NTNU
Norwegian University of
Science and Technology

Teaching Sorting Algorithms in an Interactive Virtual Reality Environment

Marius Meland Bang

Master of Science in Computer Science
Submission date: July 2018
Supervisor: Frank Lindseth, IDI
Co-supervisor: Ekaterina Prasolova-Førland, IPL
Simon McCallum, IDI

Norwegian University of Science and Technology
Department of Computer Science

Teaching Sorting Algorithms in an Interactive Virtual Reality Environment

Marius Bang

June 26, 2018

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

Author
Marius Bang
mariusmb@stud.ntnu.no

Main supervisor
Frank Lindseth
frankl@idi.ntnu.no

Co-supervisors
Ekaterina Prasolova-Førland
ekaterip@ntnu.no
Simon McCallum
simon.mccallum@ntnu.no

Abstract

The use of Virtual Reality (VR) is no longer restricted to use in video games, where VR was first introduced to the mainstream consumer market. VR is continually expanding its presence into several new domains. One such domain is education. Many disciplines are currently being taught using VR, but a hitherto unexplored frontier is that of computer science. It is true that web courses designed to teach programming are gaining a lot of popularity these days, but the underlying theory that is computer science might still seem esoteric and daunting to many. Could the introduction of visualization and interactive elements in a fully immersive virtual environment make computer science theory more accessible to those without computer science backgrounds? This thesis seeks to find the answer to that question. The thesis will present a proof-of-concept in the shape of an application called SortVR. SortVR allows students with no computer science background to learn two simple sorting algorithms—selection sort and bubble sort—through the use of either of two VR headsets, Oculus Rift or HTC Vive. User testing of the application and a subsequent survey will then be used to evaluate the application. Analysis will try to uncover the feasibility of using VR to teach sorting algorithms and what elements should be included in an application designed for that purpose. It is shown that users found the design of SortVR easy to interact with and that they believe the application shows the potential of using VR in the future to teach sorting algorithms and other computer science concepts.

Contents

1 Introduction

> It was also a lot easier for online teachers to hold their students' attention, because here in the OASIS, the classrooms were like holodecks. Teachers could take their students on a virtual field trip every day, without ever leaving the school grounds.
>
> During our World History lesson that morning, Mr. Avenovich loaded up a stand-alone simulation so that our class could witness the discovery of King Tut's tomb by archaeologists in Egypt in AD 1922. (The day before, we'd visited the same spot in 1334 BC and had seen Tutankhamen's empire in all its glory.)
>
> In my next class, Biology, we traveled through a human heart and watched it pumping from the inside, just like in that old movie *Fantastic Voyage*.
>
> In Art class we toured the Louvre while all of our avatars wore silly berets.
>
> In my Astronomy class we visited each of Jupiter's moons. We stood on the volcanic surface of Io while our teacher explained how the moon had originally formed. As our teacher spoke to us, Jupiter loomed behind her, filling half the sky, its Great Red Spot churning slowly just over her left shoulder. Then she snapped her fingers and we were standing on Europa, discussing the possibility of extraterrestial life beneath the moon's icy crust. (*Ernest Cline, Ready Player One*, 2011 (1))

In his New York Times bestseller (2) *Ready Player One*, Ernest Cline describes a dystopian future where an energy crisis and a following war has turned Earth into a desolate and hostile place. The only consolidation for the remaining inhabitants is their free access to a virtual world named OASIS. The experience of being in this world—which can be accessed through visors and haptic equipment resembling our own head-mounted displays (HMDs) and motion controls—is so immersive and fulfilling that most people choose to live out their entire lives in it. While the gap between current technology and the science fiction technology described in Ready Player One remains quite large, OASIS serves as an inspiration for what the future of virtual reality (VR) and augmented reality (AR) might bring. The gap is also closing in as were seeing promising advances in VR and AR technology. Since *Ready Player One* was published in 2011, we have seen the release of Oculus Rift and Oculus Touch, HTC Vive, Microsoft Hololens, Samsung Gear VR, and Google Cardboard among others. These technologies are all promising endeavors in the pursuit of an immersive virtual world, and their developers are continuously working on the introduction of new and better versions of their products. The use domain of VR and AR is also ever expanding, as we are seeing VR being deployed in areas such as real estate (3), army training (4), and education (5)(6)(7). This thesis seeks to further understand the potential of VR in an educational context. Specifically, the use of VR to teach sorting algorithms will be examined. To this end, a VR application called SortVR has been made in the game engine Unreal Engine 4. The application educates the users in the basics of two simple sorting algorithms—selection sort and bubble sort. The sorting algorithms are taught through video, animations, and textual descriptions. The next section in this introduction will present the theory needed to understand the teaching material in SortVR and how it is presented. The final section of the introduction will present a specialization project which provides the background for this thesis. After the introduction, the rest of the sections in the thesis are:

1. **Research Goal**, which presents the purpose of the thesis and poses relevant research questions
2. **Method and Equipment**, which details the software, hardware, and techniques used in the development of the application
3. **Results**, which presents the application and the results of the user testing of the application
4. **Discussion**, which summarizes some of the challenges presented earlier in the thesis and discusses them in further detail
5. **Conclusion**, which concludes the thesis by summarizing what has been found and what remains inconclusive and thus provides the basis for future work

1.1 Theory

1.1.1 Comparison sort

Both of the sorting algorithms taught through SortVR are of the type comparison sort, which is a family of sorting algorithms with certain characteristics that makes the belonging algorithms useful for analysis and real-world applications. Comparison sort employs a comparator, which is an operator able to compare two elements (meaning two data objects of the same data type, such as integers, floating point numbers, or characters in a string) and determine which of the two should come first in a list. Typically, the elements are mapped to some numeric value and then compared using the less-than-or-equal ("$\leq$") operator or the greater-than-or-equal ("$\geq$") operator. For numerical data types, the mapping is trivial, since elements already have a numerical value. For non-numerical values, a meaningful mapping must be created. For characters in a string, for example, a possible mapping is giving each character in the alphabet a value equal to its index in the alphabet (A=0, B=1, and so on). Other operators than "$\leq$" and "$\geq$" can be used as a comparator as long as the operator obeys the two properties of total order:

1. (Transitivity) If $a \leq b$ and $b \leq c$, then $a \leq c$.

2. (Totalness or trichotomy) For all a and b, either $a \leq b$ or $b \leq a$.

Where a, b, and c are three (potentially different) elements of the same data type. "$a \leq b$" in the rules above indicates that a comparator has determined that a will come before b in a list.

Sorting algorithms utilizing comparison sort are ubiquitous due to their simple nature and versatility. Most data can be given some numerical value, and most computers are limited to comparing two numbers at a time (ignoring any parallelization). Given these factors, as well as a computer's ability to load and store numbers in physical arrays, we can see how comparison sort algorithms are relevant. Numerous comparison sort algorithms have been invented, analyzed, and improved over the years. The following sections will cover two of the most basic sorting algorithms: selection sort and bubble sort. These algorithms were chosen for their relative simplicity in terms of the number of steps in the algorithm, the time it takes to learn them, and how simple they are to demonstrate in a three-dimensional VR environment.

1.1.2 Selection sort

Pseudocode typically contains variables, set notation, and other mathematical notation. The pseudocode used in this thesis is in a similar vein as the instructions given in SortVR. Since SortVR is intended to teach algorithms to users who might not have a computer science background, the language used in the instructions is plain English with no special notation.

The selection sort algorithm uses the concept of dividing the list to be sorted into two disjoint parts: one that we know is sorted and another that assume to be unsorted. We will call the former part the sorted part and the latter part the unsorted part (the unsorted part might in reality be sorted, but this is not known to the algorithm). Elements can be marked as sorted, meaning that they have moved from the unsorted part of the array to the sorted part of the array. When all elements have moved from the unsorted to the sorted part of the array, the entire array is sorted and the algorithm terminates.

Pseudocode for selection sort:

1. Find the smallest element in the unsorted part of the array.

2. Swap the element with the first element in the unsorted part of the array.

3. Mark the element sorted.

4. If there are any elements left in the unsorted part of the array, go to step 1. Otherwise, the array is sorted.

Note that the algorithm still produces a sorted list if we find the greatest element rather than the smallest element in step one or if we swap with the last (rather than the first) element in the unsorted part of the array in step two. The table below shows the consequence of changing these two factors. The columns indicate which element is found in step one, while the rows indicate which element is swapped in step two. The inner cells indicate the order of the resulting array and the side where the sorted part of the array will be, using the convention that the first element in an array is the leftmost element and the last element in an array is the rightmost element.

	Smallest element	Greatest element
First element	Ascending, Left side	Descending, Left side
Last element	Descending, Right side	Ascending, Right side

1.1.3 Bubble sort

Like selection sort, bubble sort keeps track of two disjoint parts of the array: a sorted part and an unsorted part. These parts are equivalent to the sorted and unsorted parts of selection sort algorithm, but the way bubble sort moves elements from the unsorted part of the array to the sorted part of the array differs.

Pseudocode for bubble sort:

```
1. Compare each pair of adjacent elements in the unsorted array
2. If they are in the wrong order, swap the elements
3. When you reach the end of the unsorted array, mark the last element sorted
4. If none of the elements had to be swapped, the array is sorted. Otherwise, repeat from Step 1.
```

The version of bubble sort presented in this section is sometimes referred to as optimized bubble sort. This is because the original bubble sort is a simpler version where there is no notion of a sorted and an unsorted array. The implication of this is that more comparisons are done, because pairs of elements already known to be sorted correctly are repeatedly being compared for each iteration. If step three is omitted from the pseudocode above, the result would be equivalent to the non-optimized version of bubble sort.

Another variation to the algorithm that makes it terminate later than necessary would be to omit step four, in which case the algorithm does not keep track of the number of swaps, but runs until every element is marked sorted.

Finally, the order in which elements are compared in step one is not specified. This is done for brevity, in the hope that the user will understand what is meant by examining each pair of adjacent elements from left to right until they reach the end in step three. It is possible to compare the pairs in reverse order, as long as the first element is marked sorted in step three. The consequence of this is that for each iteration, the smallest unsorted element moves to the left side of the array (where the sorted array is located) instead of the greatest element moving to the right side.

1.2 Specialization project

This master thesis is based on the work done in a specialization project (8) completed in the fall of 2017. In this section, a short summary of the most relevant content of the project report will be presented. A brief discussion of the main differences between the application resulting from the specialization project and SortVR, including ideas that were eventually abandoned during the development of SortVR, will also be included.

The specialization project application was a much simpler application than SortVR, both in terms of features, models, and user interface. The project was however built on many of the same ideas that lead to SortVR. The main inspiration comes from a project called Computer Science Unplugged (9). Computer Science Unplugged is a collection of activities and instructional material teaching various computer science topics. The teaching material is taught without the use of computers, in a simple manner meant to be understood by students at lower levels of education without requiring a computer science or mathematical background. The simplistic, broken down approach and gamification elements were appealing and Computer Science Unplugged was therefore a great inspiration for both the specialization project application and SortVR. Initially, the idea was to map various tasks and instructionals from the Computer Science Unplugged web site (9) onto a virtual environment, in order to create an interactive environment for the user to explore. The project report proposed an interactive virtual lab environment for learning and experimenting with a multitude of interactive computer science learning activities. However, due to limited resources and time, and to simplify user testing, both the specialization project application and SortVR has a focus on teaching sorting algorithms. The main inspiration for the specialization project application was an instructional video on the Computer Science Unplugged website (9). In the video, the narrator explains two sorting algorithms (selection sort and quicksort) while the instructor in the frame demonstrates the algorithms with weights and a balance scale. By using the balance scale as a comparator, the instructor is able to correctly sort the weights in ascending order.

Figure 2 shows the starting state of the specialization project application. The user sees five elements (the blue boxes) that the user needs to sort into the five corresponding brown boxes in ascending order. The elements have different masses, which are unknown to the user. The user may not inspect the masses of the elements directly, but is able to compare two elements at a time to see which one is greater. As we have seen in the section above, this is enough information to sort the elements with comparison sort. Comparison of elements is done in one of two ways.

The first comparison method resulted from an attempt to accomplish a direct mapping of the aforementioned

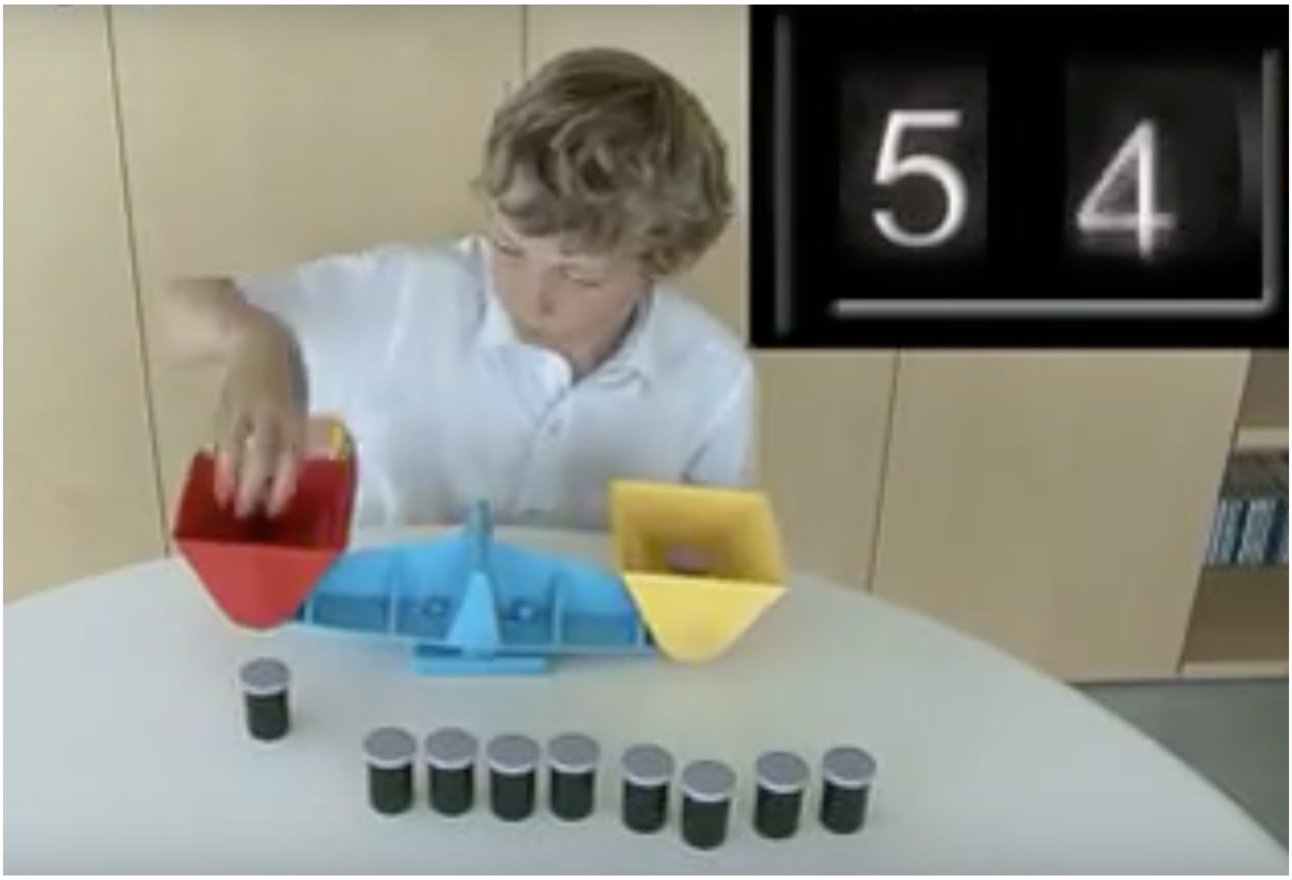


Figure 1: The video on csunplugged.com showing a student in the process of sorting an array of weights using a balance scale

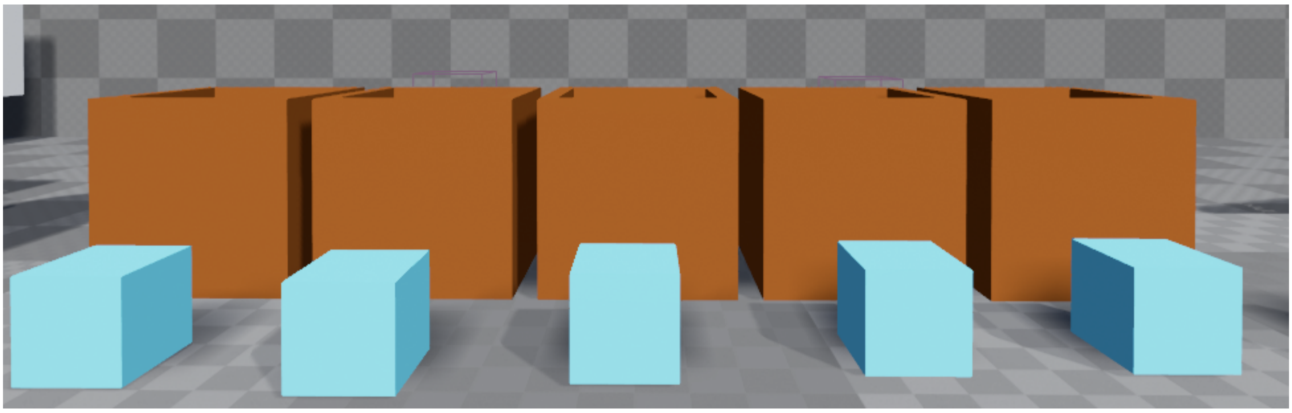


Figure 2: Starting state of the specialization project application.

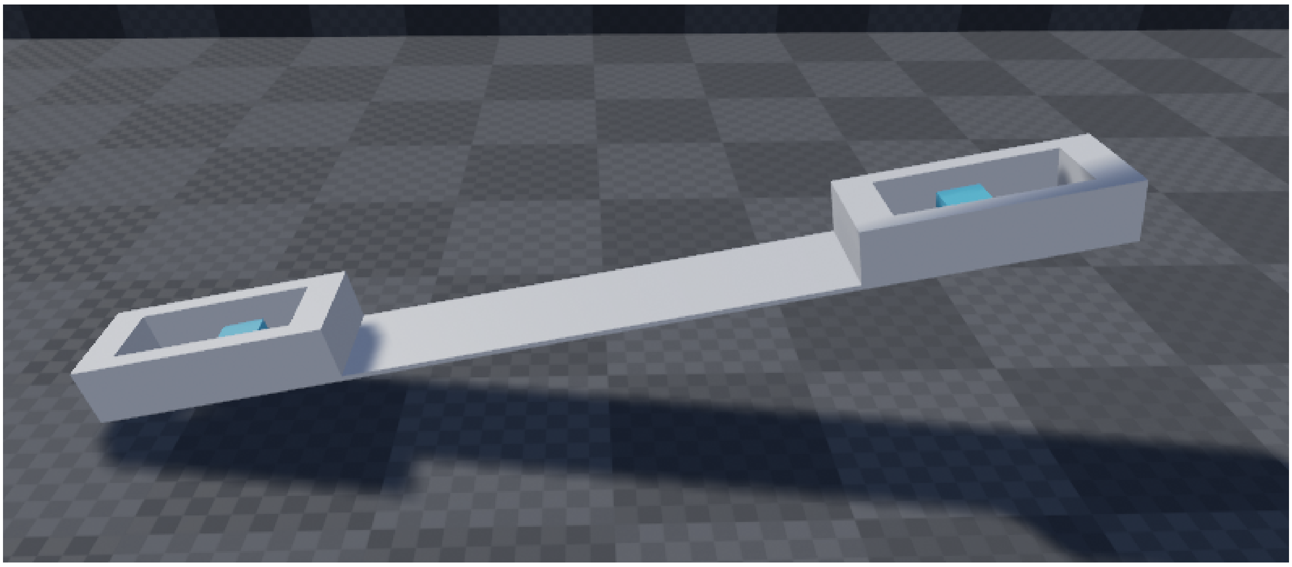


Figure 3: One of the balance board prototypes.

instructional video on the Computer Science Unplugged website. The user has access to a balance scale in the virtual environment onto which the user can place elements on either side. Just like a real-world balance scale, the balance scale would then tip over to the side with the greatest mass. During initial implementation and testing of several prototype balance scales (one of which can be seen in figure 3), many problems with the design were discovered. The method relied heavily on actual physics which made the design of the balance scale non-trivial. Elements had to be placed at the exact same distance from the center of the scale and with the same rotation in order to measure the difference between two masses accurately. This is due to the way a balance scale works by creating torque about the fulcrum (the pivot point at the center of the scale) to pull one of the ends of the scale down. The applied torque is a function of force applied and the distance from the fulcrum to the point where the force is applied, known as the arm. Since the gravitational force applied depends on the mass of the objects, we can keep the arm constant and observe the difference in masses indirectly by observing how they create a difference in torque pulling one of the sides down to the ground. In theory, the implementation of a balance scale adhering to these physical principles should be feasible, albeit demanding for a single inexperienced developer/modeller. There was however another problem with the design which lead to the idea being abandoned entirely; The act of removing elements from the array, placing them on the scale, waiting for the scale to adjust, and then placing the elements back in the array for each comparison was deemed to be all too cumbersome.

The second comparison method is seen in figure 4. The method utilizes the virtual resources at hand rather than being constrained by the limitations of the physical world. The user simply holds an element in each hand to compare the two. The heaviest element then changes its rendered material to gold, while the lightest element changes its rendered material to wood. This method is simpler and more effective than the first, especially when doing many comparisons. There is a loss of affordance since this doesn't really correspond directly to any real-world interaction, but this was seen as an acceptable tradeoff.

The user can proceed to compare elements and place elements in whatever order the user wishes until the user is satisfied with the sorting. The user can then activate a button which changes the material of the containers depending on whether the contained elements are in their correct, sorted position. If an element is in the correct position, its container changes material to gold, otherwise its container changes material to silver. This can be seen in figure 5.

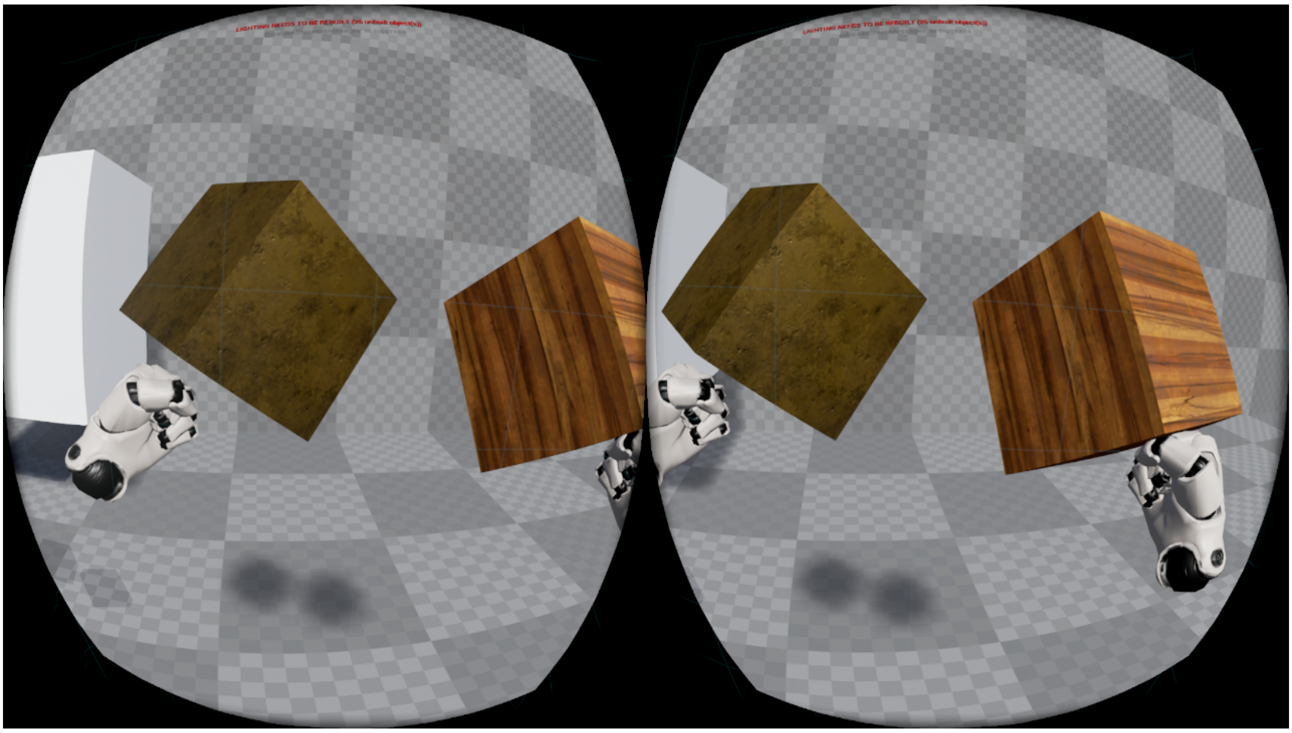
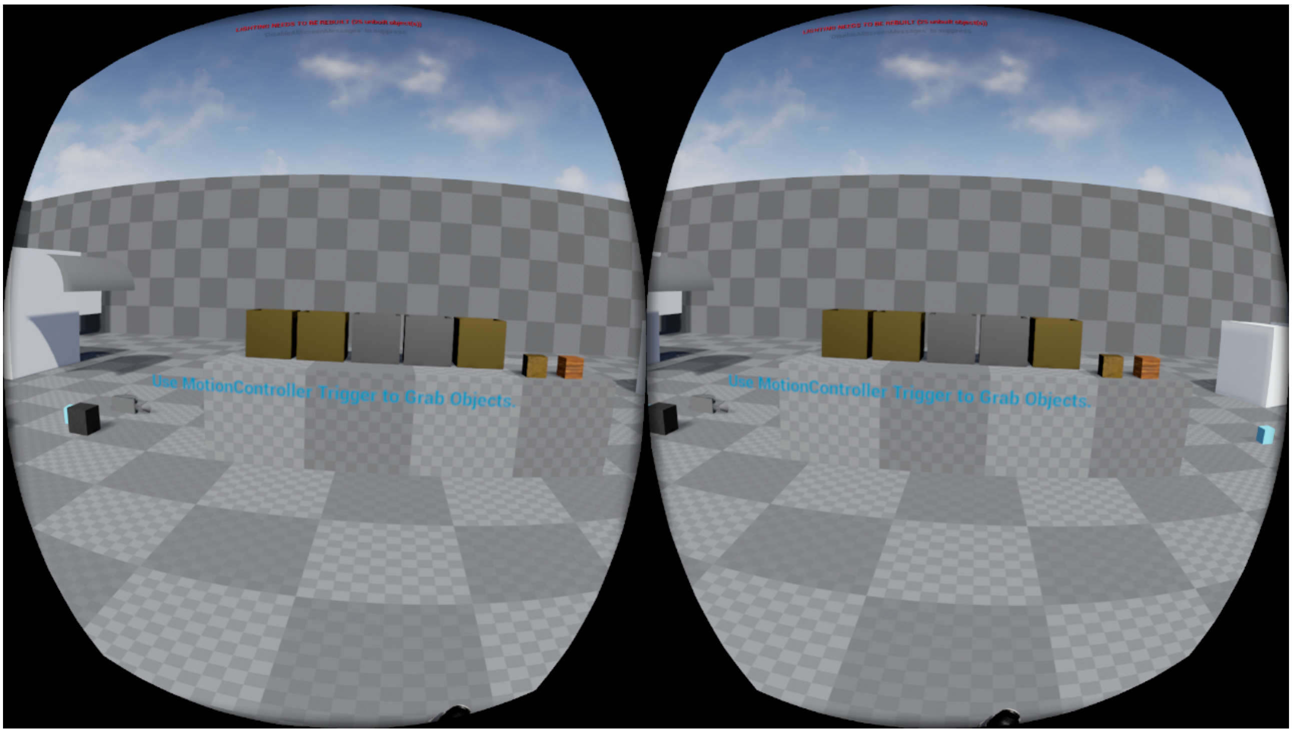


Figure 4: Comparison of two elements. The left element is the heaviest of the two.



Use MotionController Trigger to Grab Objects.

Use MotionController Trigger to Grab Objects.

Figure 5: Validation of the order of the array elements. Elements at indices 0, 1, and 4 are golden, indicting that they are in their correct places. Elements at indices 2 and 3 are silver, indicating that they are not in their correct places. Swapping elements 2 and 3 would yield a correctly sorted array.

1.3 Research Goal and Research Questions

The main goal of this thesis is to extend the work done in the specialization project, but with a more practical approach. The specialization project report provided a foundation for this thesis by examining the current state of VR in education and some advantages and disadvantages related to the use of VR in an educational context. It also included a prototype version of what was to become SortVR. This thesis will focus on presenting the finalized version of SortVR and how users respond to this application when testing it for the first time. The results of the user testing will then be used to answer the research questions posed below.

1.3.1 Research Question One

RQ1: To which degree can a virtual reality environment enable students to learn about sorting algorithms?

This thesis will examine the feasibility of using VR in an educational setting to teach sorting algorithms. The feasibility was determined by creating a prototype solution in the shape of a VR application called SortVR and presenting it to test users. The application will need to communicate the teaching material in an effective and efficient way, take advantage of the virtual reality platform, exploit gamification elements without overshadowing the pedagogy, and exhibit a satisfactory degree of usability. If the users testing the application agree that the application satisfies these criteria, the degree of feasibility of using VR in this educational setting can be considered high.

1.3.2 Research Question Two

RQ2: What artifacts and mechanics should a virtual reality environment include for teaching sorting algorithms effectively and efficiently?

If the application presented to the users yields promising results in terms of feasibility (RQ1), it is necessary to explore which aspects of the application contributes to this and which do not. It is also necessary to consider how these aspects work together. In the end, it can hopefully be determined what combination of features makes the application perform well and how it can perform better. User testing and subsequent surveying will also be important when answering RQ2.

2 Method and Equipment

This section will detail the hardware and software technologies employed in realizing SortVR. The choice of technologies will also be justified. In terms of hardware, head-mounted display (HMD) technologies used in the development of SortVR will be presented. There is obviously also need for a workstation (with peripherals like a monitor, keyboard, and mouse) for developing and testing the application (many HMDs require a connection to a computer, especially the most powerful ones), but this choice will not be detailed. Any configuration will do, as long as the technical specifications of the configuration satisfy the demands of the software and other hardware (the HMDs). In terms of software, the game engine (Unreal Engine) and the 3D modelling software (Blender) used in the process of making SortVR will be presented.

2.1 Head mounted displays

The final version of SortVR is designed to support two HMDs, Oculus Rift and HTC Vive. At the initial stages of the development, only the Oculus Rift (with Oculus Touch) was used. The only reason for this was a limited supply of HMDs. After more HMDs were acquired, development was moved over to HTC Vive. The switch was done both to make development easier (HTC Vive handles room-scale VR better) and to ensure compatibility with multiple HMDs. Moving to development with HTC Vive turned out to be unproblematic, since Unreal Engine has button and trigger events that are mapped to buttons and triggers for both Oculus Rift and HTC Vive, rather than having HMD-specific implementations.

Oculus Rift is more restrictive in terms of allowed movement than HTC Vive. This is clearly a disadvantage to both the user and the developer, but also an advantage in the challenge it presents for the developer to restrict movement. In other words, if the solution developed works with the restrictions of Oculus Rift, it will surely also work for HTC Vive. The difference in the handling of room-scale VR of the two HMD's ended up making little difference in the final version of SortVR, since the interface requires almost no movement.

2.1.1 Oculus Rift

VR headsets have existed since the early nineties (10), but it was not until 2012 Oculus VR successfully convinced the mainstream consumer (gamers in particular) of the potential of VR. Part of the success can be attributed to clever marketing through the use of the crowdfunding website Kickstarter (11), where backers funded an astounding 2.4 million USD, greatly surpassing the goal of 250,000 USD (12). Belief in the future of Oculus VR and their HMD was further strengthened as social media giant Facebook acquired the company in 2014 for two billion USD (13). Though development kits were made over the years, the first consumer version of the headset would not see release before 2016. Its introductory price was 399.99 USD. Later that year, motion controllers were developed for Oculus Rift. These motion controllers, called Oculus Touch, are now bundled with the headset (prior to the release of Oculus Touch, Oculus Rift came bundled with Microsoft's Xbox One Controller). The latest development of Oculus VR at the time of writing is Oculus Go (14), which is a standalone (i.e. it requires no computer, wires or sensors) version of the Oculus Rift headset.

Figure 6: Oculus Consumer Version 1.

The consumer version of the Oculus Rift was used in the making of SortVR. It features a PenTile OLED display with a 1080x1200 resolution per eye, a 90 Hz refresh rate and a 110° field of view. It also has integrated headphones which provide 3D audio effects. The headset employs a tracking system called "Constellation". It

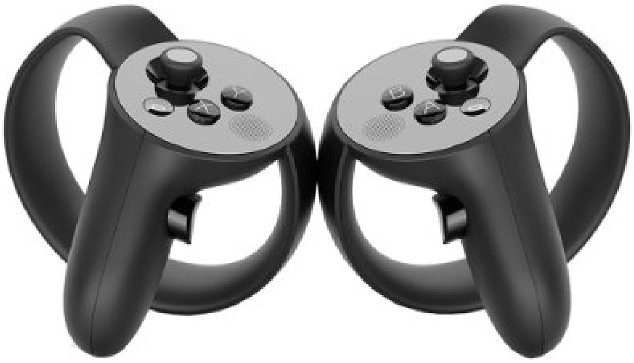


Figure 7: Oculus Touch, the motion controllers bundled with Oculus Rift. The controllers are designed to resemble traditional video game controllers and have an ergonomic fit to the user's hands. There is a left and right controller, which are mirrored versions of each other.

works by having IR LEDs on the headset emit light which is picked up by sensors (there are usually two, one on either side of the user's monitor) to determine the headset's position and rotation. The maximum tracking area recommended by Oculus VR is 8 x 8 feet, which is significantly smaller than that of HTC Vive. Another restriction is that at least one additional sensor (i.e. a total of three or more) needs to be purchased to provide 360° tracking (15). Finally, a restriction which was discovered during the development of SortVR is that the sensors are designed to be mounted on a desk, but a desk tends to block the IR signals when the HMD or motion controllers are too low to the ground. These restrictions make HTC Vive more suited for room-scale VR.

2.1.2 HTC Vive

HTC Vive is a VR Headset developed by HTC and Valve Corporation. It was unveiled at HTC's Mobile World Congress in 2015 and released in 2016 to compete with Oculus Rift. Its introductory price was 799 USD, which is twice the introductory price of Oculus Rift. HTC Vive comes bundled with two base stations (which provide tracking) and two motion controllers. HTC has recently (2018) released a new headset, HTC Vive Pro, in addition to the original headset with improved technical specifications.

The display of HTC Vive has the same technical specifications as that of Oculus Rift. It is a PenTile OLED display with a 1080x1200 resolution per eye, a 90 Hz refresh rate and a 110° field of view. The tracking system employed, called Lighthouse, works in about the opposite way of Oculus Rift's tracking system. Two base stations (ideally placed elevated and tilted down in two opposing corners of the dedicated play area) emit IR signals which are picked up by multiple sensors on the headset and controllers to determine their positions and rotations to sub-millimeter precision. This tracking system enables 360° tracking of a 15 x 15 feet play area. With these specifications, HTC Vive facilitates a much better room-scale VR experience than Oculus Rift.

2.2 Game engine

A game engine is software that abstracts the lower level operations required to create, deploy and run a video game. Core functionality provided by a game engine typically includes graphics rendering, physics calculations, sound, and networking, to name a few. Game engines also provide templates and huge libraries of built-in macros for handling game logic. Historically, games before game engines were made as singular entities for every individual game, optimized for the underlying hardware architecture. Proprietary game engines were eventually developed in-house at game developers with enough resources to do so and with a desire to have a re-usable platform for making their games. As the popularity and maturity of game engine software grew,

Figure 8: Sensor for Oculus Rift.

several companies (such as id Software and Epic Games) have released licensed versions of their own game engines. The increase in availability, the amount of official and unofficial documentation, and versatility of game engines on the market today makes them popular among indie developers with limited resources. Another huge strength of game engines is that they enable rapid prototyping which allows developers to easily test out new ideas. A simple game can be made in a matter of hours, even by inexperienced developers.

When it came to the choice of which game engine to use to realize SortVR, two viable candidates existed: Unity and Unreal Engine. Both are well suited for the purpose and after some research into the two as well as some informal talks with the supervisors of this thesis, it became apparent that the choice mostly comes down to personal preference. It was initially proposed that this project—along with several concurrent related master projects—were to be integrated with a common platform which was also supposed to include an existing VR master project. The existing VR master project is an application called IVR which is meant to facilitate group collaboration efforts with network communication. In the application, multiple users can connect to the same server and work together by interaction with a common environment, for example by writing on a blackboard or by showing each other images. Since IVR was made with Unreal Engine, it was decided that this project would also use Unreal Engine, to ease any possible future fusion of the two. The status and future of the aforementioned common platform is unclear at the time of writing.

2.2.1 Blueprints Visual Scripting

In order to create the game logic for a game, Unreal Engine supports two programming languages: C++ and Blueprints. C++ is a fairly old (1983) programming language used in the creation of many games before (and after) the era of game engines, and is thus a suitable alternative to experienced programmers who want, and are used to having, low level control of the game implementation. Unreal Engine itself is also written in C++. Blueprints is a new visual programming language released by Epic Games along with Unreal Engine 4 in 2014. The simple and intuitive nature of the visual language is designed to bridge the divide between technical artists, designers, and programmers. As Alan Willard from Epic Games puts it (16):

> I could say: 'I'm going to convert this pillar into a blueprint [in the Engine] and add some sort of trap to it.' It means I can really go in and start enhancing my world with interaction that just would not have been possible without a technical artist, a designer and a programmer and now any one of those three can do all of it, provided they have the assets handy. The fact that I can just go in and say, 'If you're within X distance of this thing, start to glow and take my distance to it, normalize it zero to one and then just lerp [oscillate] between two different brightness values, so as I reach for something it gets hot'...that would have been something do-able but very difficult for anybody except a gameplay programmer. And he wouldn't have known how to set up the assets, but now any one of the three could do it.

Blueprints utilizes a system of nodes which are interconnected with links between pins on the nodes. Control

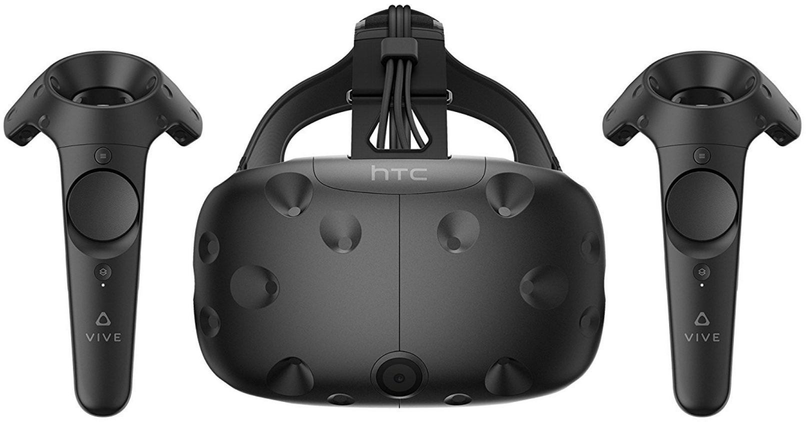


Figure 9: HTC Vive with one motion controller on either side of the HMD. Note that the motion controllers are identical. Unlike the Oculus Touch controllers, there is no specification of which controller is the left controller and which is the right.

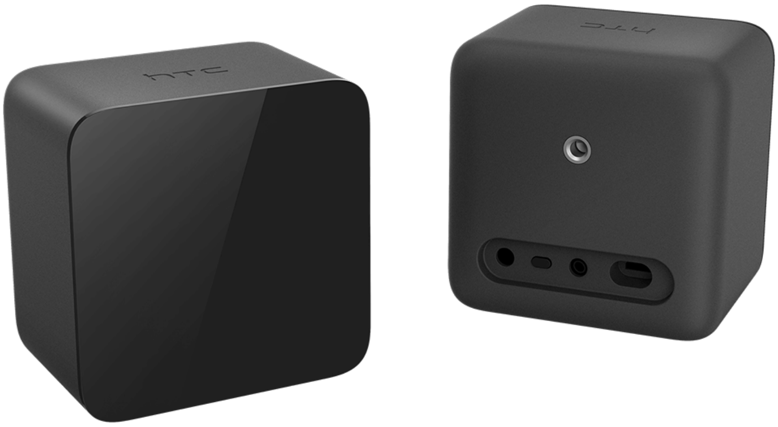


Figure 10: Base stations as seen from the front (left) and the rear (right).

flow in a program is quite literally handled as a flow of execution (which can even be seen visually when the program is running). Execution starts at an event node in the event graph, which corresponds to some event in the game. Each in-game actor (game object) with associated game logic has an event graph. The level itself also has its own blueprint and event graph, for any game logic which cannot reasonably be associated with an actor. Examples of events are *Begin Play*, which triggers after construction of the actor; *Begin Overlap*, which triggers when the collision box of an actor overlaps the collision box of another actor; and *Tick*, which triggers once every frame. In addition to these and many other pre-defined events, the developer also has access to create custom events. Once an event has been triggered, execution flows through the execution output pin, through a connection, and into an execution input pin of another node. The nodes handle some computation and are more or less analogous to a line of code in a traditional programming language. Nodes could for example be *Branch* nodes, which are similar to if statements, or user defined functions. A node executes whatever action it is responsible for, and sends an execution signal through one of its execution output pins. The execution continues to flow in this fashion through a chain of nodes until a node has no valid pin to send execution to. Nodes also have input pins and output pins for handling variables such as integers, floats, and strings.

One of the limitations of Blueprints, which became apparent during the development of SortVR, is the way delays in execution work (pauses where a program or part of a program remains idle, sometimes referred to as sleeping). Delays will only work in an execution path in the event graph, and not in functions or in combination with certain built-in macros. This had a major impact on the implementation of sorting animations. A lot of delays had to employed for the animations, since they had to run over time and in a specific order which

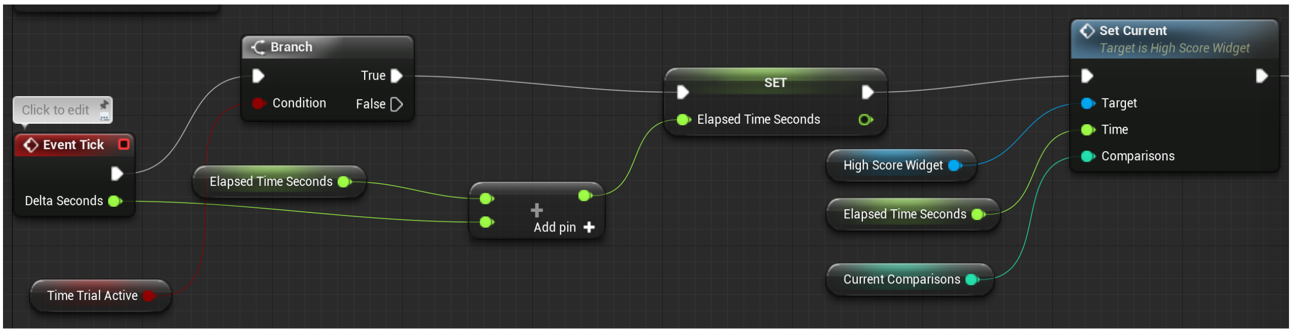


Figure 11: Part of the blueprint for handling time trial logic. White pins and connections show execution paths. Variables are color coded according to data type.

had to be determined at runtime. Functions could not be used, as the engine does not allow *Delay* nodes in functions (functions in Unreal Engine are meant to return ”instantaneously”, meaning that they do actual work and return as soon as possible). Logic for animations was therefore placed in the event graph. Loops, which are essential to the execution of sorting algorithms, do allow delay nodes in their internal logic. The problem here, however, is that that loops are built-in macros which, like functions, are designed to execute as fast as possible. The result is that the loop node sends executions signals to its internal nodes rapidly and without concern for any *Delay* nodes between iterations. In the case of SortVR, this would mean that many elements would start swapping animations at the same time and immediately cause the program to enter an illegal state. Manual loops had to be created to overcome this problem. The sequence of nodes were as follows: *Branch* node with loop condition → Chain of nodes handling internal loop logic → *Delay* node.

Blueprints was the only language used through the entire project. Blueprints was mostly chosen due to the developer’s aversion to C++ and the fact that Blueprints was a new, exciting, and intuitive approach to game development. The language was also surprisingly powerful and versatile. In the case that Blueprints would not be able to handle whatever was required, development would have continued in C++, but this was luckily never an issue. Epic Games has done a thorough job with the language, as Blueprints seems to support most of the features supported by C++. It has also already become well-documented, both officially (17) and unofficially. Finding which nodes to use when trying to implement a new feature was therefore simple once basic knowledge of Blueprints, the game engine, and game terminology had been acquired.

2.3 Modeling software

Much like 2D games need sprites to create on-screen graphics, 3D games require 3D models in order for the player to have something to look at and interact with. There are a few ways of going about adding these 3D models to your game. The simplest is using in-engine templates. With this method, shapes can be added in-engine by a simple drag-and-drop operation to place the shapes in the level you are currently working on. The downside to this method is that one is restricted to using primitive shapes such as cubes, spheres, and cones. The designer has limited ability to manipulate these by changing the transform of the shape (i.e. its location, rotation, and scale). These shapes work for simple interaction and experimentation, and they are also well-suited for creating background and surfaces such as walls and floors. For more complex shapes, like real-world objects and humans, other options must be explored. The simplest option is to use pre-made assets. Unreal Engine has an associated asset store where these assets can be obtained. Some of the assets are free, while others can cost up to hundreds of dollars. The Unreal Engine asset store has a somewhat more limited selection compared to that of Unity, but the selection is still quite rich. It was however not possible to find the necessary models for this project. Already during the specialization project, the limitations of the asset store became apparent as trouble was had finding an appropriate balance scale. When the asset store fails to meet the needs of the game, one more option remains—making the models yourself. In this case, it is necessary to use separate 3D modeling software. Blender was chosen for this purpose, due to it being free of charge and well-documented (18) (both officially and unofficially). Other options include Maya and Autodesk 3ds Max.

2.3.1 Assets

Many assets were modelled during this project and the specialization project, but only two made it into the final version of SortVR. Figure 12 shows the array container, meant to hold array elements and represent a position in the array. Figure 13 shows the greater-than/less-than sign used as a comparator to tell the user which is greater of the two elements they are currently holding. Many other assets were also modelled, including several prototypes of a balance scale, but all of these were eventually scrapped.

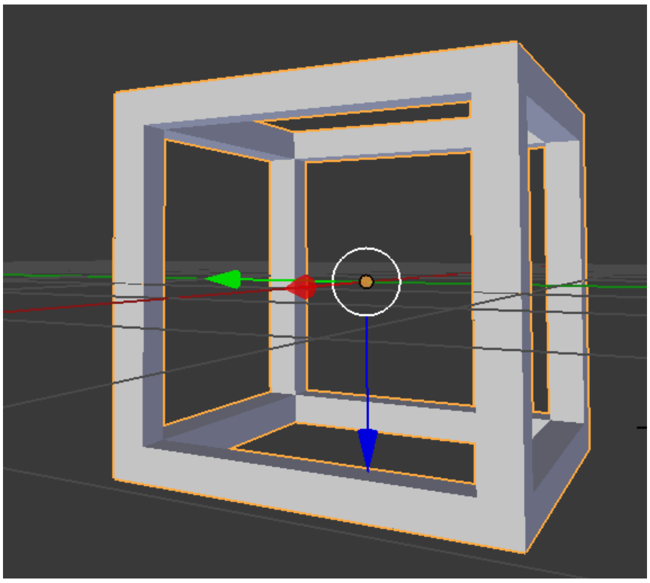


Figure 12: The 3D model for the array container.

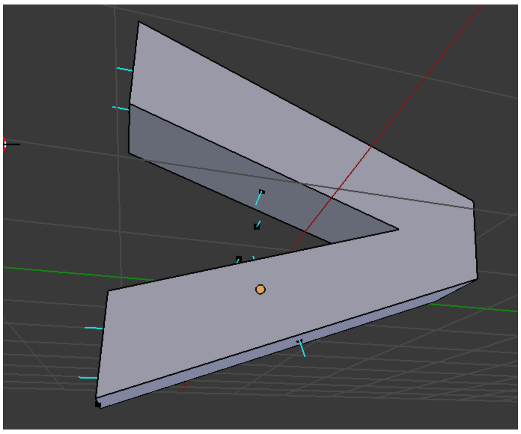


Figure 13: Greater-than sign, which doubles as a less-than sign.

2.3.2 Collisions

An important aspect in video game design is the handling of collisions, which is the logic executed when two objects are occupying, or trying to occupy, the same spatial position. Collision detection relies on so-called collision boxes (also known as hit boxes or colliders), which are invisible shapes around game actors. These are constantly using physics and geometry calculations to determine when they will intersect with other collision boxes and usually triggers the execution of some logic when collision is detected. When using the combination of Blender and Unreal Engine such as was done for this project, there are two options to creating collision boxes. One option is to use Unreal Engine directly, while the other relies on exporting the collision boxes with the assets from Blender. Using Unreal Engine allows you to add collision boxes with primitive shapes (cubes, spheres and capsules) and some automatically generated complex shapes. Unreal Engine does however not handle concave meshes well (models with a ”hollow” interior). For some concave collisions, it might be necessary to add multiple convex collision boxes. Using Blender to create collision boxes allows for simpler creation of concave (or convex) collision boxes with a tight fit to the mesh. To make these collision boxes is a simple process of making a copy of the model you want a collision for, renaming the copy, and exporting it along with the model. While this last option provides a simple way to create an accurate collision box, it was only used for the array container asset. Since collisions with complex shapes are more computationally expensive than collisions with primitive shapes, complex shapes should only be used when they're absolutely necessary.

3 Results

The main results of this thesis are (1) an application for teaching sorting basics in VR, SortVR, and (2) the results of the user testing of the final implementation of the application. The experiences made during implementation and initial testing by the developer combined with the feedback given by the test subjects will be used to answer the previously posed research questions. The implementation itself will be presented through a series of screenshots and descriptions which will describe every aspect of the game in turn, including all features and possible interactions between user and system. The results of the user testing will be presented, comprising a user survey and informal feedback collected.

3.1 SortVR

3.1.1 Controls

The controls in the game are kept as simple as possible, with only three buttons mapped to functionality. Many of the controls were also inherited from the Unreal Engine VR template. These inherited controls enable exploration of VR environments, they are conventional and are likely familiar to an experienced VR user.

The movement is the typical teleportation technique which is common for VR games. To move, a user must point to the desired destination and press down on the analog thumb stick (Oculus Touch) or touchpad (HTC Vive). The user can also choose desired rotation by holding down in that direction on the analog stick or touchpad before releasing when moving. The application also supports room-scale VR, meaning that the user can walk freely within the boundaries of the available physical space. Since this space is limited, however, teleportation might sometimes be necessary to fully explore an environment. Rotation can of course also be accomplished by simply turning, but again there are physical limitations, particularly with the Oculus Rift which loses tracking capabilities as the user turns their back to the front sensors.

Another inherited control is the control for grabbing. Any actor (game object) that allows it (by means of implementing an interface) can be picked up and moved around by holding the main triggers on either Oculus Touch or HTC Vive. Releasing the trigger will release the object. The controls for picking up and releasing actors is intuitive and offers great affordance and usability, since it follows convention and allows the user to use an action that is similar to what the user would do to pick up an object in real-life by gripping and releasing it.

Finally, in addition to these inherited controls, one more button was mapped to functionality. By pressing the grip button on HTC Vive or the A button on Oculus Touch while pointing on a position in the array, the user can change the color of that position. The color will toggle between white and green. This function is meant to enable the user to keep track of which elements in the array the user already has sorted.

3.1.2 Starting state

Figure 14 shows the starting state of the game, i.e. what the user sees as soon as the application starts. The image and all following screenshots were taken in an in-engine simulation of the game. Using an HMD and running the game in VR Preview mode allows the user to see two hand models which are not visible in simulation mode. The hand models each have attached a magenta colored laser pointer, used for selecting menu items, selecting elements (the blue boxes) from the array, and marking an array position as sorted/unsorted.
Three artifacts in the level are visible to the user from this position. The array, the main menu, and a high score list. These will now be presented in turn (the high score list will be presented along with its associated menu element, Time Trial).

3.1.3 The Array

The array is represented by eight semi-transparent containers which are meant to represent individual positions in the array. The containers are cubes with most of their sides removed in order to increase visibility of the contained elements and the surrounding environment. The transparent material was also chosen for this purpose.

Attached to each of the eight corners of each container (four for the leftmost container) is an actor called a connector. This is a remnant from when it was intended to let the user create arrays by spawning and connecting containers dynamically. To connect a container, the user would have to hold two containers close together, making sure that at least one pair of connectors are overlapping. The free container would then snap to the array and become part of it. The reverse action was also possible, meaning that a user could split the array by picking up one or more containers in an array. This feature is still fully functional and included in the final version. In fact, the containers are not initially connected at the first frames of the game but is rather connected dynamically using the feature. The feature was however not mentioned or emphasized during user

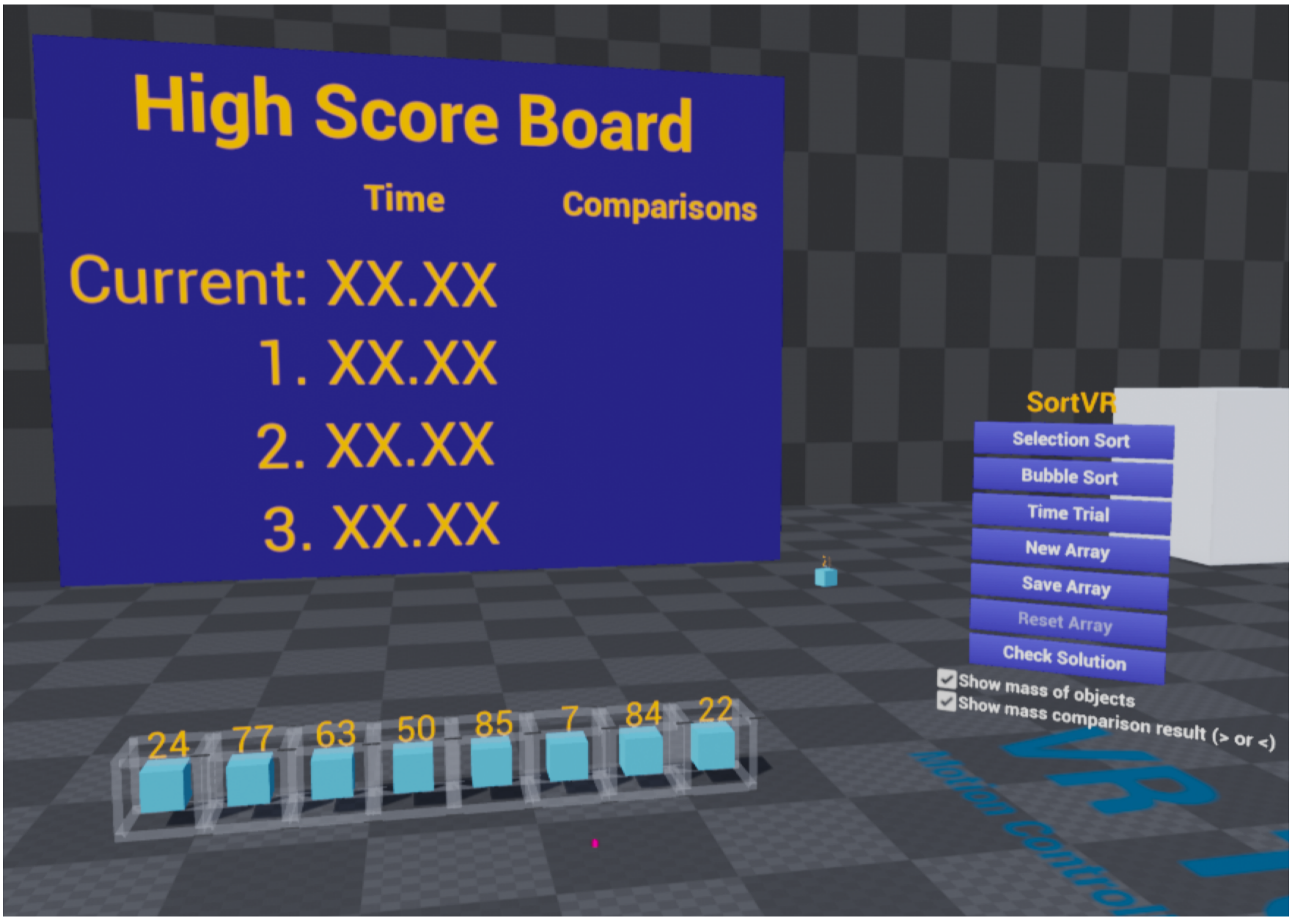


Figure 14: The starting state of SortVR.

testing, since it was deemed to be challenging to test and possibly distracting from the actual purpose of the application.

The elements themselves are simply blue boxes which were already included in the VR template. Their masses are set randomly at the start of the game to some integer (represented as a float in blueprints due to masses being represented by floats in the game engine) between 1 and 100. The mass of each element is displayed as yellow text on a widget attached to the element. This mass display can be toggled on and off in the menu.

The array in figure 14 can be represented as follows using Python syntax for lists: [24 77 63 50 85 7 84 22].

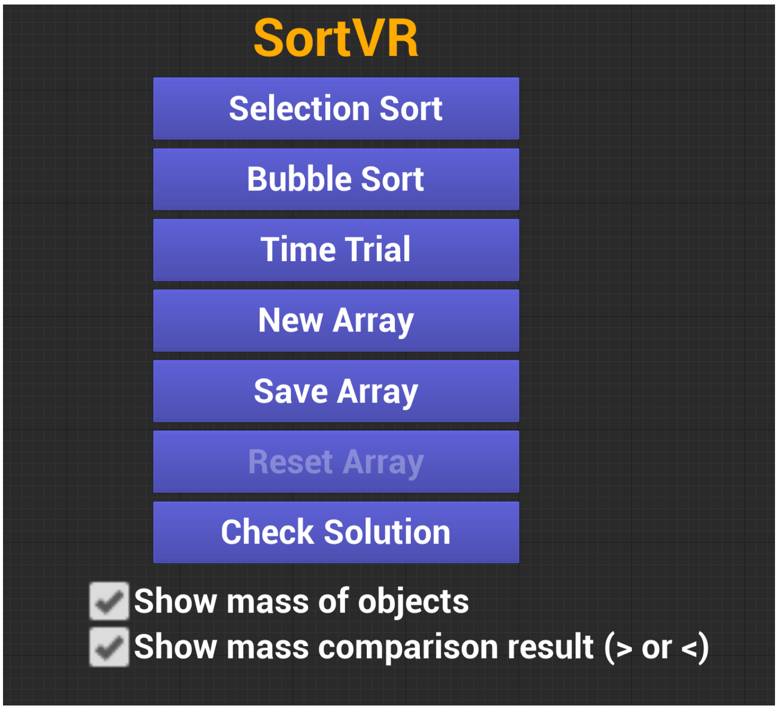
3.1.4 Main menu

Figure 15 shows the main menu, which comprise several buttons with associated functions triggered when the buttons are pushed. Some of the buttons lead to new menus. The user can select and trigger any of the buttons by hovering over it with the laser pointer and using the trigger (on either Oculus Touch or HTC Vive). To increase usability and make it easier for the user to see which element is selected, the button currently selected by the user is highlighted green. The functionality associated with each button in the menu will now be explained in turn.

Selection Sort
Triggering this button replaces the main menu with a submenu for selection sort. This submenu will be detailed in a section below.

Bubble Sort
Triggering this button replaces the main menu with a submenu for bubble sort. This submenu will be detailed in a section below.

New Array
Like the text on the button implies, this re-initializes the array with new masses. The masses are again random integers (stored as floats) between 1 and 100.



☑ Show mass of objects
☑ Show mass comparison result (> or <)

Figure 15: The main menu.

Save Array
Triggering this button will store the current state of the array to memory. State information stored includes the masses of the elements along with their respective positions in the array (more concretely, the masses are stored in a single array and the positions are implicitly given by the position of their masses in this array). If the button has been previously triggered, the currently stored array state is overwritten by the new.

Reset Array
As can be seen in figure 15, the text on this button has a lower opacity than the text on the other buttons in the menu. A classic convention of GUI design is used here to indicate that the button has no functionality in its current state. The button will remain in this state until the user has triggered the "Save Array" button above. When the "Save Array" button is triggered, the text on the *Reset Array* button gains the same opacity as the text on the other buttons, indicating that it is now functional. Triggering the button when it is in this state will restore the array to the state it was in when the "Save Array" button was last triggered. The saved representation of the array as described in the "Save Array" button description above gets retrieved and is used to restore the masses of the elements to the stored values.

Check solution
Triggering this button will allow the user to see whether the elements of the array is currently in sorted order.

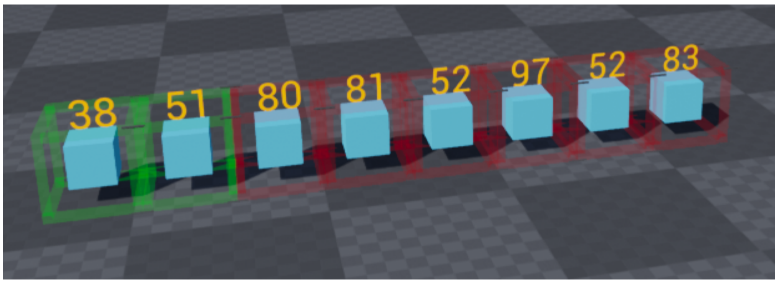


Figure 16: Pressing the *Check Solution* button will cause the array containers to change color according to whether the elements they contain are placed in the correct, sorted order. In the image above, only elements 0 and 1 are correctly sorted.

In the blueprint, this is done by duplicating the array, sorting the duplicated array, and then comparing the masses in the duplicate array with their counterparts in the original array. If the mass of an element at position

i in the original array is equal to the mass at position i in the duplicate array, the container at position i is colored green. Otherwise, the container of the element is colored red. This means that a correctly sorted array will have only green containers, while incorrectly sorted arrays will have two or more red containers. After a short delay, the array returns to its original state, meaning that the containers are again colored white. An example of what the array looks like when this button is triggered can be seen in figure 16.

Show mass of objects

When this checkbox is triggered, the checkbox goes from a checked state to an unchecked state and the masses displayed above each element become hidden. Triggering the checkbox again moves the application back to its original state, i.e. the checkbox goes from a unchecked state to a checked state and masses become unhidden.

Show result of comparison (> or <)

When this checkbox is triggered, the checkbox goes from a checked state to an unchecked state and the comparator symbol which is normally present when the user holds up two elements becomes hidden. Triggering the checkbox again moves the application back to its original state, i.e. the checkbox goes from a unchecked state to a checked state and the comparator symbol becomes unhidden.

3.1.5 Selection Sort

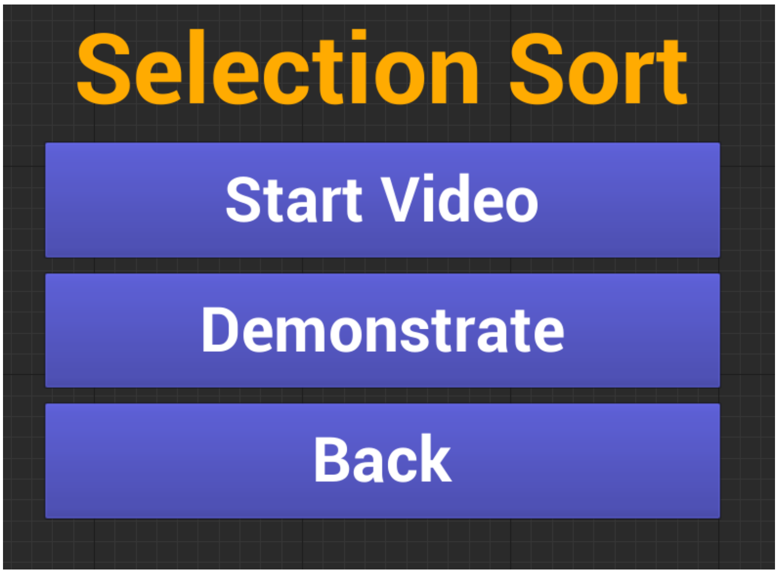
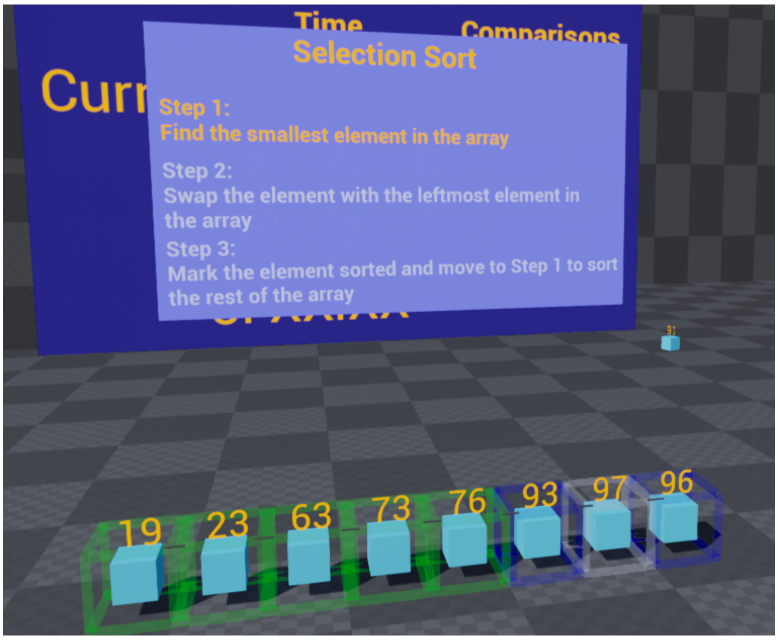


Figure 17: The submenu for selection sort.

If this button is triggered, the submenu shown in figure 17 will replace the main menu. Each of these buttons have functions as explained below.

Start Video/Stop Video
Pressing this button once starts a video explanation of selection sort and changes the text displayed on the button to "Stop Video". The video is an embedded YouTube video (19) which will be displayed on a Web Browser Widget floating in the environment. Pressing the button again will stop the video and return the video player and the button to their original state (changing the button text back to "Start Video" in the process). Video players usually come with more functionality, such as the ability to scroll through the video, change or mute the volume of the video, skipping, and adjusting playback speed. These were not implemented as initial testing revealed that the video feature was unsatisfactory and it was therefore not considered worthwhile to add more functionality to the player. Specifically, the problem was that the low resolution of the video when using an HMD made it hard to see details such as explanatory text. The video was kept as a proof of concept for the users to evaluate. Its intended functionality of explaining the sorting algorithms was left to the real-time demonstration offered by the *Demonstrate* option.


Selection Sort

Step 1:
Find the smallest element in the array

Step 2:
Swap the element with the leftmost element in the array

Step 3:
Mark the element sorted and move to Step 1 to sort the rest of the array

Figure 18: The selection sort demonstration in progress.

Demonstrate

Pressing this button starts a real-time demonstration of selection sort. Figure 18 shows this demonstration in progress. Three colors represent the different states of the array containers. The white containers contain unsorted elements, the green containers contain sorted elements, and the blue containers contain elements which are currently being compared. The steps taken by the algorithm, which can be seen on the widget in the background, are the same as those given by the pseudocode in the introduction section. As each step is executed by the algorithm, the corresponding text is highlighted. The steps taken are repeated below, each step with a description of how it is realized visually in the demonstration.

Step 1: Find the smallest element in the array:
The two first (leftmost) elements of the unsorted part of the array are compared, coloring the containers they belong to blue. After a small delay, the container of the greatest element is colored white again while the container of the smallest element remains blue. The smallest element is then compared with all remaining elements of the array in a similar fashion, toggling the colors of the compared elements' containers blue and back to white along the way. If an element smaller than the currently smallest element is found, the newly found element takes the place of the formerly smallest element (coloring the container of the formerly smallest element back to white) and proceeds to compare the new smallest element with the rest of the array. When the end of the array is reached, all elements have been compared once and the container of the smallest element is still colored blue.

Step 2: Swap the element with the leftmost element in the array:
The container of the smallest element switches color from blue to white. The smallest element is then swapped with the leftmost element in the unsorted array by means of linear interpolation (often referred to as lerping) between four points for each of the two elements, using ticks between frames as delta value for the interpolation. The delta value increments a value alpha on the blueprint lerp node. The alpha value increments steadily from 0 to 1 and is then used to find the next world location to place the elements each tick. In practice, this creates an animation of the elements moving at a constant velocity from one point to the other. The elements first ascend from the array (to avoid collisions, they are moved to different heights), then move lengthwise along the array (in effect swapping (x, y) coordinates with each other), then descend back into the array.

Step 3: Mark the element sorted and move to Step 1 to sort the rest of the array
The element that was just moved to the immediate right position of the sorted part of the array becomes part of the sorted part of the array itself. The container of the element switches color from white to green. As the algorithm dictates, the demonstration moves back to step 1 to sort any remaining elements. If all the elements are sorted, the demonstration terminates after a short delay. The end of the demonstration is marked by all containers switching colors back to white and the un-highlighting of the steps on the text widget.

This demonstration was originally meant to be used in conjunction with the video explanation as a supplement to take advantage of the virtual environment. The video was supposed to explain the algorithm to the user, while the demonstration was supposed to show the algorithm in effect on any array of the user's choosing. After the video player was deemed inadequate for its intended purpose, extra functionality in the form of the associated dynamic text widget was added.

Back
Triggering the "Back" button will take the user back to the main menu.

3.1.6 Bubble Sort

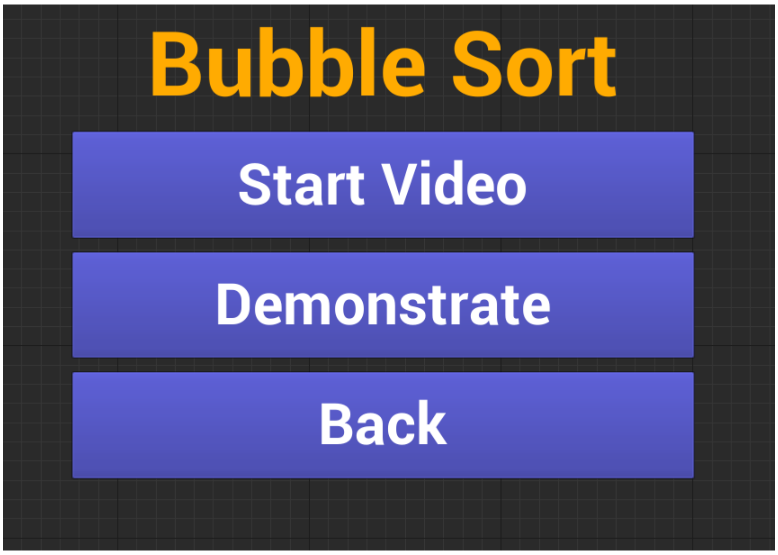


Figure 19: The submenu for bubble sort. The interface of the menu is identical to that of selection sort.

Triggering this button will replace the main menu with a submenu specific to bubble sort. The text on the buttons in the submenu for bubble sort, as seen in figure 19, is the same as the text on the buttons in the selection sort submenu. Much of the functionality, as described below, is also the same.

Start Video/Stop Video

The functionality of this button is the same as the button associated with selection sort. The video (20) is naturally different, but comes from the same creator on YouTube.

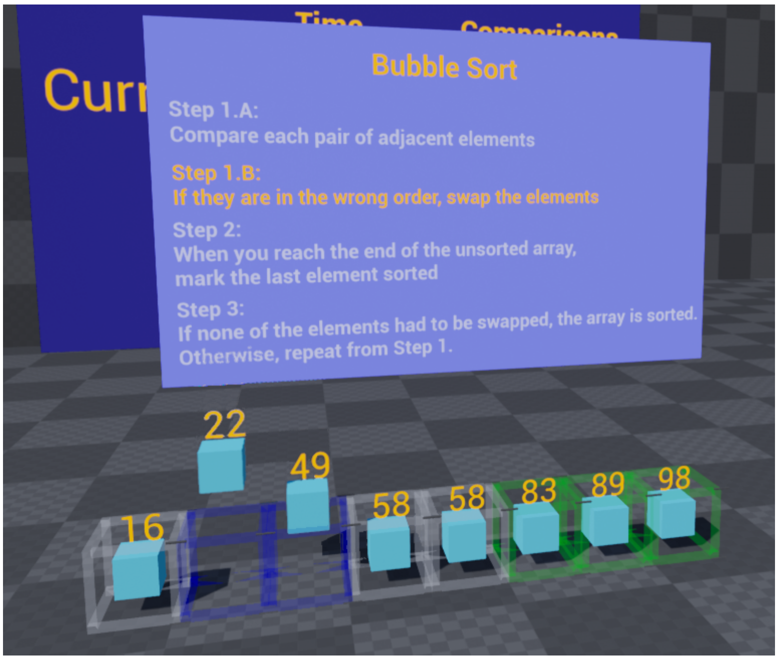
Demonstrate
Pressing this button starts a real-time demonstration of bubble sort. Figure 20 shows this demonstration in progress. As with the selection sort demonstration, three colors are used for the containers. The significance of the colors remains the same. This demonstration also has an associated text widget which keeps track of which step the demonstration is currently taking by means of highlighting the text. The steps taken are repeated below, each step with a description of how it is realized visually in the demonstration.

Step 1.A: Compare each pair of adjacent elements.
The visualization of the comparison is done in the same manner as for the selection sort demonstration, with compared elements having their containers turn blue.

Step 1.B: If they are in the wrong order, swap the elements.
The elements are again moved using linear interpolation (lerping). The blueprint event for swapping elements created for the demonstration of selection sort is reused here.

Step 2: When you reach the end of the unsorted array, mark the last element sorted
As with the selection sort demonstration, one element is marked sorted for each iteration of the loop by coloring its container green.

Step 3: If none of the elements had to be swapped, the array is sorted. Otherwise, repeat from Step 1. Unlike the demonstration for selection sort, this demonstration has the ability to terminate early before every array container has been colored green. There is no integer counter or boolean keeping track of whether any swaps have been made, but the user can easily see when swaps are made and a loop iteration with no swaps



Bubble Sort

Step 1.A:
Compare each pair of adjacent elements

Step 1.B:
If they are in the wrong order, swap the elements

Step 2:
When you reach the end of the unsorted array,
mark the last element sorted

Step 3:
If none of the elements had to be swapped, the array is sorted.
Otherwise, repeat from Step 1.

Figure 20: The bubble sort demonstration in progress.

can also be recognized by how Step 1.B is never executed. If there were no swaps made, all containers are colored green to signify the sorted status of the array. After a short delay, the color of the containers are reset to white and the all steps on the text widget are un-highlighted. If there were swaps made, the demonstration continues from Step 1.

Back
Triggering the "Back" button will take the user back to the main menu.

3.1.7 Time Trial

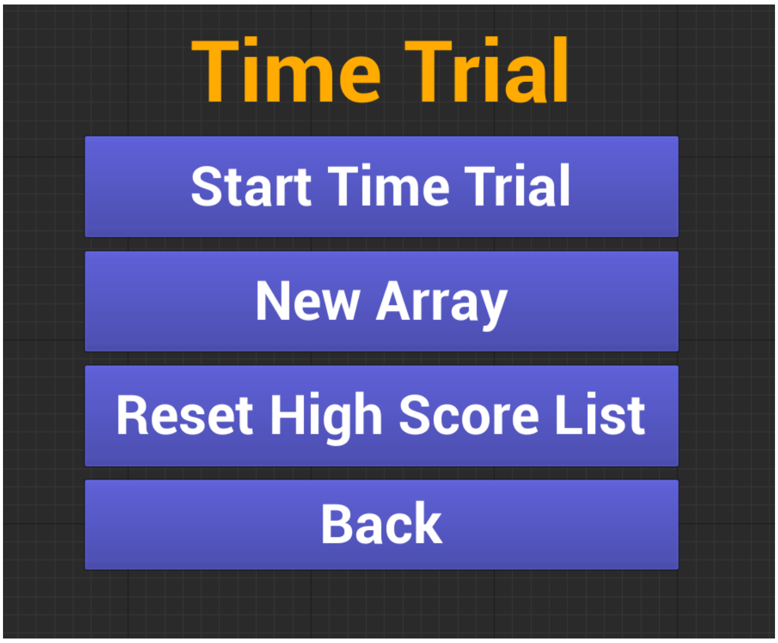


Figure 21: The submenu for the Time Trial mechanic.

Triggering this button brings up the time trial submenu shown in figure 21. The time trial is a gamification element which allows users to compete against themselves and other users to sort an array as fast as possible and with as few comparisons as possible. Upon activating a time trial and sorting an array, the time taken to sort the array is compared with the three best sorting times for that array. If the sorting time is better than any of the three best sorting times, the time will be placed on the high score list along with the number of comparisons used to sort the array.

Start Time Trial/Reset
When this button is triggered, the time trial commences and the text on the button changes from ”Start Time Trial” to ”Reset”. A timer on the high score board keeps track of the time elapsed during the current sorting attempt. This timer keeps running until one of two conditions are met: 1) The user sorts the array to completion, or 2) The user triggers the button (which now reads ”Reset”) again. In the first case, the elapsed time in seconds is recorded and compared with the three best times currently displayed on the high score board. If the time is lower than any of the times on the high score board, it takes up a position on the high score board and moves the scores in the list down accordingly. In the second case, both the button and the array are reset to their initial state. This means that the button once again reads ”Start Time Trial” and all elements in the array are put back in their original position. Note that in the first case, the button still reads ”Reset”. In order to start a new time trial, the user must therefore trigger the button once more to reset the time trial to its initial state. This is to allow fair comparison of time trial scores—it doesn’t make much sense to compare the times taken to sort two completely different arrays.

New Array
Like the text on the button implies, this re-initializes the array with new masses. The masses are again random integers (stored as floats) between 1 and 100.

Reset High Score List
Triggering this button will reset the state of the high score board, meaning that all the stored scores are deleted and their textual representation on the board are replaced with placeholders. This state, which is also the initial state of the high score board, can be seen on figure 14.

Back
Triggering the ”Back” button will take the user back to the main menu.

3.2 User testing

User tests of the application was done in a quite informal manner, with few instructions or restrictions given before or during the test. The tests were conducted this way to facilitate free exploration of the environment at the user’s own pace. This way of testing also gave a good measure of the usability and independence of the application. The users were however informed that they were to feel free to ask questions and give feedback during the test. This enabled a good dialogue between the tester and the user, and also gave insight into the thought process of the user. The thought process of the user could then be compared to that of the tester (who also developed the system) in order to find any discrepancies, especially when it came to aspects of the system initially thought to be intuitive.

Prior to the tests, some essential information was presented. The users were informed of the control system, by means of the tester showing the motion controller and pointing to the two buttons (the trigger and the grip button) used in the application. Interaction with the elements was demonstrated by the tester showing how elements can be retrieved from the array, compared to other elements, and placed back into the array. The tester also showed how the array containers can be marked sorted (coloring them green) and how the buttons in the menu are triggered. Finally, users were informed that the videos in the application are quite long (about four minutes) and that they were not expected to watch them through. After this introduction to the application, the user was given the HMD, motion controls, and headphones. Finally, they were instructed to do whatever they felt like. Most users then went through all the features of the application without any need for further instructions. If some features were left out, the user was told to test those specific features before concluding the test.

Tests were conducted with ten users who were all unfamiliar with the application and the tester prior to the test. Tests lasted for 5-10 minutes, which was sufficient to evaluate all aspects of the application. Users were for the most part alone in the room along with the tester. Spectators were allowed in the room, but these did not interfere with the testing (especially since the HMD and headphones partly isolated the user from the physical environment).

During the tests, feedback and questions from the users were noted. After the test, the users were asked to fill out a survey detailing their experience. The survey and responses can be found in the appendix. The rest of this section will by dedicated to presenting some of the major results of the feedback received from the users, both in terms of the survey and any informal comments made and questions asked during the test.

3.2.1 Test Group

The beginning of the survey collected metadata about the ten test users. Many of the test users had some affiliation with the campus hackerspace, which implies an interest for, understanding of, and experience with novel technology such as VR. This was also reflected in the survey results. Out the ten test users, six had a lot of experience with VR equipment, two had tried it a few times, and two had never tried it prior to the test. The users were all university students, representing all grades one through five. All of the test users replied that they were studying computer science or information technology, suggesting a high level of domain knowledge (the teaching material was already known to some degree).

3.3 Likert statements

One section of the survey had the respondents rate their agreement with eleven statements related to their experience with SortVR and the future of VR in an educational context. To rate the statements, respondents were asked to use a standard Likert scale which is common practice for surveys. The Likert scale had five points to choose from: Strongly Disagree, Disagree, Neutral, Agree, and Strongly Agree. In addition, participants were allowed to answer a sixth option which read "Not applicable/Don't know/Don't wish to answer" to be used as a catch-all for any responses not mappable to the scale. This last option was not used by any of the respondents.

By assigning a numerical Likert score to each of the five options on a scale, an average Likert score can be calculated. The mapping used in this thesis will be: "Strongly Disagree" = 0, "Disagree" = 1, "Neutral" = 2, "Agree" = 3, "Strongly Agree" = 4.

The statements will be presented below with their average Likert scores, which will be used to gauge the general opinion of the respondents and to reason about the responses given. Some statements are closely related and the statements will therefore be divided into four categories: *Familiarity with the technology*, *Usability of the application*, *Potential use of VR in education*, and *Independence of the application*. Each of the statements will be presented with one section presenting the rationale for including the statement, and another section discussing the result and its implications. In general, the rationale of all the below statements is to answer RQ1. As mentioned in the research goal section, the users' opinion of the application is one of the most important factors in gauging the educational value of VR, which is what RQ1 concerns.

3.3.1 Familiarity with the technology

As mentioned in the section about the test group, many of the users were familiar with VR technology. The two first statements of the survey also confirms that the respondents are in general interested in learning about and using new technology.

S0: I am interested in (new) technology and its application.

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	Average Likert Score
0	0	0	3	7	3.7

Rationale

Technophiles are more likely to be positive to any novel use of technology and their responses might therefore be somewhat biased. This statement seeked to discover how much effect this bias has.

Discussion

The average Likert score of 3.7 is extremely high, which is not unexpected given what is already known about the test users. The survey results are likely to be biased in favor of using VR technology in future education. The results from evaluating the application might also be somewhat skewed since the users are familiar with the technology used and therefore finds the application easier and more rewarding to use. On the other hand, users with such a high technological literacy might also hold higher standards and it might take more to impress them. At the least, there will be much less of the awestriking effect that usually accompanies trying VR for the first time.

S1: I consider myself ”computer savvy” (good with computers).

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	Average Likert Score
0	0	0	3	7	3.7

Rationale

The technical literacy of the test group is sure to influence how well the test users can handle being placed in an unfamiliar environment where technical tools must be utilized to solve a problem. Detecting how good the test group is at technical problem solving is important due to how it affects the usability evaluation of the application.

Discussion

This statement received the same average Likert score as the preceding statement (S0), which is perhaps not surprising. Technophiles who are interested in technology are also likely to have experimented with it as a result of an eagerness to learn more. Technical literacy is also common among—and perhaps even implicitly demanded from—anyone pursuing a career in computer science or information technology. Students affiliated with the university hackerspace are an even more niche group with a particularly strong interest.

S2: I play video games on a regular basis.

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	Average Likert Score
0	0	1	3	6	3.5

Rationale

SortVR is made in a game engine and shares many characteristics with a video game. The application presents a dynamic virtual environment in which the user (or player) must interact with game objects and menus through the use of controller peripherals. This similarity makes it essential to understand what experience the test users have with video games. Experience with video games among the test users will affect the survey results much in the same way as the previous two statements (S0 and S1). A user with a high level of video game experience will find it easier to accept and understand the conventions used in the application and will find the whole experience of interacting with the virtual environment more intuitive.

Discussion

The average Likert score is again high for this statement. A correlation between the score of this statement and those of the two previous statements (S0 and S1) is quite likely. Gamers are more likely to take an interest in the technology related to the platform they are playing games on and the technology used to create the games. Conversely, technophiles are likely to take an interest in what the new technology they are interested in can do—such as making and playing video games.

3.3.2 Usability of the application

S3: The program was easy to use.

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	Average Likert Score
0	0	0	5	5	3.5

Rationale

Whether a software system is easy to use is the big question when it comes to measuring usability. Usability is a ubiquitously important quality attribute for any system meant to support users with potentially diverse backgrounds. This system is meant to support a wide range of users, even users with no knowledge of computer science concepts and users with limited VR and video game experience. It is therefore important to facilitate usability through simple and intuitive user interaction offering high affordance which accomodates the user's expectations.

Discussion

An average Likert score of 3.5 and no responses at level 2 or below indicates that the system exhibits high usability. This is also backed up by the tester's experience that few directions were needed in order for the user to understand how to use the application. As previously mentioned, the respondent's responses to statements S0, S1 and S2 could affect their response to this statement. Technically competent and/or gaming users are more likely to find a new system easy to use. More test respondents with different backgrounds are necessary to discover the true usability of the system.

S4: The program was exciting to use.

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	Average Likert Score
0	0	5	3	2	2.7

Rationale

Another aspect of usability to consider is whether the user feels good when interacting with the system. Having a system that is easy to use is not enough on its own; A user must also find the experience rewarding in some way. If the experience is sufficiently rewarding, the user is more likely to return to using the application in the future. It is also useful to gauge how rewarding the application is to be able to compare it to traditional teaching methods which do not employ gamification elements.

Discussion

The application scores above average (average being 2, the neutral midpoint on the scale) on this statement, with half the users given a neutral score. None of the users found the experience displeasing. This is certainly satisfactory, but there is potential for improvement. More and better game elements could be included to raise the score, although care must be taken that the gamification does not entirely overshadow the pedagogy. This statement is also affected by responses on the preceding statements. A user who interacts with the system with ease experiences a better flow and the user will consequently feel that the experience is rewarding. Conversely, a user who struggles to interact with the application will experience obstructions to the flow and will consequently feel frustrated. On the other hand, users who are unfamiliar with VR and video games might find the novelty of the experience rewarding in itself.

3.3.3 Potential use of VR in education

S5: I was familiar with the teaching material (sorting algorithms) prior to the test.

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	Average Likert Score
0	0	1	3	6	3.5

Rationale

This statement is meant to discover the knowledge gap of the test user and the potential learning the application can provide. A low score indicates a large knowledge gap that the application can potentially fill. A high score indicates that the gap is small and that the potential amount of learning the application can provide is low.

Discussion

The high score given to this statement is again unsurprising based on the nature of the test group. All computer science and informatics students at NTNU take an obligatory algorithms and datastructures class (TDT4120 Algoritmer og Datastrukturer) during the fall semester of their second year. Sorting algorithms are a part of the curriculum for the course. The curriculum examines sorting algorithms in much greater detail than what SortVR does (more algorithms are taught, and students are taught to analyze aspects such as time complexity and space complexity). The two first year students are also evidently familiar with the teaching material. This could mean they were taught the algorithms during an earlier course such as the first year class TDT4110 ITGK (introductory course to information technology). It could also be due to the students being interested in the subject and that they have acquired the knowledge independently.

The average Likert score of 3.5 indicates that the amount of learning the application can provide is small. It is therefore harder to gauge how the application can perform in an educational context with users who are unfamiliar with the teaching material.

S6: The program taught me something.

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	Average Likert Score
1	4	2	3	0	1.7

Rationale

Clearly one of the most important aspects to discuss concerning the application is whether it actually is successful in educating students effectively and efficiently. The application must communicate the teaching material clearly and without presenting too many distractions that can overshadow the pedagogy. A high Likert score on this statement is necessary to state that the application can be utilized as a supplement to—or even replacement of—current teaching methods. The score must however be seen in context with the rest of survey. The previous statement (S5), which measures the test groups familiarity with the taught material, is especially important. A group that is already familiar with the taught material cannot be expected to have learned much.

Discussion

Most users found a low gain of knowledge resulting from the use of the application, which lead to this statement scoring below average. Such a low score might seem alarming for an application whose main purpose is education, but seeing the score in context helps to understand why it is so low. Since the test group exhibits such a high familiarity with the teaching material prior to the test, it is only natural that they didn't learn much from the application. Half the users do however answer neutral or agree, implying that the application has at least some level of educational value. The statement does however not specify what the user has learned, so there is a possibility that the users have learned something other than what was intended. In retrospect, it might have been better to phrase the statement to explicitly mention sorting algorithms. Based on statements S5 and S6, it is hard to draw any firm conclusions on whether the application has the intended educational value. More tests would need to be conducted—preferably with users with a lower domain knowledge—in order to gain a better understanding of the educational value of the application.

S7: VR is a good approach to teaching sorting algorithms.

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	Average Likert Score
0	0	2	4	4	3.2

Rationale

Many of the previous statements indirectly attempt to find the feasibility of using SortVR to teach sorting algorithms by measuring the usability of the system and its potential for learning. SortVR is however only a case study; This thesis aims to find not only the feasibility of using SortVR, but the feasibility of using VR to teach sorting algorithms in general (which is what RQ1 concerns). The most direct approach to find this feasibility is by asking the test participants of their opinion on the matter. Opinions are naturally subjective and the responses to this statement can therefore not be used completely independently, but at least the opinions reflect the students' willingness to use such a system in the future.

Discussion

A fairly high average Likert score of 3.2 with no responses below the neutral level indicates that the users believe using VR for the purpose of teaching sorting algorithms is indeed a feasible solution. The responses given further indicates that there could be future students who would be inclined to use a VR solutions such as SortVR instead of traditional teaching methods, given that the current interest in VR does not decline. It could again be argued that the test group, which consists of many VR enthusiasts, could have given this statement an unnaturally high Likert score, but that does not mean that the result should be discredited entirely. The mere existence of VR enthusiasts provides a clear indication that VR has potential in the future.

S8: VR is a good approach to teaching other computer science concepts.

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	Average Likert Score
0	0	0	4	6	3.6

Rationale

The specialization project that eventually lead to this master thesis concerned itself with not only sorting algorithms, but the entire field of computer science. What became SortVR was just intended as a prototype for a small part of a larger whole, namely a virtual sandbox-like environment where users could explore and educate themselves in a large variety of computer science concepts. These concepts would be taught in the same vein as sorting algorithms are taught with SortVR, focusing heavily on visualization through animations and interactive components. This statement attempts to find the users' opinion on utilizing VR in other areas of computer science, allowing us to find the extendability of SortVR into its larger, encompassing knowledge domain.

Discussion

This statement receives an even higher average Likert score than the preceding statement. It is clear that the test group believes that VR is suitable for teaching sorting algorithms, but there might be other areas of computer science that might provide an even better fit for VR. There are certainly many concepts that could be more easily understood through visualization, such as network flow and shortest path problems. These concepts are also perhaps better suited for gamification elements. Network flow could for example be taught by puzzles involving flow of liquid whose interactive components could be the combination of valves, joints, and pipes.

S9: I believe in the future of VR in an educational context.

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	Average Likert Score
0	0	0	4	6	3.6

Rationale

This statements extends even further than the two preceding statements, and concerns not only sorting algorithms or computer science, but education in general. With the responses to this statement, it is possible to understand the feasibility of using VR to supplement or replace the traditional teaching methods currently being employed. Furthermore, the responses provide a context to better understand the responses collected for the two preceding statements (S7 and S8). For example, if respondents believe in the future of VR in education but exhibits low faith in the use of VR education in the domains of sorting algorithms and computer science, it raises the question of whether further research into the aforementioned domains is worthwhile.

Discussion

Responses again yield a high average Likert score for this statement, a clear indication of the potential of VR to supplement or replace current teaching methods. How this can be done and at what scale VR should be employed is a matter of future research, but for now it is at least possible to conclude that there exists a strong willingness in the test group to use VR in education and that this willingness is likely to exist also in future students.

3.3.4 Independence of the application

S10: Educational VR programs such as SortVR can be used independently without instruction or supervision.

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	Average Likert Score
0	1	0	5	4	3.1

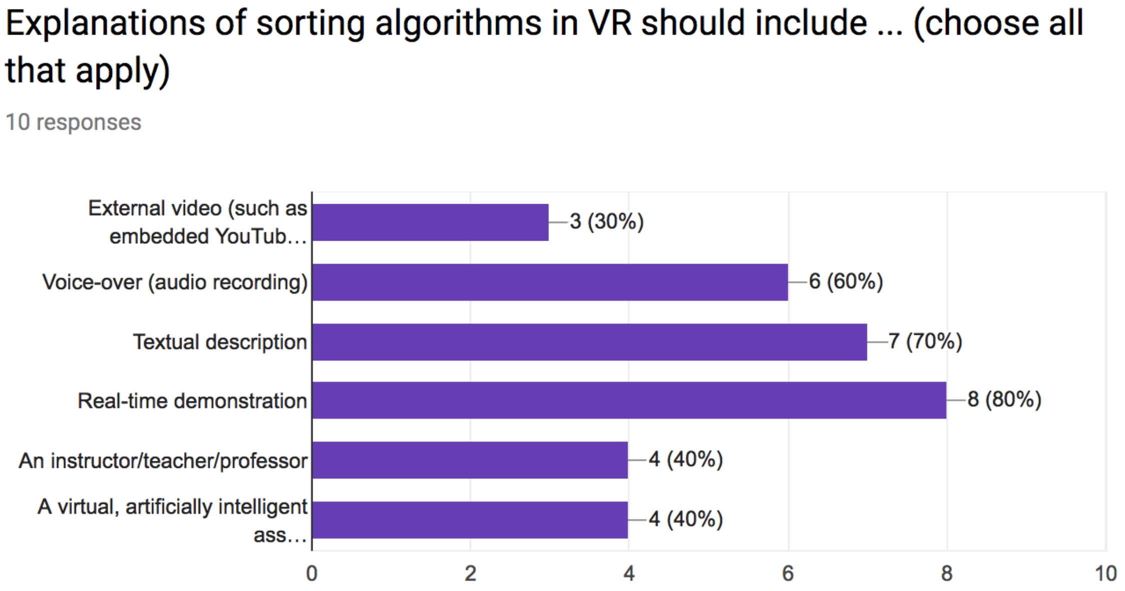
Rationale

Throughout the specialization project and the work on SortVR, discussions have been had on the inclusion of multi-user capability to the system. Two potential uses could arise from such a capability. Firstly, it could allow cooperation between students in either interactive tasks or simply the exchange of domain knowledge to guide each other in the learning process. Secondly, it could allow a user in the role of a teacher. The teacher could then do what teachers normally do in physical classroom environments, namely assisting the students in completing the tasks at hand and providing further explanation of the teaching material. The discussion ended with a decision to attempt to keep the application focused on providing an isolated experience independent on the need for external instruction or cooperation. Since adding the described multi-user capability is a possible feature to be added, this particular statement also concerns RQ2. Instructors and assistants will also be discussed further in later sections.

Discussion

All but one user gave this statement a high Likert score. The decision to focus effort on creating a self-contained experience might then seem to have been correct. There might be many students who will require instruction in the use of the VR equipment itself, but once these students have been introduced to the equipment, the responses to this statement show that it is not always necessary to have instructors present during VR education.

3.4 Wanted features



Explanations of sorting algorithms in VR should include ... (choose all that apply)

10 responses

Figure 22: Survey results for wanted features.

The Likert statement part of the survey which was presented in the previous section focused on discovering the answer to RQ1. The final part of the survey, whose results will be presented hereunder, concerns itself with RQ2. The users where asked to specify which features they believed to be critical for an interactive virtual environment designed to teach sorting algorithms. The survey provided a list of six different features (some present in the current solution and some missing) and asked the respondents to select all the features they believed should be included in the environment. The purpose of this part of the survey is twofold. First, it discovers what features in SortVR work and which could be improved or removed entirely. Second, it provides a foundation for research in future works, as the list of features can be used as a guide to what users wish implemented in similar educational VR solutions in the future. It is not guaranteed that an inclusion of all the most popular features will lead to a good solution, but it is certainly a good place to start. Results of the wanted features part of the survey can be seen in figure 22. These results will be discussed throughout the rest of this section.

3.4.1 External video versus application specific instruction

Results indicate that there is not a high demand for external video embedded into the application. This is probably due to the low quality of the video implementation. Given a higher resolution on the videos and a more advanced video player with more functionality, there might have been a higher demand. However, given that other features in the list that could easily replace the functionality of the video player scored so high, it might be more reasonable to push research in that direction rather than focusing on developing a better video player. The voice-over, textual description, and real-time demonstration features were all favored by 60 per cent or more of the respondents. There are two conclusions possible to draw from this (they may both hold true):

(1) Some combination of these features could lead to a good future solution for teaching sorting with VR. The essential part then is figuring out what the right combination of these features is. According to multimedia principles of learning (21), humans are able to process information through two separate channels at the same time—one auditory and one visual. Further, the principles declare that the capacity of these channels are limited. Seeing then as textual descriptions and real-time demonstration both utilize the visual channel, it is important to balance the information load these two are attempting to output. It is also a good idea to minimize the amount of text on the screen in a VR application. There are two reasons for this. First, the resolution in an HMD is much more limited than that of a screen designed to handle a lot of text (such as a computer monitor or tablet). This can be compensated by enlarging text, but this obviously makes the text take up more space and might then require a lot of straining and cumbersome neck twisting for the user. The other reason why limiting the amount of text is a good idea is that including a great deal of text defeats the purpose of using VR rather than using a traditional textbook.

(2) Since respondents have just completed user testing of SortVR prior to taking the survey, the wanted features survey results are an indication that the textual description and real-time demonstration included in SortVR worked well. This conclusion is also backed by informal feedback gathered from the test users. Several users reported that they really liked the visual sorting demonstration, but one user wanted more control over the demonstration (ability to pause, stop, and adjust the speed of the demonstration, for instance).
The demonstration and its associated text, which are both processed by the visual channel (21), are presented at the same time, but none of the users mentioned a sensory overload. Seemingly, the application takes heed of the principle of limited channel capacity, but the absence of sensory overload may well be caused by a strong familiarity with the domain knowledge presented.

3.4.2 Demand for instructors or assistants

Less than half of the respondents wants an instructor or assistant present in the virtual environment. This again confirms what was discussed in the above section concerning the application's potential for independence. Most test users believe that VR applications such as SortVR can be independent standalone experiences. There is however a certain demand for instructors and assistants in the test group which shouldn't be ignored entirely. Moreover, this demand might be even higher in other groups of students, specifically in groups of students with low knowledge of either VR equipment handling or sorting algorithms. An introductory course in the handling of VR equipment can be used to fix the first knowledge gap. The second knowledge gap should ideally be handled by the application itself, without any need for extra instruction, but whether this is the case needs further research with groups of low domain knowledge. It can be argued that adding instructors or assistant will likely not affect the pedagogical value of the application negatively. Therefore, adding an instructor or assistant can't hurt as long as the benefit of the instructors or assistants.

If instructors or assistants or deemed necessary, a decision must also be made regarding to employ a real human instructor or a virtual, artificially intelligent assistant. Employing a human instructor is cheaper upfront, since all that is necessary is to implement multi-user capability through networking functionality offered by Unreal Engine (this multi-user capability can then also be used to allow students to cooperate). As the author of this thesis has a background in artificial intelligence, there have been discussions to integrate artificial intelligence in the form of a virtual assistant. Virtual assistants such as Apple's Siri (22) and Amazon's Alexa (23) are becoming increasingly more commonplace. These are however results of years of research and labor from huge teams, and are still far from exhibiting a satisfying level of general artificial intelligence. Developing an assistant with a more specific purpose—such as answering questions about sorting algorithms—is much cheaper, but the cost is still significant. The upside is that virtual assistants have much lower costs once employed than university professors. The idea of using a virtual assistant for this project was eventually abandoned due to the heavy resource demand and the belief that the development of the virtual assistant would take away too much focus from the development of the VR application itself.

3.4.3 Demand for more feedback during execution of the algorithms

One of the most frequent comments from the users was to include more feedback during the user's execution of either of the algorithms. With the current state of the application, users are free to sort the array in any way they wish without receiving any feedback from the application. Many users wanted to be able to select an algorithm to execute and then execute the algorithm while the application keeps track of progress and gives feedback whenever the user did something wrong. Thought was given into making this feature in the later stages of the project, but it was hard to find a reasonable way to implement the feature with the time remaining. Part of the reason why initial solution did not include the feature was the desire to create a constructivist environment where the user could learn through exploring and experimenting with the environment, as was described in the specialization project report (8). It may have been a better idea to adopt a behaviorist approach to the teaching of sorting algorithms, based on the repetitive nature of following the steps to a sorting algorithm and the feedback given by the users. Since there are many choices left to the implementer of an algorithm, there are also several ways of completing either of the sorting algorithms in SortVR which are all technically correct. This introduced complexity and ways to handle it are left to the discussion section.

4 Discussion

4.1 Use of comparator, hide/unhide vs separate mode?

Sorting algorithms, when implemented in some programming language and executed, typically rely on comparing only two numbers at a time. The option of hiding the masses of the elements and enabling the comparator symbol is supposed to emulate this situation. This functionality was perhaps not made clear to the test users of the application. Some even said they didn't notice the comparator until the end of the test. Instead of having the two check box options in the bottom of the menu ("Show mass of objects" and "Show result of comparison (> or <)"), it may have been a more suitable approach to implement a separate mode. Using video game terminology, this could be a "hard mode" for users wanting a more challenging experience. Swapping the options out with a separate mode also makes sense because only two of the four combinations (corresponding to the two modes) are meaningful. The different combinations of the options and their effect on the system can be seen in the table below:

	Comparator shown	Comparator hidden
Mass shown	Comparator symbol unnecessary, since the user can easily compare any two numbers directly by looking at the displayed masses	Easy mode
Mass hidden	Hard mode	Meaningless. Only way to sort is to guess an order and verify.

4.2 Complexity of tracking individual variations of sorting algorithms

As mentioned in the user testing section, many users were interested in executing the sorting algorithms with more feedback. Specifically, they wanted feedback when they did a step in the algorithm wrong. There are however many ways of executing algorithms which are technically correct. Algorithm descriptions are often quite vague and the specific implementation details are design decisions left to the developer. The possible variations of selection sort and bubble sort are mentioned in the introduction. If a tracking and feedback system for the individual steps is to be introduced, there are two ways of handling the problem of many possible execution paths. The first is to restrict the user to one specific path for each sorting algorithm. The advantage here is that the solution is cheap. The disadvantage is that this restriction could lead to frustration among students who have understood the principles behind a sorting algorithm, but not the specific path of actions the application requires. The second way of handling the complexity problem is allowing for every single path of actions that lead to a sorted array while still making sure that the principles of the algorithm are followed. This method allows more freedom for the student, but at a higher implementation cost. What further complicates matters is the option of using a comparator while hiding the actual numbers associated with each element. Consider the following example to see why: A user wants to find the smallest element in an array. In the "Easy Mode" mentioned above, the user can find the smallest element by simply scanning the displayed numbers visually (at least with small arrays). In "Hard Mode", however, the user needs to pick up each element at least once to find the smallest one. These two modes therefore represent two completely different execution paths that the feedback system would need to consider.

4.3 Interacting with objects

The way in which the user interact with the elements in the array in has changed over time. Initially, the pick-up and drop interface from the VR template was simply adopted. To move an element, the user had to move their hand to the element, press and hold the trigger, move to the desired position, and drop the element by releasing the trigger. This was simple and intuitive for the most part, but was quite cumbersome in the event of a large array, especially if the array was large enough to require teleportation. Movement in VR has always been a headache for developers, and being able to avoid it entirely is therefore desirable. Therefore, the current solution was introduced, which requires no movement at all. Ray-casting is used to determine what element or array container the user is pointing to, which is used to allow the user to remotely pick up and drop items. Ray-casting was shown (during initial tests and user tests) to be both effective, precise and intuitive.

4.4 Conducting tests on users experienced with VR

There are advantages and disadvantages to having a test group with such a high level of experience with VR. The main advantage is that discussion of aspects of both using and developing VR applications becomes quite fruitful. Users experienced with VR can explain both positive and negative experiences that they've had with VR in the past, as well as any challenges they have met. These users are also able to compare the application they are testing to other VR applications and alert the tester of any unconventional choices in the user interaction design. The main disadvantage of having a group of experienced VR users is that it becomes harder to measure the feasibility of using VR solutions such as this at for education a large scale when most students are inexperienced with VR. There were luckily two test users who were trying VR for the first time, but it would be preferable to test the application with more inexperienced VR users.

4.5 Conducting tests on users with domain knowledge

In addition to the users having experience with VR, they also exhibited a high understanding of the taught material (sorting algorithms). Again, this presents advantages and disadvantages. With their level of understanding, they can evaluate the presented material and validate the correctness of the material in addition to how well it is presented. A disadvantage, as mentioned in the survey results, is that it becomes harder to gauge whether the application actually is able to teach anything to students with no knowledge of sorting algorithms. Ideally, the application would need to be tested with more users with no pre-existing knowledge of sorting algorithms to fully understand the learning potential.

5 Conclusion

This thesis sought to find the feasibility of employing VR in an educational setting, specifically to teach sorting algorithms in an intuitive and interactive way. It is clear, from the experience gained through the development of SortVR and the subsequent user testing, that VR does indeed have great potential as an educational tool. Whether SortVR has the potential to teach sorting algorithms to students without domain experience remains somewhat inconclusive and more testing with such students is required to fully understand SortVR's teaching potential. The degree of usability the application exhibits combined with the users' clear belief that VR can be used to teach sorting algorithms does however indicate that SortVR is onto something. Users also believe VR has potential to educate students in other computer science concepts, which provides a basis for future work.

The thesis also sought to find which features a VR application designed to teach sorting algorithms needs to have in order to convey the teaching material in a way that is effective and efficient. It is evident from the users' response to the application that the most appreciated features are those that fully take advantage of the virtual environment. The demonstration feature of SortVR was shown to be popular, while video integration has some potential for improvement before it can be employed. In conclusion, more interactivity and direct feedback through behaviorist tasks seem to be what the users demand.

It is the hope of the author that the results found in this thesis will provide a basis for future research into VR, so we may bring ourselves ever closer to our very own OASIS.

References

[1] Ernest Cline. Ready player one, 2011.

[2] New york times. New york times bestseller list. https://www.nytimes.com/books/best-sellers/2016/12/25/trade-fiction-paperback/.

[3] Start VR. Start vr revolutionising real estate with virtual reality. https://startvr.co/revolutionising-virtual-reality-real-estate/.

[4] The Verge. The norwegian army is using the oculus rift to drive tanks. https://www.theverge.com/2014/5/5/5682942/the-norwegian-army-is-using-the-oculus-rift-to-drive-tanks.

[5] William Winn. A conceptual basis for educational applications of virtual reality. http://www.hitl.washington.edu/research/education/winn/winn-paper.html~, 1993.

[6] Beijing BlueFocus. A case study - the impact of VR on academic performance, 2017.

[7] Brian Greene. AbelanaVR lecture at the World Science Festival. https://www.youtube.com/watch?v=Fes8bdN5ddQ, 2017.

[8] Marius Bang. Creating an interactive virtual reality environment for teaching computer science, 2017.

[9] University of Canterbury. Computer science unplugged. http://csunplugged.org/.

[10] Nathan Cochrane. Vfx-1 virtual reality helmet by Forte. http://www.ibiblio.org/GameBytes/issue21/flooks/vfx1.html, 1994.

[11] Kickstarter crowdfunding website. https://www.kickstarter.com/, .

[12] Oculus Rift on Kickstarter. https://www.kickstarter.com/projects/1523379957/oculus-rift-step-into-the-game, .

[13] Kyle Orland. Facebook purchases vr headset maker oculus for 2 billion. https://arstechnica.com/gaming/2014/03/facebook-purchases-vr-headset-maker-oculus-for-2-billion/.

[14] Oculus Go. https://www.oculus.com/go/, .

[15] Oculus roomscale—tips for setting up a killer VR room. https://www.oculus.com/blog/oculus-roomscale-tips-for-setting-up-a-killer-vr-room/, .

[16] Stephen Totilo. How unreal engine 4 will change the next games you play. https://kotaku.com/5916859/how-unreal-engine-4-will-change-the-next-games-you-play.

[17] Blueprints visual scripting – documentation. https://docs.unrealengine.com/en-us/Engine/Blueprints.

[18] Blender reference manual. https://docs.blender.org/manual/en/dev/index.html.

[19] Selection sort video. https://www.youtube.com/watch?v=3hH8kTHFw2A.

[20] Bubble sort video. https://www.youtube.com/watch?v=RT-hUXUWQ2I.

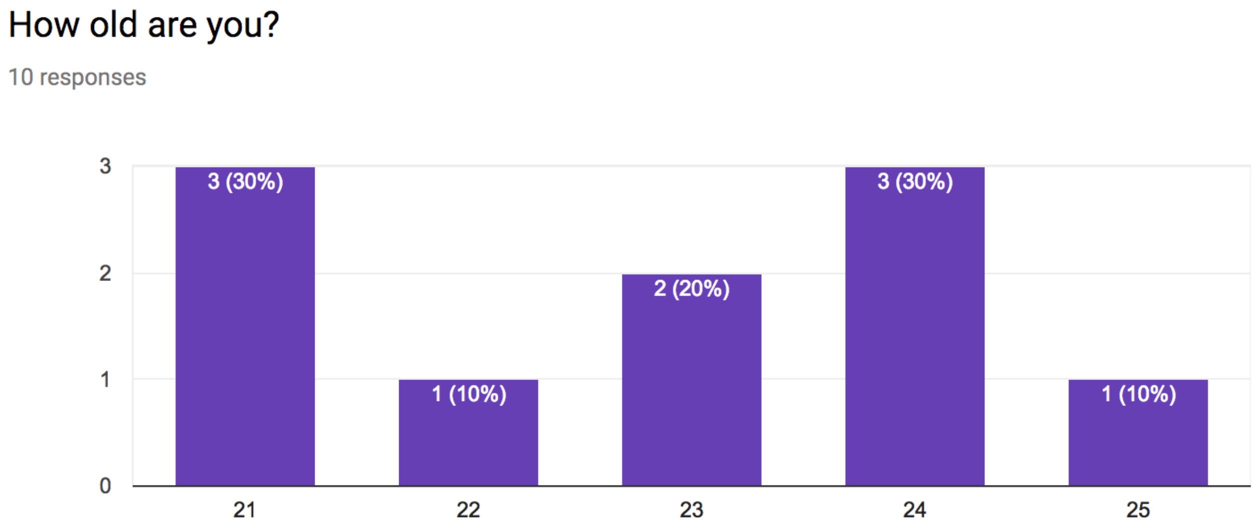
[21] R. E. Mayer. Multimedia learning. psychology of learning and motivation, 2002.

[22] Virtual assistant Siri. https://www.apple.com/ios/siri/.

[23] Virtual assistant Alexa. https://developer.amazon.com/alexa?cid=a.

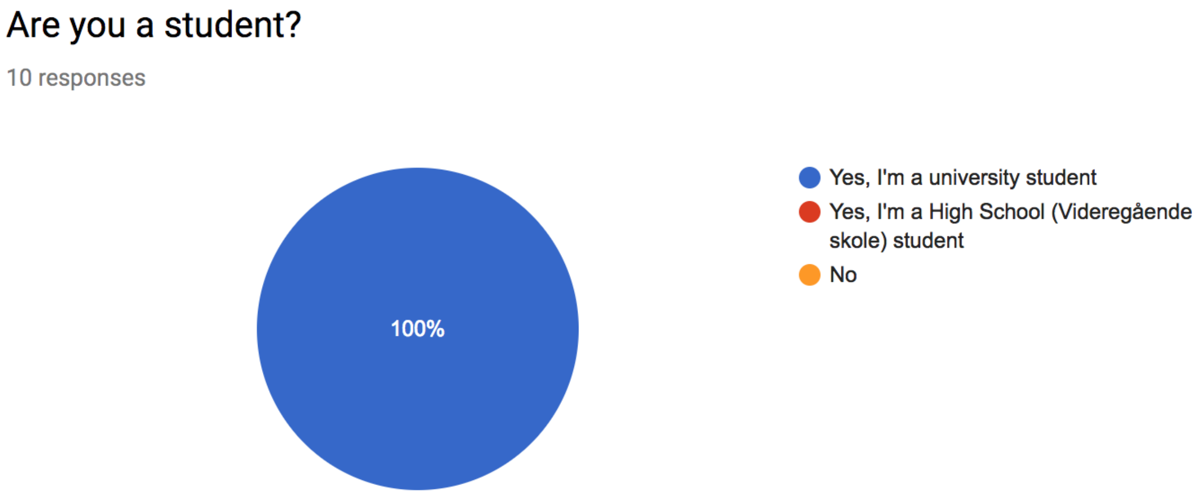
6 Appendix

A User Survey Response Summary



How old are you?

10 responses

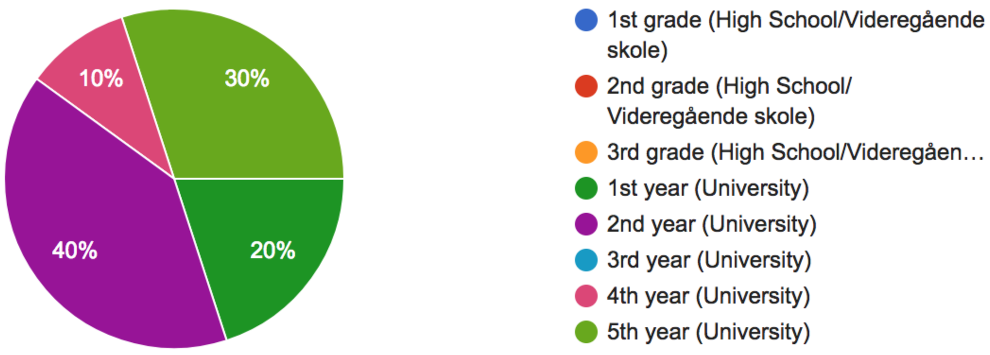


Are you a student?

10 responses

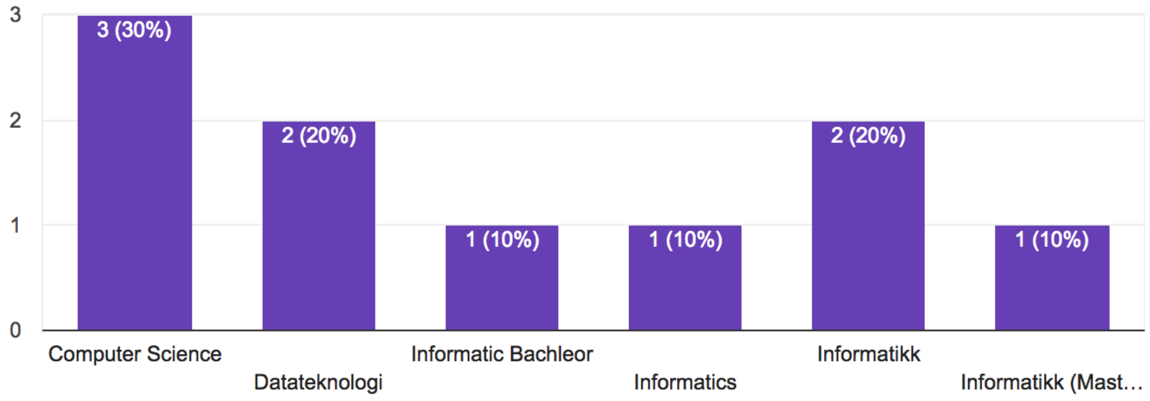
If you are a student, which grade/year are you in?

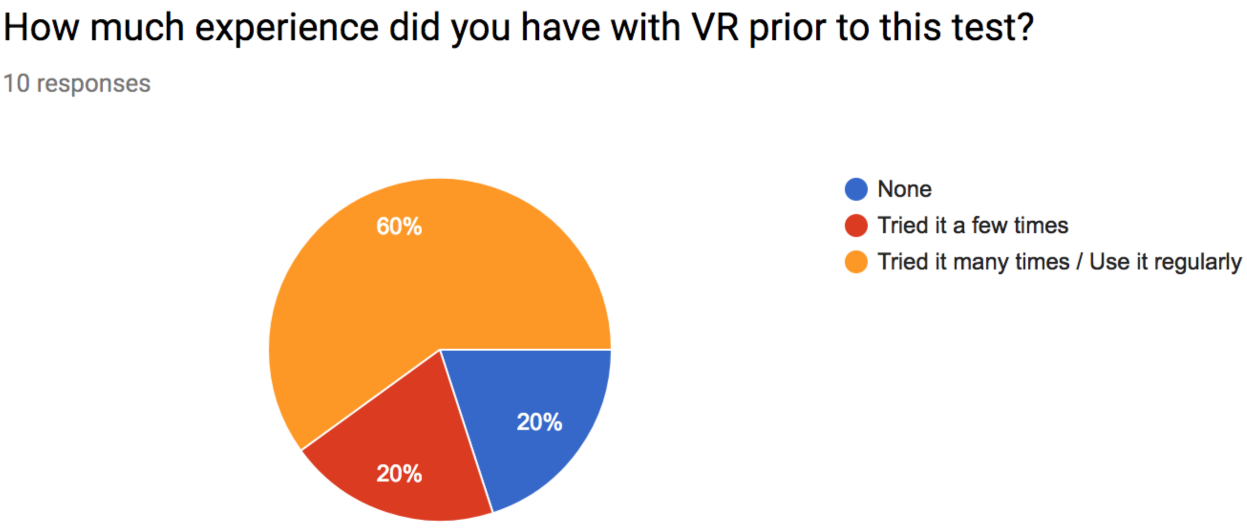
10 responses



If you are a student, what are you studying? Alternatively, if you are in High School (Videregående skole), what's your specialization?

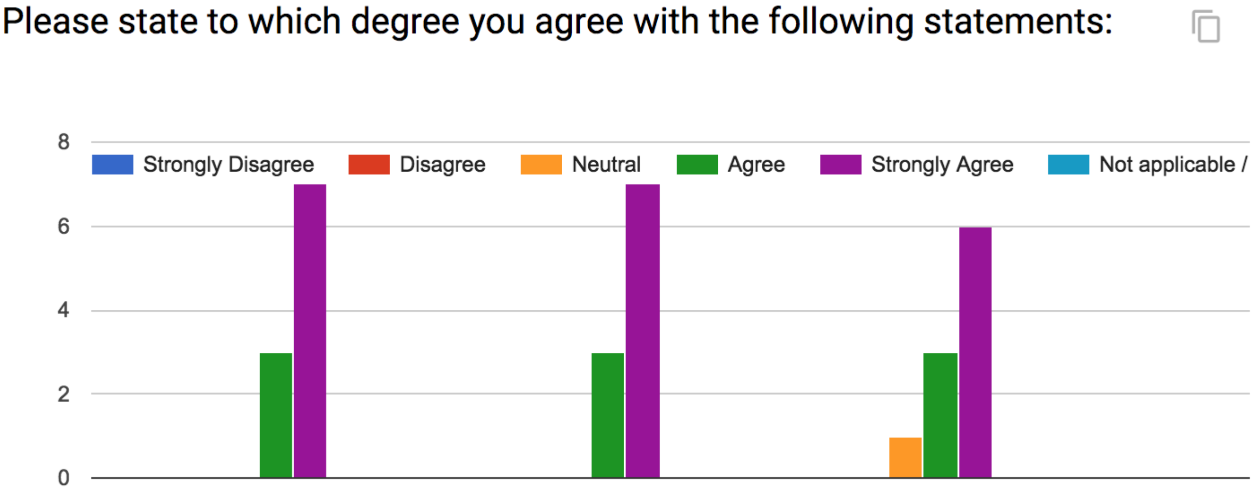
10 responses





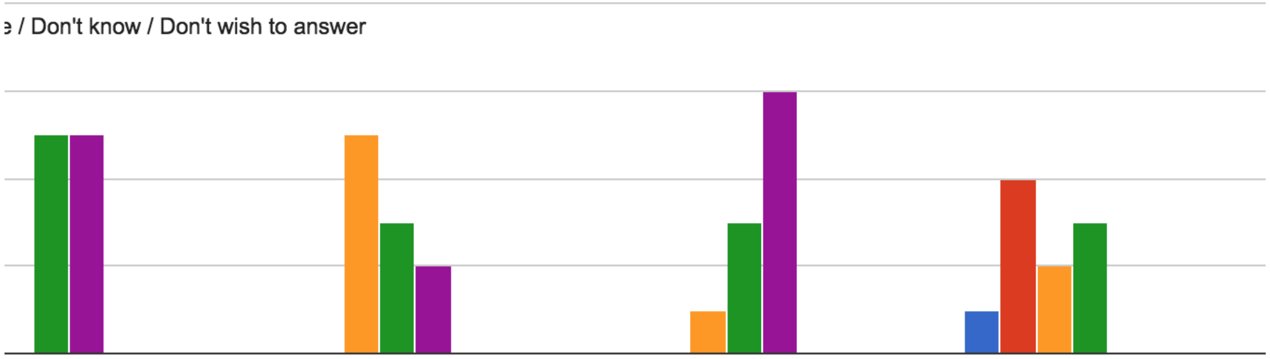
How much experience did you have with VR prior to this test?

10 responses

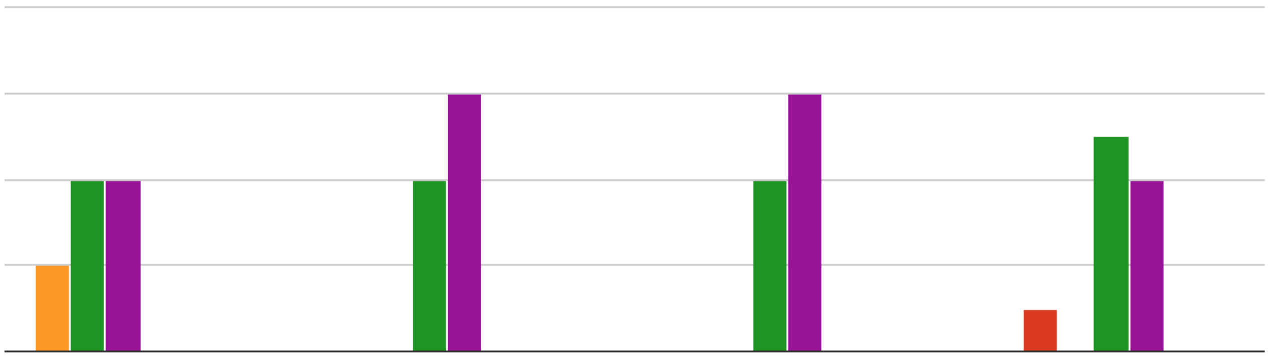


Please state to which degree you agree with the following statements:

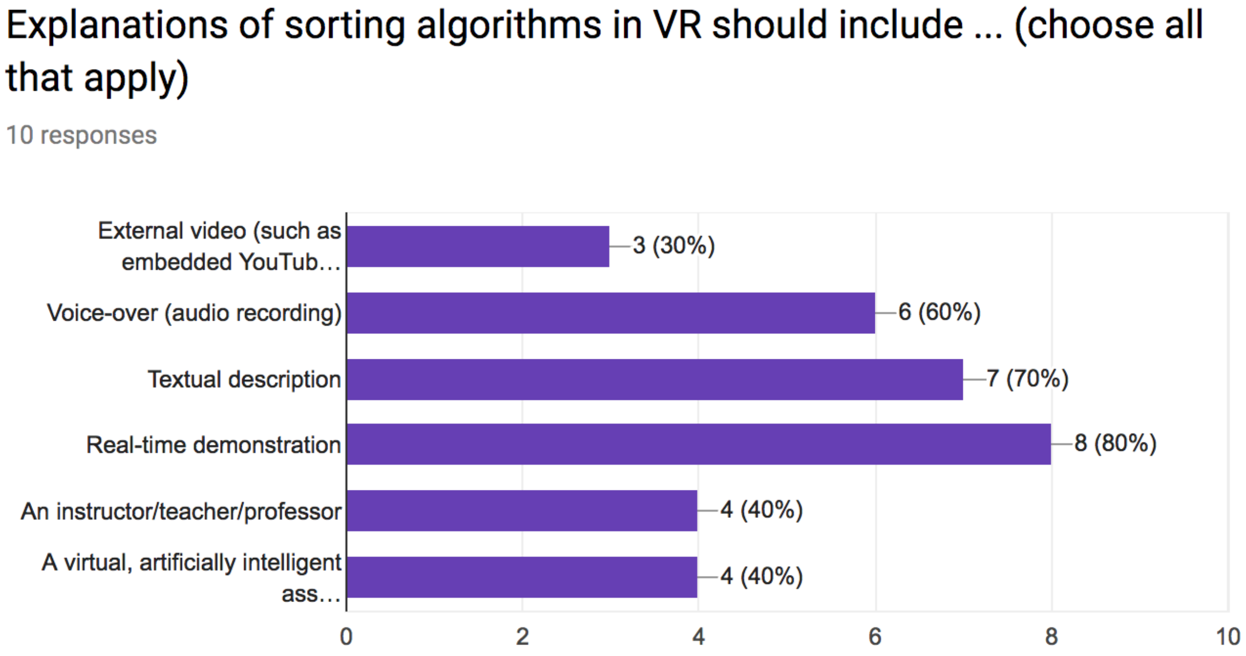
Statements 0-2



Statements 3-6



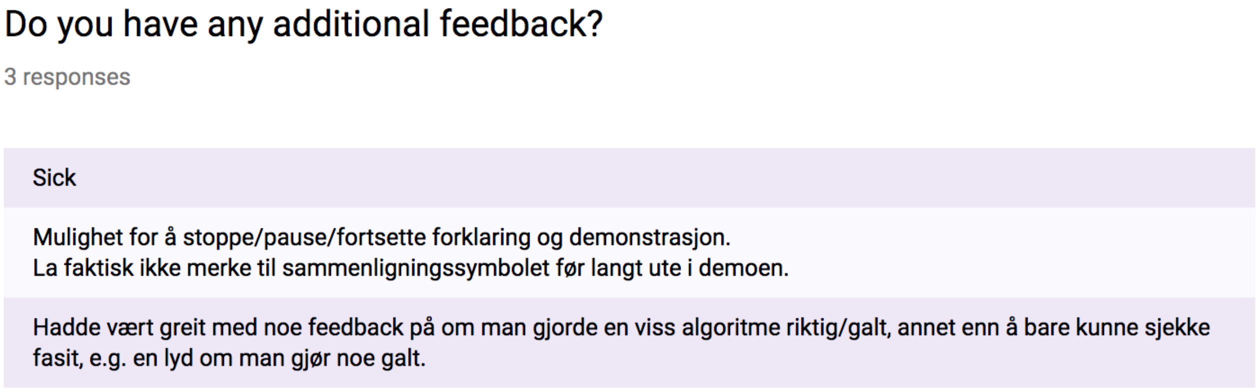
Statements 7-10



Explanations of sorting algorithms in VR should include ... (choose all that apply)

10 responses

External video (such as embedded YouTub…
Voice-over (audio recording)
Textual description
Real-time demonstration
An instructor/teacher/professor
A virtual, artificially intelligent ass…
3 (30%)
6 (60%)
7 (70%)
8 (80%)
4 (40%)
4 (40%)
0
2
4
6
8
10



Do you have any additional feedback?

3 responses

Sick

Mulighet for å stoppe/pause/fortsette forklaring og demonstrasjon.
La faktisk ikke merke til sammenligningssymbolet før langt ute i demoen.

Hadde vært greit med noe feedback på om man gjorde en viss algoritme riktig/galt, annet enn å bare kunne sjekke fasit, e.g. en lyd om man gjør noe galt.