - the "Headsight". It had a video screen for each eye and a magnetic motion tracking system to track the movements of the head. It was developed for the military to remotely get a view of dangerous and hazardous situations. Movements of the user's head would move a remote camera, which allowed the user to look around in the environment.

1965: The year when the computer scientist Ivan Sutherland presented his vision of the *Ultimate Display* - "Make that (virtual) world in the window look real, sound real, feel real and respond realistically to the viewer's actions"[21]. His concept was a virtual world which could be experience through a head-mounted display, where the user would not be able to tell the difference between the virtual- and actual reality. The concept included also interaction with objects.

1968: The year when the first virtual reality head-mounted display was created by Ivan Sutherland and his student Bob Sproull - named "The Sword of Damocles". The device was connected to a computer rather than a camera and could only display simple virtual wire-frame shapes, hence having a quite primitive usage. When the user moved the head, the 3D models changed perspectives due to the tracking system. It was too heavy for the user to wear it comfortably, thus it was attached to the ceiling.

1985: VPL (Visual Programming Lab) was founded by Jaron Lanier and Thomas Zimmerman. This company developed equipment such as the DataGlove, Audio Sphere, and EyePhone HMD, and is known to be the first company to sell VR goggles and gloves.

1987: All these years of development in the field of virtual reality still had not received a name. This changed when the term *Virtual Reality* was coined by Jaron Lanier.

1991: Virtuality Group launched a range of arcade VR machines and games, which were accessible to the public. The player had to wear a head-mounted display, which let them experience the immersive stereoscopic 3D visuals with less than 50ms latency. Some of the machines were even connected in a network, meaning multiplayer games were available.

1995: Nintendo released the "Virtual Boy" in Japan and North America, but discontinued its production and sales the following year due to commercial failure. It played 3D monochrome video games and was the first portable console to display 3D graphics. The reason for its failure was due to the lack of software support, it was uncomfortable to use, and the lack of colored graphics (only black and red).

2009: Palmer Lucky felt that there was a lack of HMDs in the marked which could give him the immersive experience he was looking for. The HMDs available in the marked suffered of low field of view, high latency, was expensive, and bulky to use. He started to design a prototype HMD in 2009, which aimed to take 3D gaming to the next level. His first prototype was finished in 2010, which featured 90° field of view, built-in haptic feedback, and low latency.

2012: The Oculus Rift Kickstarter campaign was launched by Palmer Lucky, who currently had developed his 6th generation VR unit called "Rift". The campaign raised more than $2.4 million[50].

2014: Facebook bought Oculus Rift in March 2014. According to the CEO of Facebook, Mark Zuckerberg, he saw the potential of Oculus Rift's non-gaming experiences as well, such as VR applications with inbuilt social networking[10].
The same year, HTC and Valve Corporation started their cooperation on developing the HTC Vive[14].

2016: The first consumer versions of Oculus Rift and HTC Vive were both released in 2016, and in the same year the number of active virtual reality users worldwide increased drastically[33].

- today: The popularity of virtual reality has grown larger since 2016, and various HMDs have become available in the marked. The next section will show some of the hardware released, and show their differences and pros/cons.

2.3 Hardware

Virtual reality hardware comes in various models which can be categorized into mainly three categories: PC tethered-, standalone-, and mobile VR. Pros and cons can be found in each category - therefore it is important to think about what kind of applications you are planning to use before buying one. The following should be considered:

Degrees of freedom: The main unit for virtual reality is the HMD, which has the goal of tracking movements of the user and displaying the virtual environment to the user through its screens. Some VR packages also includes controllers, which also depends on the degree of freedom (DoF) in order to track properly. Even though all the HMDs in the three categories can track some or all the user's movements, how they actual work are different. The DoF tells what kind of movements the HMD support, and the higher DoF the more movements can be tracked. Figure 2.1 shows the tracking difference between 3DoF and 6DoF.

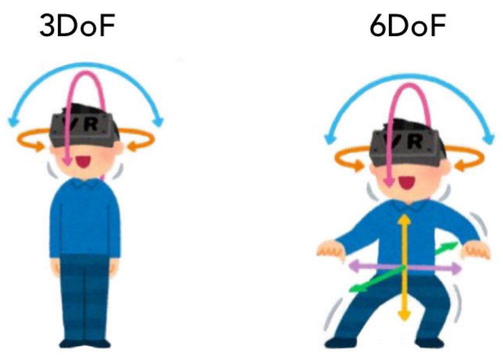


Figure 2.1: Degrees of freedom illustration[9].

Screen resolution: The screens are placed inside the HMD and close to the eyes in order to create the immersive experience. The higher resolution of the screens, the better quality of images you will see, and it becomes harder to see the pixels, which leads to a smoother experience for your eyes.

Field of view: The HMDs available today still have a reduced field of view in comparison with the normal field of view of humans. A normal human vision has a 200° - 220° binocular- and a 114° monocular field of view of view[15], while HMD usually lies in the range 90° - 110°.

Cost: There is a significant difference in price among the three categories, where PC tethered is the most expensive due the extra sensors, controllers, and the need of a computer with hardware that can handle it. The standalone VR category can be considered the mid-range for cost, while the mobile VR category can be considered the cheapest option in case you already got a smartphone.

Play area: The surrounding space required to utilize the hardware. This boils down to the DoF supported by the HMD, and how the movement actions of an application have been implemented. HMDs using outside-in tracking requires a minimum play area due to the sensor placements. Applications which uses teleportation for movement reduces the area needed, making it possible to stand still during gameplay.

2.3.1 PC Tethered Virtual Reality

PC tethered VR requires a computer that is powerful enough to power up, handle the computations, and transfer data to the HMD. These HMDs mainly use *outside-in* tracking, meaning they require external sensors placed somewhere in the room so they can track the movements of the HMD and controllers. The hardware of this category supports 6DoF movements due to the outside-in tracking system.
The current available HMDs are usually tethered, which means that the headset is physically connected to a computer by cables. Some HMDs support wireless adapters, meaning you can experience virtual reality without having to worry about the cables getting in your way.

We will see further into details of the Oculus Rift and HTC Vive, since CourseVR will be playable using these via a plugin called SteamVR. Section 2.3.4 will present two wireless adapters that can be used for these HMDs.

Oculus Rift

The Oculus Rift comes with two sensors and two touch controllers, as shown in Figure 2.2. Its HMD has two OLED panels with a total of 2160 x 1200 pixels (1080 x 1200 pixels per eye), has a refresh rate of 90Hz, and a 110° field of view. The headset has a single custom motherboard with an ARM processor which controls the chips for the LEDs. The key component of the Oculus Rift is a magnetometer, a gyroscope and an accelerometer which together are called the "Adjacent Reality Tracker". This component tracks the headset accurately in all three dimensions.

The tracking system is called *constellation*[23], which has pretty much the same function as the Nunchucks controllers of the Nintendo Wii. The infrared LEDs embedded in the headset are being tracked by the sensors, which results in the three-dimension location of the headset. The developer kit 2 had a problem where the user couldn't turn 180° away from the sensors, due to all the LEDs going out of sight of the sensors. The consumer version addressed this problem, offering a full 360° experience.

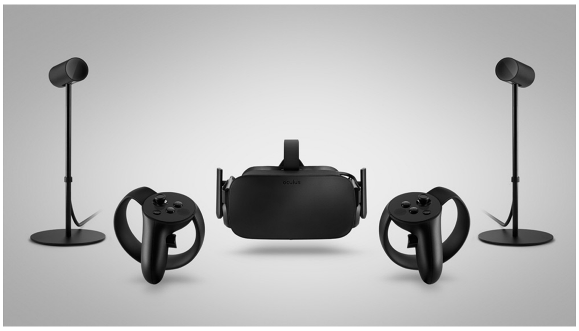


Figure 2.2: Oculus Rift headset, controllers, and sensors[41].

HTC Vive

The HTC Vive also comes with two sensors and two controllers, as shown in Figure 2.3. The HTC Vive has OLED panels with a total resolution of 2160 x 1200 pixels (1080 x 1200 pixels per eye), a refresh rate of 90Hz, and a 110° field of view. The later version, HTC Vive Pro, released in 2018 comes with upgraded OLED panels with 1440 x 1600 pixels per eye. The sensor boxes are called *Lighthouses*[14], which fills the room with infrared light. The tracking sensors on the headset and the controllers receives this infrared light, which is used to detect the location of them in the room.

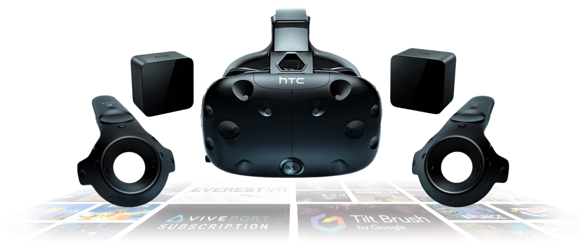


Figure 2.3: HTC Vive headset, controllers, and base stations[45].

2.3.2 Standalone Virtual Reality

Standalone VR HMD has the benefit of working without any need of a computer nor any extra sensors. The drawback is the reduced quality of the image compared with a PC driven HMD. The tracking ability of the HMDs in this category is affected by the lack of external sensors, rather using a *inside-out* tracking system, which results in either less movement tracking or a higher price tag. Usually these HMDs comes with 3DoF or a hybrid of 3DoF/6DoF, but some even offers 6DoF. The battery capacity is another important factor to consider before buying one.

Oculus Go

The Oculus Go (Figure 2.4) comes with a 5.5inch Fast-Switch WQHD LCD display with a total of 2560 x 1440 pixels (1280 x 1440 per eye). The refresh rate of the screen can vary between 60Hz and 72Hz, depending on the usage, and a 100° field of view. It is equipped with a Qualcomm's Snappdragon 821 processor, which is also to be found in the HTC U Ultra, LG G6, Google Pixel phones and more. It uses a gyroscope sensor to track how you twist, turn, and rotates your head, thus tracks movements with 3DoF. It includes a simple controller which can be used as a laser pointer to navigate within applications. Other Bluetooth controllers can also be used, such as the bluetooth controllers for the Nintendo Switch, PS4, XBox one and so on. It has A 2600mAh battery equipped which provides up to 2.5 hours of usage[32]. It has 3GB RAM, and there are two models with different storage size available: 32GB for €219, and 64GB for €269[28].

Figure 2.4: Oculus Go with controller[32].

HTC Vive Focus & Plus

This was HTC's first attempt to develop a standalone VR HMD, which was originally planned to operate on Google's Daydream VR platform. Somewhere in the development process they backed down from their partnership with Google and dropped the idea of developing it for the Daydream, and rather push it on the Viveport version for China[8].
The HTC Vive Focus features a Qualcomm Snapdragon 835 SoC processor, a 3K AMOLED (2880 x 1600) display with a refresh rate of 75Hz, and 110° field of view. It has a 4000mAh battery, 4GB RAM, and 32GB storage. It features an inside-out hybrid tracking system, with 6DoF for the HMD, and 3DoF for the controller. However, due to the limitation of tracking for the controllers, an upgraded version called HTC Vive Focus Plus (Figure 2.5) was introduced which features a full 6DoF experience[36].

Figure 2.5: HTC Vive Focus Plus with controllers[36].

Oculus Quest

The Oculus Quest (Figure 2.6) has two OLED panels with a display resolution of 1440 x 1600 per eye, 72Hz refresh rate, and 95° field of view. It is featured with a Snapdragon 835 processor, which is the same used in the Samsung Galaxy 8, Google Pixel 2 phones and more. It features a 6DoF tracking for both the HMD and its two included touch controllers[30]. It performs the tracking by scanning all objects in its play area, then creating a 3D map which it combines the data from its gyroscope and accelerometer once every millisecond[11]. The battery has a capacity of 2-3 hours, depending on the usage. It has 4GB RAM, and a storage of either 64GB or 128GB. The 64GB model costs €449, and the 128GB model costs €549[27].

Figure 2.6: Oculus Quest with controllers[27].

2.3.3 Mobile Virtual Reality

The last category is the cheapest VR equipment in the marked, which only requires a smartphone to be inserted into the HMD shell. Hence, the technical specs of this setup mainly depend on the smartphone being used. The quality and extra properties of the smartphone VR cases range from being made of cardboard with lenses to models which looks similar to the standalone VR HMDs. Some models include a controller, but third-party Bluetooth controllers can also be used.

Google Cardboard

The Google Cardboard (Figure 2.7) was first seen at Google I/O 2014, and had its first release in February 2015 as small budget HMD. A version 2.0 was released later the same year, with some of the specifications upgraded. All display specifications are given by the smartphone that is inserted, which is only limited to a screen size of 4.0" to 6.0". It has a 37mm biconvex lens and has a focal length of 50mm and 45° field of view[48]. The main material used is cardboard, as the name suggests, hence it only weights 181g. Various models have been released since then, with a price tag as low as €5 and upwards.

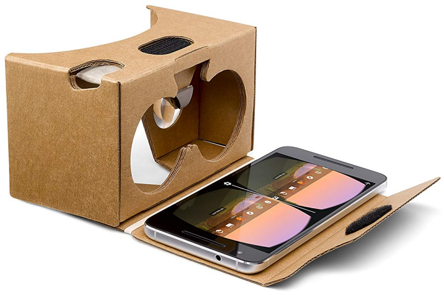


Figure 2.7: Google Cardboard[2].

Samsung Gear VR

The Samsung Gear VR (Figure 2.8) is a mobile VR HMD which is manufactured by Samsung and powered by Oculus. It is compatible with Samsung Galaxy S6 and above, Note 8/9, and requires the smartphone to have Android Lollipop 5.0 and the Samsung Gear VR application installed[31]. It uses a spring loading system and comes with an adapter for USB type-C or micro USB, hence it can fit various smartphone sizes. When starting it up for the first time, you will need to install the Samsung Gear VR application, which will help you to set up the headset and the included controller. It weights 345g, has a 101° field of view, and costs about €100.

Figure 2.8: Samsung Gear VR with controller[31].

2.3.4 VR Accessories

Wireless Adapters

The following products can be used to make tethered HMDs wireless.

TPCast: There are still no sign of an official wireless adapter for the Oculus Rift. When VRFocus asked Nate Mitchell, the head of the Oculus Rift team, they got the answer that they did not have any plans of developing a wireless adapter yet[13]. There exist some third-party solutions though, for instance the TPCast wireless adapter which can both be used for the Oculus Rift and HTC Vive (no Pro support). The technical specs for the TPCast device is shown in Table 2.1. This wireless package main components are the following; the adapter which is attached on top of the HMD, a battery and a unit to place it in, a wireless router, and a base station. It also comes with a small bag and a belt which you can place the battery in and attach to your waist. The TPCast wireless adapter costs about $ 319.

Play area	16ft x 16ft
Signal	60GHz
Battery	Up to 5 hours
Latency	Less than 2ms
Weight	90g

Table 2.1: TPCast wireless adapter technical specs.

HTC Vive: The HTC Vive wireless adapter can be bought to remove the cabled connection between the HMD and computer. It is powered by Intel® WiGig which offers a near-zero latency VR experience[46]. It has an easy-swap battery pack which can be attached to your belt or trousers. The technical specs for the wireless adapter is shown in Table 2.2. The wireless pack's four main components are the following; the adapter which is attached on top of the HMD, the battery, a PCIe card to be mounted in the computer, and a wireless link box to be attached somewhere visible to the adapter in the play area. The official price for the wireless adapter is $299.99 for the HTC Vive, and $359.99 for the HTC Vive Pro. HTC Vive Pro users must also buy an additional adapter attachment kit, hence the extra cost.

Play area	20ft x 20ft
Signal	60GHz
Battery	Up to 2.5 hours
Latency	Near-zero
Weight	129g
Requirements	1x free PCIe slot

Table 2.2: HTC Vive wireless adapter technical specs.

Controllers

As already mentioned, most of the HMDs in the previous sections includes one or two controllers. Some of these have limited tracking ability due to less than 6DoF, such as the controller for the Oculus Go, and mobile VR such as the Google Cardboard does not even have a controller. Third-party Bluetooth controllers is a solution, but here is one that can enhance the experience.

Nolo: A motion tracking kit which can be utilized to increase the tracking ability to 6DoF. The package includes two controllers, a base station for front-facing room scale, and a headset marker for seamlessly movements within the virtual world - as shown in Figure. It costs $ 199, and can be bought as an expansion by those who wants to experience 6DoF, but does not have a VR headset which supports this2.9[47].

Figure 2.9: Nolo VR controller[25].

2.4 Software

Developing a game or an application requires some tools to get the job done. There exist many software programs to be used for developers, unless you want to build all the tools yourself. There are many game engines to choose among when you are going to develop an application, and due to the increased popularity of VR, many of these support VR development as well. This section will explain what a game engine is, present some of the game engines used for development of VR applications, and give some information about plugins and assets.

2.4.1 Game Engine

A game engine is a software development tool which lays a framework for creating video games. Game engines creates abstractions for the developers, in form of giving them the tools for game physics, rendering graphics, collision detection, memory management and so on. Game engines usually provides templates and libraries of prefabs and assets. Reuse of code is common thing to do in the software industry, hence having a game engine saves time and money. Companies that have successfully built a solid *full-figure option*[7] can sell licences of their game engine to other developers who are in need of it. Game engine licences are quite expensive for professional usage, thus the developers must consider the value of buying one. Sometimes a game engine offers most features, but maybe not all of the needed tools, meaning the company needs to rewrite and disabling parts of the game engine to make it a better fit for their development.

Unity

Unity is a popular game engine among novice developers and professionals. The CEO of Unity claims that Unity is used to create half of the world's games[39], something that might be right considering that Unity dominates the indie and lower budget marked. It has a broad support, with over 25 platforms - including VR, AR, Mobile, Web, PC, and console. As for programming languages, Unity supports C#, UnityScript, and Boo. It offers a community and official tutorials for developers, and there are also many individual developers who create tutorials which can be found online. Unity offers three versions for their game engine - pro: $125/month for professionals and studios, plus: ˜$25/month with 1 year prepaid or $35/month for hobbyists, and personal which is a free version for beginners who have a revenue or funding which does not exceed $100,000 per year. Unity offers many assets and plugins in the Asset store, which either can be bought or downloaded for free.

Monobehavior lifecycle: MonoBehaviour is the base class which every Unity class inheritance from. The flowchart in Figure 2.10 shows the various of stages a instantiated gameobject goes/loops through until it is destroyed.

During the initialization of a gameobject it goes through the *Awake* method up to the *Start* method, before the first frame update, if the script instance is enabled. Awake is always the first method to be called, after the scrips has been loaded into the game, and can be used to initialize variables. *OnEnable* is called if the gameobject is active when instantiated. And then Start is called, which can be used similar to how Awake is used. Awake and Start are only called when a gameobject is instantiated, while OnEnable is called every time the game object is reactivated.

There are three main update methods in the lifecycle, namely the *Update*, *Late-Update*, and *FixedUpdate*. The difference between these update methods is that Update is called once per frame, while FixedUpdate is called on a reliable timer which is independent of the frame rate. Hence, FixedUpdate should be used for actions that involves physics, such as changes to a rigidbody, because then it is certain that it will be executed in sync with the physics engine. The LateUpdate is called after the Update method, which makes it useful for calculations that are needed after the Update method has been called. LateUpdate can therefore be used to ensure that calculations in the Update method are completed, before starting new calculations.

Trigger and collision can be detected by a gameobject which has a rigidbody, hence under the physics part, by using the *OnTriggerXXX* and *OnCollisionXXX* methods. Trigger can be enabled by checking the trigger box of the collision component on the gameobject.

The decommissioning phase is the final stage for a game object, which begins whenever destroying a game object, or the application quits.

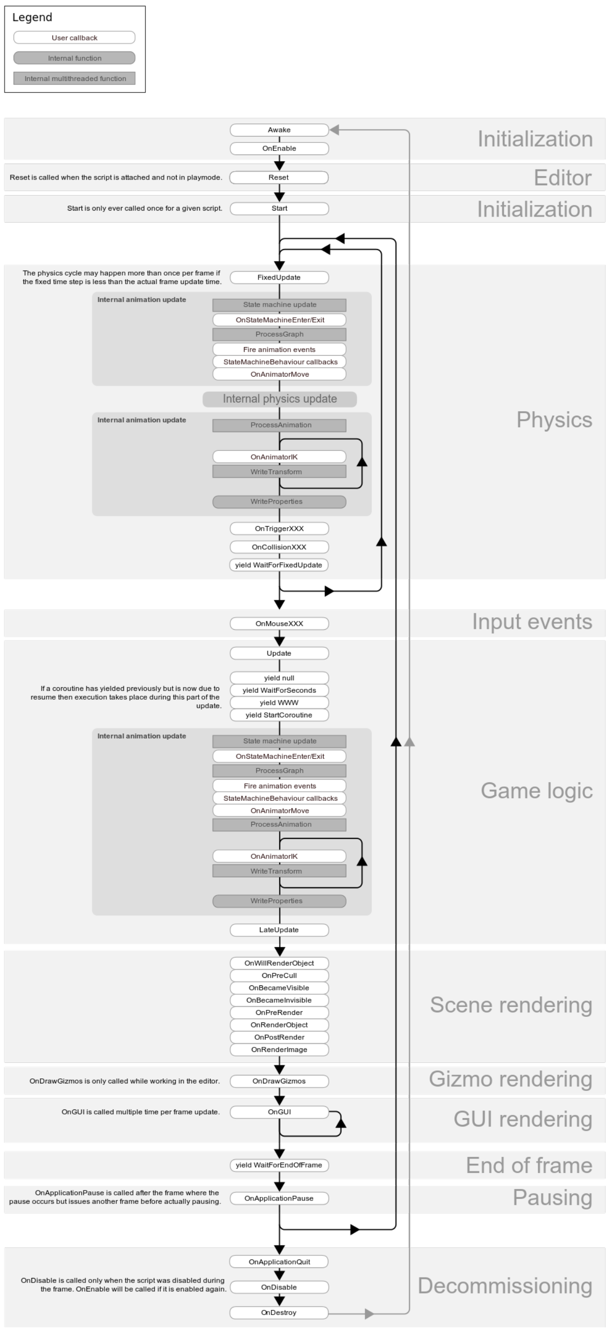


Figure 2.10: Monobehaviour flowchart[40].

Prefabs: Unity's prefab system makes it simple to create, configure, store, and reuse gameobjects. A prefab is made by dragging a gameobject from the scene into the assets. A copy of a prefab within the scene is an instance of the prefab.
Unity discovered issues with their prefab system when they asked a range of indie and AAA studios about how they work with prefabs. The main issue they found were that as a developer you had to drag a prefab into a scene to modify it, and then delete it again. Another big issue was the apply button for a prefab instance in the inspector that could be clicked by an accident, which then applied all changes of this prefab instance to the prefab itself, and was hard to reverse. Unity 2018.3 addressed these issues by introducing new features, which included nested prefabs, prefab mode, and prefab variants[26]. Prefab mode makes it possible to edit a prefab asset in an isolated environment as shown in Figure 2.11, which is not connected with the scenes of the project.

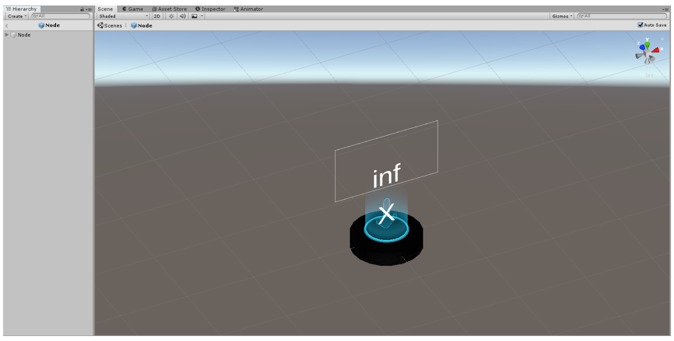


Figure 2.11: Prefab mode.

Instances of a prefab now has an improved visualization of property and object overrides, and this gives the ability to apply overrides on multiple levels of granularity, such as per property, per component/gameobject, or as in previous versions, the entire prefab instance.

It is recommended to create a prefab when a gameobject configured in a particular way is used in multiple places in a scene or across multiple scenes in a project. Common example of prefabs are environmental assets, NPCs (Non-Player Characters), and the player itself (for spawning in multiple scenes).

Animation: Gameobjects can be animated by attaching an *Animator* component to the object. There are two views in the Unity editors that can be used to create and arrange animation clips. The first view which is shown in Figure 2.12 is the animation view, which is used to create animation clips. This view provides the tools to create animations by clicking the record button, and then add animation properties, such as changes in position, rotation, scale, and so on. The used properties will be listed in the left side of the view and marked on the timeline.

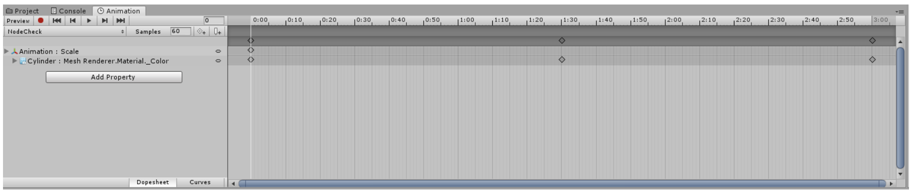


Figure 2.12: Animation view.

The steps done in the first view automatically sets up the relevant components and references within the second view, which is the animator view (Figure 2.13). The states are automatically added, but the edges between the states must be set by the developer. Conditional transitions between the animation states can be made by adding condition variables in the column to the left. An extra menu will display in the inspector column on the right side when one transition is selected. Clicking the (+) button and selecting the variable gives the opportunity to set what conditions must be fulfilled to move from the first state to the second. The transition conditions support boolean, integer, float, and trigger.

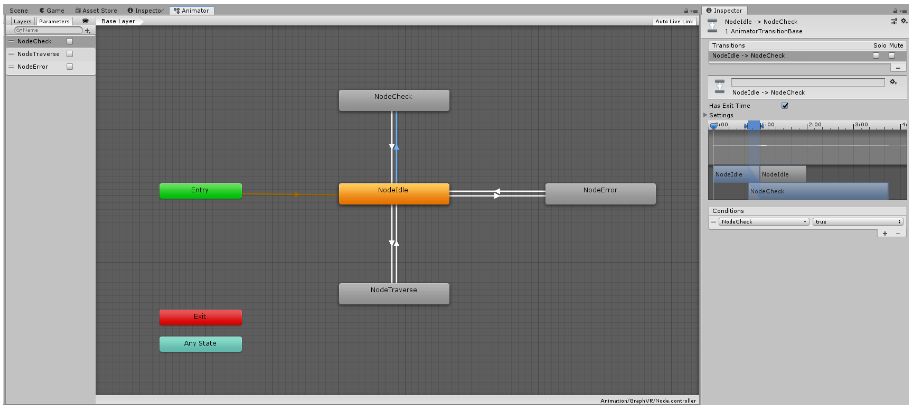


Figure 2.13: Animator view.

Unreal Engine 4:

UE4 is known for its beautiful graphics, and is currently the latest version of the Unreal Engine. UE4 is often used by professional developers. The programming language used is C++, in additional with a visual scripting system called *Blueprints* which lets the user create classes and wires these together. It mainly supports PC and console games but offers also support for other platforms as well. It is free to use until the day you start to ship your game or application, then you need to pay 5% royalty on gross revenue after the first $3000 per product, per quarter[34].

2.4.2 Plugins & Assets

Unity Asset Store

The asset store makes it easy for Unity developers to download for free or buy assets made by other developers. There are many categories to choose among, such as 3D, 2D, audio, templates, tools, and many more.

SteamVR

The SteamVR plugin can be found in the asset store, which is an SDK which allows developers to target a single interface that will work with all major VR headsets from seated to room scale experiences. It uses an action based key binding, which by adding an action to a button, trigger, trackpad on the HTC Vive, will easily work on an Oculus Rift as well. It also provides access to tracked controllers, chaperoning, render models for tracked devices.
In the SteamVR package you can find some tutorial pdf's, prefabs, and even some scenes where you can test the different interactable objects and action bindings. Changing or creating new actions can be done in the SteamVR Input inside Unity. Binding or changing these actions can be done in the browser via steam's binding UI page. It requires a Steam account and having SteamVR installed on the Steam platform as well, running in background in order to fix the key bindings.

Sound effects

Sound effect is an easy way to make a game or application more exciting to use, and can be used as to motivate the player through curiosity, which will be explained in more details in Section 2.5.1. Sound effects can be found in Unity Asset Store, and on web pages such as open game art.

2.5 Theory

The theory background to the development of CourseVR will be presented in this section. How this theory will be used for the development of CourseVR will be explained in the next chapter.

2.5.1 What Makes Things Fun to Learn?

Thomas W. Malone provides a set of heuristics and guidelines in the article "What Makes Things Fun to Learn?"[19], for how to design computer games in order to make them fun in an educational context. He proposed that *challenge*, *fantasy*, and *curiosity* were the essential characteristics of intrinsically enjoyable situations, such as computer games.

Challenge

A game should provide challenging tasks for any players for the game to become interesting. According to this study, this can be accomplished by having goals, some uncertain elements, and boost the self-esteem of the player rather than decreasing it.

Goal: An appropriate learning curve should be implemented by giving the player simple goals in the beginning, which later on becomes harder as the player has gained experience. The player should be able to understand the goals based on the environment, which contains goals of appropriate difficulty. Furthermore, fantasy goals are often better than simple use of skills, e.g. solving math problems will move the character towards the goal in a fantasy, rather than just solving arithmetical exercises. Finally, the player should receive some sort of feedback so that they can tell whether they are getting closer to the goal or not.

Uncertain outcome: Uncertain outcome spices up the game so that the player will not become bored, and stays focused. This can be achieved by (1) variable difficulty levels, (2) Multiple level goals, (3) Hidden information, and (4) Randomness.
An uncertain outcome can be implemented in various ways. Variable difficulty level which is either automatically determined by the program based on the player performance, chosen by the player themselves, or determined by the opponent's

skill.
The game can have multiple goals, such as score keeping based on the players actions throughout the game. Speeded response can also be used, which is based on how fast the player solves the task, either as fast as possible, before a deadline or the player's opponent.
Provoking curiosity can be accomplished by selectively revealing hidden information to the player. Randomness is a final way to create an outcome of a game uncertain.

Self-esteem: The self-esteem of humans can be the reason for continuing a challenging activity. As with any challenging activity, success in a video game can lead to a positive boost in our self-esteem and make us feel better about ourselves. On the other hand, we can become discouraged by failure, resulting in quitting the activity. An appropriate balance of challenge should be provided to the player in order to make the player feel motivated to continue playing.
A problem occurs due to a tension between giving a clear performance feedback and their self-esteem, hence feedback should be presented in a way that minimizes the damage to the self-esteem.

Fantasy

Fantasy can help an educational game to become more interesting, by replacing abstract symbols with fantasy objects. This can also make it easier to understand difficult concepts. Fantasies can be split into two categories based on their dependencies between the player's skill and the fantasy, namely *intrinsic*- and *extrinsic* fantasies. As shown in Figure 2.14, extrinsic fantasies only depends on the skill, while intrinsic fantasies goes both ways.

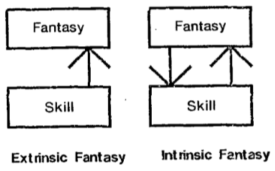


Figure 2.14: Intrinsic- and extrinsic fantasies.[19]

Curiosity

Computer games can awake the player's curiosity by *sensory* and *cognitive* methods, in order to increase the motivation to learn. Sensory curiosity can be stimulated by visual and audio effects such as (1) decoration, (2) to enhance fantasy, (3) as a reward, or (4) as a representation system.
Cognitive curiosity can be stimulated by giving the player just enough information for them to get a desire to bring a better "form" of their knowledge structure.

2.5.2 GameFlow

The article "GameFlow: A model for Evaluating Player Enjoyment in Games"[35] starts by explaining what "flow" is: "*Flow denotes the mental state of operation in which a person is fully immersed and time seems to 'fly by'*". Further, it goes into detail of what gameflow is, and summarizes it with the following eight elements:

Concentration: The player should be fully focused on the game, and solving the task at hand. The game should have meaningful tasks, have an appropriate workload, and minimize non-related interactions.

Challenge: The game should match the player's skill level in order to avoid the player becoming bored or anxious.

Player skills: The player should be able to improve their skills through playing the game, hence learning should feel as part of the game. This can be achieved by giving hints and tips during gameplay and creating real-world analogies.

Control: The player should have a feeling that their action impacts the world, and there should not be only one optimal way of facing a challenge.

Clear goals: The goals of a game should be easy to understand, and each level should have multiple goals which tests the player in different levels of difficulty.

Feedback: The player should be given feedback when both winning and losing. This can help the players to become motivated to continue playing the game.

Immersion: The game should provide deep and effortless involvement, emotionally engaging entertainment, and an "escape" from the daily life.

Social interaction: Social interaction is not an element of flow, and can both be constructive and destructive. For instance, in a multiplayer game it is useful to provide a way for social interaction for cooperation, matchmaking and so on.

2.5.3 Design Principles

When designing the user interface and interaction of objects, then it is smart to use design principles to make the application intuitive and easy to use. This section will present some relevant design principles that are often used when designing a computer application's user interface.

Affordance: The attribute of an object that makes people aware how to use it, according to Don Norman[29]. Good affordance of an object will give the user a clue of how to use it. For example, a button invites the user to push it down.

Proximity: According to Gestalts principles, the human brain creates assumptions based on what our eyes see. Elements are perceived as being more related when placed closer together, rather than those which are spaced further apart from each other. Furthermore, elements that are placed within the same region are perceived as a group[12].

User control and freedom: Mistakes can be performed by the user on a system. This should be addressed by giving the user an option to return to the previous state before the mistake were done, according to Jakob Nielsen[24]. For instance, support to undo an action.

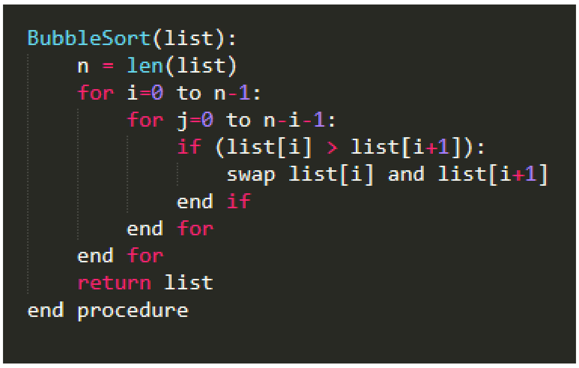
2.5.4 Algorithms

The sorting algorithms which SortingVR featured, and which will be further developed in CourseVR will be explained here. CourseVR will also include some graph algorithms, which also will be explained in this section. All of the following algorithms are part of the curriculum of the course TDT4120 - Algorithms and Data Structures[1] and have many practical applications to solve problems of our lives.

Sorting Algorithms

A sorting algorithm takes an input array of unsorted data of the same type, for instance integers, doubles, floating point numbers, strings, characters, and so on. Then it rearranges the data, following the instructions of the algorithm such that the array becomes sorted. Sorted data can makes it easier to find, analyze, or visualize the content of the data we are working with. Sorting algorithms can be used as a basic building block in order to make other problems become easier.
For instance, given an array of n soundtracks, and you want to find out whether there exist any duplicates. The problem can easily be solved when all the soundtracks are sorted, because then duplicates will appear next to each other.

Bubble sort: Bubble sort is a simple sorting algorithm which keeps comparing two values at a time, let's say value A and B, then if value A is larger than B then swap these. This will result in the biggest value "bubbling" its way to the last index. Now it does the same process repeatedly, until the whole list is sorted. See Figure 2.15 for the pseudocode, and Table 2.3 for the complexity for the given cases.



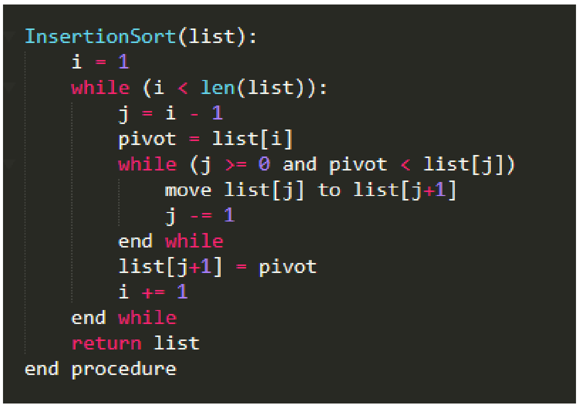
```
BubbleSort(list):
    n = len(list)
    for i=0 to n-1:
        for j=0 to n-i-1:
            if (list[i] > list[i+1]):
                swap list[i] and list[i+1]
            end if
        end for
    end for
    return list
end procedure
```

Figure 2.15: Bubble sort pseudocode.

Case	Comparison	Swap
Worst-case performance	O(n^2)	O(n^2)
Average performance	O(n^2)	O(n^2)
Best-case performance	O(n)	O(1)

Table 2.3: Bubble sort cases[51].

Insertion sort: Insertion sort starts by declaring the value of index 0 sorted, then picks the next element (index 1) as pivot and starts comparing the pivot with all the sorted elements which lies to the left of the pivot. If the pivot is less than the element it is comparing with, then move the comparison element one index to the right and continue comparing with the next index. If the pivot element is larger than the comparison element or it has reached the bottom of the list, then insert the pivot element back into the list. See Figure 2.16 for the pseudocode, and Table 2.4 for the complexity for given cases.



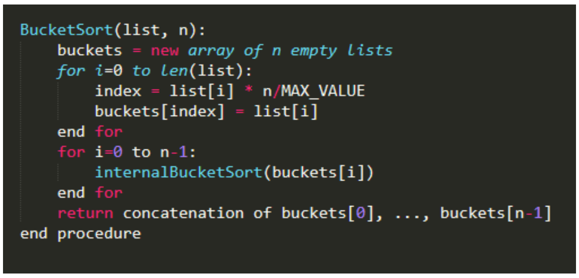
```
InsertionSort(list):
    i = 1
    while (i < len(list)):
        j = i - 1
        pivot = list[i]
        while (j >= 0 and pivot < list[j])
            move list[j] to list[j+1]
            j -= 1
        end while
        list[j+1] = pivot
        i += 1
    end while
    return list
end procedure
```

Figure 2.16: Insertion sort pseudocode.

Case	Comparison	Swap
Worst-case performance	O(n^2)	O(n^2)
Average performance	O(n^2)	O(n^2)
Best-case performance	O(n)	O(1)

Table 2.4: Insertion sort cases[51].

Bucket sort: Bucket sort sets up an array of n empty "buckets". Then the values of the unsorted list get moved into the bucket which has a capacity range that satisfies the element's value. When all values have been placed into the buckets, then each bucket is sorted internally. Usually insertion sort is used for the internal sorting method. Now that each bucket is individually sorted, then by concatenating all buckets we get a sorted list. See Figure 2.17 for the pseudocode, and Table 2.5 for the complexity for given cases. We see that bucket sort has the same worst-case as the other sorting algorithms explained here. In case we have a worst case, and in additional all values are not well distributed, ending up in the same bucket, then we get the same result as insertion sort's worst-case.



```
BucketSort(list, n):
    buckets = new array of n empty lists
    for i=0 to len(list):
        index = list[i] * n/MAX_VALUE
        buckets[index] = list[i]
    end for
    for i=0 to n-1:
        internalBucketSort(buckets[i])
    end for
    return concatenation of buckets[0], ..., buckets[n-1]
end procedure
```

Figure 2.17: Bucket sort pseudocode.

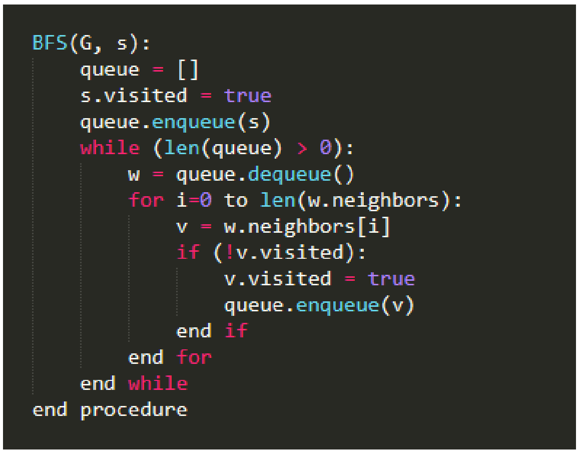
Case	Complexity
Worst-case performance	O(n^2)
Average performance	θ(n + k)
Best-case performance	Ω(n + k)

Table 2.5: Bucket sort cases[51].

Graph Algorithms

A graph algorithm takes an input graph G alongside some more information in order to analyze G. There are many applications in real life for graph algorithms, hence there exists many variants to choose among.
Various traversal algorithms exist, such as pre-, in-, and post order traversal, BFS, and DFS. These graph algorithms take a graph G and a start node s as input, and then traverses all the nodes from there.
Shortest path is a common problem in daily life, such as finding the shortest route from Trondheim to Oslo. In this case we can map the cities as nodes and the roads between cities as edges. This problem can be solved by a shortest path algorithm which uses the map of Norway as its graph G, and Trondheim and Oslo as start- and end nodes respectively. Then search the graph from the start node until it either finds the end node or stops due to no possible path between the nodes. Examples of shortest path algorithms are Dijkstra, Floyd–Warshall, DAG-shortest path, Bellman-Ford, and many more. Some of the differences of these algorithms are what conditions they allow, for instance whether they support negative weights or cycles.
Graph algorithms have other applications as well, such as finding a Eulerian Path/Circuit, Hamiltonian Path/Circuit, minimal spanning tree, and many more.

Breadth-First Search: BFS is a graph algorithm that starts traversing a graph G from a start node s. From node s it enqueue all its (unvisited) neighbors to a queue, then it dequeues the node which has been waiting the longest in the queue, following the FIFO (First-In-First-Out) principle. BFS is mainly a traverse algorithm but can also be used to find the shortest path on an unweighted graph. Pseudocode is shown in Figure 2.18.

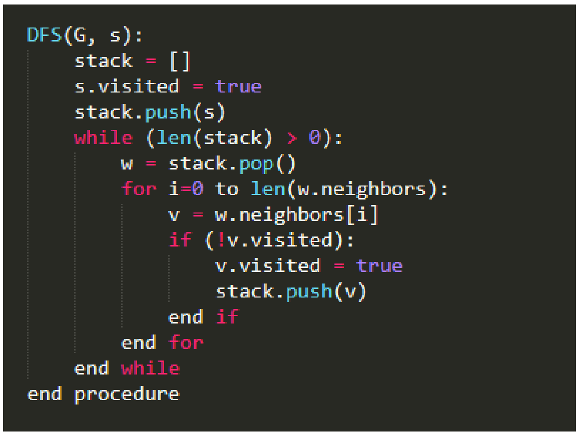


```
BFS(G, s):
    queue = []
    s.visited = true
    queue.enqueue(s)
    while (len(queue) > 0):
        w = queue.dequeue()
        for i=0 to len(w.neighbors):
            v = w.neighbors[i]
            if (!v.visited):
                v.visited = true
                queue.enqueue(v)
            end if
        end for
    end while
end procedure
```

Figure 2.18: The pseudocode for BFS.

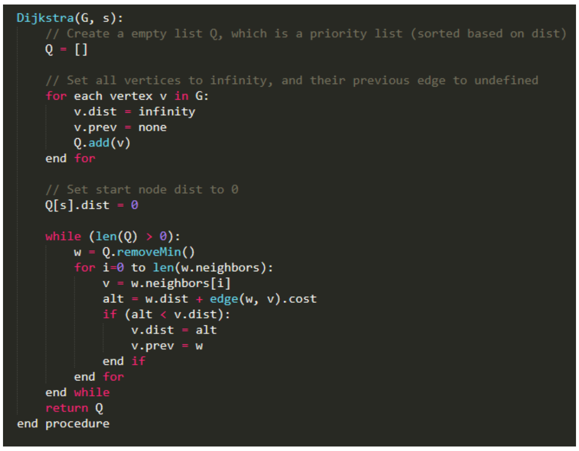
Depth-First Search: DFS has a similar code compared with BFS, but it uses a stack instead of a queue. A stack follows the LIFO (Last-In-First-Out) principle. Starting from node s, it pushes all its neighbors to the stack, then pops off the node that was last pushed. Then it repeats this until all nodes has been traversed. DFS can also be used to solve a shortest path problem on an unweighted graph. Pseudocode is shown in Figure 2.19.

Dijkstra's algorithm: Is a greedy algorithm that finds the shortest path from a start node S, to an end node E or all other nodes of a graph G. It uses a priority queue which sorts the nodes by their distance, where the node with the lowest distance has the highest priority. Hence, it will always choose the node with the lowest distance from the priority queue. The node which gets removed from the priority queue, w, relaxes all its untraversed neighbor nodes, and in case a shorter distance was found from w to neighbor v, then node v's distance and previous edge is updated. This is done until it reaches node E or the whole graph. Pseudocode is shown in Figure 2.20.



```
DFS(G, s):
    stack = []
    s.visited = true
    stack.push(s)
    while (len(stack) > 0):
        w = stack.pop()
        for i=0 to len(w.neighbors):
            v = w.neighbors[i]
            if (!v.visited):
                v.visited = true
                stack.push(v)
            end if
        end for
    end while
end procedure
```

Figure 2.19: The pseudocode for DFS.



```
Dijkstra(G, s):
    // Create a empty list Q, which is a priority list (sorted based on dist)
    Q = []

    // Set all vertices to infinity, and their previous edge to undefined
    for each vertex v in G:
        v.dist = infinity
        v.prev = none
        Q.add(v)
    end for

    // Set start node dist to 0
    Q[s].dist = 0

    while (len(Q) > 0):
        w = Q.removeMin()
        for i=0 to len(w.neighbors):
            v = w.neighbors[i]
            alt = w.dist + edge(w, v).cost
            if (alt < v.dist):
                v.dist = alt
                v.prev = w
            end if
        end for
    end while
    return Q
end procedure
```

Figure 2.20: The pseudocode for Dijkstra's algorithm.

Chapter 3

Methodology

The implementation of a well-designed educational VR application requires knowledge about the hardware, software, and related theory. This chapter will try to combine these, using the research conducted in the previous chapter, as well as the results gained from developing SortingVR during the specialization project. CourseVR will be the new name of the application, since other types of algorithms is planned, and making the application open for other concepts for future development.

3.1 Hardware and Software

This section will go into details of what hardware and software that will be used for the development of CourseVR. Most of the choices were taken during the specialization project.

3.1.1 Head-Mounted Displays

The functional requirements of SortingVR required HMDs that supported tracking of both the head and hands; thus, it was decided to develop for either HTC Vive or Oculus Rift. It was first considered to develop for the Oculus Rift alone but taking the price of these HMDs and the target population into consideration, it would be

more ideal to create an application which could be utilized by multiple platforms. Realizing that the SteamVR plugin can be used to accomplish exactly this, it was decided that SortingVR would at least support the HTC Vive and Oculus Rift.
The virtual environment should be implemented in a way which feels natural to the users, in order to avoid losing immersion. Immersion is one of the elements of gameflow[35], which provides an "escape" from our daily life. This element was also a crucial part of the decision of what type of HMD to develop CourseVR for, due to the requirements for 6DoF. Also, the mentioned HMDs has extra buttons which can be used for enhanced interaction.

3.1.2 Game Engine

Research was conducted for which game engine to use during the specialization project. The result of this research was either to use Unity or Unreal Engine 4, which both had the necessary tools in order to develop a VR application. Unity was chosen over UE4, since Unity supported development using C# as programming language, which the author of this thesis was more familiar with.

3.1.3 Integrated Development Environment

An integrated development environment (IDE) is a software that provides tools which makes it easier for the programmer to develop software, such as code completion, refactoring, debugging tools, syntax highlighting, version control support and more.
Unity itself does not have an IDE in its editor, hence a third-party is needed. Visual Studio gets installed by default when installing Unity on a Windows or macOS system, but Unity also supports Visual Studio Code (Windows, macOS, Linux) and JetBrains Rider (Windows, macOS, Linux)[38].

3.1.4 Video Recording and Editing

The videos which will be presented later in this thesis were recorded while playing CourseVR in the editor, using Unity Recorder. The videos were edited using Microsoft Photo Video Editor, and the voiceover for the videos were created using a text-to-speech program called Balabolka.

3.2 Requirements

This section will go into details of the requirement specifications for this project. These will be split into non-functional- and functional requirements. The non-functional requirements specify the criteria that can be used to judge the operations of the application, while the functional requirements define specific functionality and behaviors of the application.

3.2.1 Non-Functional Requirements

Table 3.1 contains the non-functional requirements for the application that will be developed.

NFR	**Type**	**Description**	**Priority**
1	Usability	The application should be easy and intuitive to use for any users	High
2	Extensibility	It should be easy to add new algorithms or other concepts in the same architecture	High
3	Teachablity	The users of should gain a better understanding of the teaching materials through playing the application	High

Table 3.1: Non-functional requirements.

3.2.2 Functional Requirements

The functional requirements table has been separated into three tables due to the size of the application. Table 3.2 gives an overview of functional requirements which applies to both the sorting- and graph algorithm implementations, while Table 3.3 and Table 3.4 goes into the details of the sorting- and graph implementation respectively. Functional requirements in Table 3.2 marked with a * will be specified in the sorting- and graph table.

FR	Description	Priority
1*	Various algorithms should be implemented to give the player a broader selection of teaching materials	High
2*	Each algorithm task must be unique to ensure a new challenge for each playthrough	High
3*	The player must be able to control the problem size	High
4*	The initial state and other parameters should be adjustable by the player	High
5	Tutorial: A tutorial scene should be implemented, which gradually makes the player familiar with the controllers	High
6	A way of introducing the teaching materials to the player should be implemented (e.g. video)	High
7	Demo: A new demonstration system should be implemented which combines the demo and step-by-step teaching modes from SortingVR	High
7.1	The player should be able to control the demo, such as pause and increase/reduce the speed.	High
7.2	Step-by-step: A functionality which activates when the demo is paused, giving the player control of the pace and direction of the demo.	High
8	User test: A playable mode must be implemented, where player can practice and test themself	High
8.1	Various difficulty levels should be implemented to gradually increase the challenge for the player	High
8.2	The player should be given informative feedback of their performance during a user test.	High
8.3	A score system should be implemented to give the player more challenge through subgoals	Medium
8.4	The player should be given detailed information after a user test, which explains incorrectly performed actions.	Medium
9	Pseudocode must be presented in the background to support the player	High
9.1	The pseudocode should be highlighted and updated in real time	High
9.2	The player should be given the option to make the pseudocode update in multiple steps.	Medium
9.3	Various colors should be used as a representation system	Medium
9.4	Voiceover should be used to explain the pseudocode, and give instructions to the player	Low
10*	Customized support for the algorithms should be implemented to improve the visualization of the algorithm state	High
11	Sound effects should be used to create representation systems	High
12	A progression bar should be implemented to give an illustration of how many steps are remaining of the active task.	High
13*	The player must be able to interact with the virtual environment	High
14	The player must be able to move around using teleportation	High
15*	Game objects should never fall out of reach for the player.	High
16	The controllers should display tooltips of what the buttons can do	Low

Table 3.2: Overall implementation functional requirements.

FR	Description	Priority
1.1	The algorithms developed in SortingVR must be updated according to the new functional requirements	High
1.2	More sorting algorithms should be implemented	Low
2.1	Sorting elements must be instantiated with random values	High
3.1	The player must be able to choose the number of elements used in a sorting task	High
4.1	The player can select a sorting case, such as best- and worst case	High
4.2	The player can select whether duplicates can occur	High
4.3	The number of buckets in bucket sort should be adjustable by the player	Medium
10.1	The holder prefab should give visual feedback of the sorting element's state	High
13.1	The player must be able to pick up and rearrange the sorting elements	High
15.1	Sorting elements should return to the sorting table if dropped on the floor or thrown to unintended locations	High

Table 3.3: Sorting implementation functional requirements.

FR	Description	Priority
1.1	Traversing algorithms must be implemented	High
1.2	Shortest path algorithms should be implemented	High
2.1	Some randomness should be used for graph building	Medium
2.2	Edges should be instantiated with random weights (shortest path)	High
3.1	The player must be able to adjust the size of the graph	High
4.1	CourseVR must be able to set the input node(s) automatically	High
4.2	The player should be able to select input node(s) themself	High
4.3	Various graph structures should be implemented to ensure variation of tasks, e.g. tree, grid, and random	High
4.4	Both undirected- and directed edges should be selectable.	High
10.1	A list visualization of queue/stack/priority list should be implemented to visualize the data structure used by the graph algorithm	High
10.2	Graph objects should change appearance according to the state of the graph algorithm	High
10.3	Node animation should be used to give directional help to the player	High
13.1	The player must be able to add a node to the data structure used by the algorithm, or performing other actions on it	High
13.2	The player must be able to traverse the nodes	High
15.1	Standalone items used within the graph scene must be available for the player when they are required. Falling on the floor should return them to the player	High

Table 3.4: Graph implementation functional requirements.

3.3 Decisions, Ideas, and Feedback from SortingVR

The usability results from SortingVR gave some insight of what worked as intended, what functionality that needed some improvements, and missing features. This section will explain the decisions and introduce the ideas that were made to create an educational application. The feedback from the usability tests will also be briefly explained, in order to show what CourseVR will continue to work with.

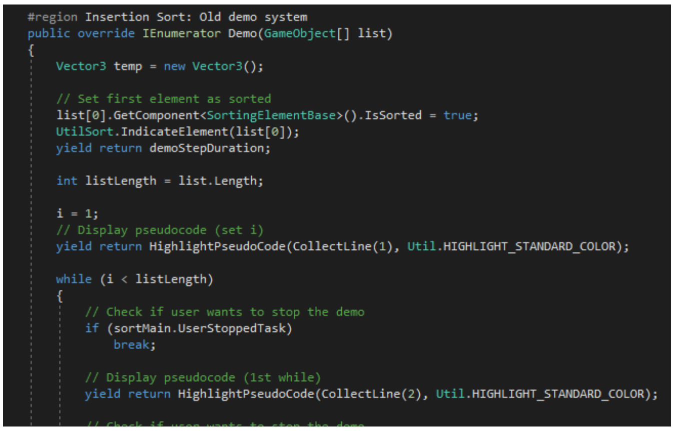
3.3.1 Decisions

Some decisions were made during the development of SortingVR in order to improve the structure of the software architecture. The plan for SortingVR was to create an application where the user could learn about sorting algorithms, but with a plan of including other types of algorithms as well in the future, thus it became important to write code such that it would be easy to expand in the future. Following programming conventions was stated as important, even though there was only one person in charge of the development. This was chosen for better comprehensibility of the code for later work and by others, and to make it easier to debug.

3.3.2 Ideas

Some ideas of how to present the teaching material in a intuitive way for the user was made. Three different teaching modes were implemented, which each had their own purpose for providing a way to learn about algorithms. The three teaching were the following:

Demo: A VR demonstration of how the algorithm work. The demo system was the first teaching mode of SortingVR to be implemented. It was created by taking the code of an algorithm and insert code blocks for moving the sorting elements in the virtual environment, and also create a pausing mechanism between each movement. Figure 3.1 shows the first few lines of the insertion sort demo implementation. The pausing mechanism was implemented by starting a coroutine on a method of type IEnumerator interface, which yields and returns pause duration in form of WaitForSeconds from time to time. This way of presenting a demo worked poorly due to no control given to the user while the demonstration was ongoing, and loads of redundant code. Section 3.3.3 will go into details of how CourseVR will address these issues.



```
#region Insertion Sort: Old demo system
public override IEnumerator Demo(GameObject[] list)
{
    Vector3 temp = new Vector3();

    // Set first element as sorted
    list[0].GetComponent<SortingElementBase>().IsSorted = true;
    UtilSort.IndicateElement(list[0]);
    yield return demoStepDuration;

    int listLength = list.Length;

    i = 1;
    // Display pseudocode (set i)
    yield return HighlightPseudoCode(CollectLine(1), Util.HIGHLIGHT_STANDARD_COLOR);

    while (i < listLength)
    {
        // Check if user wants to stop the demo
        if (sortMain.UserStoppedTask)
            break;

        // Display pseudocode (1st while)
        yield return HighlightPseudoCode(CollectLine(2), Util.HIGHLIGHT_STANDARD_COLOR);
```

Figure 3.1: Old demo code example.

User test: The user test was the next teaching mode of SortingVR that was implemented. An instruction system was made in order to create the user test, which will be explained below. The user test let the player put their hands to work, giving them the freedom of doing anything they wanted with the sorting elements. But in order to progress, they had to move the sorting elements in a correct manner which followed the instructions of the algorithm. If an incorrect action was performed, then feedback would be given to the user though changing appearance of the props.

Step-by-step: The lack of controls in the demo implementation was the reason why this teaching mode was created. This was an easy teaching mode to add due to the working instruction system which had already been created for the user test.

Instruction System

A system that could pause and wait while the player performed an action were needed for the user test. Figure 3.2 shows a simple illustration of the system that was implemented. It generates a set of instructions which are used to update game objects involved during a user test. Each instruction contains data about the next state of an object and/or which pseudocode line to update. The instruction progresses automatically during a user test based on the actions performed by the player, or manually during a step-by-step session by the increment/decrement actions (see Section 3.4.1). During the user test it will not send out the next instruction until the correct action has been performed by the player.

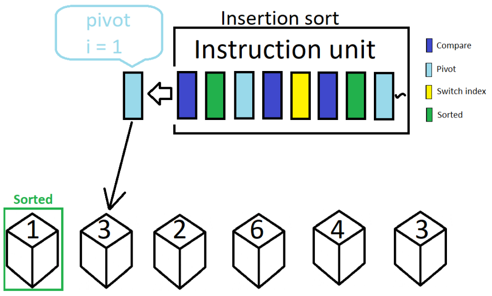


Figure 3.2: An illustration of the instruction system.

3.3.3 Feedback and Solutions

The feedback received during the usability testing of SortingVR will be presented here.

Tutorial: SortingVR did not have any sort of tutorial that helped the player to build up their player skills[35]. A ”help corner” was added within the algorithm rooms, but since the player spawned in the main menu scene, they were not able to learn the basic functionality of the controllers right away. This was considered a problem, at least for those who had no prior knowledge about VR applications. Therefore, it is planned to create a proper tutorial where the controllers, functionality, and gameobjects will be explained.
It is also planned to implement a tool which the player can watch introduction videos of algorithms, such that those who are new to the teaching material or don’t remember much about it, have a chance to learn the basics.

Settings menu: The transition from the menu to a sorting task was a problem due to no ”physical” button for start/stop, rather a button on the controller was used for this. Also, the settings menu and sorting table were positioned in different positions and directions, thus caused some confusion. Another problem was the lack of feedback from the buttons in the menu, making the player become unsure whether the buttons had been clicked or not.
These issues will be solved by replacing the SteamVR settings menu that was used, and create a new one which includes more feedback in form of colors, sound effects, and text. A ”physical” start button will be added to the menu in order to make it clearer how to start. The menu will be placed in the same location of the sorting table, so that they are visible in front of the player when starting/stopping a task.

Adjustable height: The heights of the prefabs and objects in SortingVR were implemented with the idea of being seated in a chair. This resulted in some problems for those who played the game while standing, making it harder to interact with objects and concentrate on the pseudocode.
This will be fixed by making it possible to adjust the Y-position of objects, such as the sorting table.

Misreading values: The value on the surface a sorting element was not underlined. For instance, this could cause a value of 6 to be mistaken by 9, and vice versa. This was not reported a big issue, but should be fixed to prevent misunderstandings in the future.

Teaching mode: A problem with the demo system as mentioned earlier was the lack of controls for the player. The introduction of the ”step-by-step” teaching mode slightly fixed this issue, but also introduced more work for the developer. This meant that there were in total three teaching modes with two different systems to maintain, which was rather time consuming whenever adding a new algorithm. It is therefore decided to scrap the old demo system, and go all in for the instruction system by adding a auto-play function to the step-by-step teaching mode, which will make the implementation and maintenance more efficient. The new demo will automatically progress through the generated instructions, or enable the step-by-step functionality whenever paused.

All-in-one-room: Each sorting algorithm were separated in each their isolated scene, even though there were no big difference among these scenes. Hence, the player had to move unnecessary between the scenes in order to switch algorithm. Therefore, it is decided to create a single room for each type of algorithm in CourseVR to make it more convenient to use. This means there will be in total two algorithm teaching scenes, the sorting- and graph scene.

3.4 Implementation

Now that the details of SortingVR has been presented, it is time to look into the implementation of CourseVR. The functionally of the controllers and the implementation of the sorting- and graph algorithms will be explained in detail in this section.

3.4.1 Working with SteamVR

The controller functionality is featured through the usage of the SteamVR plugin. SortingVR used the default action set provided by SteamVR, in additional with some extra actions that were added. The default action set alongside the extra functionality added will be explained in this section.

SteamVR has models for the hands holding the controllers, and the models also have animations whenever the buttons on the controllers are clicked. This makes your virtual hands and controllers feel more natural, since when clicking a button on the controller will be visible inside the application as well.

Default Actions

The default actions can be used without any need of changing the input of SteamVR. By simply adding SteamVR's player prefab to the scene you will be able to explore the scene using the VR headset. Some other prefabs and scripts must be added to the scene or to gameobjects in order to use more of the default action set. The default action set provides the following actions:

Grab, throw & interaction: SteamVR provides a script called *Interactable* which makes a gameobject interactable if added to it. An interactable gameobject gets highlighted when the controller touches it, as shown in Figure 3.3, which makes it easier to distinguish which gameobjects that are interactable.
Presenting hand-sized objects in a virtual environment will make the user want to interact with them, due to affordance. Making them highlight when touched will enhance the feeling that the cube indeed can be picked up through feedback[29].

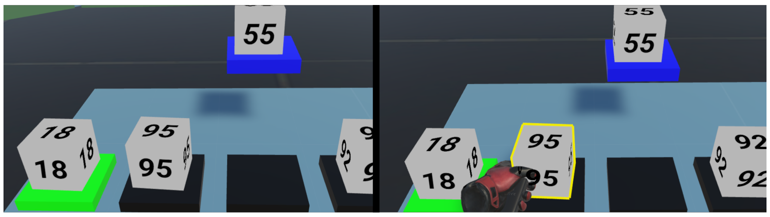


Figure 3.3: Illustration of highlighting of interactable gameobjects. Source: SortingVR

Interactable gameobjects can be interacted with by clicking the trigger button (HTC Vive) or the rear trigger (Oculus Rift). Grabbing can be performed by doing the same action, but then by holding the button, move the object, and then release the button to drop it. The grip buttons (HTC Vive) or side trigger button (Oculus Rift) can do the same grabbing action, but one click grabs and holds the object until the same button is clicked again.

SteamVR also provides a script called *Throwable* that makes a gameobject become more natural in the sense of physics. This script requires the gameobject to have a rigidbody, interactable, and velocity estimator components, which will be automatically added in case the gameobject misses any.

Teleport: Teleportation makes it easier for the user to move around within a scene, and between scenes. The first benefit of teleportation is the simple fact that the user can stand still while playing, hence reducing the chance of bumping into anything near the play area. The second, and main benefit is to reduce the chance of motion sickness.

SteamVR provides three prefabs, namely *Teleporting*, *TeleportPoint* and *TeleportArea*, which easily can be added to the scene to give the user teleportation ability. *Teleporting* is required to enable teleporting within a scene, and can be placed anywhere in the scene. *TeleportPoint* is a single point which is useful to place where the developer wants the player to stand in a stationary position. *TeleportArea* on the other hand can cover a tiny area or the whole scene just by changing the scale of the object. It is up to the developer to decide which one is more useful for the situation within the application.

By clicking down and holding the trackpad (HTC Vive) or the thumbstick (Oculus Rift), a parabola line becomes visible as aim. The color of this parabola changes based on where the user aims, giving informative feedback whether the action can be performed. A red parabola means the user cannot teleport, while a green one indicates the user can move to the new location. This is shown in Figure 3.4.

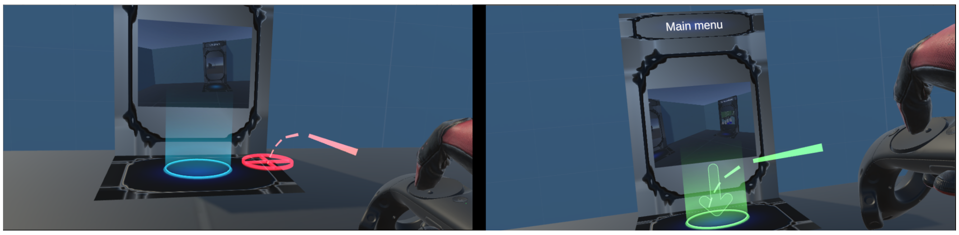


Figure 3.4: Teleport aim.

Extra Actions

This section will present the actions that were added in additional to the default action set.

Demo & Step-by-step: It will be necessary to switch between the demo and step-by-step. The menu button (HTC Vive) on the right controller will be used for this functionality.

Increment & Decrement: This functionality was added to the default action set during the development of SortingVR. It was and will continue to be used to progress through the step-by-step feature. The player can progress one instruction at a time by clicking the grip button (HTC Vive) on the right controller, and backtracking by clicking the same button on the left controller.

Increase & Decrease: This functionality also uses the grip buttons, and gives the player control of the demo speed of an algorithm.

Pivot holder movement: Insertion sort has been implemented in a way where there is an extra position outside the list, represented by the pivot holder (see Section 3.5.4). This functionality is active while the player takes a user test in insertion sort, which allows them to move the pivot holder to the left and right. This functionality also uses the grip buttons.

Pointer: It was necessary to come up with practical way for the player to interact with the nodes in a graph algorithm. For instance, when traversing a graph using BFS we start by adding the start node to a FIFO list, then enqueue each neighbor from this node, then traverse the node which has waited longest in the queue.
This pointer functionality will serve two purposes; let the player choose the input node(s), and check nodes by pointing towards it and shoot it with a laser beam. This action will be activated when the player wants to choose start (and end) node(s), which is optional, or during a user test on a graph exercise.
The trigger button (HTC Vive) or the rear trigger (Oculus Rift) on the right controller is used for this functionality.

Graph extra: Some of the supportive tools within the graph scene might block the view of the pseudocode, hence an action to move these objects slightly to the sides was needed. The menu button (HTC Vive) on the left controller will be used for this purpose.

3.4.2 Application Scenes

A small presentation of the scenes will be given here. For an overview of how the scenes are connected, please take a look at the scene diagram in Appendix B.1.

Starting Scene

The player first spawns in the starting scene, in the center of the two green arrows shown in Figure 3.5. An information panel with a welcome message is positioned in front of where the player spawns, making it the first thing for them to see. The video panels to the left and right displays a short preview of the main menu and the tutorial scene respectively. The preview video combined with the texture on the floor, and the moving and rotating green arrows is used to increase the affordance[29] for the player to move towards the scene they want to enter. A simple step towards either of the arrows will transport the player to the chosen scene. This gives the player control[35] of whether to take the tutorial or not, giving the experienced players an opportunity to skip the tutorial.

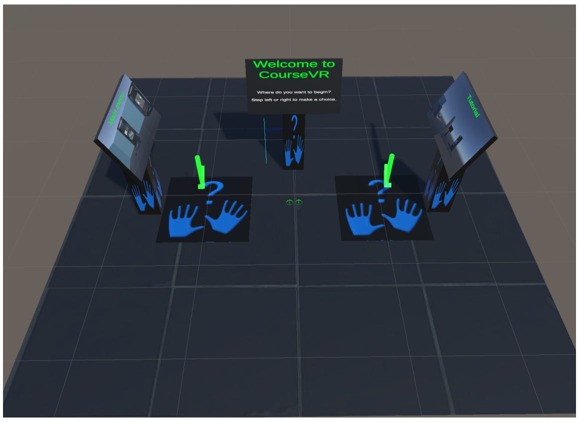

Welcome to CourseVR
Where do you want to begin?
Step left or right to make a choice.
Tutorial

Figure 3.5: Starting scene

Tutorial Scene

A tutorial scene was implemented where the users can learn the basic functionality of the controllers, and become familiar with game object and how to interact with them. The scene is split into different rooms with a somewhat "dungeon" feeling to them, as shown in Figure 3.6.

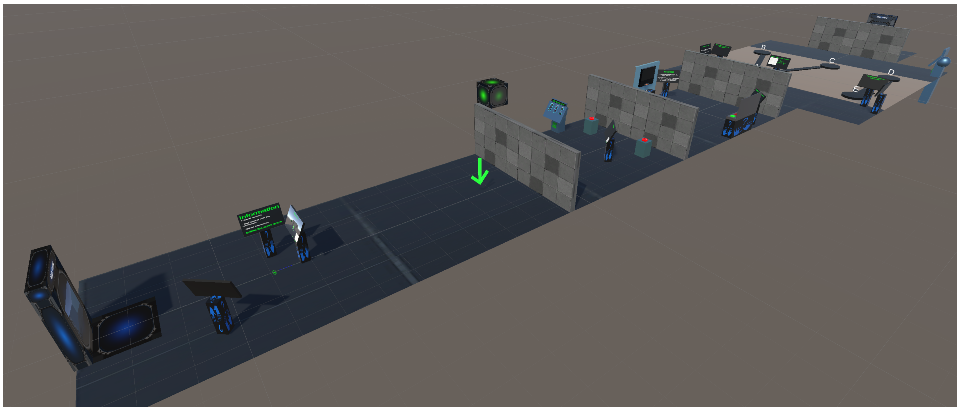


Figure 3.6: The tutorial scene.

The tasks of each room gradually become harder, starting with no prior knowledge required. As the player progresses, they need to apply what they have learned so far in additional to the new content presented. The door leading to the next room unlocks when the current room's tasks are completed. Each room and their tasks have a theme, and these are explained by a video panel playing a "how to do ..." video of the task. There are also some signs with text that the player can read to learn more. A green bouncing arrow leads the way from start to finish. Both the arrow and the information text titles are displayed in green, which is used to guide the player through the tutorial.

Main Menu Scene

The main menu connects all the scenes of the application using the portal prefab, as shown in Figure 3.7. The scene diagram in Appendix B.1 shows an overview of the main menu and its connections with the other scenes.

The portals leading to the other scenes have a short looping preview video that shows the content of the scene, in additional to the title above. When the player moves in front a portal leading to another scene, then the color of the portal changes and a sound effect starts to play as feedback. After three seconds, the player will be teleported to the chosen scene, unless stepping away from the portal before this timer. This gives the player control and freedom, giving them some time to prevent a mistakenly teleport.

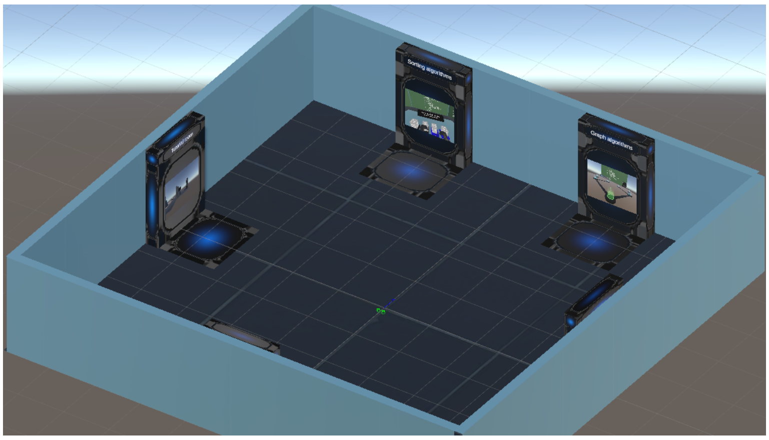


Figure 3.7: The main menu scene.

Algorithm Scenes

The details of the sorting- and graph scenes will be presented in Section 3.5 and Section 3.6 respectfully.

3.4.3 Introduction to the Teaching Materials

Some users might be completely new to the teaching material, or need a quick refresh before putting their hands to work. The monitor prefab was implemented to provide extra support for the player, optionally before starting using the teaching modes explained in the next section. The monitor prefab will be explained in Section 3.4.5.
The tutorial scene focuses on getting the player familiar with the basics of the controllers, and presenting some of the content of the application. The feedback from SortingVR showed the importance of making the application as intuitive as possible, thus the monitor will also serve the purpose of explaining the implementation details which were not included in the tutorial scene.
The introduction videos of the teaching material were found on Youtube, while the recordings of each algorithm's demo and user test was recorded with the Unity Recorder plugin.

3.4.4 Teaching Methods

CourseVR provides two teaching modes as according to what were concluded with in Section 3.3.3.

Demo

The demo gives a visual presentation of the algorithm. The new version of the demo gives more control to the player by letting them pause it whenever they want and adjust the demo speed. Pausing the demo enables the step-by-step functionality, giving the player the opportunity to progress at their own speed. The step-by-step feature will let the player perform forward steps, and for some algorithms, also backward steps.

User Test

The user test gives the player an opportunity to test their skills while putting their hands to work. The user test was implemented so that the fantasy and the skill of the player depends on each other, resulting in an intrinsic fantasy[19]. The feedback given back to the player based on their action, might help them learn from both their correctly performed actions and mistakes.
It was unclear how the user test was going to be presented during the planning phase of the specialization project. The decision was set between two ideas of implementation; (1) a fun and educational game with a score system, where the player could challenge themself by getting a highscore, or (2) an educational game with an informative feedback system. SortingVR was developed using the second option, with plans of adding the first option as well sometime in the future.
During the development of CourseVR it was decided to extend the user test with sub goals as described in the first option above, as an optional score system. The score system can be enabled/disabled in the settings menu, and when enabled, the player will be able to see their correct action streak, their highest streak, and score during the user test. After the user test is finished, they will get a score based on the time spent, the chosen difficulty, and number of correct/incorrect actions performed. The score system was implemented in order to give the player more challenge in form of sub goals[19].

The challenge should be appropriate to the player's skill level; hence the user test has four difficulty levels for the player to choose among, ranging from beginner to examination. The beginner level provides all implemented support, while intermediate and advanced give some support. At last, the examination level gives no help at all, except for some exceptions. The idea is to let the player put their hands to work through the user test, which could improve their understanding of the teaching material as a result of the interaction in a VR environment. The immersive and interaction experience can help to insert motorics memory in the sensory motorics part of the brain, making it easier to learn and remember the concept[20].

3.4.5 Support and Feedback Prefabs

The prefabs implemented to support and give feedback to the player, and which can be utilized by any algorithm implementation will be explained in this section.

Pseudocode

The main purpose of CourseVR is to make the player get a better understanding of algorithms, hence the main goal for the player is to follow the steps by the pseudocode in order to solve the problem. With this in mind, the pseudocode is the main support given to the player.
It automatically updates the values of the variables within the pseudocode, according to the instructions from the instruction system, during a demo of an algorithm progresses. The user test also provides this support when an appropriate difficulty level has been chosen by the user.

The pseudocode will highlight the current active line of code for the demo and user test (beginner), to provide a connection between the state of the algorithm and the pseudocode. This connection can be of help to fill the "gap" of knowledge to make the player bring a better "form" to their knowledge structure, also called *cognitive curiosity* according to Thomas W. Malone[19].
The highlight color depends on the code line. Green is used for the lines where the player must perform an action to progress. Blue and red are used to indicate whether a condition is fulfilled or not, respectively, for an if-statement and for-/while-loop conditions. Magenta is used otherwise.

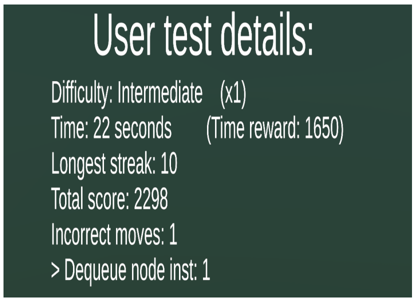
An extra "step" feature was also implemented. This feature updates some pseudocode lines in multiple steps. This was added to help the player to remember each pseudocode line. It is an optional feature which can be enabled/disabled in the settings menu. For instance, a line of an incrementing variable i, with a current value of 0, will be shown in three steps:

Step 1: $i = i + 1$
Step 2: $i = 0 + 1$
Step 3: $i = 1$

The pseudocode had to be further updated in CourseVR since the player will be moving around more in the graph algorithms. This was not a problem in the earlier version of the application, since the player would only stay in one location during playthrough. Thus, now it updates the font size, line space and location based on the player's distance and position relative to the pseudocode.

Blackboard

The blackboard gives informative feedback to the player, such as during and after a user test. This prefab is mainly used when the score system is enabled. An example of the performance feedback given to the player through the blackboard prefab is shown in Figure 3.8, when the score system is enabled.



User test details:

Difficulty: Intermediate (x1)
Time: 22 seconds (Time reward: 1650)
Longest streak: 10
Total score: 2298
Incorrect moves: 1
> Dequeue node inst: 1

Figure 3.8: Example of the feedback given after a user test.

Progress Tracker

The progress tracker is a gameobject that informs the player of how many steps remains of a demonstration or user test. As Thomas W. Malone proposed in ”What Makes Things Fun to Learn?”[19], the player should be able to tell whether they are getting closer to the goal.

Monitor & VHS

As mentioned in Section 3.4.3, this prefab's purpose is to introduce the teaching material for the player, such as introduction- and example videos. The prefab of a video is represented by a gameobject that resembles a VHS cassette. These cassettes are lying underneath or next to the monitor, as shown in Figure 3.9, and can be inserted one at a time. The videos are rather short, only about thirty seconds up to a few minutes.

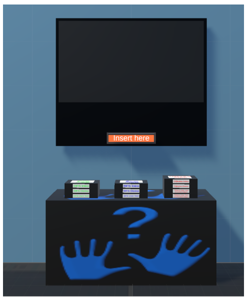
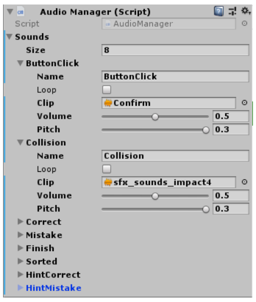


Figure 3.9: The monitor prefab.

Audio Manager

The *AudioManager* prefab was implemented to create an easy modifiable sound collection. The script is attached to an empty game object, and can be placed anywhere within the scene. Figure 3.10 shows the audio manager component of the prefab, in this case used for the sorting scene.



```
Audio Manager (Script)
Script              AudioManager
Sounds
  Size              8
  ButtonClick
    Name            ButtonClick
    Loop            ☐
    Clip            Confirm
    Volume          0.5
    Pitch           0.3
  Collision
    Name            Collision
    Loop            ☐
    Clip            sfx_sounds_impact4
    Volume          0.5
    Pitch           0.3
  Correct
  Mistake
  Finish
  Sorted
  HintCorrect
  HintMistake
```

Figure 3.10: Audio manager component.

The prefab has an empty list of sound effects, but by changing the size of the "sounds" list shown in the figure, adding sound effects can be easily done by drag and drop. A sound effect can be named, changed its volume and pitch, and enable loop. A sound effect can be accessed from any class by calling the *Play* method of the *AudioManager*, with the name of the sound effect as argument.
Sound effects are used to create representation systems[19], such as when an correct/incorrect action has been performed by the player, or when a gameobject changes state. Sound effects are always enabled, regardless teaching mode or difficulty.

3.4.6 Controlling Prefabs

The prefabs which gives the player control of their situation before and during a task will be explained in this section.

Settings Menu

The player will be given the control of the task via the settings menu, such as the size of the exercise and various other options.
The prefab for the setting menu was implemented based on the feedback from the usability tests in the specialization project. A section class was implemented to keep track of all the settings menu items underneath it, in this case buttons. A section is also responsible to report to the settings menu class whenever an item has been interacted with. The buttons within a section is placed close together in groups, creating proximity which makes it easier to understand that they belong together, according to Gestalt design principles[29]. In additional, each section has a borderline surrounding it with different colors, which enhances this effect by making regions[12].
When interacting with an item, the player receives feedback through descriptive text on the display, sound effect, and maybe changed appearance of the item. The settings menu might also change due to subsection becoming visible when buttons with subsections are clicked. These feedback effects were implemented using Don Norman's design principles.
A set of various buttons were implemented, which are further explained in Appendix B.2. The settings menu implementation is not limited to buttons only, even though there are only buttons for now added. The idea with the abstract class *SettingsMenuItem* is to allow other items as well, such as slider control, touchpad and so on. This was decided for improved expandability in the future.

Demo Device

The demo device was implemented to give the player more control of the demo, since control is one of the gameflow elements proposed in the article "Gameflow"[35]. This device can adjust the speed of the demonstration, as well as pause the demo. Pausing the demo enables the "step-by-step" functionality, which provides the control of the steps to the player. If the algorithm does not support backward steps, then this button will not be activated. This device is integrated in the sorting table within the sorting algorithm scene, while it is a standalone device in the graph algorithm scene.

3.4.7 Scripts

This section will go into detail of the scripts that can be utilized by any algorithm- or other concept implementations.

Instruction Control Base

The *InstructionControlBase* script is an abstract class which holds all the instructions for the demo or user test, and updates the progress tracker. The *DemoManager* and *UserTestManager* scripts inherit from InstructionControlBase, and are initialized when a demo or user test is started respectively.
DemoManager receives the instructions as they come, automatically or manually, depending on whether the player has paused the demo or not.
UserTestManager receives updates from the interactable gameobjects when the player performs an action on them. The UserTestManager has an integer variable called *ReadyForNext*, which is incremented by one if the action was correct. It has another constant named *UserActionToProceed*, which is initialized by the algorithm when starting a user test. This constant tells how many actions the player must perform in order to progress to the next instruction. Normally this constant equal 1, but for the sake of extensibility, it can be set to any values larger than 0. Bubble sort is currently the only implementation which has this constant set to 2, due to two elements always receives the same instruction. The next instruction is accessible when ReadyForNext equals UserActionToProceed, which also will reset ReadyForNext to zero and restart the process. UserTestManager also takes care of the score of the player's performance, in case the score system is enabled.

Main Manager

The *MainManager* script is an abstract class which controls the main loop of the application. The main class for an algorithm or other concepts should inherit from this class, in order to take advantage of the main functions which are implemented in this class. It contains the tools to pause a demo, step-by-step functionality, and start/stop a task. The user test part is mostly delegated to the classes which inherit from MainManager, for instance *SortMain* and *GraphMain*. MainManager is the center of the application and connects all the important pieces together, as shown in the class diagram in Appendix B.1.

3.5 Sorting Algorithm Implementation

This section will explore the implementation of the sorting environment, prefabs, scripts, and how the players can practice and challenge themselves through the user test teaching mode.

3.5.1 Environment

The sorting scene before and during a sorting task is shown in Figure 3.11. The settings menu is placed in the center of the room, which will be replaced with the sorting table when a sorting task has started. In the left corner is the monitor prefab with some videos available. In the front of the room there are three blackboards, where the central one is used for the pseudocode, the right blackboard gives feedback during and after a user test, and the left one gives specific feedback regarding incorrectly action performed during a user test. A progress tracker is placed on top of the pseudocode for progress feedback.

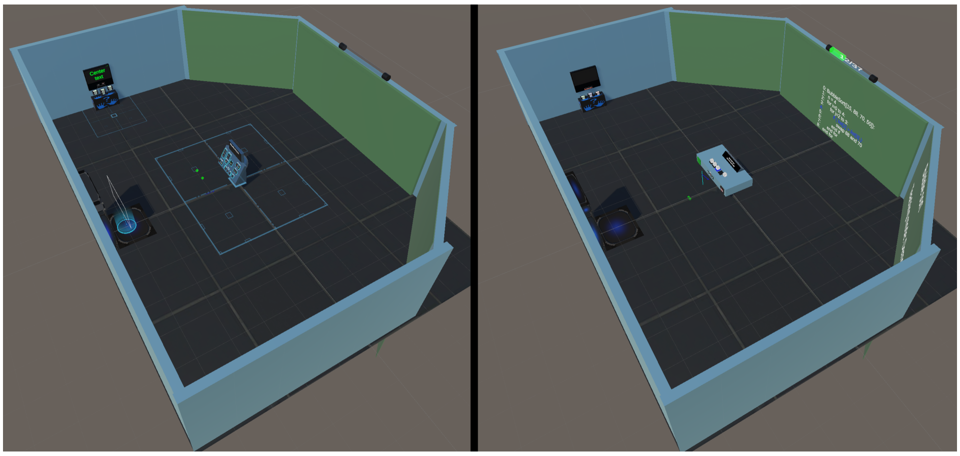


Figure 3.11: The sorting algorithm scene.

3.5.2 Prefabs and Scripts

Sorting Element

Each value of a list is represented by a cube with the value printed on its surface, as shown in Figure 3.12. The appearance of the sorting element has slightly changed based of the usability testing of SortingVR. There was one case where one of the test participants experienced the value changed due to no underline of the value. This was solved by replacing TextMesh- with TextMeshPro texts, which supports underlining text.
The minimum and maximum value of a sorting element cannot be controlled by the player yet, but can be changed in the Unity editor under the settings menu prefab, by a developer. The default value range of a sorting element is [0, 100), and the value is set randomly within this range when a sorting task is generated.

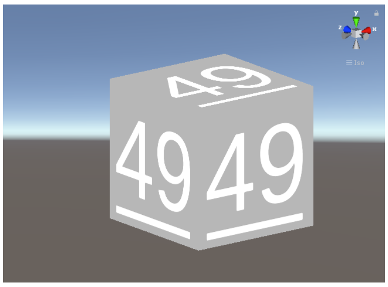
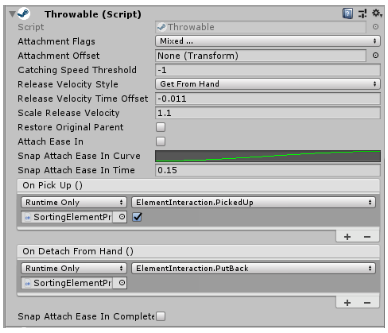


Figure 3.12: The sorting element prefab, with the value 49.

A sorting element prefab uses the following scripts from the SteamVR plugin; *Interactable* for interaction between it and the player, and *Throwable* for a natural physics feeling when thrown. Throwable also has UnityEvents for when the player picks up or detach an item with the hand, as shown in Figure 3.13. In Figure 3.13a, *On Pick Up* is used to mark the element that is picked up, which is used for the comparison action in the bubble- and insertion sort user tests. *On Detach from Hand* is used to set the transform parent of the sorting element back to the sorting table, since this changes to the hand when picked up. Both these actions are performed through the script called *ElementInteraction*, which is partly shown in Figure 3.13b.

Each sorting algorithm has its own sorting element script, e.g. *BubbleSortElement*, which has two main methods. The first method receives incoming instructions and updates the element's state, and the second method evaluates the action performed by the player during a user test. The evaluation uses the data of the current state of



Throwable (Script)

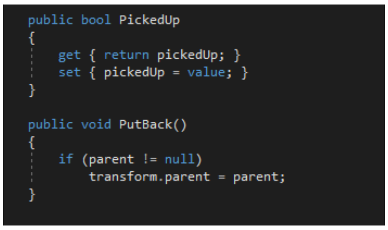
Script	Throwable
Attachment Flags	Mixed ...
Attachment Offset	None (Transform)
Catching Speed Threshold	-1
Release Velocity Style	Get From Hand
Release Velocity Time Offset	-0.011
Scale Release Velocity	1.1
Restore Original Parent	☐
Attach Ease In	☐
Snap Attach Ease In Curve	
Snap Attach Ease In Time	0.15

On Pick Up ()
Runtime Only | ElementInteraction.PickedUp
SortingElementPr | ✓

On Detach From Hand ()
Runtime Only | ElementInteraction.PutBack
SortingElementPr

Snap Attach Ease In Complete ☐

(a) *Throwable* component.



```
public bool PickedUp
{
    get { return pickedUp; }
    set { pickedUp = value; }
}

public void PutBack()
{
    if (parent != null)
        transform.parent = parent;
}
```

(b) Methods in *ElementInteraction*.

Figure 3.13: Sorting element interaction.

the element and tells whether it is placed in the correct holder. The sorting element script is a subclass of *SortingElementBase*, which is an abstract class containing all the shared data about the sorting element's state. Action performed by the player is reported to SortingElementBase, where it collects the evaluation from the subclass, and then reports the status to the *UserTestManager*.

Holder

Each index of a list is represented by a flat cube with its index printed on the front, as shown in Figure 3.14. The holder prefab has improved its functionality from SortingVR whereas it now uses both collision and triggers. During the development of SortingVR it only used collision with the sorting element to see whether it holds an element or not. Now it can register elements above itself with the trigger box (green lines forming a cube in the figure) on top of the surface. The registration of an element is not considered as if the element is placed in the index. This trigger can be used to provide hints, if implemented, to the player in the future. The sorting element must collide with the holder to be marked as placed in this index. Removing an element from the index can be performed by moving the element outside the trigger region.
The text showing the index number can also tell whether the holder is a pivot holder, which then will be marked *Pivot* instead of a number.

The holder object only gives visual feedback to the player, and cannot be directly interacted with. The color of the holder gives information and feedback about state of the sorting element placed on top of it. If the holder is black, then it is in its initial or idle state. When it changes into blue, it means the sorting element

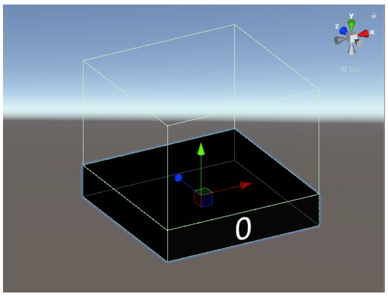


Figure 3.14: The holder prefab, with the index 0.

is being compared with, or highlighted in some cases. It turns green when the sorting element has become sorted. Cyan color is used to mark the pivot element. Lastly, the user test also uses the color red to give feedback to the player about an incorrect action. Incorrect actions will be reported to the *UserTestManager*.

Bucket

The bucket prefab is only used in the bucket sort implementation. It is designed in the Unity editor, and somewhat resembles a bucket. The bucket ID and value range is marked on the front of the prefab, as shown in Figure 3.15.

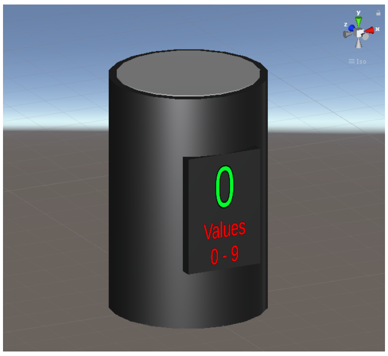
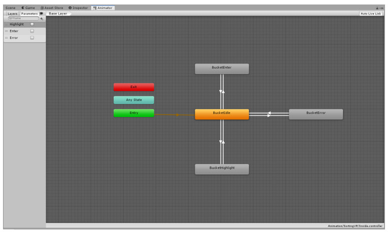


Figure 3.15: The bucket prefab, with ID 0 and holds values from 0 to 9.

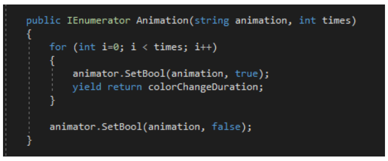
The number of buckets can be set by the player in the settings menu, by clicking the + or - buttons under the *#Buckets* section. When either button is clicked, the number of bucket variable is incremented/decremented by one, and then checked by a while-loop with the condition showed in Equation (3.1), where *max* is the maximum value of a sorting element, and *N* is the next value for the number of buckets. The while-loop continues to increment/decrement by one until the condition is not satisfied. By default, the maximum value of a sorting element is set to 100, and the minimum and maximum number of buckets are 2 and 10 respectively, hence the available number are: 2, 4, 5, and 10 buckets.

$$\text{max} \ \% \ \text{N} \ != 0 \tag{3.1}$$

The bucket prefab uses two triggers to register sorting elements, one when the sorting elements are distributed among the buckets, and another for when the sorting elements have been sorted within the bucket and are placed on top of the bucket prefab.
The "inside" of the bucket changes colors depending on the state of animation, as shown in Figure 3.16, where 3.16a shows the animation states, and 3.16b shows the code which allows easy modifiable animations. The string *animation* contains the name of the boolean for the transition conditions in 3.16a, and the integer *times* can change how many times the animation should be performed.
The inside of the bucket will flash green if the sorting element entered the correct bucket, and red if it was incorrect. The *BucketHighlight* state flashes blue, and is used when displaying the sorted elements of a bucket. And lastly, the idle state is the grey color shown in Figure 3.15.



(a) Bucket animation states.



```
public IEnumerator Animation(string animation, int times)
{
    for (int i=0; i < times; i++)
    {
        animator.SetBool(animation, true);
        yield return colorChangeDuration;
    }

    animator.SetBool(animation, false);
}
```

(b) Method used to changed animation state.

Figure 3.16: Bucket animation.

The pseudocode uses the bucket ID to tell which bucket a sorting element should be put inside, but for improved understanding of the algorithm it was decided to also display the value range. Also, the value range is necessary for user tests above beginner difficulty, due to no pseudocode update.

3.5.3 User Test Difficulty

The support given to the player in the sorting algorithm implementation is presented in Table 3.5. The table has slightly changed since the development of SortingVR, switching column 1 and 2. The rationale for this change was due to the fact that the pseudocode is the main support to learn the algorithm, hence giving it a bigger weight than the holder feedback. This also makes sense in the form of difficulty, which we gradually want to become harder.

Difficulty	Pseudocode	Holder feedback	Pseudocode highlighting
Beginner	Yes	Yes	Yes
Intermediate	Yes	Yes	No
Advanced	Yes	No	No
Examination	No	No	No

Table 3.5: The support provided for each difficulty level in a user test for any sorting algorithms.

3.5.4 Sorting Algorithms

The sorting algorithms are implemented in such a way, that they can see the array as objects, instead of values only. Each value can be picked up and rearranged using the player's hands.

Starting a Task

When entering the sorting algorithm scene then the player can select the algorithm and teaching mode in the settings menu, which is in the center of the room. Each teaching mode has their own sub menu - where the player can set the initial speed of the demo, or the difficulty level of the user test. The player can select the size of the array themself, ranging from 2 to 8 sorting elements. The initial state of the array can also be selected among none, best case, and worst case. Duplicates can also be enabled/disabled. In the optional field, the player can enable pseudocode step (see Section 3.4.5), whether the line number should be visible, and whether to use the user test score system.
When the player is ready, clicking the "ready" button will start the sorting task, and the settings menu will be replaced with the sorting table. The sorting exercise will start when the "start" button on the right side of the table has been clicked.

Since the demo does all automatically, the algorithms below will be explained using the user test.

Bubble Sort

The explanation of the implementation below can be read, or if preferred, this video can be watched.

The bubble sort implementation only uses the props on top of the sorting table, namely the sorting elements and the holders.
All the holders are black to begin with, but starts changing color as the algorithm starts. The sorting starts from the left side of the array, with index 0, where it highlights the two first sorting elements by making them jump slightly above the holders, and the holders turns blue. The player must pick up the sorting elements to ensure the system that they have been checked, and then put them back to the same holders to finish off the comparison action. If the element in index 0 was bigger than the one in index 1, then the player must swap the positions of these. Now the color of the holders will change back into black, and the comparison continues with the next two elements in index 1 and 2. This is performed until the largest element has bubbled its way to the last holder index, which then is sorted and the holder turns green. Then the whole procedure is performed again, until all the elements are sorted, which means all the holders are green.
This implementation of bubble sort is optimized, hence sorted elements will not be compared with again. But it does not count the number of swaps for each round, hence it will not notice in case all sorting elements are already sorted, until all loops have been performed. Hence, there are still room for more optimization.

Insertion Sort

The explanation of the implementation below can be read, or if preferred, this video can be watched.

The insertion sort implementation has one extra holder where the pivot element will be placed. The pivot holder is placed slightly behind and above the array index holders. The pivot holder can be controlled as mentioned in Section 3.4.1.
The first index holder will change into green right away, displaying that the first sorting element is sorted. The next step is to set a variable i equal 1, and then move the sorting element which stand on top of holder i to the pivot holder. The player

does not need to set variables, as these updates themself, but the sorting elements must be rearranged by the player. The pivot element will now start comparing its value with the value of each sorting elements to the left of itself. The comparison element is marked by its holder turning blue. The player must now perform the comparison action on the comparison element, the same way as in the bubble sort implementation. Whenever the pivot element is less than the comparison element, the player must move the comparison element one holder to the right. If the pivot element is greater or equal to the comparison element, or has reached the end of the array, then the player must insert the pivot element into the array again. Now the variable i will increment by 1, and the same procedure is performed with the next pivot element. This is performed until the whole array is sorted, and all the holders are green.

Bucket Sort

The explanation of the implementation below can be read, or if preferred, this video can be watched.

Bucket sort is the sorting algorithm which utilizes the most extra props, but only requires the player to move sorting elements around while not performing any comparisons. It is worth mentioning that the buckets will be placed differently in the demo and user test. In the demo, the buckets will be placed between the sorting table and the pseudocode, in order to make it easier to concentrate on the sorting elements, buckets and pseudocode at the same time. In the user test they are placed to the left and right of the sorting table. It was changed in the user test to make it easier for the player to move elements to them.
The buckets are instantiated when the start button is clicked on the sorting table. It starts to calculate the bucket index for the sorting elements in index 0, moving to the right. In case pseudocode is not visible due to higher difficulty level, then the player must calculate the bucket index using the Equation (3.2), where *value* is the value of the sorting element, N is the number of buckets and MAX is the maximum value of a sorting element.

$$\text{index} = \text{value} * \text{N} \ / \ \text{MAX} \tag{3.2}$$

The sorting element to be thrown into a bucket will make a small jump, and the holder will change into blue for a short period to highlight the element, then turn back into black. The player must now move the sorting element to the correct

bucket, which has the index calculated with Equation (3.2). This can either be done by throwing the sorting element, or pick it up and move/teleport near the bucket and drop it inside. The bucket will change its color using animation, either green or red, as explained in Section 3.5.2.
When all the sorting elements has been placed inside one of the buckets, then each bucket will automatically sort the elements it is holding, and then display them on top of itself. The buckets will use an animation, changing its color to blue while the displaying is going on. The next step will be to move the sorting element back to the sorting table, starting with the smallest element. There are three ways for the player to move the sorting elements back to the sorting table, but two of them are to be recommended. The first way is to pick it up and move/teleport back to the sorting table and place it in the correct holder, smallest element into the holder with index 0 and the biggest element into the largest index. The second option is to drop the sorting element on the floor, which will make them spawn on top of the sorting table, where it can be picked up again and placed into the holder. The third way is to throw it back, but even though it hits the correct holder it will not freeze in the position, and will probably fall on the floor. It is optional how many sorting elements the player brings back to the sorting table at a time, but they must be placed in the holder one at a time. In case pseudocode is active, the sorting elements can be placed on the holder whenever the pseudocode says so.

The sorting process which takes place inside each bucket happens automatically, using insertion sort, and requires no action by the player. For the full experience, this should have been implemented as well. It was not prioritized, since the player can learn insertion sort as well in this application.

3.6 Graph Algorithm Implementation

This section will explore the implementation of the graph environment, prefabs, and how the players can practice and challenge themselves through the user test teaching mode. Details of how the graphs are generated will also be presented.

3.6.1 Environment

The graph algorithm scene had to be larger than the sorting room, due to the idea of having the player traverse the graph themself. The scene is split into three parts: the starting-, graph-, and the supportive area, as shown in Figure 3.17.

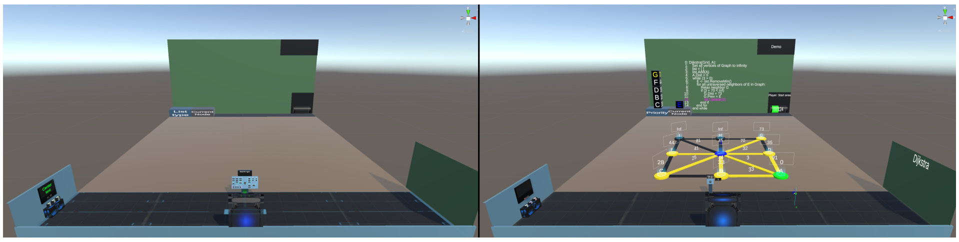


Figure 3.17: The graph scene.

The starting area is where the settings menu, portal (main menu), monitor w/videos, and a blackboard is placed. The player spawns and can freely teleport around in this area.
The next area is where the graph is generated, and where the player's movement is limited to the nodes only.
Behind the graph area is the support area, where the pseudocode, list visualization, position manager, and progress tracker are located.

3.6.2 Prefabs and Scripts

Node

The node prefab is made of a cylinder base with a teleport point on top of it. There are two text fields above the node, one for the name of the node and the other to display the distance. Figure 3.18 shows a node named *X* with a distance equal infinity. The distant text field will only be visible during a shortest path problem. The text rotates such that it is always readable for the player.

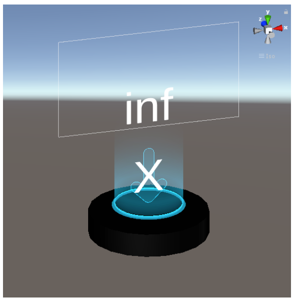


Figure 3.18: A node prefab.

The figures of the animations views in the previous chapter (Figure 2.13 & 2.12) shows how the animations were implemented for the node prefab. Animations are used to indicate what action the player should perform on the node. All the animations apply to all the implemented graph algorithm, but the following example will explain how the node transitions between the states using BFS:

A node in the *NodeIdle* state has the same color as in Figure 3.18. When it is time to enqueue that node it will change the boolean value *NodeCheck* to true and enter the state, which will start the animation, making the node switch back and forth between the idle color and yellow. This animation indicates that the player should shoot the node with the *Pointer* action in order to enqueue the node. When shot, the boolean changes to false and the condition for the state will not be fulfilled anymore, thus returning to the idle state. The appearance of the node will now stay yellow, until it is time to traverse (dequeue) the node. When the time has come to dequeue a node from the queue, then the boolean *NodeTraverse* will become true, and the *NodeTraverse* animation will begin. This animation makes the node switch back and forth between the idle color and blue. Traversing the node by teleporting to it will stop the animation, and the node returns to the idle state. The color of the node will now change into green, meaning it has been traversed. The last state, *NodeError*, will only be entered in case of an incorrect action performed on the node, which will make the node blink red a few times before it automatically returns to the idle state and color.

Edge

An edge is a stretched and rotated rectangle between two nodes. A text field is placed above it which displays the cost of traversing it. This text field is only visible for shortest path problems. The edge cost can be adjusted the same way as the sorting element prefab (Section 3.5.2), and has the default range set to [0, 100]. The edge cost of an edge prefab is randomly set when a shortest path task is started.
There are three different edge prefabs and three classes used. The prefabs are *undirected-*, *directed-*, and *symmetric edge*, as shown in Figure 3.19. There is one base class named *Edge*, which *UndirectedEdge* and *DirectedEdge* in-heritage from. A graph problem using directed edges will sometimes instantiate a directed edge as a symmetric edge, making a path both ways between the two nodes. The chance of a direct edge to become a symmetric edge can be set in the editor, using randomness when instantiated.

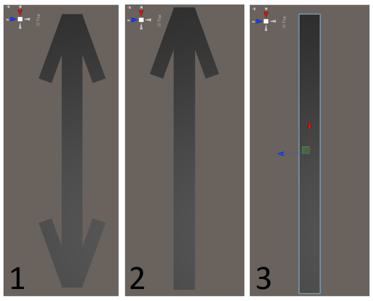
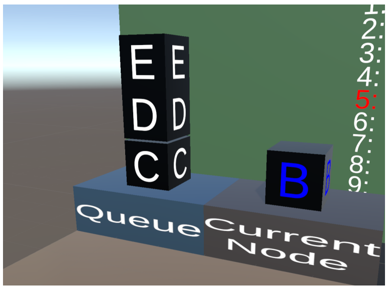


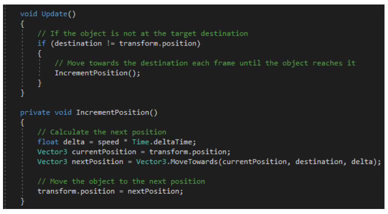
Figure 3.19: The edge variants. 1) Symmetric, 2) Directed, and 3) Undirected.

List Visualization

Graph algorithms use different types of data structures to traverse a graph, such as a Breadth-First Search (queue), Depth-First Search (stack), Dijkstra's algorithm (priority queue). Understanding the data structure of an algorithm is important in order to fully understand the algorithm. Hence, a representation system was implemented to support the player.
A listed node is represented by a cube with the node's ID printed on its surface. For a shortest path problem, it also displays the current distance to the node. All the listed nodes stack on top of each other, while the current active node is moved out of the list, as shown in Figure 3.20a. Figure 3.20b shows the update part of the *MoveObject* script, which is used to smoothly animate the movements of the node representations in the list.



(a) List visual in-game.



```
void Update()
{
    // If the object is not at the target destination
    if (destination != transform.position)
    {
        // Move towards the destination each frame until the object reaches it
        IncrementPosition();
    }
}

private void IncrementPosition()
{
    // Calculate the next position
    float delta = speed * Time.deltaTime;
    Vector3 currentPosition = transform.position;
    Vector3 nextPosition = Vector3.MoveTowards(currentPosition, destination, delta);

    // Move the object to the next position
    transform.position = nextPosition;
}
```

(b) Node representation movement code.

Figure 3.20: List visualization

The node standing in the *Current Node* position in Figure 3.20a is destroyed when it has performed its tasks. The node representations are structured so that it should resemble the data structure, as follows:

- **Queue:** Figure 3.20 (a) shows a queue for BFS, where the node B is the currently dequeued node, and C has waited longest and will be the next to be dequeued. Node E is the last element that was enqueued to the queue.
- **Stack:** Pushing a node will place it on top of the stack, and popping a node will move the node representation which was last pushed on top of the stack to the current node position.
- **Priority list:** A priority list adds the node representation according to its distance, placing it such that the list is always sorted. In a shortest path task, a node can be relaxed, which results in a lower distance for the node. In this case, the node representation also gets updated, changing the displayed value, and maybe also changes its position within the list.

Pointer

The pointer prefab is always located on the right controller of the player. The laser pointer has two purposes - either selecting the input nodes for a graph task, and/or used during a user test for checking nodes. Functionality was explained in Section 3.4.1.
It uses *RayCast* from Unity's physics class, which casts a ray from a starting point, in this example the controller, in the direction of where the controller is aiming, for a range set by the *Pointer* script. The ray can hit any colliders in the game, except for those put under a *LayerMask*, which will be ignored. The ray returns a reference to the object it hit. If nothing were hit, then *hitInfo* of the ray return false.
The next step is to create a laser beam following the trace of the ray. This was implemented with the *Line Renderer* component found in Unity. When hitInfo of the ray hit something, the Line Renderer creates a starting point at the controller position and an end point at the ray hit position, and then draws a red line between those points. When the red line becomes visible it looks like a laser beam, making it perfect to point out nodes.

Calculator

The calculator prefab was implemented to test the player's understanding of the Dijkstra algorithm. It can be used as a normal calculator, but for the Dijkstra implementation it was slightly modified to give feedback to the player as well.
The player must fill in the current node's distance + the weight of the edge leading

to the node being relaxed. Then the player needs to decide whether this value is less or greater than the current distance of the relaxing node, and lastly fill in the distance of the relaxing node. At the end, click the equal sign to check the answer. A correct answer will make the display become green, and then the task will continue, and the calculator will be removed. An incorrect answer will change the color of the display to red, and an error message tells what went wrong. After a few seconds, the calculator resets so that the player can try again.

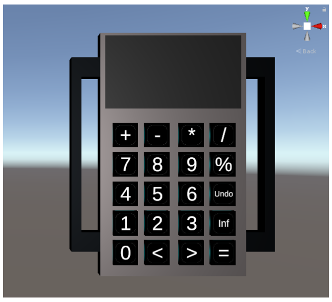


Figure 3.21: The calculator prefab.

It was decided to create a genuine calculator with more than just addition, as shown in Figure 3.21. The rationale for this is due to the effect of having more option available requires more thinking, compared with a calculator with only an addition button - which would make the choice kind of obvious. An undo action was implemented to give more control to the player, as well as error correction[29].

3.6.3 User Test Difficulty

Table 3.6 shows the support given for each difficulty level for user tests. The pseudocode with/without highlighting is given as support the same way as in the sorting implementation.

The animation support was added due to some issues that the player might stumble upon during a graph user test. The graph nodes must be added/relaxed in the same order as the instructions were generated, something which is impossible for those who does not know how it was implemented. Hence, the node has animations which makes them blink with a specific color to indicate whether it should be added/relaxed (yellow) or traversed (blue).
This issue was considered to be fixed by making a new type of node script, which would make it possible for players to add/relax nodes in their own order, while following the algorithm's rules. This was tested, but not finished. The nodes in the

tutorial scene uses a new set of nodes which allows this, but these were not tested for any algorithms.
Direction animation is not given in during examination, even though what has just been explained. The rationale for this was due to the fact that this would reveal what action should be performed on a node, hence destroying the image of taking an exam.
An idea to fix this issue in the future, is to present information to the player which order the neighbors of a node should be added/relaxed. Another way is to use a passive color instead of yellow/blue, which gives the player directional help without telling what action to use.

Difficulty level	Animation	Pseudocode	List visualization	Pseudocode highlighting
Beginner	Yes	Yes	Yes	Yes
Intermediate	Yes	Yes	Yes	No
Advanced	Yes	Yes	No	No
Examination	No	No	No	No

Table 3.6: The support provided for each difficulty level in a user test for any graph algorithms.

3.6.4 Node Building

Real world problems can often be represented by graphs, such as family trees, roads between cities, relationships between people on social media and so on. In order to increase the variation of the graph problems and its applications, it has been implemented various ways to build graphs. A grid-, tree-, and random pattern were implemented to make various graph problems. The graph structures mentioned in the subsections can be selected by the player in the settings menu.

Grid Graph

Creates a graph consisting of nodes and edges in a grid pattern, with M rows and N columns. The player can choose the number of rows and columns to be a value from 1 to 5 in the settings menu. Figure 3.22 shows a shortest path problem using Dijkstra's algorithm on a grid graph.

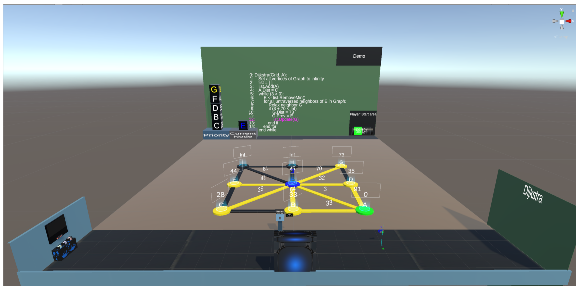


Figure 3.22: Grid pattern.

Tree Graph

Creates a graph consisting of nodes and edges in a tree pattern. The depth of the tree can be set from 0 (root only) to 3, and the number of leaves per node can be set to binary and ternary. Figure 3.23 shows a depth-first search on a binary tree with a depth equals 3.

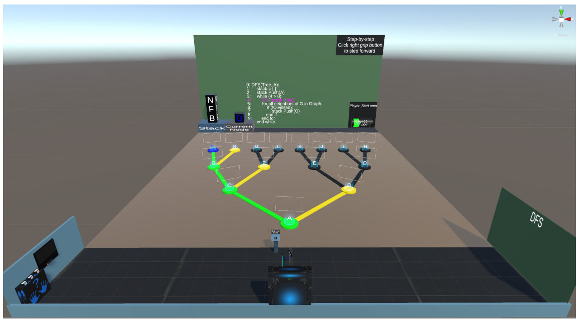
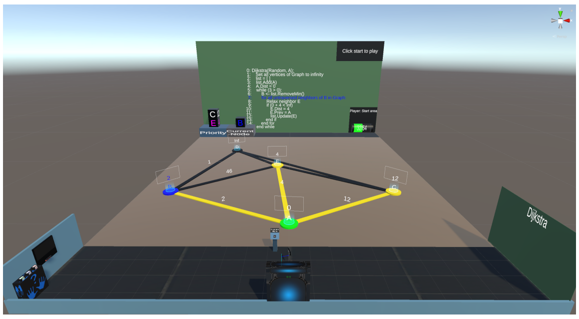


Figure 3.23: Tree pattern.

Random Graph

Creates a graph with the number of nodes and edges chosen by the player. The nodes are randomly placed within the graph area, only limited by the distance between nodes. The distance between the nodes cannot be set by the player, but a developer can change it in the editor. Figure 3.24 shows a shortest path problem using Dijkstra's algorithm on a random generated graph.



```
Dijkstra(Random, A):
    Set all vertices of Graph to infinity
    list = [ ]
    list.Add(A)
    A.Dist = 0
    while (3 > 0):
        B <- list.RemoveMin()
        for all untraversed neighbors of B in Graph:
            Relax neighbor E
            if (0 + 4 < Inf)
                E.Dist = 4
                E.Prev = A
                list.Update(E)
            end if
        end for
    end while
```

Figure 3.24: Random pattern.

3.6.5 Edge Building

The graph can be built using undirected- or directed edges, as explained in Section 3.6.2. In additional, there are four edge building options which changes the way the edges are added to the graph. These were mainly implemented and tested for the grid structured graph. Figure 3.25 shows the four building options, and in Appendix B.2 a small description can be found.

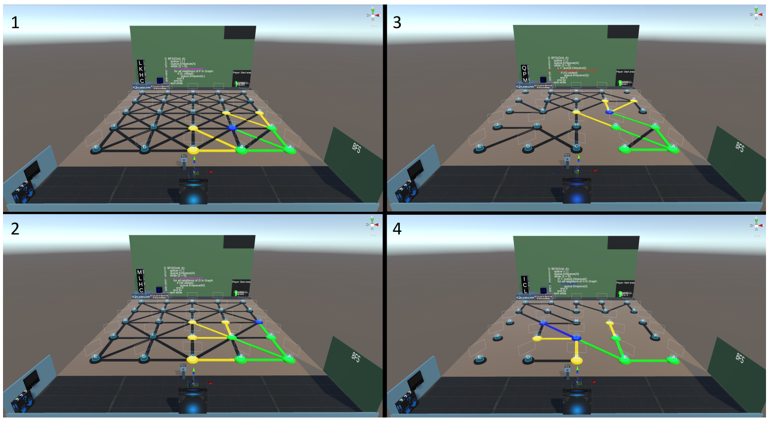


Figure 3.25: 1) Full, 2) Full no crossing, 3) Partial, and 4) Partial no crossing

3.6.6 Graph Algorithms

The graph algorithms are implemented in such a way, that the player becomes a "part" of the algorithm. The idea is that the player will be standing on top of the active node, where they can see and interact from the node's point of view. Interaction is performed by moving together with the algorithm, and perform actions on the nodes on the behalf of the algorithm.

Starting a Task

When entering the graph algorithm scene, the player can select the algorithm and teaching mode in the menu in front of the spawn point. The menu has some of the same sections as in the sorting menu, such as teaching mode and the optional section. The optional section has one extra button, the ”select node(s)” button, which gives the player the opportunity to select the input node(s) themself or leave it to the application. The application will always choose the start node as the first generated node, and the end node as the last, in case it is disabled.
To the left of the algorithm category there is a ”graph task” section which cannot be used, but will display what kind of task the chosen algorithm will perform. The idea of this section was to add more functionality for each algorithm, but for now BFS and DFS can only perform traversal tasks, and Dijkstra’s algorithm can perform shortest path tasks.
When a shortest path algorithm is chosen, an extra option will be available, for now only available for Dijkstra. This makes it possible to choose whether to find the shortest path from a start- to end node, or a start node to all the other nodes. This is currently only available for the demo.
On the right side of the menu there are buttons to choose how the graph will be generated. The player can choose among three types of graph, which were explained in Section 3.6.4. Furthermore, the player can choose what kind of edge to be used, and how they will be built, which was explained in Section 3.6.5.
When the player is ready, clicking the ”ready” button generates the graph, and the settings menu will be replaced by a pillar with a button. The next action depends whether the player chose to select the input nodes or not. If not, then the start button will be active, and it can be clicked right away. In this case, the input node(s) will flash white to show which were chosen by the application. If node selection was enabled, then the start button cannot be clicked, and the text tells the player to choose the input node(s). This can be done by using the *Pointer* action. Selecting an input node manually will also make the node flash white, to give feedback to the player that it has been chosen.
Now that all has been set, it is time to explain each algorithm by using their user test as an example.

Breadth-First Search

The explanation of the implementation below can be read, or if preferred, this video can be watched.

The node that was selected as start node will start to flash yellow, using animation, which indicates that it should be enqueued to the queue. The player must use the pointer action to do so, which will stop the animation and become yellow, meaning it has been added. The list visualization will generate a node representation of the added node. The starting node will now start to flash blue, meaning it is time for the player to dequeue it. Dequeuing is done by teleporting to it. Dequeuing the node will also move the node representation out of the queue and into the "current node" position. The node will now become green, meaning it has been traversed. Now, it is time to enqueue all the neighbors that has not been visited yet. The next node to enqueue will start the yellow animation, the same way as the start node did. Enqueuing a neighbor node will add a node representation of the node in the list visualization as well. When all the unvisited neighbors have been enqueued, then it is time to dequeue the node that has waited the longest in the queue and perform the same procedure again. This will go on until all nodes has been traversed, and the whole graph has become green.
In case directed edges are used, then the search may be aborted if the queue becomes empty before all nodes has been traversed. The traversal algorithms have not been implemented to choose a new start node when this happens, hence it might only traverse a subgraph of the whole graph.

Depth-First Search

A video explanation of DFS can be watched by clicking this link.

DFS works pretty much the same way as BFS, but it uses a stack instead of a queue, hence the traversal order will be different from what BFS produced. Replacing the words *enqueue* with *push*, *dequeue* with *pop*, and *queue* with *stack* in the BFS explanation will pretty much explain DFS. Due to the nature of a stack, the node that has waited shortest, not the longest as in BFS, will be removed from the stack.

Dijkstra's Algorithm

The explanation of the implementation below can be read, or if preferred, this video can be watched.

Dijkstra's algorithm was the hardest graph algorithm to implement, due to more functionality than the traversal algorithms.
All the nodes get their distance updated to infinity when the start button is clicked. The previous edge variable is already set to null when the nodes are instantiated. The first action the player must do is to add the start node which is flashing yellow. Adding the first node will update the distance to 0 and create a node representation. Now there is only one node in the priority queue which Dijkstra's algorithm uses. This node will now start to flash blue, meaning the player must remove it from the priority queue, since it is the node with the lowest distance. Removing it from the priority queue is done by teleporting to the node. The node representation will also move out of the priority queue, and into the "current node" position. The player must now *relax* all its untraversed neighbors. The player can relax a node by shooting the node with the pointer action. The correct node to relax will flash yellow. This will make the calculator spawn in front of the player. Here, the player must insert the slightly modified if-statement, which used to determine whether to update the node's distance and previous edge. The modified part the player must take into consideration, is whether the current node's distance plus the edge cost is less than or greater than the distance of the relaxing node. In the first case it will be: "0 + edge cost <inf".
If the correct values are inserted into the calculator, and the equal-sign button is clicked, then the display will become green and the calculator will disappear, until the next node is shot with the pointer. If the inputted values, operator, or less/greater sign were wrong, then the display will become red and show an explanatory message. Then the player can try again. When all the untraversed neighbors have been relaxed, then the node with the smallest distance will start to flash blue, and the player must teleport to it and do the same procedure again. This is done until the end node is traversed, which then starts the backtracking procedure.

All the node representations which were displayed in the list visualization will now disappear to prepare for the backtracking, and the nodes from the end node leading to the start node will visualized. The end node and its previous edge will become magenta colored, leading the way back to the start node, going one node at a time. The player must now traverse the nodes back to start to finish of the exercise.

See Appendix B.2 for more details of the Dijkstra implementation.

3.7 Evaluation

Learning benefits, drawbacks, and usability data had to be collected in order to answer the research questions. Hence, CourseVR had to undergo some testing and evaluation. This section will present the testing methodology, while the results will be presented in Section 4.2.

3.7.1 Test Group

There were in total eight test participants who tested CourseVR for its teaching ability and usability. None of the test participants were familiar with CourseVR prior to the test.

3.7.2 Test Execution

The tests were conducted at IDI's XR-lab in IT-building 458 Gløshaugen, where CourseVR was developed. A small presentation about the project and test were given to the test participants before the test started. Before equipping the HMD, they were asked whether they had used VR before. In case they did not have much or any experience, then some information about how to put it on and adjust it was explained. They were informed to feel free to think loud, ask questions, and give feedback during the test. They were also informed that they were allowed to take a break or stop the test, in case they felt any discomfort. In this way, the tester and the participant could have an open dialogue throughout the test. Making the test person think loudly helped to discover parts of CourseVR which was intuitive to the developer but was seen differently by the test participant.

The tutorial had been implemented to guide the participant to gain some player skills, hence no information was given to the participant regarding the buttons of the controllers. This was done in order to evaluate the tutorial as a part of the test. The only instruction they were given prior to the test, were that they could freely explore the application, but that it was desirable that they at least tested one algorithm from both the sorting- and graph scene.

All the participants were asked to answer an online questionnaire after they had tested the application. The questionnaire was split into four sections: background of the participant, test summary, system usability scale (SUS) questionnaire, questions regarding education in VR, and a feedback section, which will be discussed in the next chapter.

3.7.3 System Usability Scale

SUS has references with over 1300 articles and publications, hence it has become an industry standard[44]. SUS is a reliable tool for measuring the usability of various products and services, such as hardware, software, websites, and applications. It consists of 10 statements about the application, where each statement has five Likert[22] responses: 1) *Strongly disagree*, 2) *Disagree*, 3) *Neutral*, 4) *Agree*, and 5) *Strongly agree*. All the participants will answer each statement from 1 to 5, choosing the statement which they felt fit their experience.

How to Calculate and Interpret the Score

Interpreting the score can be done by converting each participant answers together to a raw SUS score. This is done by subtracting 1 from the score of odd numbered statements and subtracting the value of even numbered statements from 5, and then sum all these to get the raw SUS score. By multiplying the raw SUS score by 2.5 we end up with the final SUS score. Even though the final score ranges from 0 to 100, it must not be mistaken by percentages. Conducted research has concluded that a score above 68 would be considered above average, while anything below 68 is considered below average[44].

Chapter 4

Result

This chapter will present the results of the implementation- and user tests.

4.1 Implementation Results

Videos will be presented to give a better picture of the algorithm implementation. The videos in Table 4.1 presents some of the supportive tools of the application, while the videos in Table 4.2 shows an overview of the scenes.
Each algorithm has two videos, one for each teaching mode. The first video shows what the player can see and interact with during a demo. The second video shows how the player can practice with the user test on all difficulty levels. The user test videos are the same videos that were linked to in the previous chapter.

The whole playlist of all the videos can be found here as well. Two class diagrams and one scene diagram can be found in Appendix B.1.

Support explanation	Url
Pseudocode	`https://youtu.be/ZSd3neOLhaU`
Monitor	`https://youtu.be/vF_LokzqQ_c`

Table 4.1: Videos of the supportive tools.

Scene	Url
Start & tutorial	https://youtu.be/4ksZOwIYAQA
Sorting scene overview	https://youtu.be/6In_Ua4-ndc
Graph scene overview	https://youtu.be/QtUE6hqovhQ

Table 4.2: Explanation videos of the scenes.

4.1.1 Sorting Algorithm Results

The videos in Table 4.3 contains videos of all the sorting algorithms implemented.

Algorithm	Demo url	User test url
Bubble sort	https://youtube/...	https://youtube/...
Insertion sort	https://youtube/...	https://youtube/...
Bucket sort	https://youtube/...	https://youtube/...

Table 4.3: Videos of the sorting algorithms.

4.1.2 Graph Algorithm Results

The videos is Table 4.4 contains videos of all the graph algorithms implemented.

Algorithm	Demo url	User test url
BFS	https://youtube/...	https://youtube/...
DFS	https://youtube/...	https://youtube/...
Dijkstra's algorithm	https://youtube/...	https://youtube/...

Table 4.4: Videos of the graph algorithms.

4.2 Evaluation Results

Details of the questionnaire result will be presented in this section. See Appendix C for all the details of the questionnaire and the responses.

4.2.1 Participants' Background

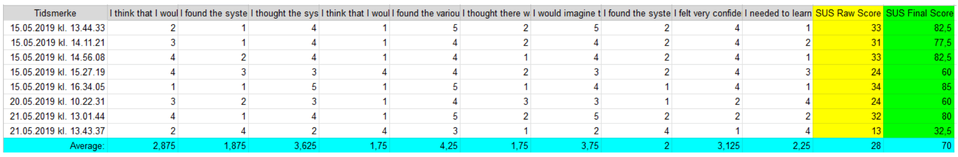
There were in total eight people who tested the application, who were all computer science- or informatics students from NTNU. Everyone except one had taken the course TDT4120 Algorithm and Data Structures. All the participants considered themselves a computer savvy, with 75.0% *strongly agree* and 25.0% *agree*. The prior experience using virtual reality was well distributed with an average of 3.25, which is slightly above *neutral*. Knowledge about the teaching material had an average of 3.88, which is slightly under *agree*.

4.2.2 Summary of the Tested Content

All of the participants took the tutorial. All the algorithms except DFS had their demo tested. All algorithms had their user test tested at beginner difficulty level, a few on intermediate, and none for higher difficulties. On average, each participant watched about 2 demos and tried 2-3 user tests on beginner difficulty. About half of the participants tried a user test on intermediate difficulty.

4.2.3 System Usability Scale Results

The results of the usability part of the questionnaire will be presented in this section. Figure 4.1 shows the feedback of the SUS part, in terms of the individual score, raw SUS score, final SUS score, and the averages.



Tidsmerke	I think that I woul	I found the syste	I thought the sys	I think that I woul	I found the variou	I thought there w	I would imagine t	I found the syste	I felt very confide	I needed to learn	SUS Raw Score	SUS Final Score
15.05.2019 kl. 13.44.33	2	1	4	1	5	2	5	2	4	1	33	82,5
15.05.2019 kl. 14.11.21	3	1	4	1	4	2	4	2	4	2	31	77,5
15.05.2019 kl. 14.56.08	4	2	4	1	4	1	4	2	4	1	33	82,5
15.05.2019 kl. 15.27.19	4	3	3	4	4	2	3	2	4	3	24	60
15.05.2019 kl. 16.34.05	1	1	5	1	5	1	4	1	4	1	34	85
20.05.2019 kl. 10.22.31	3	2	3	1	4	3	3	1	2	4	24	60
21.05.2019 kl. 13.01.44	4	1	4	1	5	2	5	2	2	2	32	80
21.05.2019 kl. 13.43.37	2	4	2	4	3	1	2	4	1	4	13	32,5
Average:	2,875	1,875	3,625	1,75	4,25	1,75	3,75	2	3,125	2,25	28	70

Figure 4.1: System usability scale results.

Chapter 5

Discussion

This chapter will discuss the results presented in Section 4.2.3, and the rest of the feedback which can be found in Appendix C. Feedback from the participants, and observations from the testing sessions will also be discussed. Next, the requirements that were made in Section 3.2 for the application will be evaluated. The last section will look at the research outcome and try to answer the research questions.

5.1 Questionnaire

The feedback found in Appendix C will now be analyzed and discussed.

5.1.1 Test Participants

There were eight test participants who tested the application, with an average knowledge of the teaching material higher than planned. More candidates were desirable, and also people with less knowledge of the teaching material, but it was hard to find good candidates due to the time of the year, meaning many people were busy. But regardless their knowledge, they can anyways give feedback using their experience throughout the years at NTNU, to tell how they felt about virtual reality versus traditional learning methods.

5.1.2 System Usability Scale Results

The SUS results presented in Section 4.2.3 will be further investigated and discussed in this section. The average of the final SUS score for CourseVR was 70.0, which means the application can be considered above average as for usability[44], but with room for improvements. Each question will now be discussed to see what lies behind the final score.

Q1: I think that I would like to use this system frequently

The answers for this question was almost evenly distributed from *strongly disagree* to *agree*, with an average score of 2.88 - which is slightly under *neutral.* This was expected, considering that 7 out of 8 had already taken TDT4120 - Algorithm and Data Structures, which teaches all the application's content and much more. More answers to this question might be revealed when analyzing the education part of the questionnaire.

Q2: I found the system unnecessarily complex

It seems like the majority of the participants did not find CourseVR unnecessarily complex, with half of the answers put into *strongly disagree.* The average score for this was 1.88, which is slightly under *disagree.* No strongly convincing patterns were found when looking at the individual answers, to see whether prior experience with VR had any impact on the score.

Q3: I thought the system was easy to use

Half of the participants chose *agree* for this statement. The average score was 3.63, which lies between *neutral* and *agree*, leaning towards *agree.* Some feedback in the comment section gave some tips of what can be done to make CourseVR easier to use. More direct information about what to do, and also reduce unnecessary information for easier gameplay was suggested.
This seems to be right from what was observed during the user tests. The parts of the implementation that was not fully explained lead to some time to think before performing an action. When the technique of how to perform the action or task was discovered, then the obstacle was conquered and became easier. This

will be further explained in Section 5.1.4, which also takes the verbal feedback into account.

Q4: I think that I would need the support of a technical person to be able to use this system

The result for this statement supports what was discussed in Q3 regarding the learning curve of the application. 75.0% answered *strongly disagree* and 25.0% answered *agree*, resulting in an average score of 1.75, which is slightly under *disagree*. Most of the participants found CourseVR easy to be self-taught, while one-fourth suggests there are still room for improvements to make the application more intuitive.

Q5: I found the various functions in this system were well integrated

This statement scored high, with only one answer in *neutral* and none below. The average score was 4.25 which lies slightly above *agree*. A well-integrated application is important, especially in an educational context. This might imply that the participants thought the sorting- and/or graph algorithms were implemented and presented in a correctly working matter.

Q6: I thought there was too much inconsistency in this system

This statement also scored in a positively way, with only one answering *neutral* and none above. This means the majority of the participants disagreed with this statement, resulting in an average score of 1.75 which is slightly under *disagree*. Reuse of gameobjects through prefabs might be the reason to this score. With a total of six algorithm implementations, they were bound to share props to make the development phase more efficient. The learning curve for the first algorithm might be steeper than the next algorithm, due to the consistency of appearance and interaction among prefabs used.

Q7: I would imagine that most people would learn to use this system very quickly

The graph for this statement makes a nice bell-shape, with a mean score of 3.75, which lies slightly under *agree*. SortingVR scored 2.67 for this statement, which might imply that the tutorial scene helped improving this score. This further supports that CourseVR can be self-taught, but still has room for improvements to make the application more intuitive.

Q8: I found the system very cumbersome to use

Seven people answered below *neutral* while one answered *agree*, resulting in an average score of 2.00, which equals *disagree*. As already mentioned, overall it seems like the participants found CourseVR not too complex and easy to use after the interaction system was understood. This is a good result, considering that CourseVR contains much content spread over multiple scenes, hence requires some movement to experience it all.

Q9: I felt very confident using the system

With five participants answering *agree*, two for *disagree*, and one for *strongly disagree*, the result of this statement is kinda split in two directions. Looking at the comments and what were observed during the user tests, it seems like there are a few parts of CourseVR which does not appear intuitive enough for the player. This might have caused some stress in order to understand the mechanics of the application for a few participants. Some of the feedback says that they wanted more instruction of what to do, while some also said that there was too much information at a time - causing some visual noise. A golden middle way between these suggestions should be found and implemented to improve this score.

Q10: I needed to learn a lot of things before I could get going with this system

The answers for this statement was also a bit mixed. The average score was 2.25, which is slightly above *disagree*. This statement can be interpreted in more than one way. If we first consider that it was answered based on the need of learning

about virtual reality and its usages, then the answers seem to be that CourseVR managed to teach most of the functionality through the tutorial. But due to the shape of the graph, it could also imply that some users felt that there was a lot of content to learn in the tutorial before even performing a sorting- and/or graph exercise. This was how the statement were meant to be interpreted, but afterwards it became clear that it might have been interpreted in other ways.
Changing the perspective into whether CourseVR managed to teach the algorithm material in a useful way from introduction to practice, we need to consider the background of the test participants. None of the participants answered below *neutral* for how familiar they were with the teaching material, hence the majority felt confidence with their knowledge about sorting- and graph algorithms. Which means most of the participants felt that there was not too much to learn before starting, while a few felt otherwise.

5.1.3 Evaluation of the Education Part

The questionnaire contained a section about what the user experienced during the test session, and what they feel about using virtual reality in an educational context. The statements follow the same structure as the SUS part, having five responses for each statement - ranging from *strongly disagree* to *strongly agree.*

Q11: I experienced discomfort during the test. (e.g. motion sickness)

CourseVR is heavily dependent on having the player to move around, especially in the tutorial- and graph scene. Movement in VR can result in motion sickness if not implemented in a good way, and an educational application should not make the player feel any discomfort, hence it felt natural to ask this question.
There was nobody who asked to quit the test session due to motion sickness, but one participant reported that 15 minutes was tough for the eyes.
Looking at the responses we see that this fits with the seven participants who answered *strongly disagree*, and one *agree.* This will be further discussed when we look at the usability as a whole for the application, which is one of the non-functional requirements.

Q12: The application taught me something

The most important task of an educational application is to be able to present the teaching material such that the user can learn from it. This statement was created in order to be able to answer the research question. The average score for this statement was 3.38, which is above *neutral*. This is a higher score than expected, considering the high familiarity with the teaching materials among the test participants. It is hard to draw any conclusion to what exactly each participant meant with their response, due to the unstructured nature of the test and this statement being very broad.
More testing with people who are not familiar with the teaching materials should be conducted, in order to get a better answer to this question. More specific questions should also be planned to make it easier to find the educational value of each implemented part of the application.
This will be further discussed when looking at the teachability of the application, which is another non-functional requirement of CourseVR.

Q13: I feel that the application was exciting to use

This question was created to see whether the test users found CourseVR appealing from the elements put into the application, based on the heuristics from "What makes things fun to learn?"[19]. The application should not only be easy to use, but should also make the user feel excited and challenged.
With only one *neutral* vote and the rest above, the average score for this statement was 4.13, which is slightly above *agree*. This supports the theory that was used in order to create challenging tasks, even for people who are familiar with the teaching materials.

Q14: I would recommend this application to someone who is taking an algorithm course

A good indication when evaluating the educational benefits of an application, is to see whether the participants would recommend it for other people. This is something that anyone can answer, familiar or not with the teaching materials. Those who are familiar with the teaching material can use their experience of how they learned it, and then compare it with their VR experience. People who are unfamiliar can tell from their experience using VR, and how they felt.

All participants for this test group were all familiar with sorting- and graph algorithms, hence they can compare it with their own experience, e.g. lectures of TDT4120.
This statement got an average score of 3.88, which is almost *agree*. Most of the responses were positive to this, with three votes for both *agree* and *strongly agree*. One vote was given for *strongly disagree*, which can be concluded due to the usability of the application, where the same participant felt that CourseVR was unnecessary complex and not easy to use.

Q15: I would prefer to use a VR application over traditional learning methods (books, videos, lectures etc.) for learning about algorithms or other concepts.

People have different preferences when it comes to learning techniques. RQ3 is involved with this statement, and it was therefore asked to see the response of the test group.
As expected, the responses for this was not unambiguously, having three votes for both *strongly agree* and *neutral*, two votes for *disagree*. This results in an average score of 3.50, which is in the center of *neutral* and *agree*. Considering the feedback given verbally after each user test, and the comments left in the questionnaire, then most people were positive to the application, and found VR engaging and interesting. There are a few things that were mentioned by those who were bit negative to this statement, such as the cost of VR hardware, and concerns about discomfort. Economics is one thing, but the concerns about discomfort is a tough one in an educational situation, since the user would prefer to feel well while studying.

Q16: I feel that virtual reality is a good approach for learning about algorithms

Another statement to see what the test participants felt about learning algorithms through VR, in order to answer the research questions. This thesis uses algorithm concepts to investigate the benefits and drawbacks of using VR as a learning tool, but it also wants to learn more about the full potential of VR. This is a personal opinion, which takes the test users' eagerness to use a similar educational VR application in the future.
All the answers were put into *neutral* or above, resulting in an average score of 4.00, which equals *agree*. This indeed supports that it is feasible to use VR for

learning about algorithms.

Q17: I think there is a potential for virtual reality in an educational context

The last statement builds on top of Q16, to see how the test group felt about using VR in other teaching domains. There exist many other concepts that might take advantage of the immersive experience that VR provides.
This statement received a higher average score than Q16, with the majority of responses in *strongly agree*, resulting in an average score of 4.88. This further supports that VR is a good educational tool according to the test group. There are many hard subjects or parts of the teaching concepts that University students must go through, such as physics, statistics, calculus and so on. Taking advantage of the immersiveness and interaction of a VR application, might be of help to overcome common obstacles found in these subjects.

5.1.4 Feedback

The verbally- and written feedback from the test participants will now be discussed. Most of the participants left a written comment in the questionnaire, see Appendix C, but some other things were also brought up after the user test was finished.

Feedback from the Participants

Many of the participants liked the way CourseVR was implemented, having an optional tutorial, various algorithms with many options and settings, informative and performance feedback, and support systems. The pseudocode support was brought up by all the participants, saying it was nice to have the code being highlighted and updated throughout an algorithm exercise.

Start button: The starting process of an algorithm was pointed out to have a flaw due to the line of sight. When the ”ready” button was clicked in the settings menu, then some participants thought the exercise started right away, due to the start button on the sorting table or pillar (graph) fell out of sight. Hence, some time were spent not knowing whether the task had started or not, until they found the start button. Feedback suggested that the moving the buttons to a better position would help a lot, such as placing the buttons on top of the sorting table instead of having them in front, and giving a warning message to the user if the graph was entered before starting the task.

Better instructions: Some actions brought up some confusion of what to do, especially the comparison action for bubble- and insertion sort was not intuitive enough, according to the participants. The comparison action did not tell in any way that the comparison elements had to be picked up and placed back to the holders, hence some of the participants assumed there was no action. This made them go directly to the swap action, which caused the holders to go red and making them confused. Some of the participants discovered how it worked themselves, but many had to be given some hints in order to progress.

Animations: More use of animation were also wanted by some of the participants, especially in the sorting scene where the sorting elements just swap places instantly. The pseudocode in the sorting scene was also getting some criticism due its position relative to the sorting elements, making it hard to focus on both things at the same time. The feedback suggested that making the sorting elements move more continuously, using animation, would be more appealing to the eyes. This might also make it easier to concentrate on the pseudocode and the sorting elements at the same time.

Menu details: One participant said it would be useful if more explanation was given in the settings menu, such as an preview in form of an image, or game objects. As for now, there is a textfield display on the top of the settings menu, which gives a short description of the last button clicked. A brought-up example was the graph types, which only told it was a grid, tree, or random graph. In order to see how it actually looked like, the user had to click ”ready” to generate the graph.

Graph colors: It was reported that the colors used for nodes and edges in a graph was not intuitive from the beginning. There should at least have been an explanation of the colors within the tutorial, the same way as the holder colors were presented.

Extra Suggestions from the Participants

There were some suggestions for new ideas to improve CourseVR as well. These were the following:

Restart button: A restart button was considered a missing feature. This would make it easier to restart an algorithm task without having to stop the session, click the ready button on the menu and then the start button again.

Reduce overflow: A few participants also noted that there were too much going on at the same time, causing information ”overflow”. One suggested that creating some sort of head-up display (HUD), or a virtual device such as an iPad to take care of the information that is not important for the time being.

5.1.5 Observations & Thoughts

Some observations from the user tests will be presented in this section.

Button Interaction

It was observed that some of the test participants tried to interact with the buttons in the tutorial in other ways that intended, such as aiming with the teleport aim at the button or trying to push the button physically with the controller.
The first case where the participants tried to use the teleport aim to click a button seemed to be a result of the users having tested another VR application just a few minutes before. CourseVR and another VR based education application were tested simultaneously the first day, where the other application used a laser pointer to interact with buttons. Hence, those who had tried the other application first tried to apply a similar action when they saw the first buttons in the tutorial, without taking a close look at the video panel. When they realized that it did not work, then they watched the video panel and quickly understood what went wrong.
The second case where some who tried to physically push the button with the controller. There were three reasons reported by the participants: what feels natural for us, how the functionality was presented, and VR experience.
When we interact with buttons in real life, we click it using our hand or finger, hence it would be more intuitive to do so in a virtual environment as well.
Also, the video panel prefab has a short video of how to perform an interaction, and a small figure with some text in the lower right corner. The figure was made small in order to not block too much of the video, thus making it hard to examine for those who encountered this problem. By only watching the video it might appear as the button is clicked by simply touching it with the controller.
Lastly, it is easier for those who have played various VR games compared with beginners to adjust themselves to the application. This we can see from the first

case explained above, which were people with less VR experience, hence trying to apply what they just had learned in another application.

Reflection:
Having implemented a tutorial was indeed a good idea, which helped to prepare the users for the real tasks inside the sorting- and graph scene. This observation found a few usability issues which should be addressed.
The videos which presents the interaction inside the tutorial can be upgraded by adding voices that explains how to perform the action, such as which button to click on the controller. In this case the figure and texts become less important in some aspects, and could be removed. But taking into consideration that users with hearing disability also should be able to play the application, then the figure and text should be enlarged and maybe placed on another panel next to the video panel.
Making the buttons more natural by adding a push feature, is an idea that was thought of earlier in the development process. This was not implemented though, rather following the way which the SteamVR plugin had implemented their buttons. Using animation for the button prefabs has also been considered, but was not added due to other features that were prioritized.

Button Experiment

A button experiment was added to the tutorial, shown at 2:02 in the tutorial scene video, where two pillars with a red "button-like" object on top, are presented in front of the door leading to the next room. These buttons are not used anywhere else in the application, and does not have any information on them.
The majority of the participants said when seeing these, that they resembled buttons, hence clicking these will probably open the path, which indeed they do. Most of the participants used what they had learned in this room, by first touching it and then clicking the trigger button. A few participants were observed to physically push it down, but realizing it did not work they proceeded by doing the correct action. One of the participants who tried to push it down added that the virtual environment gives rise to many ways of interaction, compared with a normal computer game where you only click a button on the keyboard or mouse to perform an action.

Reflection:
The appearance of these buttons was not similar with the buttons introduced in the first menu, and taking the opportunity of interaction within VR into consideration, hence trying other ways of interaction is not strange at all. This makes it even more

important to make game objects in the virtual environment intuitive.

Start and Restart

As some of the feedback given in Section 5.1.4 has already mentioned, the issues regarding the start button of an algorithm task was observed as well.

Reflection:
This is an issue that was overlooked during the development of the application due to a few reasons. The first reason is that the intermediate feedback text which are located on the sorting table in the sorting scene, and in the top right corner of the pseudocode in the graph scene, which tells the user to "click start to play". The participants reported that there was much information received at the same time, hence this might be the reason why this was not noticed.
The second reason was that it was considered it would be natural for the user to see the buttons upon start, or look for it due to the first reason. For instance, the idea of having the height-adjustment buttons on the front side of the sorting table was obtained from desks in real life which has this feature. Hence, all buttons were placed in this position.
The third reason is that the introduction videos provided in each algorithm scenes would present all the possible interaction for each teaching mode. The location of the start button would become clear if they watched either a demo or user test video before starting a task. It was not many participants who utilized the videos, hence not providing much help at all. The monitor prefab was introduced in the tutorial scene, but might not have been used due to its placement in the algorithm scenes. Furthermore, the videos might have been considered as algorithm introduction videos only, hence these might have been ignored due to the participants already being familiar with the teaching material. At last, the schedule for testing was kind of tight, hence the videos might have been ignored to spend more time testing the core functionality of the application.

The suggestions of moving the buttons on top of the sorting table to make them more visible is a good idea though, and should be considered in future development of the application. The idea of having a restart buttons seems also like a good idea, which makes the process of beginning a new task with the same settings less dependent on the settings menu.

Intermediate Instructions

As also mentioned in the feedback section, some parts of the implementation did not give clear enough instructions of what do to during a user test. The highlighting of the pseudocode and other feedback could for some action be clear, but such as the comparison action in the bubble- and insertion sort implementation, did not give clear instruction of what needed to be done. It was observed that the comparison elements were picked up and then swapped right away, which resulted in an error due to the way the comparison action had been implemented for the sorting algorithms.
For the actions used to interact with nodes in a graph, it was observed that the pointer- and teleport action were sometimes mixed during the startup phase. This happened mainly in the tutorial, but also a few times inside the graph scene.

Reflection:
There are many ways to address this problem. SortingVR did not have the comparison action, but during its usability tests it was observed that some participants picked up and put back the sorting elements, even though it was not required. The lack of a comparison action for the sorting algorithms were noted as a missing feature, and was therefore implemented in a similar manner that had been earlier observed.

What first comes to mind is to add more informative instructions during the user test, such as when the elements should be compared in bubble sort, then the feedback display on the sorting table could say "pick up elements **a** and **b** and put them back to perform a comparison". And then if they should be swapped it could say "swap elements **a** and **b**.
Adding voiceover was planned, but not implemented due to low priority. Voiceover can be integrated into the instruction system, making the instruction voice to be played when updating the gameobjects used, such as the pseudocode.
Another planned feature was to add tooltips to the controllers, explaining what each button can do. This was also not implemented due to low priority.
The simplest way to inform the users of how to interact within CourseVR is actually just to prompt them to watch some of the videos before starting a task. This does not require any changes in the code, but will of course make the users spend more time before they can actually start playing the game. But considering that the algorithms within an algorithm scene shares much of the same playstyle, watching one or two videos will probably drastically make the learning curve less steep.

The idea of having a HUD interface or a device with other kinds of information, which was suggested by one of the participants, is also a good idea to give other sorts of information at hand range for the user. For instance, information about the different colors used for sorting elements, nodes, and pseudocode could be explained in such a device. Such a device could also be used to clear up the differences of the actions used for a graph task.

Start Scene

It was observed that the participants who said they had much VR experience tried to use teleportation within the start scene, which did not work as expected.

Reflection:
The choice of not adding teleportation in this scene might have been a small mistake, even though nobody complained about it. The idea was to leave this scene without any controller functionality, since it is a small room that requires only a step to load the main menu or tutorial scene, and that inexperienced players would not know how to use it. This choice of action did not take experienced players into consideration and might have affected their immersion due to the controller not working as expected. This can easily be fixed though, by just adding the teleportation prefab from SteamVR and a couple of teleportation points.

Immersion & Surroundings

Observations of a couple of participants who was totally immersed into the virtual world was noted. It only happened a few times within the first two minutes of the user test, that the surroundings were forgotten, thus they bumped into a desk or chair. They were told to only do small movements, and rather rely on the teleporting functionality for longer distances, which worked out fine afterwards.

Reflection:
This is not the participants fault for forgetting, since SteamVR can be configured to display borderlines within the application using the room scale setup. The room scale setup had been configured to *standing-only* during the development, due to two reasons. First, while developing and knowing how the application works, it was never needed to use anything but the teleport functionality. The second reason was due to the neighbor students at the lab sitting too close, therefore making it hard to draw borders without getting into other students' space, or else the play area becoming too small.

5.2 Requirements

In this section we will take a look at the non-functional- and functional requirements which were made in Section 3.2, and see whether they have been accomplished. The non-functional requirements will be discussed in bigger details in the section below, since these can be used to judge the usability, extensibility, and teachability of the application. The functional requirements will be briefly discussed in Section 5.2.2.

5.2.1 Non-Functional Requirements

The usability and teachability will be discussed based on the results from the user tests. The extensibility will be explained based on the development from SortingVR to CourseVR.

Usability

The usability reflects on how easy and intuitive the application is to use, which was tested by eight students with a spread level of experience with virtual reality. The usability of CourseVR was measured by the SUS statements in the questionnaire, which overall scored positively. The average final SUS score of 70 suggests that CourseVR is above average according to conducted research of SUS[44]. But this score is slightly above the border score of 68, which indicates that there is still room for improvement, as we also can see from the feedback and observation just discussed.
As stated earlier in Section 5.1.2, even though lack of instructions of what to do were reported, the majority of the participants responded that they do not think they would need support of a technical person to be able to use the application. Furthermore, most of them were positive that people would learn to use CourseVR quickly.

When starting the application, the participants stood upon a choice of either taking the tutorial or go directly to the main menu. All of the participants chose to take the tutorial, and were all positive for being given this choice. Comparing SortingVR which did not have a tutorial with CourseVR that has a tutorial, one could indeed see the benefits it brought. For example, those who were influenced by the application they had tried before CourseVR, discussed in Section 5.1.5, could be guided back on the right tracks, using the content of the tutorial, instead of

having an instructor to explain it to them.
As further discussed in Section 5.1.5, virtual reality gives many ways to interact with things. Implementing features that feels natural to us, like pushing a button with the controller, could improve the immersion of the application, and also make it more intuitive for the users.

For the sorting algorithms, the biggest issue was the lack of information about the comparison action, as stated in Section 5.1.4 and discussed in Section 5.1.5. The other actions were more intuitive, such as moving or swapping sorting elements.

The graph implementation has the most vary interactions, where the player must switch between shooting nodes with the laser pointer and teleport to them, compared with the sorting implementation which only requires the player to pick up and rearrange sorting elements. The colors used for a graph task is not introduced anywhere in the application, which was reported as a missing feature in Section 5.1.4. Even though the video panels in the tutorial scene (4:25 in this video) shows how to interact with the nodes, it was not considered intuitive to understand the meaning of the colors. Adding this information in the tutorial- and/or graph scene seems reasonable, in a similar matter to how the colors for the holder prefab is presented, in order to make it more intuitive.

Playing an VR application over an extended period of time might be tiresome for some players. Since CourseVR requires much movements, especially within the graph scene, a question regarding this issue was asked in the questionnaire. The result was very good, without anyone answering that they felt any motion sickness at all. One comment said that it would be really tiring in the long run, as just after 15 minutes this participant felt it straining for the eyes. It is worth noting that this participant also tried another VR application before testing CourseVR. Even though nobody else mentioned any issues regarding discomfort in the questionnaire, some of the other participants mentioned as well that it was a bit tiresome for the eyes after the test. They added that it was not painful though, and that it is probably something that they could become used to over time.

Extensibility

The development of SortingVR during the specialization project was mainly focusing its architecture around sorting algorithms. Two NFR regarding extensibility was used for this project, which addressed that it should be easy to add sorting algorithms with high priority, and adding other algorithms with medium priority.

SortingVR turned into an application which had a base prepared for adding new sorting algorithms, with some parts being more generalized to fit other algorithms or concepts as well. In the end of the specialization project, it was decided that the project would expand with other types of algorithms, in order to explore other ways of interaction in VR. In the beginning of this semester, it was decided to expand SortingVR with some graph algorithms, which made extensibility even more important.
The name SortingVR was changing into CourseVR, and all the classes which could be more generalized by changing the name and slightly rewrite the code was done so. More abstract classes and inheritance was added to further improve the extensibility of the application. For instance, instead of having only one base class for all the algorithms, now there is an abstract class named *TeachingAlgorithm* acting as the base, which stores all the common data and methods among all algorithms. The abstract classes *SortAlgorithm* and *GraphAlgorithm* inherit from TeachingAlgorithm, which takes care of all the specific data and methods for sorting- and graph algorithms respectively. In the bottom of this tree hierarchy are the classes for each algorithm implemented for this application, which overrides the abstract methods from the abstract classes. See the first class diagram in Appendix B.1 for details of this example.
A main loop system for the teaching modes was needed, in order to avoid having redundant code for each new concept added to the application. The abstract class *MainManager* was added to take care of the main delegation, which the classes *SortMain* and *GraphMain* inherit from. The demo teaching mode is all taken care by MainManager, while the sub classes must override abstract methods that takes care of the user test loop.
The interfaces *ITravese* and *IShortestPath* were planned to be used more than it is today, as briefly described in Section 3.6.6. These interfaces can be used to further expand the use cases for the graph algorithms, for example, making BFS perform shortest path on a graph. But in order to do so, the limitations of such an implementation must be taken care of. For the BFS example, the graph must be generated using unweighted edges.

Teachability

CourseVR is built in such a way that players who are unfamiliar with VR and/or the teaching material can be taught this from scratch. The tutorial presents the basics of the controllers with short videos played on a video panel, and some information is presented through figures and text. The monitor prefab was implemented so that the player can watch introduction videos found on YouTube of the algorithms, as well as videos from within the application itself. These methods of learning can

give a basis for the player to build further on top of when they start to use the core features of the application.
The core features of CourseVR is the content within the algorithm scenes. Here they can watch a demo, to learn how the algorithms have been implemented within the application. Furthermore, they can practice their skills in the user test, which has multiple difficulty levels, in order to gradually make the task more challenging. A score system was also implemented for the user test, so that the player can add even more goals, in order to take the challenge to the next level.
The freedom given to the player in the sorting scene, has its pros and cons. The positive aspect with this is that the player can explore by themself, and interact however they like with the cubes. The negative aspect is the fact that the playfulness a virtual environment presents, could lead to unintended actions, such as throwing the sorting elements off the sorting table. This was addressed so that the elements will be brought back to the sorting table automatically.
The graph implementation on the other hand, is more structured in the way that the player cannot perform any unintended actions on the graph itself.

The evaluation of CourseVR was conducted by students who were familiar with the teaching material, which makes it hard to tell whether CourseVR has the ability to teach people who are unfamiliar with the concepts. Based on the received responses and knowing that the majority had taken a course like TDT4120, supports that CourseVR can at least be useful to refresh the memories of sorting- and/or graph algorithms, due to the fact that half of the responses told that they learned something from the application.
The responses indicate that CourseVR has the potential to become a useful tool for students who are learning about algorithm, even though it has not been tested properly by the target audience. The feedback discussed in Section 5.1.3 points towards that CourseVR was exciting to use, and that most of the participants would recommend it to a friend who is taking an algorithm course. Some of the test participants said after the user test, and one left a comment, saying it would have been useful to have such an application as support while taking a course like TDT4120.
They added that it was nice that CourseVR has two teaching modes with various supportive tools, and optional controls. The pseudocode with highlighting was pointed out to be very useful, in order to get a better understanding of the algorithm. Though, the positioning of the pseudocode relative to the sorting table could be improved for better concentration, as well as using more animated movements, as mentioned in Section 5.1.4.

5.2.2 Functional Requirements

Table 5.1 will discuss the functional requirements, which will not include the FRs that were marked with a * in Section 3.2. These will be specified in Table 5.2 & 5.3.

FR	**Overall result**
5	A tutorial scene was implemented as described in Section 3.4.2, and presented in this video.
6	A monitor prefab was implemented to introduce the teaching material, as well as how CourseVR works. This is explained in Section 3.4.5, and presented in this video.
7	A new demo system was implemented using the instruction system only, which provides the same features as the old demo and step-by-step feature of SortingVR
7.1	The demo can be controlled by the player through the buttons within the application, and the controllers.
7.2	Step-by-step is activate when the demo is paused. Forward steps are available for all algorithms, while the sorting algorithms also can perform backward steps. The graph algorithms are almost ready for backtracking as well, but for now it is left out.
8	User tests for the sorting algorithms were implemented during the specialization project, but have been further improved and optimized. User tests for the graph algorithms were implemented as well.
8.1	Four difficulty levels have been implemented, with various support given for each level and algorithm. See Table 3.5 & 3.6 for more details about the support given for sorting- and graph algorithms respectively.
8.2	The blackboard is used for some feedback during a user test, in additional to some customized support, see FR 10 (Table 5.2 & 5.3).
8.3	A score system can be enable in the settings menu, which will give a performance feedback after a user test - calculating the score based on the difficulty, time spent, correct streak, and wrongly performed actions.
8.4	A dictionary keeps tracks of all the mistakes a player does during a user test, using the instruction ID that is present at the time a wrong action is performed. A the end of a user test, these IDs will be presented together with the number of times the mistake(s) occurred.
9	The pseudocode was implemented during the development of SortingVR but has been further improved and optimized. See Section 3.4.5 or watch this video for more details.
9.1	Highlighting of pseudocode was already implemented in SortingVR
9.2	The pseudocode can now be updated in multiple steps, if enabled in the settings menu where it is called "pseudocode step".
9.3	Various colors are now being used for the pseudocode, but are not explained in the application. The pseudocode video in FR9 explains it though.
9.4	Voiceover was not added, but should be considered with higher priority in future work.
11	Various sound effects have been used to create representation systems, which were mainly found on this page
12	A progression bar prefab was created and added to the algorithm scenes, which is controlled by the instruction system.
14	Teleportation was added during the development of SortingVR, using prefabs from the SteamVR plugin. These are still used.
16	Tooltips was not added, but should be considered with higher priority in future work. Tooltips might reduce the need of a tutorial scene, rather giving tips where the player is.

Table 5.1: Overall FR results.

Most of the functional requirements in Table 5.2 were already implemented during the development of SortingVR.

FR	**Sorting result**
1.1	All the sorting algorithms implemented in SortingVR are up to date
1.2	Effort was rather put into the graph algorithms; hence no new sorting algorithms were added
2.1	A random array is generated each time a sorting task is started. Each sorting element has a random value in the range [0, 100) by default.
3.1	The array size can be adjusted by the player, with a length of 2 to 8 sorting elements. Larger array length is not hard to fix, but it was considered any larger arrays would be tiresome, hence setting the limit to 8.
4.1	The player can choose *best-* or *worst case* for a sorting task, as well as *none* which will not affect the order of the elements.
4.2	Duplicates can be enable/disable in the settings menu.
4.3	The number of buckets in bucket sort can now be selected by the player in the settings menu. More details in Section 3.5.2.
10.1	The color of the holder prefab changes color based on the state of the algorithm, and is provided to the player during a demo and user test up to *intermediate* difficulty level. Sound effects are also used for representation system of correct and incorrect actions. See more details in Section 3.5.2.
13.1	Interaction with sorting elements was implemented using the SteamVR plugin. See Section 3.5.2 for all the details.
15.1	Sorting elements that collides with the floor or falls outside the scene (below ground level), will return to the sorting table. This mechanism can be used to the player's advantage during a bucket sort user test, as shown at 2:15 in this video.

Table 5.2: Sorting algorithm FR results.

FR	Graph result
1.1	Two traversal graph algorithms were implemented: DFS and BFS.
1.2	One shortest path algorithm was implemented: Dijkstra's algorithm.
2.1	Various edge building modes were implemented to add some randomness to how a graph is generated. See more details in Section 3.6.5.
2.2	Edges used for a shortest path problem will be given a random value in the range of [0, 100] by default.
3.1	Three graph structures with two adjustable parameters have been implemented. These can be selected and configured in the right section of the settings menu.
4.1	Node selection can be enabled/disabled in the settings menu, where the button is labeled "Select node(s)" in the optional section. When disabled, CourseVR will choose the input node(s) automatically.
4.2	Same as FR 4.1, but when enabled, the player can select the input node(s) after the graph has been generated.
4.3	As briefly mentioned in FR 3.1, three graph structures: grid-, tree-, and random pattern, were implemented. These are described in Section 3.6.4.
4.4	Undirected- and directed edges can be selected by the player in the menu.
10.1	The list visualization (Section 3.6.2) was implemented to present the inner workings of graph algorithms, displaying the state of a queue for BFS, stack for DFS, and priority queue for Dijkstra's algorithm.
10.2	Nodes changes color based on its state, as described in Section 3.6.2. The edges changes color naturally based on the state of the nodes which they are connected to.
10.3	Node animation was created to give some directional guidance, as described in Section 3.6.3. As mentioned, a new type of node was started, but not finished. This new node could improve the experience, by using animation on all the possible nodes that can be interacted with, instead of only one which follows a strict order made by generated instructions
13.1	The pointer class, described in Section 3.6.2, allows interaction with nodes, following the pseudocode terms *enqueue*, *push*, *add*, and *relax*.
13.2	Traversing a node can be done by teleporting to the *TeleportPoint* on top of the node. This will perform the pseudocode terms: *dequeue*, *pop*, and *removeMin*
15.1	The demo device (Section 3.4.6) used by graph demos, and the calculator (Section 3.6.2) used by Dijkstra's algorithm, will spawn in front of the camera of the player when they are needed, or thrown on the floor or outside the scene. The position of these devices will freeze upon spawning, and become affected by gravity when picked up by the player.

Table 5.3: Graph algorithm FR results.

5.3 Research Outcome

In this section, the results of CourseVR will be discussed in the light of the research questions defined in Section 1.3. The main research question was defined as follows:

> *Can virtual reality be utilized for educational purposes in order to improve the learning process, compared with traditional teaching methods?*

What has been discussed and found through this thesis is that the immersive experience is what separates VR from traditional teaching methods. Through CourseVR the user can get a hands-on experience with sorting- and graph exercises. Making such an exercise in the physical world would be time consuming and inefficient, since you would need to put together materials to create a task, such as placing cubes on the table and play with them. Drawing the states of an algorithm on paper is a simpler way to do the same thing, but at the same time loses some of the hands-on experience. A computer application would let you spend less time on preparing the materials, but still cannot provide the same type of interaction VR provides. Restarting such exercises would mean to fix new appearance of the materials, or get a new piece of paper to draw on. Both VR and a computer application have the benefit of simply generating a new exercise with a button click. When all that is said, an application requires that it is first developed before it can be used, and for VR, the extra necessary hardware must be bought and setup before it can be used. The development phase takes a longer time, but when a base software architecture has been implemented, then reuse of code and prefabs can reduce the amount of time required to implement a new concept.
VR isolates the user from the surroundings, which both has its pros and cons. The positive aspect with this is that you can fully focus on the task at hand and get rid of distractions around you. The negative aspect as mention in Section 5.1.5, is that you can bump into the surroundings around the play area, though this be fixed to some extent by properly setting up the room-scale environment. Furthermore, when isolated from your surroundings it can be harder to communicate with those who are in the same room. CourseVR only uses some sound effects and no background music, which makes it possible for nearby people to still be able to talk to the user. For some games that have background music, e.g. Beat Saber[5], it becomes harder to communicate to other people in the same room. SteamVR got a "knock on the door" mechanism which can be used by other people who wants to communicate with the user who is wearing a HMD.
Furthermore, the feeling of being present within the virtual world and being able to interact with virtual objects using the controllers as your hands, becomes more

natural compared with using the keyboard and mouse. As stated by one of the test users in Section 5.1.5, VR provides more ways to interact with objects. If the interaction within a VR application is implemented based on the physical world, then interaction also become more intuitive to the user.

Reading books, watching videos, and attending lectures are some of the methods we use today, which mainly follows a one-way passive communication. When that is said, many lecturers can be good at creating a dialog between the students, opening a two-way active communication. But this is not always possible, hence leaving the students to watch, take notes, and learn from it. Also, in a lecture you can ask questions and get an answer.
A virtual world can provide the same functionality to some extent. Reading text and watching videos is possible in VR, but can be tiresome for the eyes over longer periods of time, as discussed in Section 5.2.1. The main benefit of VR though, is as already mentioned, the possible interaction. A two-way dialog between the user and the application can be created with an intrinsic fantasy, where the user can learn from putting their hands to work, and receive feedback based on their performance.

This thesis also wanted to see whether those who have experienced VR, would prefer to use a VR application instead of the more traditional teaching methods, in order to learn about algorithms or other concepts. The test participants had mixed responses to this question, as mentioned in Section 5.1.3. From the responses we see that almost half would indeed prefer to use virtual reality over reading books, watching videos, and attending lectures. The rest, more than half, were either neutral or a bit against this proposal. The cost of VR hardware and concern of discomfort were brought up as reason for those who were negative to such a proposal.
The cost of tethered virtual reality HMD, which was used for this project, is indeed a drawback when considering the target audience. This was the conclusion that was drawn in the feasibility study of virtual study halls, which is summarized in Section 2.1.1. A possible way to make it more accessible for students is to develop mobile VR version, which was part of the conclusion of the feasibility study. Using mobile VR means the interaction must be implemented in a new way, due to less degree of freedom. The mobile version could be implemented so that it supports accessory equipment, such as the Nolo controller which was described in Section 2.3.4.
The discomfort experienced in VR is tougher to address. Motion sickness is something that many people relate to VR, but when properly taken care of it reduces the chances for it to occur. How movements in the application is implemented, and what hardware is being used, are some of the factors that cause motion sickness. If

movements in the application happens while your body stands still, then the brain gets confused and you might feel nausea. The only discomfort that was reported, was mainly from one participant who said it was tiresome for the eyes. This can vary from person to person and can probably be a bigger issue for those who are inexperienced with virtual reality. The author of this thesis can relate to this issue in the beginning of this project, but became more used to it as time went by.
All the participants were either neutral or positive that VR is a good approach for learning about algorithms, and the majority was very positive that VR has a potential within an educational context. This might indicate that they want to see more concepts from other courses to be implemented in VR.
It leans towards that VR cannot replace the traditional teaching methods yet, but maybe as the technology becomes better, and cheaper, it might become more adapted into the school systems in the future. Also, more available education VR applications might cause more people to want to invest. For now, VR can be used as supplementary support tool for people who finds the traditional learning methods not that useful and has the required hardware. A solution could be to develop application that could be utilized by students at the University, such as at Hackerspace or IDI-XR-lab at NTNU, as an extra helping hands, next to the student assistants.

It is hard to draw a conclusion out of the conducted tests, whether VR can be used to improve the learning process compared with traditional teaching methods, since the tests were conducted by students who were familiar with the teaching material. Also, more testing should have been conducted to get a firmer answer. The discussion in Section 5.2.1, and their positive attitude towards VR in an educational context, indicates that there is a positive trend among the participants that VR indeed can be a useful tool in order to learn more efficiently.

Chapter 6

Conclusion and Future Work

This chapter will draw a conclusion for the results gained through developing and testing CourseVR. Future development of the application and further research will be presented as well, based on what was learned through this project.

6.1 Conclusion

This thesis has explored the benefits and drawbacks of turning educational content into a virtual reality application, in order to evaluate its value and possibilities in comparison with traditional teaching methods. The development of such an application has focused on algorithms, more specifically sorting- and graph algorithms. The results gained through this project, especially from the user testing of CourseVR, shows that virtual reality has good potential as a supplementary educational tool, but might still need some years until it will be broadly adapted into the school systems. Both the feasibility study conducted in 2017, and the feedback for this project shows an interest of using virtual reality for educational purposes, but also some concerns about discomfort while using it, and the price for such hardware. Whether virtual reality can be utilized in order to improve the learning process is hard to answer with the data collected, but it leaves a positive impression that it might indeed be the case. This provides a reason to continue to work with VR projects in the future, in order to challenge these concerns and make it more accessible for those who feel they would like to use virtual reality as a supportive tool in education. The answer to RQ4 can be concluded with a yes, it can indeed be

developed a VR application in order to teach the user about algorithms, and there seems to be a bigger interest for learning about other concepts as well.
CourseVR shows some of the possible ways of implementing a system that lets the user watch visualization of algorithms, and the opportunity to put their own hands to work, in order to learn about algorithms. There are many things to consider when developing an educational VR application, the biggest one, probably being to transform a concept into an intuitive and useful way where the user can learn through interaction. There is also the learning curve for inexperienced VR users who must first learn the basics of the functionality of the HMD and controllers. Implementing a tutorial scene proved to a good solution to teach the basics, but better solutions probably exists.
If the interest of VR continues to increase over the years, maybe some time in the future we might see VR become more adapted into school systems, for more various ways for students to learn.

6.2 Future Work

The first idea for future work is to further improve CourseVR based on the feedback that was described and discussed in the previous chapter.
Furthermore, here are two idea for how to make CourseVR more attractable from a larger audience. This can be done by expanding the content within the application, and creating a version which is available for more platforms, such as the mobile VR headsets that was presented in Section 2.3.3.

Improve CourseVR Based on Feedback

The main issue of CourseVR was to create intuitive actions and making it clear of what the user should do at all time. In additional, the virtual environment provides many sources of information, hence this also became an issue, where the users felt overflown by information. Following the suggestions given by some of the test participants should be considered if continuing the development of CourseVR. The idea of making a virtual tablet with all kinds of information seems like a good direction to address this. The tablet can contain all the unnecessary information which the user can check out anywhere at any time within the application.
Voiceover and tooltips should also be considered in order to improve how information and instructions are given to the user.

Expand courses

The development from SortingVR to CourseVR has only focused on computer science concepts, which can be considered as a prototype for one course. A future plan for CourseVR could be to include other courses, like physics, calculus, statistics, and so on, to make more teaching material available on one VR platform. A scene diagram for a possible future vision of CourseVR can be found in Appendix B.1.

Mobile VR

CourseVR has explored the possibility of creating an educational VR application for tethered VR hardware, which is expensive and requires more work to setup. A mobile VR application on the other hand could make it more affordable and available for those who already owns a smartphone. This solution would mean a simple and cheap HMD, such as Google Cardboard, could be used. This rises new challenges though, since the interaction used in CourseVR could not automatically be used. A solution could be to create a mobile version that also supports third-party devices, such as the Nolo[47], which enables a 6DoF experience even for mobile VR.

Bibliography

[1] Algdat, ????
URL https://algdat.idi.ntnu.no/

[2] amazon.com, ???? Accessed: 25-05-2019.
URL https://www.amazon.com/Google-87002822-01-Official-Cardboard/dp/B01L92Z8D6

[3] Bang, M. M., July 2018. Teaching sorting algorithms in an interactive virtual reality environment.
URL https://brage.bibsys.no/xmlui/handle/11250/2563027

[4] barnard, D., August 2018. History of vr - timeline of events and tech development. Accessed: 26-05-2019.
URL https://virtualspeech.com/blog/history-of-vr

[5] beatsaber.com, ???? Accessed: 20-06-2019.
URL https://beatsaber.com/

[6] Besecker, B., July 2015. How does virtual reality work? Accessed: 26-05-2019.
URL https://www.marxentlabs.com/how-does-virtual-reality-work-part-i

[7] Blow, J., 2004. Game development: Harder than you think. Queue 1 (10), 28.

[8] Carbotte, K., February 2019. Htc vive focus plus includes 6-dof controllers, zero wires. Accessed: 25-05-2019.
URL https://www.tomshardware.com/news/htc-vive-focus-6dof-controller-standalone-vr,38656.html

[9] Cherdo, L., February 2019. Vr headset buying guide: how to choose a virtual reality hmd? Accessed: 26-05-2019.
URL https://www.aniwaa.com/guide/vr-ar/vr-headset-buying-guide/

[10] Dredge, S., July 2014. The guardian - oculus rift sold to facebook. Accessed: 17-03-2019.
URL https://www.theguardian.com/technology/2014/jul/22/facebook-oculus-rift-acquisition-virtual-reality

[11] Endicott, S., April 2019. Oculus quest: Everything you need to know. Accessed: 25-05-2019.
URL https://www.androidcentral.com/oculus-quest

[12] Gkogka, E., January 2018. Gestalt principles in ui design. Accessed: 25-06-2019.
URL https://medium.muz.li/gestalt-principles-in-ui-design-6b75a41e99

[13] Graham, P., September 2018. No official plans to make oculus rift wireless confirms nate mitchell. Accessed: 19-05-2019.
URL https://www.vrfocus.com/2018/09/no-official-plans-to-make-oculus-rift-wireless-confirms-nate-mitchel

[14] Holly, R., May 2016. Everything you need to know about htc vive. Accessed: 09-12-2018.
URL https://www.vrheads.com/everything-you-need-know-about-htc-vive

[15] Jay, March 2016. Field of view for virtual reality headsets explained. Accessed: 12-12-2018.
URL https://vr-lens-lab.com/field-of-view-for-virtual-reality-headse

[16] Kong, T. X.-K., Kruke, A. M., June 2018. Teaching computer science algorithms through virtual reality.
URL https://brage.bibsys.no/xmlui/handle/11250/2567688

[17] Lim, G., December 2012. Einstein – “i never teach my pupils…”. Accessed: 30-03-2019.
URL https://glennlimthots.wordpress.com/2012/12/12/einstein-i-never-teach-my-pupils/

[18] Lorentzen, M., May 2017. Sprøyter milliardbeløp inn i ny it-satsing: ≪pokémon-teknologi≫ skal revolusjonere arbeidshverdagen. Accessed: 30-03-2019.
URL https://e24.no/digital/equinor/statoil-vil-sproeyte-milliardbeloep-inn-i-ny-it-satsing/24003839

[19] Malone, T. W., 1980. What makes things fun to learn? heuristics for designing instructional computer games. In: Proceedings of the 3rd ACM SIGSMALL symposium and the first SIGPC symposium on Small systems. ACM, pp. 162–169.

[20] Mangen, A., Velay, J.-L., 2010. Digitizing literacy: reflections on the haptics of writing. In: Advances in haptics. InTech.

[21] Mazuryk, T., Gervautz, M., 1996. Virtual reality - history, applications, technology and future, 1–3.

[22] McLeod, S., 2008. Likert scale. Accessed: 11-06-2019.
URL https://www.simplypsychology.org/likert-scale.html

[23] nield, D., March 2016. How oculus rift works: Everything you need to know about the vr sensation. Accessed: 09-12-2018.
URL https://www.wareable.com/vr/how-oculus-rift-works

[24] Nielsen, J., April 1994. 10 usability heuristics for user interface design. Accessed: 25-06-2019.
URL https://www.nngroup.com/articles/ten-usability-heuristics/

[25] Nolo-Amazon, ???? Vr headsets accessories console controllers 3d virtual reality system set cell phone mobile vr game headset station controller cardboard vr gaming glasses video motion tracking kit for mobile and pc. Accessed: 25-05-2019.
URL https://www.amazon.com/Headsets-Accessories-Controllers-Controll
dp/B073QL4J61

[26] Nyholm, M., June 2018. Introducing new prefab workflows. Accessed: 18-05-2019.
URL https://blogs.unity3d.com/2018/06/20/introducing-new-prefab-workflows/

[27] Oculus.com, ????. Accessed: 25-05-2019.
URL https://www.oculus.com/quest/?locale=nb_NO

[28] Oculus.com, ????. Accessed: 25-05-2019.
URL https://www.oculus.com/go/

[29] Preece, J., Rogers, Y., Sharp, H., 2015. Interaction design: beyond human-computer interaction. John Wiley & Sons.

[30] Rogers, S., May 2019. Oculus quest: The best standalone vr headset. Accessed: 25-05-2019.
URL https://www.forbes.com/sites/solrogers/2019/05/03/oculus-quest-the-best-standalone-vr-headset/#7e5c61238ed8

[31] Samsung.com, ???? Gear vr with controller (sm-r325). Accessed: 25-05-2019.
URL https://www.samsung.com/no/wearables/gear-vr-r325/

[32] Shilov, A., May 2018. Oculus go now available: Mainstream standalone vr headset starts at $199. Accessed: 25-05-2019.
URL https://www.anandtech.com/show/12715/oculus-go-standalone-vr-headset-available

[33] Statista.com, 2018. Number of active virtual reality users worldwide from 2014 to 2018 (in millions). Accessed: 21-10-2018.
URL https://www.statista.com/statistics/426469/active-virtual-reality-users-worldwide/

[34] Sweeney, T., March 2015. If you love something, set it free. Accessed: 10-12-2018.
URL https://www.unrealengine.com/en-US/blog/ue4-is-free

[35] Sweetser, P., Wyeth, P., 2005. Gameflow: a model for evaluating player enjoyment in games. Computers in Entertainment (CIE) 3 (3), 3–3.

[36] Takahashi, D., March 2019. Htc will sell vive focus plus standalone vr headset for $800 starting april 15. Accessed: 25-05-2019.
URL https://venturebeat.com/2019/03/25/htc-will-sell-vive-focus-plus-standalone-vr-headset-for-800-in-mid-

[37] Tuva Strøm Johannessen, E. H. U., Mars 2019. Åf dropper tegninger i bro-prosjekt: – 3d-modellen inneholder all informasjon. Accessed: 30-03-2019.
URL https://www.tu.no/artikler/af-dropper-tegninger-i-bro-prosjekt-3
460886?utm_source=newsletter-tudaily&utm_medium=email&utm_campaign=newsletter-2019-03-30

[38] Unity, July 2018. Accessed: 02-06-2019.
URL https://docs.unity3d.com/Manual/ScriptingToolsIDEs.html

[39] Unity, 2018. The world's leading real-time engine. Accessed: 10-12-2018.
URL https://unity3d.com/unity

[40] Unity, March 2019. Order of execution for event functions. Accessed: 28-05-2019.
URL https://docs.unity3d.com/Manual/ExecutionOrder.html

[41] unrealengine.com, ???? Developing for oculus rift. Accessed: 09-12-2018.
URL https://docs.unrealengine.com/en-us/Platforms/Oculus

[42] Urke, E. H., July 2018. Equinor ansetter gamere til å ferdigstille oljeplattformer - kan spare store kostnader. Accessed: 30-03-2019.
URL https://www.tu.no/artikler/ferdigstiller-oljeplattformer-med-dat
441178

[43] Urke, E. H., Mars 2019. Hologrammer avdekket hundrevis av feil på johan sverdrup - hololens 2 kan gi enda større muligheter for industrien. Accessed: 30-03-2019.
URL https://www.tu.no/artikler/hologrammer-avdekket-hundrevis-av-fei
459415

[44] usability.gov, ???? System usability scale (sus). Accessed: 08-12-2018.
URL https://www.usability.gov/how-to-and-tools/methods/system-usability-scale.html

[45] vive.com, ???? Vive vr system. Accessed: 09-12-2018.
URL https://www.vive.com/us/product/vive-virtual-reality-system/

[46] Vive.com, ???? Vive wireless adapter. Accessed: 19-05-2019.
URL https://www.vive.com/us/wireless-adapter/

[47] VR, N., January 2018. Nolo home: The world's first-ever 6-dof mobile vr application platform. Accessed: 25-05-2019.
URL https://www.youtube.com/watch?v=tgMq449V5Dg

[48] vrbound.com, ???? Accessed: 25-05-2019.
URL https://www.vrbound.com/headsets/briztech-vr/google-cardboard-20

[49] vrs.org, ???? How did virtual reality begin? - virtual reality society. Accessed: 22-03-2019.
URL https://www.vrs.org.uk/virtual-reality/beginning.html

[50] Wikipedia.org, December 2018. Wikipedia - oculus rift. Accessed: 17-03-2019.
URL https://en.wikipedia.org/wiki/Oculus_Rift

[51] Wikipedia.org, last edited: 2018. Sorting algorithms. Accessed: 09-10-2018.
URL https://en.wikipedia.org/wiki/Sorting_algorithm

[52] www.vrs.org, ???? History of virtual reality. Accessed: 26-05-2019.
URL https://www.vrs.org.uk/virtual-reality/history.html

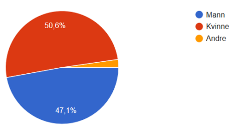
Appendices

Appendix A

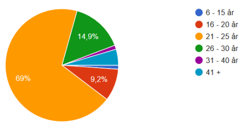
Technology Management Feasibility study

Online questionnaire feedback

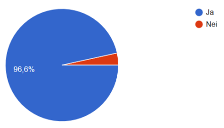
Kjønn (87 svar)



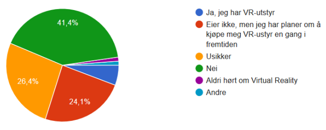
Hvilken aldersgruppe tilhører du? (87 svar)



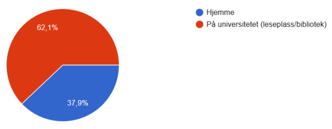
Er du, eller har du vært, student? (87 svar)



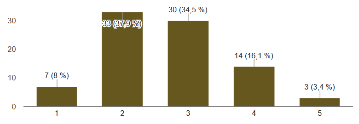
Eier du, eller har du planer om å kjøpe deg VR-utstyr slik som Oculus Rift, HTC Vive, eller annet?
(87 svar)



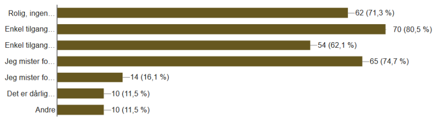
Hvor foretrekker du å lese? (87 svar)



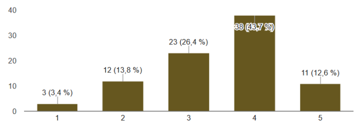
Hvor effektivt studerer du hjemme idag? (87 svar)



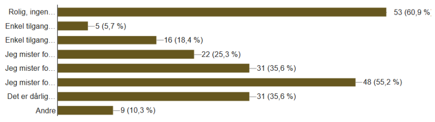
Hva er fordeler og ulemper for deg ved å lese hjemme? (87 svar)



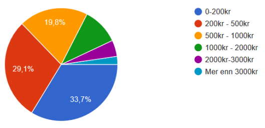
Hvor effektivt leser du på lesesalen/biblioteket? (87 svar)



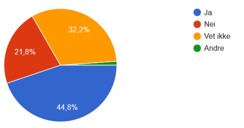
Hva er fordeler/ulemper ved å lese på lesesal/biblioteket? (87 svar)



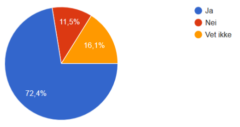
Hvor mye hadde du vært villig til å betale selv for en permanent, virtuell leseplass som beskrevet tidligere?
(86 svar)

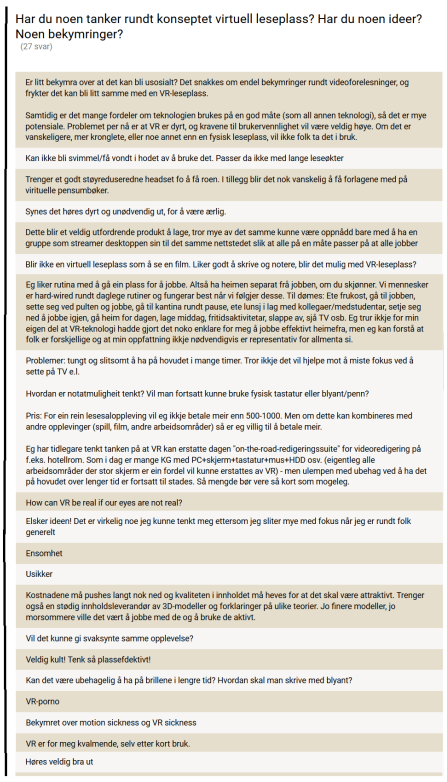


Virtuell lesesal ved hjelp av VR-teknologi (87 svar)



Virtuell pensumbøker ved hjelp av VR-teknologi (87 svar)





Har du noen tanker rundt konseptet virtuell leseplass? Har du noen ideer? Noen bekymringer?
(27 svar)

Er litt bekymra over at det kan bli usosialt? Det snakkes om endel bekymringer rundt videoforelesninger, og frykter det kan bli litt samme med en VR-leseplass.

Samtidig er det mange fordeler om teknologien brukes på en god måte (som all annen teknologi), så det er mye potensiale. Problemet per nå er at VR er dyrt, og kravene til brukervennlighet vil være veldig høye. Om det er vanskeligere, mer kronglete, eller noe annet enn en fysisk leseplass, vil ikke folk ta det i bruk.

Kan ikke bli svimmel/få vondt i hodet av å bruke det. Passer da ikke med lange leseøkter

Trenger et godt støyreduseredne headset fo å få roen. I tillegg blir det nok vanskelig å få forlagene med på viriuelle pensumbøker.

Synes det høres dyrt og unødvendig ut, for å være ærlig.

Dette blir et veldig utfordrende produkt å lage, tror mye av det samme kunne være oppnådd bare med å ha en gruppe som streamer desktoppen sin til det samme nettstedet slik at alle på en måte passer på at alle jobber

Blir ikke en virtuell leseplass som å se en film. Liker godt å skrive og notere, blir det mulig med VR-leseplass?

Eg liker rutina med å gå ein plass for å jobbe. Altså ha heimen separat frå jobben, om du skjønner. Vi mennesker er hard-wired rundt daglege rutiner og fungerar best når vi følgjer desse. Til dømes: Ete frukost, gå til jobben, sette seg ved pulten og jobbe, gå til kantina rundt pause, ete lunsj i lag med kollegaer/medstudentar, setje seg ned å jobbe igjen, gå heim for dagen, lage middag, fritidsaktivitetar, slappe av, sjå TV osb. Eg trur ikkje for min eigen del at VR-teknologi hadde gjort det noko enklare for meg å jobbe effektivt heimefra, men eg kan forstå at folk er forskjellige og at min oppfattning ikkje nødvendigvis er representativ for allmenta si.

Problemer: tungt og slitsomt å ha på hovudet i mange timer. Tror ikkje det vil hjelpe mot å miste fokus ved å sette på TV e.l.

Hvordan er notatmuligheit tenkt? Vil man fortsatt kunne bruke fysisk tastatur eller blyant/penn?

Pris: For ein rein lesesaloppleving vil eg ikkje betale meir enn 500-1000. Men om dette kan kombineres med andre opplevinger (spill, film, andre arbeidsområder) så er eg villig til å betale meir.

Eg har tidlegare tenkt tanken på at VR kan erstatte dagen "on-the-road-redigeringssuite" for videoredigering på f.eks. hotellrom. Som i dag er mange KG med PC+skjerm+tastatur+mus+HDD osv. (eigentleg alle arbeidsområder der stor skjerm er ein fordel vil kunne erstattes av VR) - men ulempen med ubehag ved å ha det på hovudet over lenger tid er fortsatt til stades. Så mengde bør vere så kort som mogeleg.

How can VR be real if our eyes are not real?

Elsker ideen! Det er virkelig noe jeg kunne tenkt meg ettersom jeg sliter mye med fokus når jeg er rundt folk generelt

Ensomhet

Usikker

Kostnadene må pushes langt nok ned og kvaliteten i innholdet må heves for at det skal være attraktivt. Trenger også en stødig innholdsleverandør av 3D-modeller og forklaringer på ulike teorier. Jo finere modeller, jo morsommere ville det vært å jobbe med de og å bruke de aktivt.

Vil det kunne gi svaksynte samme opplevelse?

Veldig kult! Tenk så plassefdektivt!

Kan det være ubehagelig å ha på brillene i lengre tid? Hvordan skal man skrive med blyant?

VR-porno

Bekymret over motion sickness og VR sickness

VR er for meg kvalmende, selv etter kort bruk.

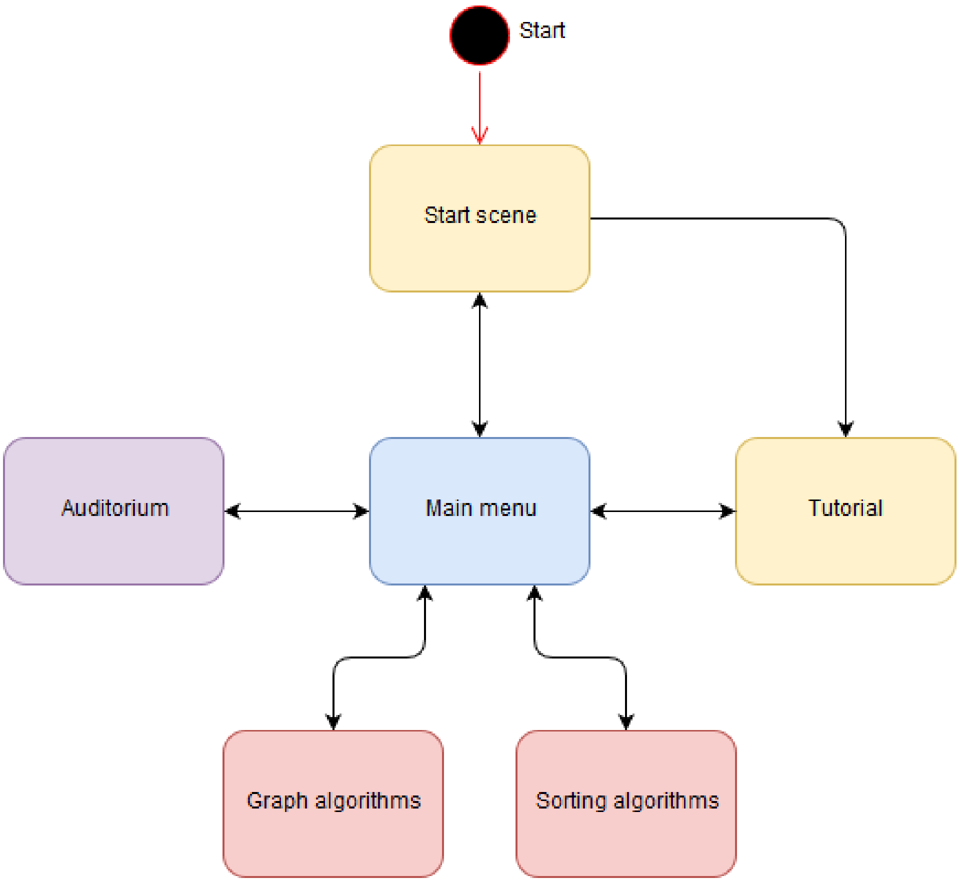
Høres veldig bra ut

Appendix B

Implementation

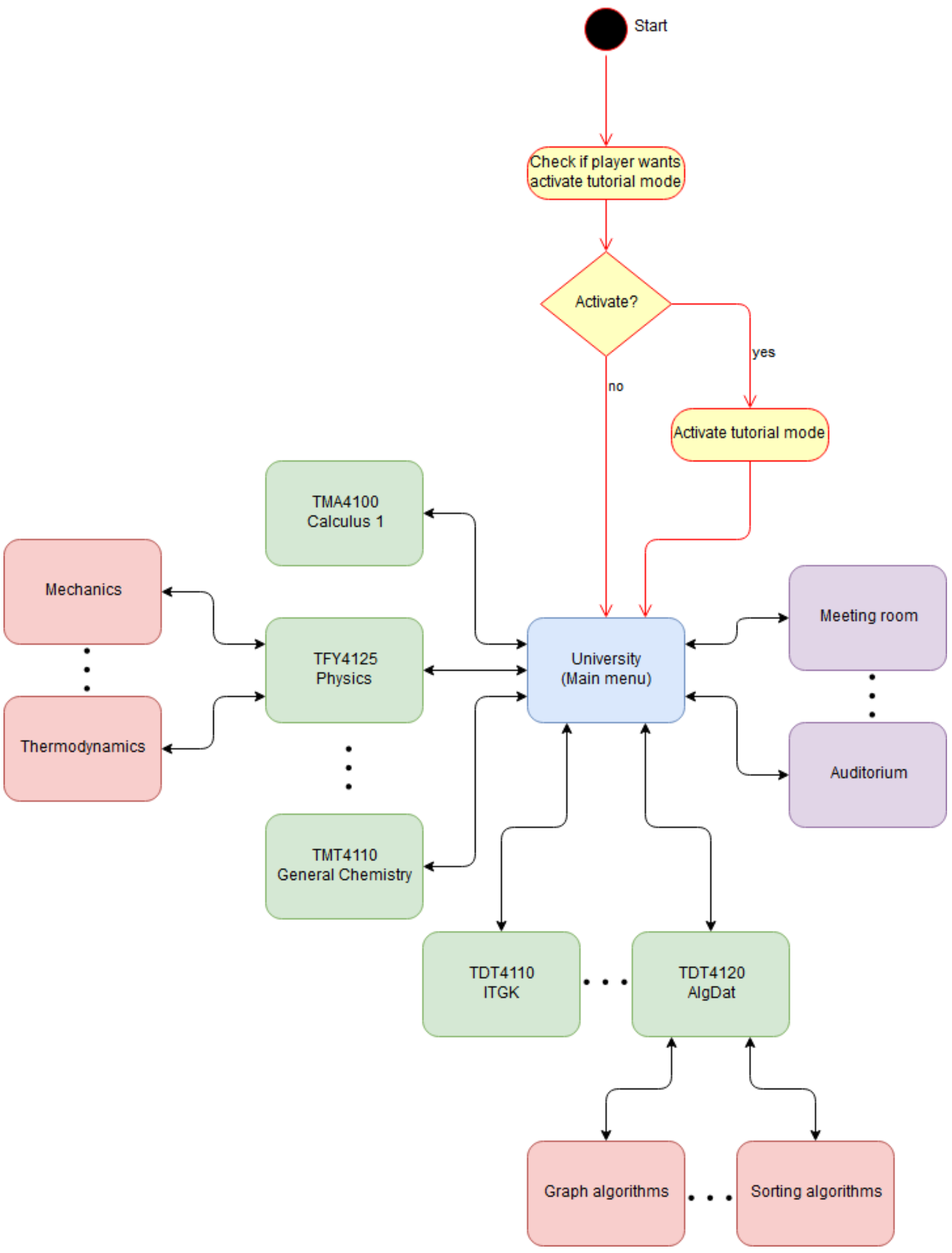
B.1 Diagrams

Scene diagram

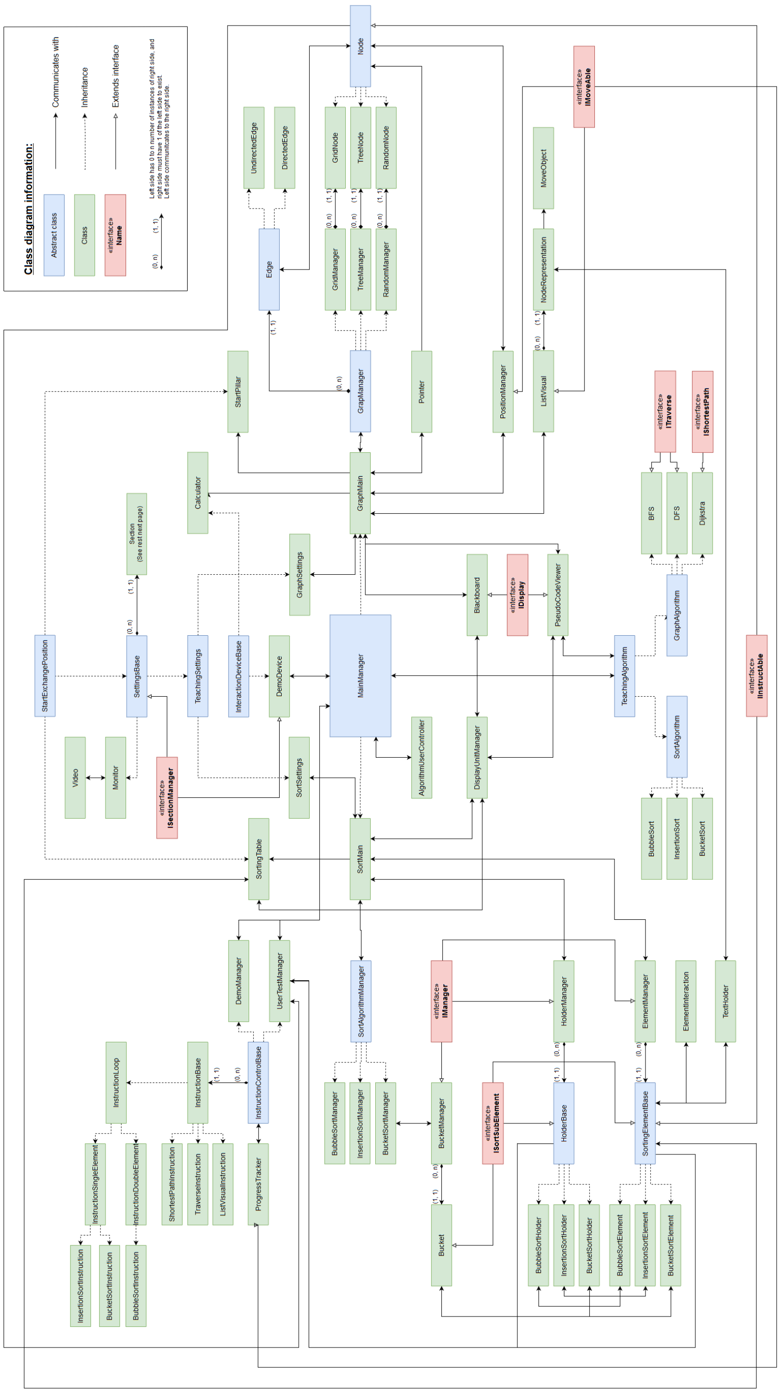


Note: The auditorium scene will not be introduced or discusses anywhere in the thesis. This scene only contains a up-scaled version of the monitor prefab, hence does not contain anything new to be presented.

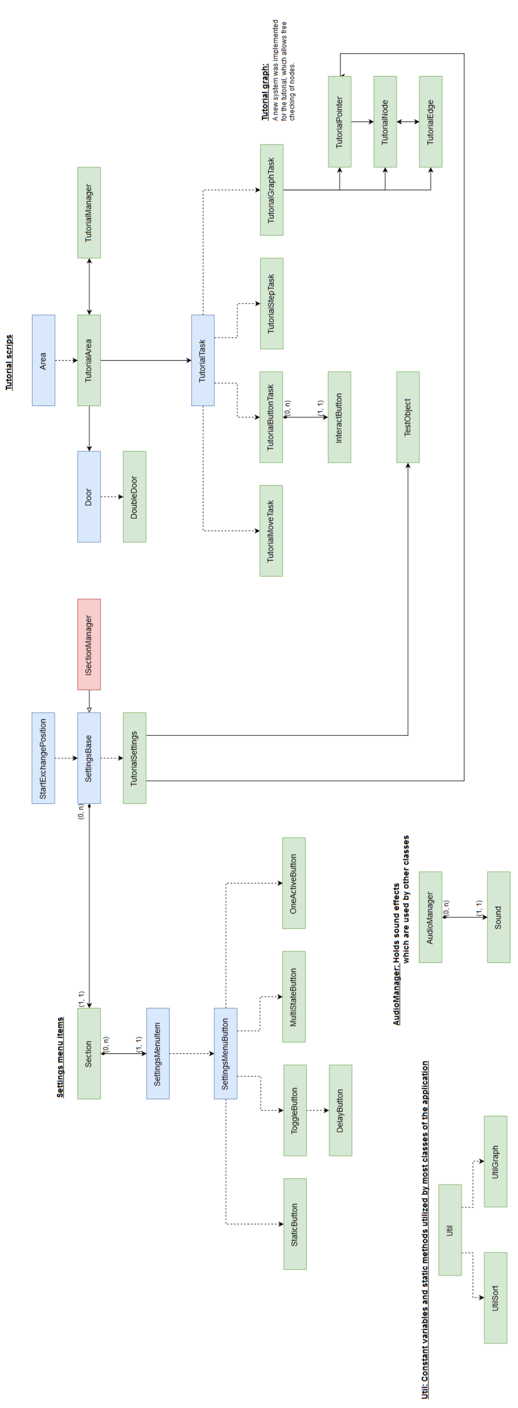
Future scene diagram



Class diagram part 1



Class diagram part 2



B.2 Implementation Notes

Settings Menu Buttons

Note for Section 3.4.6.

- **Radio button:** A set of buttons, where only one can be active at a time. All buttons have the same appearance, except the active one. This gives the users all the possible choices right away.
- **Static button:** A normal button which never changes its appearance.
- **Toggle button:** Toggles between two states (On/Off). It changes appearance based on the active state.
- **Delay button:** Is a subclass of the toggle button. It works the same way as the toggle button, but it also adds a duration after a click where the user can't click again.
- **Multiple state button:** An expanded version of the toggle button, which can contain as many states as wanted. Each state can have different appearance. The radio button shares the exact same usage, though this one does not display all the possible choices at the same time. Each state needs to be iterated from 0 to N, then it starts from 0 again.

Edge Building Note

Note for Section 3.6.5.

1. **Full:** A node has an edge to all its neighbors in a grid graph.
2. **Full no crossing:** Same as Full, but no crossing edges.
3. **Partial:** Randomness is used when building the edges. The chance is set by the developer in the Unity editor.
4. **Partial no crossing:** Same as Partial, but with no crossing.

Dijkstra Note

Note for Section 3.6.6.

- The node representations in the list visualization shows both the ID and distance of the nodes.
- The distance of the current node is displayed above the progress tracker, to the right of the pseudocode.
- If a node already has been relaxed once, but not traversed, then it will have a node representation. If relaxed again and it gets a lower distance, then the node representation will also update itself within the list. (In this case the node will become cyan colored while updating itself).
- All the edge costs will not display from the beginning in order to avoid too much information.
- When it is time to traverse a node, then all edge costs will be removed to improve the view.
- When a node is traversed, then all the edges leading to its neighbors and all the previous edges' cost will be displayed.
- In case the priority queue becomes empty before the end node has been found, then the exercise will finish off since there are no paths leading from the start to the end node.

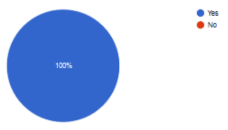
Appendix C

Questionnaire

Participant background

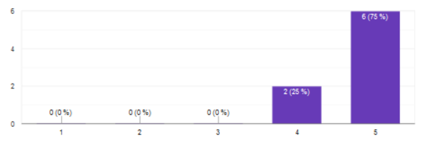
Are you a student?

8 svar



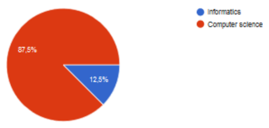
I consider myself well informed about or proficient in the use of modern technology, especially computers.

8 svar



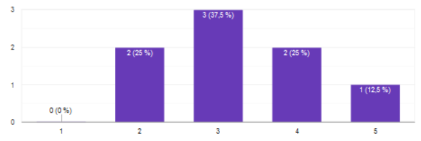
What are you studying?

8 svar



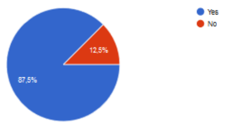
Did you have any experience using virtual reality prior to this test?

8 svar



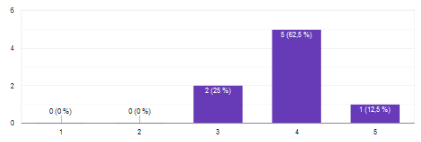
Have you taken the course "AlgDat" ? (TDT4120 - Algorithms and Data Structures)

8 svar



Were you familiar with the teaching materials prior to the test? (Sorting- and graph algorithms)

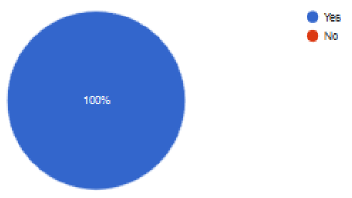
8 svar



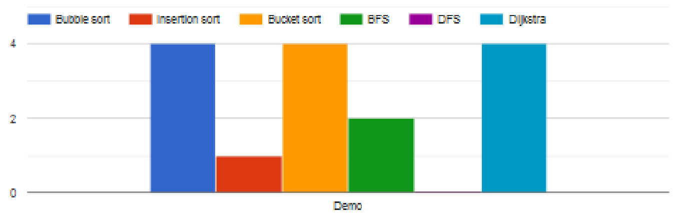
Summary of tested content

Did you take the tutorial?

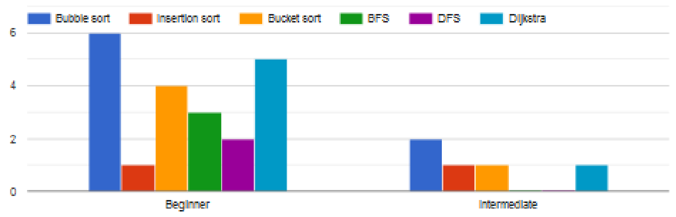
8 svar



Did you watch any of the demonstrations? (Check all that apply)



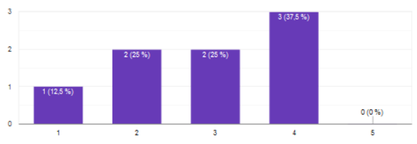
Select the algorithm(s) and difficulty level(s) you tested. (Check all that apply. NB: default difficulty is beginner)



System Usability Scale

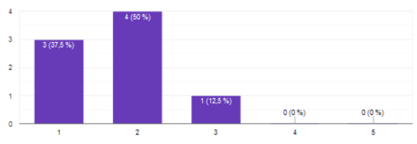
I think that I would like to use this system frequently

8 svar



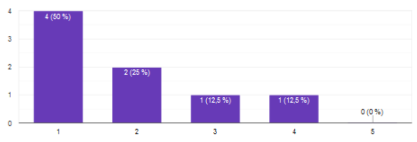
I thought there was too much inconsistency in this system.

8 svar



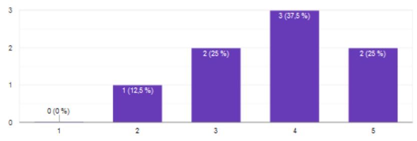
I found the system unnecessarily complex

8 svar



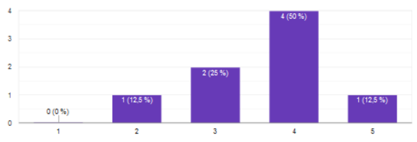
I would imagine that most people would learn to use this system very quickly

8 svar



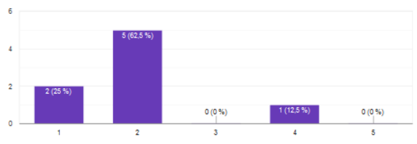
I thought the system was easy to use

8 svar



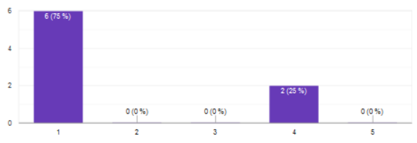
I found the system very cumbersome to use.

8 svar



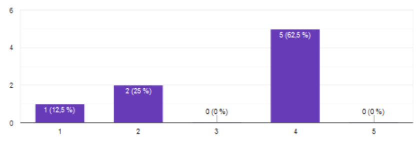
I think that I would need the support of a technical person to be able to use this system.

8 svar



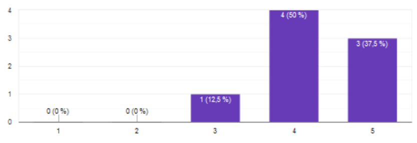
I felt very confident using the system.

8 svar



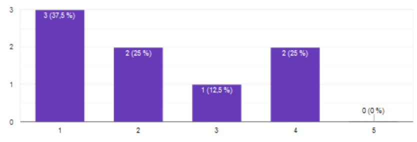
I found the various functions in this system were well integrated.

8 svar



I needed to learn a lot of things before I could get going with this system.

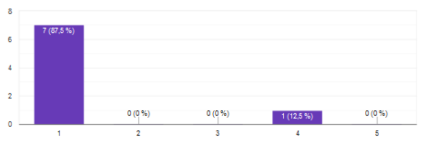
8 svar



Virtual reality and education

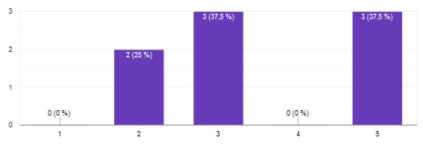
I experienced discomfort during the test. (e.g. motion sickness)

8 svar



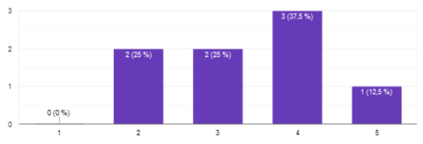
I would prefer to use a VR application over traditional learning methods (books, videos, lectures etc.) for learning about algorithms or other consepts.

8 svar



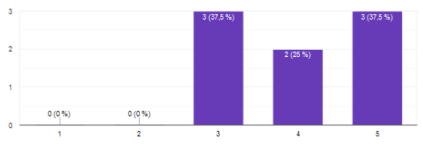
The application taught me something.

8 svar



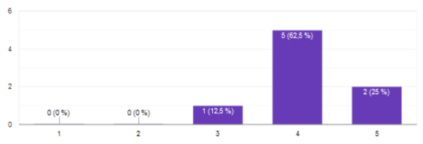
I feel that virtual reality is a good approach for learning about algorithms

8 svar



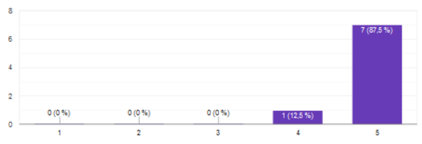
I feel that the application was exciting to use.

8 svar



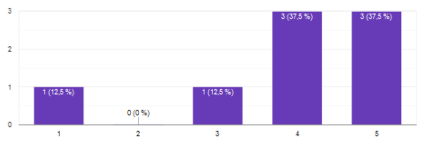
I think there is a potential for virtual reality in an educational context.

8 svar

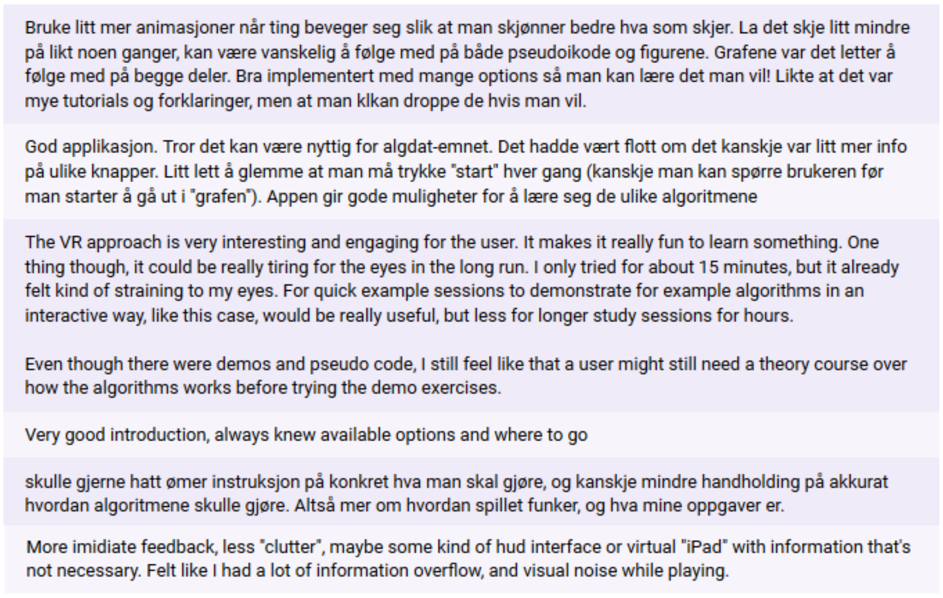


I would recommend this application to someone who is taking an algorithm course.

8 svar



Feedback



Bruke litt mer animasjoner når ting beveger seg slik at man skjønner bedre hva som skjer. La det skje litt mindre på likt noen ganger, kan være vanskelig å følge med på både pseudoikode og figurene. Grafene var det letter å følge med på begge deler. Bra implementert med mange options så man kan lære det man vil! Likte at det var mye tutorials og forklaringer, men at man klkan droppe de hvis man vil.

God applikasjon. Tror det kan være nyttig for algdat-emnet. Det hadde vært flott om det kanskje var litt mer info på ulike knapper. Litt lett å glemme at man må trykke "start" hver gang (kanskje man kan spørre brukeren før man starter å gå ut i "grafen"). Appen gir gode muligheter for å lære seg de ulike algoritmene

The VR approach is very interesting and engaging for the user. It makes it really fun to learn something. One thing though, it could be really tiring for the eyes in the long run. I only tried for about 15 minutes, but it already felt kind of straining to my eyes. For quick example sessions to demonstrate for example algorithms in an interactive way, like this case, would be really useful, but less for longer study sessions for hours.

Even though there were demos and pseudo code, I still feel like that a user might still need a theory course over how the algorithms works before trying the demo exercises.

Very good introduction, always knew available options and where to go

skulle gjerne hatt ømer instruksjon på konkret hva man skal gjøre, og kanskje mindre handholding på akkurat hvordan algoritmene skulle gjøre. Altså mer om hvordan spillet funker, og hva mine oppgaver er.

More imidiate feedback, less "clutter", maybe some kind of hud interface or virtual "iPad" with information that's not necessary. Felt like I had a lot of information overflow, and visual noise while playing.

Sindre Windsrygg

Learning Algorithms in Virtual Reality as Part of a Virtual University

NTNU
Norwegian University of
Science and Technology