Context Project - Health Informatics
Group 2: CondExt
Final Report

Matthias Tavasszy
4368401

Arjan van Schendel
4366212

Luke Prananta
4288386

Jasper van Esveld
4372581

Bart Ziengs
4391799

June 23, 2016

Preface

This report was written for the purpose of *TI2806 Contextproject*, a course given in the bachelor *Computer Science* at the *Delft University of Technology.* This project was conducted in cooperation with *CleVR*, a company that specifies in Virtual Reality Exposure Therapy (VRET) and is based in Delft. This report aims to describe our progress and achievements during the two months the project lasted. This project would not be possible without the support and helpfulness of people from both the university and *CleVR.* Firstly we would like to thank *CleVR.* Guntur, Niels and Joris thank you for the extensive introduction, feedback sessions and providing us with the necessary hardware components. Mr. Brinkman, thanks for guiding us in the right directions and making us think about proper ways we should organize our project. At last we would like to thank René Vennik for his role as our teaching assistant. Your comments and code reviews contributed to our product as it currently is.

We wish all of you the best reading our report and if any questions may arise we are happy to reply.

Kind regards,

Jasper, Luke, Arjan, Matthias, Bart

Contents

Introduction **4**

1 Product and Software Overview **6**
1.1 Software . . . 6
1.2 Essential Hardware . . . 7

2 Product Reflection from a Software Engineering Perspective **9**
2.1 Performance . . . 9
2.2 Maintainability . . . 9
2.2.1 Design Patterns . . . 9
2.3 Functionality . . . 10

3 Process Reflection from a Software Engineering Perspective **11**
3.1 Tooling . . . 11
3.1.1 StyleCop . . . 11
3.1.2 CodeMaid . . . 11
3.1.3 GhostDoc . . . 11
3.1.4 CloudBuild . . . 12
3.2 Scrum . . . 12
3.2.1 Communication with the Group . . . 12
3.2.2 Communication with the Stakeholders . . . 12
3.3 Git . . . 12

4 Description and Evaluation of the Functional Modules and Entire Product **13**
4.1 Integrating the Leap Motion with the Kinect . . . 13
4.1.1 Inverse Kinematics . . . 14
4.1.2 Application of Inverse Kinematics . . . 14
4.2 Grabbing Mechanics . . . 14
4.2.1 Collision Detection . . . 14
4.2.2 Grab State . . . 15
4.2.3 Release State . . . 15
4.3 Clipping with the Grabbed Object . . . 16
4.3.1 Single Handed . . . 16
4.3.2 Double Handed . . . 16
4.4 Failure Analysis . . . 16
4.4.1 Duplicate Work . . . 16
4.4.2 Discarded Work . . . 16

5 Interaction Design and Testing **18**
5.1 User and Usage Situation . . . 18
5.2 Interaction Goals . . . 18
5.3 User Testing . . . 19
5.3.1 Method . . . 19
5.3.2 Results . . . 19
5.3.3 Conclusion . . . 20

6 Outlook **21**
6.1 Hand Rotation 21
6.2 Incorrect Tracking 21
6.3 Loss of Tracking 21
6.4 Movement 21
6.5 Clipping 22

A Testing survey results **24**
A.1 Observations 24
A.2 Survey 24

Introduction and End-Users' Requirements

Psychosis refers to an abnormal condition of the mind described as involving a "loss of contact with reality". People that suffer from psychosis are described as psychotic. Experiencing this may exhibit some personality changes and thought disorder. Depending on its severity, this may be accompanied by unusual or bizarre behaviour, as well as difficulty with social interaction and impairment in carrying out daily life activities. The non-medicinal treatment of psychosis involves psychological therapies and social support. CleVR, the company this project was conducted for is involved in making those treatments possible by using virtual reality(VR). To mimic a real world setting as realistically as possible, four hardware components are used in this specific case. The full hardware kit consist of an Oculus Rift head-mounted display, a Leap Motion device, a Microsoft Kinect and a set of ManusVR gloves. The problem we were asked to tackle is to integrate these separate components seamlessly. That is, search for an optimal combination in which they function as one product in its entirety. Either in an synergistic or serendipitous manner.

The approach that is taken is by using so-called user stories. These are descriptions consisting of one or more sentences in the everyday or business language of the end user or user of a system, which capture what a user does or needs to do as part of his or her job function. At the beginning of the project a lot of initial user stories were composed, which evolved throughout the project. Our primary user stories that reflect our requirements include the following:

- As a user, I want to be able to put my groceries in a shopping cart so that I can take groceries with me.
- As a user I want to be able to pick up items in a physically realistic way so that I can interact with the world in a more natural manner.
- As a user I want to use the Oculus Rift so that I will be fully immersed in the virtual world
- As a user I don't want my hand to snap between different tracking methods so that I will be more immersed in the virtual world.
- As a user, I want my virtual body to move in a realistic way so that I do not feel disconnected from the virtual body.
- As a developer I want to have a good understanding of the code so that it becomes easier to edit it.

As can be seen, these different stories are formed from the perspective of a imaginary end-user, mostly the patient that is being treated for psychosis. The stories that are listed above are from the last phase of the project and are evolved from initial stories by making specific enhancements in the requirements that we established during the completion of the project. Every story is, if applicable, divided in one or more sub tasks to specify the needs in an nuanced fashion. An example is given with the user story below that is subsequently divided into two sub-parts. Sub-parts can be seen as direct tasks.

- As a user I want my virtual body to stand on the ground, not hover
    - Make sure that the lowest foot of the Kinect model always touches the ground
    - Fix the hip distort bug when the Kinect loses track of the user.

After this was implemented in the product it was evaluated with the group where was decided whether or not this was acceptable and if this needed further modifications. Another example is formed by the task "As a user I want a realistic environment to interact with so that I will get immersed in the world". To verify this we set up an testing day with people that were willing to test it en let the users fill in a questionnaire afterwards. Elaboration on testing techniques and procedures will take place in the chapter *Interaction Design and Testing*.

Chapter 1

Product and Software Overview

The product is a system that uses several pieces of special hardware, controlled by a simulation program that runs on the Unity engine. This program uses the data from the hardware to provide user-computer interaction in a virtual environment. The product also includes a simulation of a supermarket where a person can interact in. The person has the ability to pick up, hold and drop virtual grocery items, while also able to slightly move around within the environment.

1.1 Software

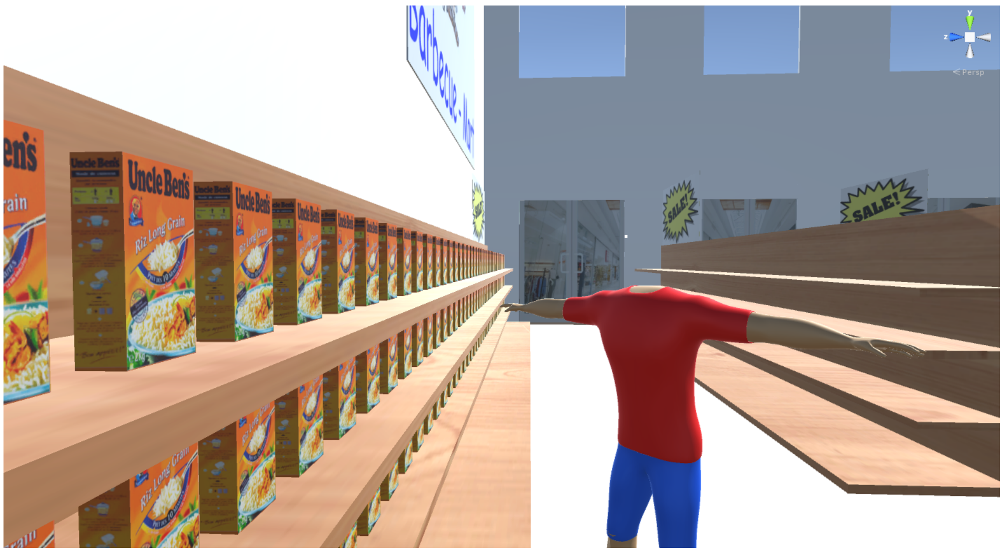


Figure 1.1: A look at the virtual environment

The code related to the simulation is implemented in C-Sharp on the Unity engine. The simulation runs on a single computer, which also acts as a connection hub for all the hardware.
The simulation is created in a Unity scene. The Unity scene includes a 3D visualization of a supermarket. It includes models of shelves filled with virtual grocery items and the scene also includes a virtual avatar, which the user controls through hardware. This is the virtual environment where the user interacts in.

In a Unity scene, each element of the scene is a Unity object. To add behaviour to certain objects MonoBehaviour-scripts are attached to the objects. These scripts are written in C-Sharp and are unique to the Unity engine. There are scripts attached to the virtual avatar that are related to translating tracking data to movements in the virtual environment. Most of these are third-party and were provided by the developers of the hardware. Additional scripts and code are written for the behaviour needed for the solution.

The virtual avatar is made of different sub-objects that represent different parts of the body. To each part are scripts attached that relate to the functionality of that part of the body. For example, scripts related to grabbing are attached to the hand objects and scripts related to body movement are attached to a overarching body object.

The simulation can accurately simulate grabbing motions and actions within a virtual environment. The user is also able to walk around in the environment within a limited range. The system determines with recognized gestures which grab to apply, a single grab with left or right hand or a double grab that uses both hands.

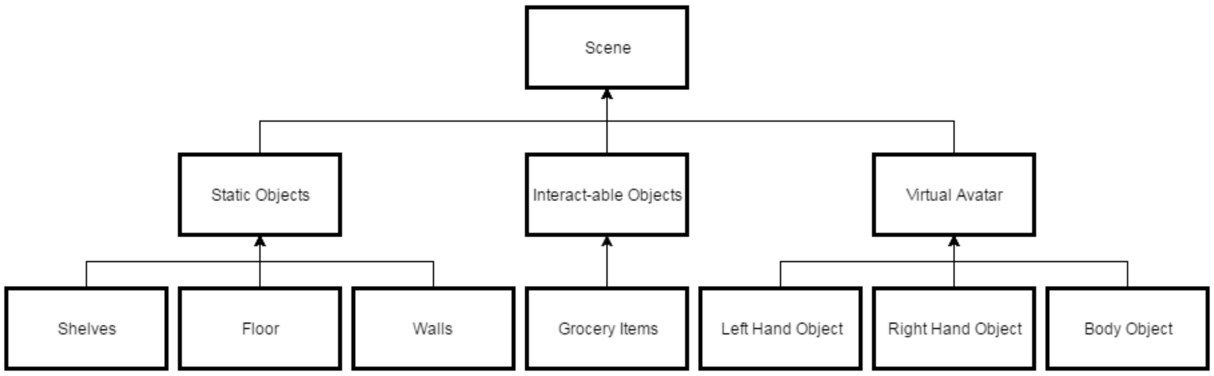


Figure 1.2: A general overview of the contents of the simulation scene

1.2 Essential Hardware

The following hardware is involved in the system:

- Oculus Rift, a head-mounted display (HMD) that provides head-tracking for the system.
- Leap Motion, a small tracking device that can track arms, hands and fingers within the vision of its infrared cameras.
- ManusVR, gloves that can accurately track the bending of fingers.
- Kinect, a device that can track full-body movements within the vision of its cameras.

The simulation uses the data from the hardware to translate real-life movements to the virtual avatar. The Leap is used to position the hands of the virtual avatar, while the ManusVR is used to correctly bend the fingers. The Kinect provides movement for the rest of the body of the virtual avatar. The Oculus HMD gives the person the ability to look around in the virtual environment. See figure 1.4.

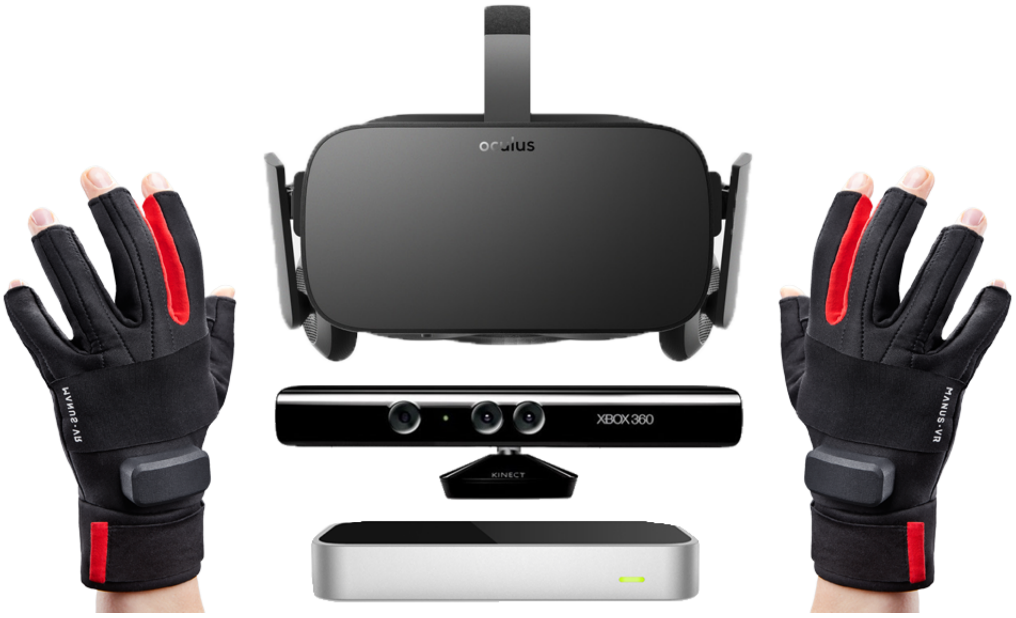


Figure 1.3: A look at the hardware

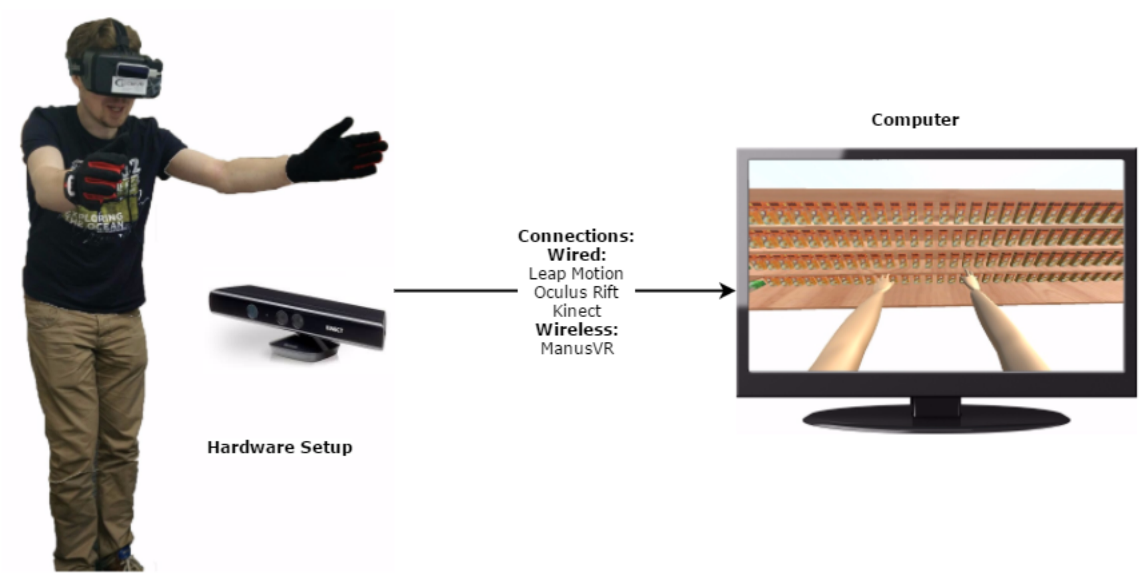


Figure 1.4: A look at the setup

Chapter 2

Product Reflection from a Software Engineering Perspective

When CleVR described the product they had in mind they wanted us to focus on a few software quality characteristics, namely: performance, maintainability and functionality. [3] In this chapter we will reflect on our decisions on these three aspects.

2.1 Performance

Performance is important because this can make or break the VR experience. We used Unity's built-in profiler to show the performance of our project and this showed that our implementation had a total impact of 2 ms per frame. About 1.9 ms comes from a function within one of the plugins we have to use and we can therefore conclude that performance won't be an issue. We made sure that time-complexity within the code is as low as possible. With this decision came a lot of refactoring, especially at the end of the project. This might not have been the most optimal decision because the performance benefits were not as big as we expected, but it still is important to keep the code running fast in case of future expansions.

2.2 Maintainability

Maintainability was another important aspect, we tried to ensure maintainability by providing documentation for all functions/classes and by implementing several Design Patterns. We wanted potential new group members to have an easy time understanding our system and in hindsight, it also provided benefits for current group members. Proper documentation made code review a lot easier and we are sure that it also improved the productivity within the group.

2.2.1 Design Patterns

Design patterns can improve the maintainability of the code. We used three different Design Patterns in our implementation for recognizing grabs. Since there can be multiple grabs on a given moment we implemented the Observer Pattern in which each observer is represented by a grab which gets notified by the subject when it needs to be updated. This subject implements a State Pattern which is used for when the subject holds one or more grabs and for when it does not. This way, the grabbed object could update itself instead of letting the hand object update all the grabbed objects individually. With this decision logic could also be more easily be separated. In the end we were content with this implementation.

The last design pattern we implemented is the strategy pattern, this is used for the actual grab itself and defines the strategy for a grab with two hands, just the right hand and just the left hand. Because we used these design patterns editing the code became a lot easier and hopefully help CleVR when implementing it in their own project. The logic for the different grabs could now be separated with strategies. This made code more clear and code became also more easy to modify. We also find this implementation satisfactory.

2.3 Functionality

In terms of functionality, we are satisfied with the current features. We were able to include a single hand grab and double hand grab. This was something CleVR really wanted to see and thus became our highest priority. With the implementation of the double hand grab came also the ability to move a grabbed object from one hand to the other. While this feature is not perfect by any means, having it implemented in some form was still a good step forward.

Because of our prioritization, certain features were not got less attention, unfinished or skipped entirely. An example of an feature that was included but not highly prioritized is the shopping cart. The idea was to let the user move the cart and drop their groceries in the cart. We were able to finish the model and code to control it.

A functionality that did not see the light of day was to use the Manus VR rotation tracking instead of the Leap Motion rotation tracking. We noticed that the Leap Motion rotation tracking struggled when the thumb was out of sight. Using the Manus VR to track the rotation of the hand would be more accurate and if we had more time we would've liked to implement this.

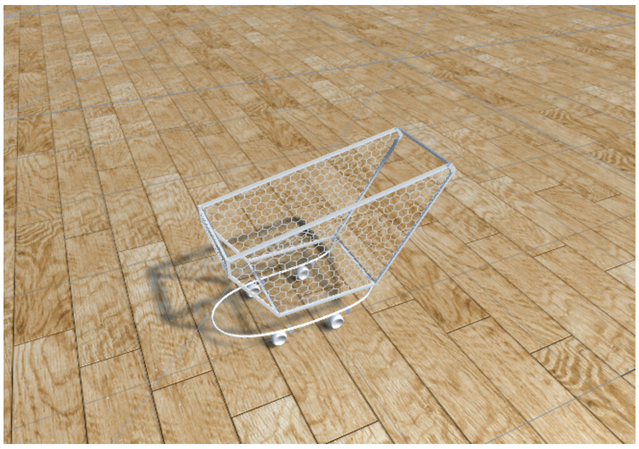


Figure 2.1: The shopping cart

Chapter 3

Process Reflection from a Software Engineering Perspective

Here we will reflect on the problems we encountered during the development process and how we managed those problems.

3.1 Tooling

The development process has had a big impact on the maintainability, quality and functionality of the final product. To help us during the development process we have made use of several tools, for example StyleCop, GhostDoc, CodeMaid, VSSpellChecker and CloudBuild. Some of these tools we did not use all that often (like CodeMaid) but others proved to be very useful (like StyleCop). In the following subsections we will go over some of these tools (VSSpellChecker is very straight forward and we will therefore not discuss it) and how we used them.

3.1.1 StyleCop

StyleCop is the visual studio equivalent of CheckStyle. Just like CheckStyle it enforces the readability standards within Csharp. Besides that StyleCop also ensures that comments meet certain requirements like documentation on return values and parameters. We used StyleCop to improve readability during the development process, this not only helped us during the development but will also help CleVR when we deliver the final product.

3.1.2 CodeMaid

CodeMaid is a very handy tool that helps solve StyleCop errors (white space, method order etc.). CodeMaid was recommended by CleVR but we ended up not using it all that much because most of the problems that CodeMaid helped to fix were easy to do by hand. Looking back we could have explored more of the features CodeMaid has to over.

3.1.3 GhostDoc

To speed up the creation of documentation we used GhostDoc which generates the basic structure for a method or class with a simple shortcut. This made it easier for us to generate documentation however GhostDoc has more features to offer which we did not utilize.

3.1.4 CloudBuild

CloudBuild is the Continuous Integration solution for Unity. It works with any Unity project can be connecting with a git repository. For each change within the project it will than automatically build and run all tests resulting in succeeding and failing build. There is however one downside and that is the time for each build when using the free plan. The average build time for our project was more than one hour which meant that using CloudBuild for things like pull requests became basically impossible.

3.2 Scrum

During the development of our product we followed the Scrum method with sprints of one week. While we have used this method before we still came across some things we could improve. For example at the start of the project we had some problems selecting clear user stories and assigning concrete tasks. We ended up with vague tasks that would be assigned to multiple people. Another problem we had was with our priorities. These would be given correctly but we did not give an explanation to how they came to be. After the feedback we received during the meetings with our SA we would improve our backlogs each week. With backlogs improving we noticed that making a proper retrospective would become easier. This is mainly because the tasks became clearer and therefore easier to reflect on.

3.2.1 Communication with the Group

When we started the project we thought that meeting up each day would not be necessary and we could just use Skype when communication was needed. However we noticed that when we did meet we would all be more productive. After the first two weeks we tried to reserve one of the project rooms of Drebbelweg for most of the week. This way whenever one of us encountered a problem we all would be there to help and when changes in the structure of the code were needed we could directly discus them. We would strongly recommend any project group to meet in person on a regular basis.

3.2.2 Communication with the Stakeholders

We could not meet up with CleVR (the company we were working for) every week, we did however have weekly meetings with Willem-Paul Brinkman and René Vennik (SA). The weekly meetings were very useful not only because of the feedback we received on our progress, code and sprint but you also get to take a step back and see what work was performed during the last sprint. Even though we could not always meet up with CleVR we still wanted to be able to show our progress. Therefore we used Slack to share short videos showing our progress of that week. This helped the meetings we did have because CleVR already had an idea of the progress we made leaving more time discussing further development.

3.3 Git

Just like in previous projects we made heavy use of pull requests and branching. We tried to keep commits directly to the master to a minimum and use pull requests even for small changes. This made sure that at least 2 people would check the code that is pushed to the master. We did however have a problem with GitHub's statistics when it comes to lines of code. Since we also needed to push our resources (models, textures, etc.) the lines of code for each person got really skewed. Another thing we learned is to make use of the Issues functionality of GitHub, it turned out to be a great reminder since you cannot directly fix every bug you find.

Chapter 4

Description and Evaluation of the Functional Modules and Entire Product

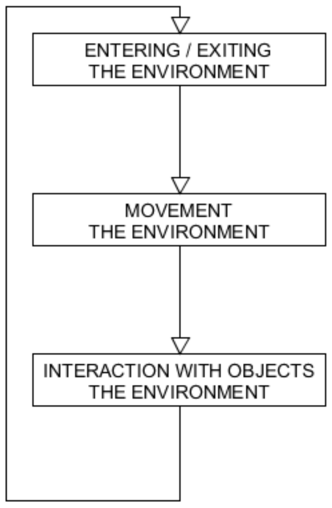


Figure 4.1: Composition of functional modules

When aiming for a composition into modules, our product can be split up into the three components that are shown in the figure above. The first model, *Entering / exiting the environment* is just as it says the act of spawning and removing the user in and out of the virtual environment. There is basically no smooth operation for this since there has to be a transition somewhere. Secondly the *movement in the environment* is very limited in the current state of the project since moving around is not possible (yet) due to the missing of proper controls to achieve this.

The final functional module, *Interaction with objects in the environment* is the most complex and testable part. This last part is also the most addressed when gaining feedback from the test users. More about this can be found in the chapter about testing and in the chapter on the outlook.

4.1 Integrating the Leap Motion with the Kinect

To combine the Leap Motion hand position with the Kinect body a certain problem needs to be overcome, namely the fact that the Leap Motion hands need to be connected to the Kinect body. The easy solution would be to just override the Kinect hand position with that of the Leap Motion, however this would result in huge amounts of distortion because the arm is disconnected from the hand.

4.1.1 Inverse Kinematics

Another solution is to use a technique called Inverse Kinematics. Inverse Kinematics can be used to solve a chain of joints so that the end joint will be as close to a certain goal as possible. It is used in many fields like Character animation as well as robotics. Our implementation of this technique is fast and therefore useful in virtual reality but not as accurate as the algorithms used in robot arms.

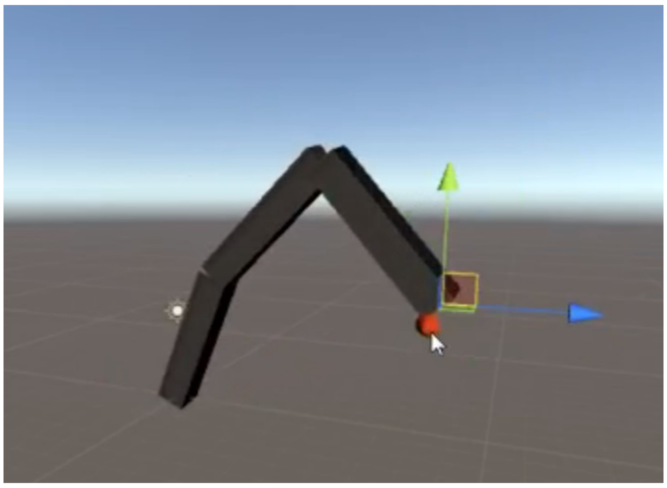


Figure 4.2: Joints solved using Inverse Kinematics

4.1.2 Application of Inverse Kinematics

With this technique it is possible to solve the arm's rotation and make the palm of the hand come as close to the goal. This goal can then be set to the hand position of the Leap Motion. The result is a hand that is accurately tracked with the Leap Motion while not causing any distortion.

4.2 Grabbing Mechanics

The grabbing mechanics of the solution can be separated in several subsections. The system can recognize based on the amount of fingers to get a proper understanding of the whole grabbing system, every subsection will be described in detail.

4.2.1 Collision Detection

Before the system can enter a grabbing state, the system must first detect potential grabs. To determine a single hand grab, the fingertip of the thumb and the fingertip of another finger of the same hand must touch or collide with the same object. To determine a double hand grab, fingertips from both hands must detect the same object.

To detect a grab, a method called "OverlapSphere" is used which is part of the Physics class provided by Unity. It returns an array with all colliders touching or inside the sphere and the position and the radius of the sphere can be set.

The system runs the method for each fingertip to determine collision with objects. Because the method returns an array of colliders, the system only looks for the closest collision. With this the system can determine if a grab is possible and which object will be grabbed.

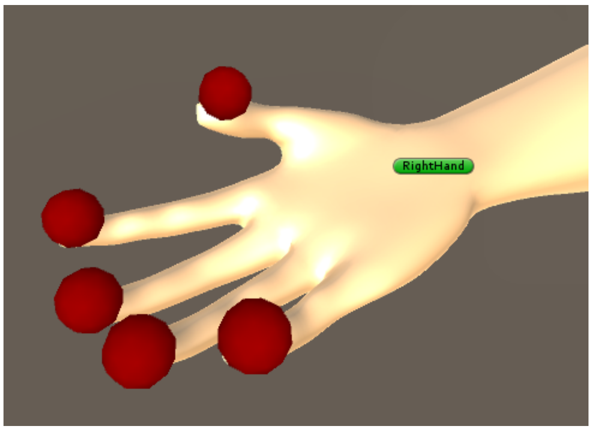


Figure 4.3: OverlapSphere applied at the fingertips, visualized by gizmos for debug purposes

4.2.2 Grab State

When the system enters a grab state, the system checks for several things. First it limits the current bending of the fingers which is described later in more detail. The system creates an empty GameObject, which will be referred as the root object, to which the grabbed object is parented to. The relative start position and start rotation of the grabbed object is calculated. This will be used to correctly calculate the relative position and rotation when updated.

Now the system can commence the grab. If the grab is single handed, the system updates the position and rotation of the root object relative to the position and rotation of the hand object for each rendered frame. With this, the grabbed object will correctly stick to the hand as in real life. If the grab is double handed, the root object is positioned at the center between the two hands and the rotation is based on the rotations of both hands.

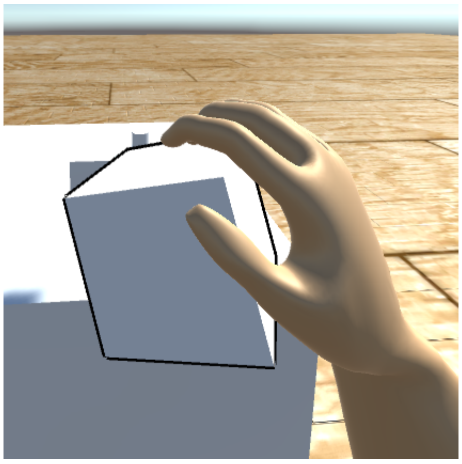


Figure 4.4: Single hand grab with fingers clamped

4.2.3 Release State

To determine a release for a single handed grab, the system checks if one of the following conditions are met: There are not enough valid fingers still touching the object or all the fingers of the hands are fully open. For a double handed grab, the system checks if the position of both hands are still in range to touch the object.

If the conditions are met, the system enters the release state. The root object previously mentioned is destroyed and the parent of the grabbed object returns to previous parent. To make throwing objects possible, the system also applies the last known velocity of the hand object to the grabbed object.

4.3 Clipping with the Grabbed Object

Clipping is the phenomenon when two items pass trough each other. To prevent the user from feeling disconnected from the virtual body clipping needs to be minimized. When the user grabs an object the chance of the fingers clipping with the object is very much present.

4.3.1 Single Handed

For single handed grabs this can be prevented with the help of a clamping method. This method makes sure that the fingers can't move past the bending value from when the finger collided with the object. This prevents the fingers from clipping through the object because the object follows the palm during a grab just like the fingers. Meaning that the bend value is the only thing that moves the finger relative to the object.

4.3.2 Double Handed

The same method is applied in a double handed grab but another problem needs to be solved. During a two handed grab we use the average position of the hands to move the object however this means that the hands themselves can move inside of the object. The solution to this problem is to use the Inverse Kinematics script and set the goal to the starting position whenever the hand moves inside the object.

4.4 Failure Analysis

Of course, during the project we had some things not going or working as planned. These failures costed us time and work, and should not be repeated in future projects. A couple of the most costly failures during our project are listed below.

4.4.1 Duplicate Work

Because of miscommunications, every now and then two people ended up doing the same work while not being aware of the other person. This resulted in duplicate work and one of the two solutions having to get discarded in favour of the other. This could have been a good thing since you have the freedom to choose the best out of two solutions, but in this case our time was very limited thus duplicate work should be avoided.

4.4.2 Discarded Work

One of the things that didn't go as planned was the implementation of the PhysicsGrab class. This was supposed to become the improvement over our previously implemented gesture grab, but right after adding the code to the project we made a more general, better and easier implementation which also supported dual handed grabbing. Because of this the PhysicsGrab code got discarded, thus the effort and time spent on the class was not utilized and could have been spent on other features. We did however used our knowledge gained by the PhysicsGrab to implement the new grabbing script, thus the time did not entirely go to waste.

Chapter 5

Interaction Design and Testing

5.1 User and Usage Situation

The user of this software will use the system in a treatment room. This means that there has to be enough space to walk around and make full use of the detection range of the Kinect. The user will be a person that suffers from a psychosis, meaning that he or her sees a distorted version of reality. [2]
The entire purpose of this software is to make the user face his or her biggest fear: public places, in particular a supermarket, in virtual reality. In order to make the user show their fear, the virtual world has to look as realistic as possible. This includes a other people, a realistic supermarket, a working and moving virtual body, and the ability to pick up items. The last two of these were the problems that this software had to tackle.

5.2 Interaction Goals

Effective and Efficient
Goal: There should be close to no latency between real body movements and virtual body movements.
Solution: When tested on the available laptops there was some minor latency. While testing on a more high-end gaming PC, (Intel i5 6600K, MSI GTX 960 4GB, 16GB RAM) there was no noticeable delay or latency. The situation in which this system will be used, a PC with at least these specifications will always be available.

Safe
Goal: All virtual reality movements have to be natural and physically possible, the patient should never be exposed to potentially 'scary' experiences.
Solution: As many impossible body movements as possible were made impossible. For example, the wrists bending the wrong way is impossible.

Utility
Goal: The user can interact with the virtual world in a natural way, meaning that they can walk around and pick up items.
Solution The user can pick up items using their hands. Walking around is possible but limited to area covered by the Kinect.

Easy to Learn and Remember
Goal: Controlling the system should feel natural, simple, and easy to remember.
Solution: Controlling the virtual body is very intuitive. All a user has to do is move their real body, and the virtual body will move accordingly. Because the movements are the exact same as people use in their whole lives, remembering them is no problem.

5.3 User Testing

5.3.1 Method

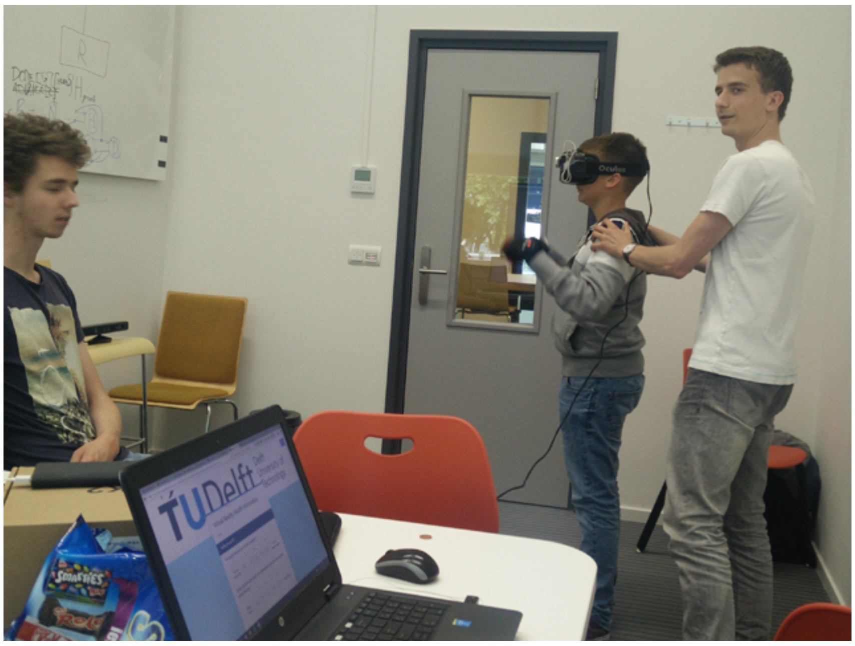


Figure 5.1: Guiding a test person to the correct position

Users that had to do the tests were put in a room, with enough moving space to be able to comfortably complete the assignments. Firstly, they were asked some questions about their experience and opinion on virtual reality before the test. After that they put on the ManusVR and the Oculus with the Leap Motion attached. They were then guided into the correct position in front of the Kinect, then given the following tasks:

1. Look around
2. Look at your hands, move them and your fingers
3. Look at and move your feet
4. Walk towards a shelf
5. Pick up an item with one hand
6. Pick up an item with two hands
7. Throw the item away
8. Put an item on a lower shelf
9. Pick an item up from a lower surface (approximately 30cm / 1ft from the ground)

When all these tests were done, users were asked to give a brief explanation about their experience, how easy the tasks were to complete, and what they though were the biggest problems in the system.

5.3.2 Results

In general the reactions to the system were very positive. Most users needed some time to adapt to the environment. This was half a minute per person on average. When given the assignment to pick up three items and put them on a lower shelf everyone succeeded at least once. While observing and speaking to the testers a couple of problems became apparent:

First of all, the length of the arms in the system is fixed. This means that people with short arms reach the virtual objects before they expect to reach them, and people with long arms reach them later than expected. Because of this some people initially had problems with reaching for objects.

Secondly, the tracking of the Leap Motion is limited to a small field of view. This means that when the hands are moved out of its field of view, the Kinect has to take over. This transition was not always as smooth as it should be, and could be described as 'snappy'.

A final, minor point that some of the testers mentioned was the restricted area that can be walked around in. They wanted to explore the supermarket environment and move further than the available two by two meters. This could be solved by using a different movement tracking device, like the Virtuix Omni [1]

The results from the survey can be found in the appendices.

5.3.3 Conclusion

After the tests and reviewing the test results, a few things can be concluded:

First of all, making a user believe that they are actually in the virtual world does not require perfectly realistic graphics. As one of the users responded : *"Resolution is less important (...) than correct picking up and precise tracking: those done wrongly disturb the illusion the most."*

That leads to the following conclusion: it is really hard to correct the hardware's limitations. The Leap Motion for example, has a very limited field of view. When tracking is lost, the system tries to use the data from the Kinect to keep the simulation realistic. The problem is, firstly, that the Kinect is way less accurate than the Leap, so it is easy for the user to notice this difference. Secondly, the transition from one tracking device to the other is not as smooth as it could potentially be, but this can be solved through future developments.

Chapter 6

Outlook

In this chapter we look at possible future improvements for the product.

6.1 Hand Rotation

Improvements could be made in hand rotation tracking. Instead of using Leap Motion data to add rotation to the hand objects, the rotation of the Manus VR could be used because it is more accurate and less sporadic.

6.2 Incorrect Tracking

Another improvement that could be made as stated before in the chapter on testing, half of the people testing the product indicated incorrectness of tracking as a problem. Thus, this definitely is an area in which improvements can be made. Solutions can be found by creating states for certain gestures or body movements where tracking is less accurate and take averages from data provided by other hardware.
Another solution involves different hardware. A system could use the Vive HMD and controllers to track the hand instead of the Leap Motion because it is accurate and more reliable. With Vive, Room VR can be used to accurately track movement within a room.

6.3 Loss of Tracking

Improvements could be made when certain hardware loses tracking. For example when the Leap Motion loses tracking, instead of an instantaneous jump between positions, you could give this process a smooth transition to make it less jarring for users.

6.4 Movement

Currently the movement of the user is limited by the Kinect to about 2-by-2 meter of free movement. The only other way to move through the environment is with the use of keyboard keys. CleVR stated they would like to use the Omni for these kind of movements which is a possible solution.

6.5 Clipping

At the moment the user's virtual body can move freely through static objects. Another option we had was to stop the virtual hand when it hits something. Something that would be doable in the future would be to let the virtual arm collide with the object and stop moving, but let it continue moving again when the arm would not be in a colliding position anymore.

Bibliography

[1] Virtuix omni. `http://www.virtuix.com`. Accessed: 23-06-2016.

[2] Oliver Freudenreich. Differential diagnosis of psychotic symptoms: medical mimics. *Psychiatric Times*, 27(12):56–61, 2010.

[3] Joost Visser. Building maintainable software. 2015.

Appendix A

Testing survey results

A.1 Observations

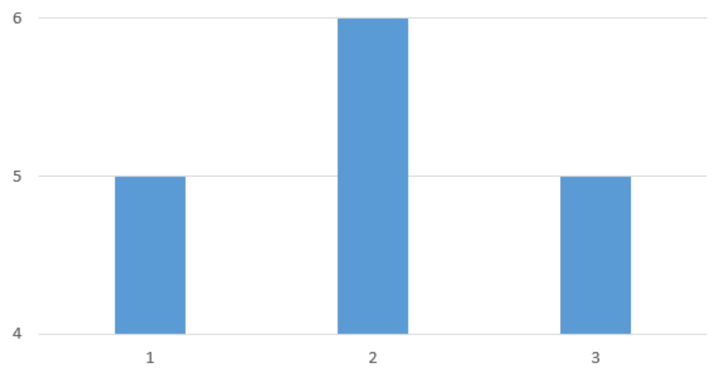


Figure A.1: Amount of items picked up

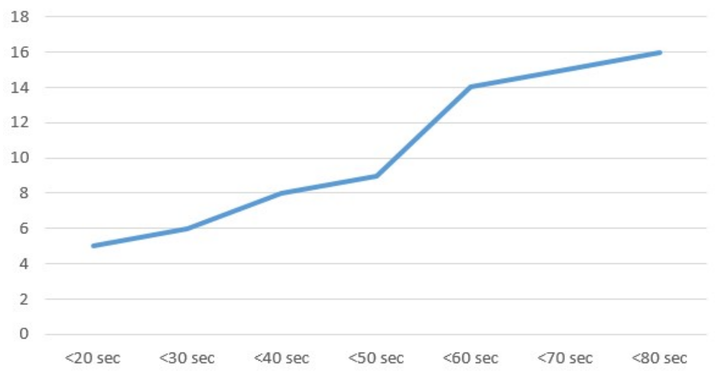


Figure A.2: Amount of users that picked up up their first item within a certain amount of time

A.2 Survey

The VR tests were held after question 7.

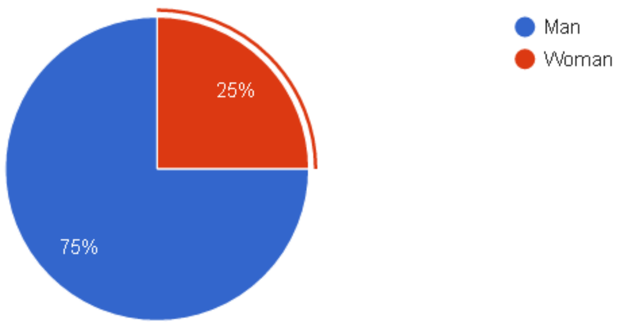


Figure A.3: Are you male or female?

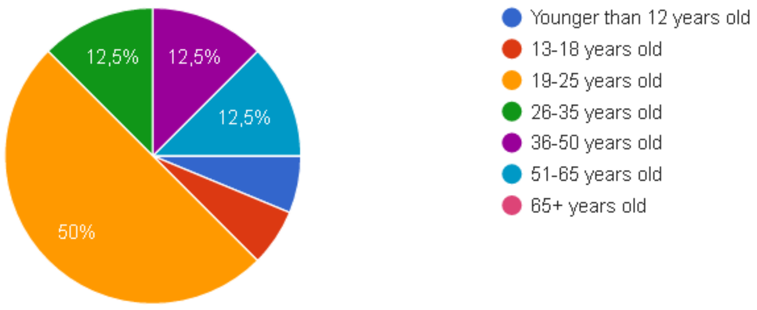


Figure A.4: What is your age?

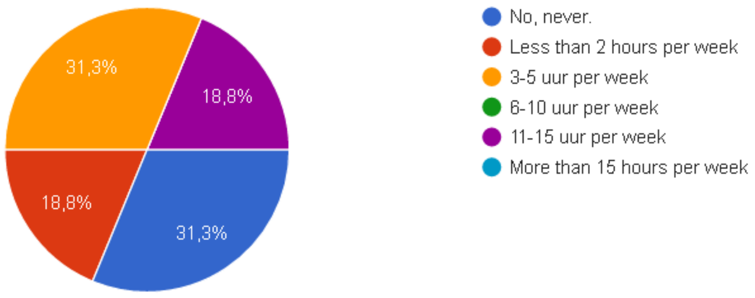


Figure A.5: How often do you play video games?

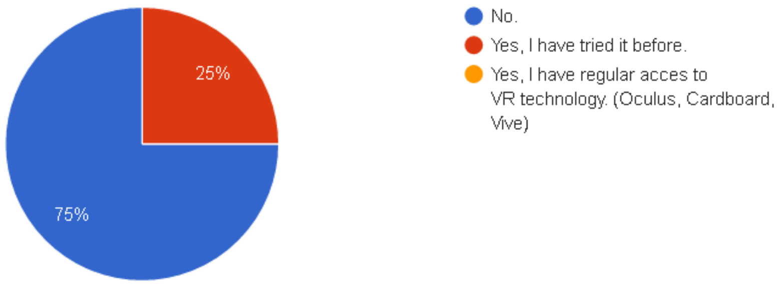


Figure A.6: Have you ever tried virtual reality before??

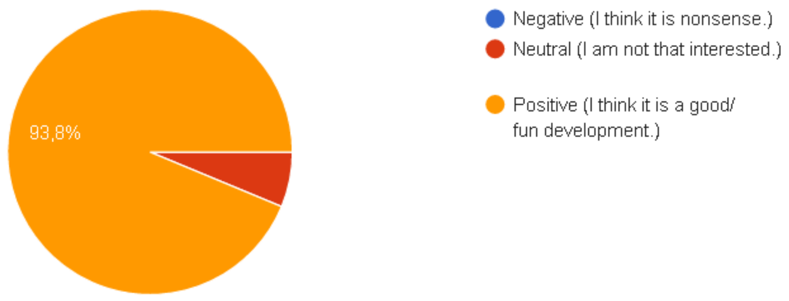


Figure A.7: What is your opinion on virtual reality?

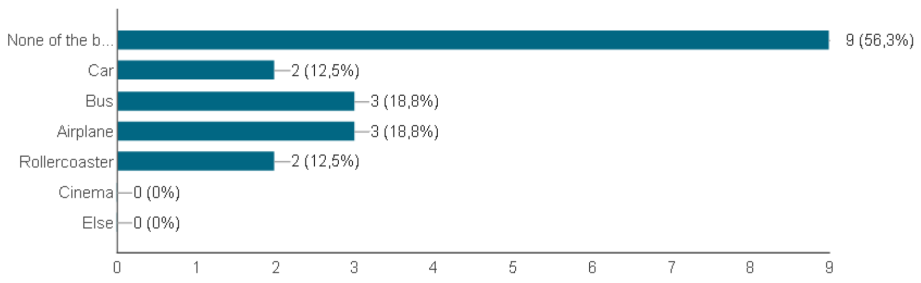
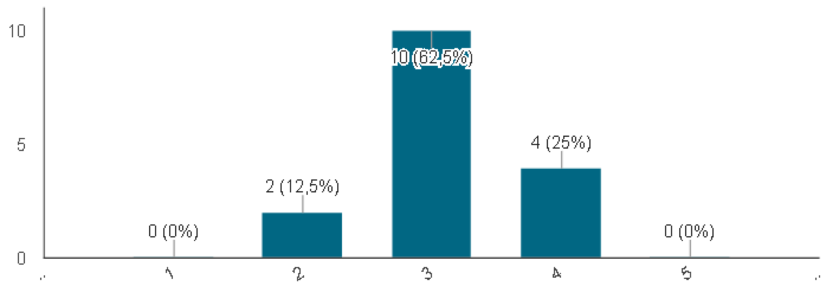


Figure A.8: Do you ever feel sick in one of these situations?



0

Figure A.9: How well do you think virtual reality will represent the real world?

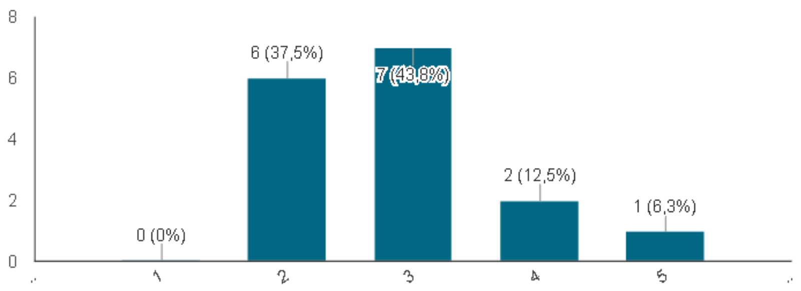


Figure A.10: How well did virtual reality represent the real world?

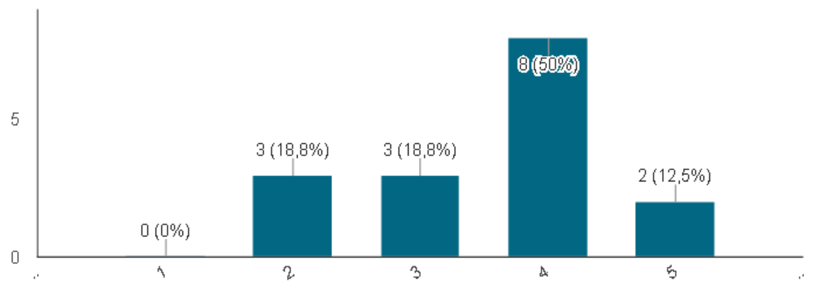


Figure A.11: During the simulation, I felt like I was really in the virtual world.

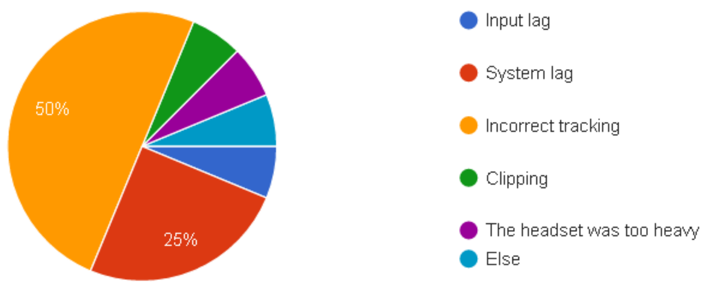


Figure A.12: What was the biggest problem you encountered during the tests?

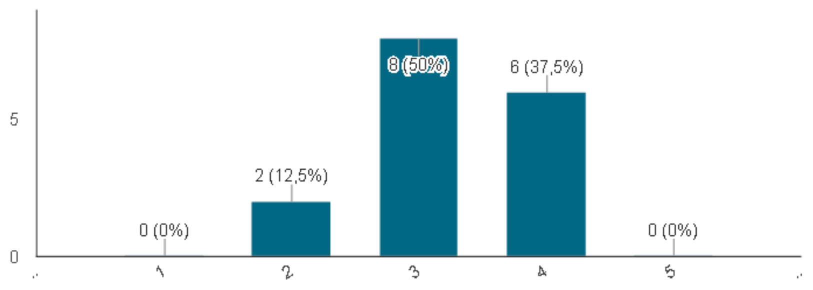


Figure A.13: How easy is it to pick up objects?

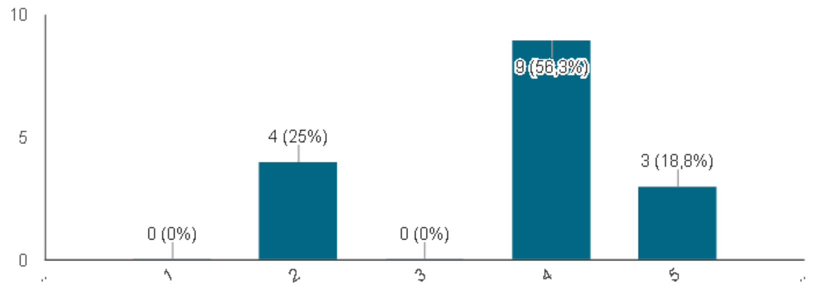


Figure A.14: How easy is it to drop objects?

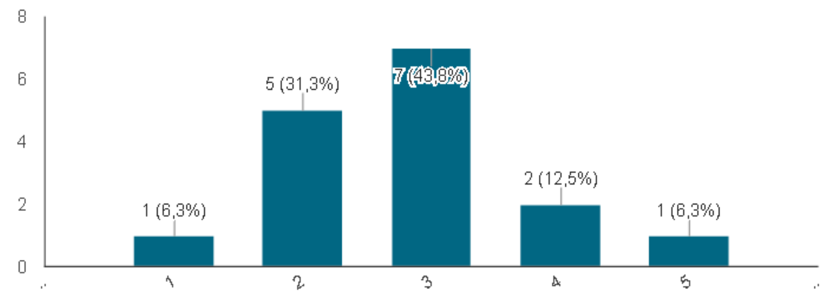


Figure A.15: How realistic are the physics in the virtual world?

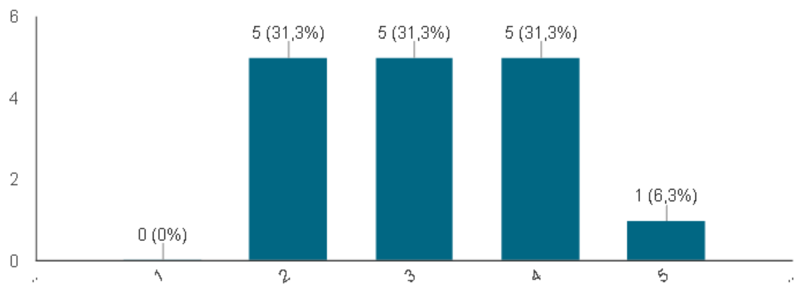


Figure A.16: The environment looks realistic.

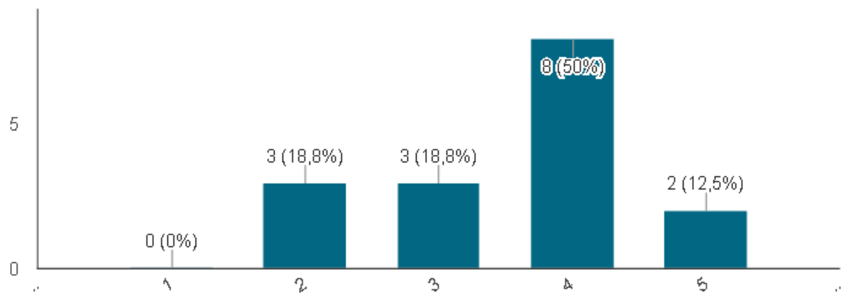


Figure A.17: The objects in the environment look realistic.

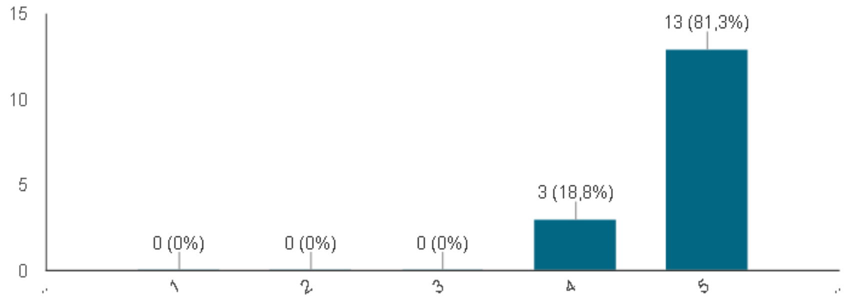


Figure A.18: I enjoyed trying virtual reality.

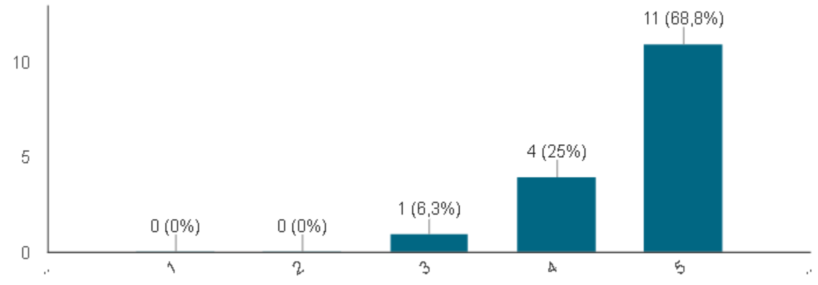


Figure A.19: I want to use virtual reality again.

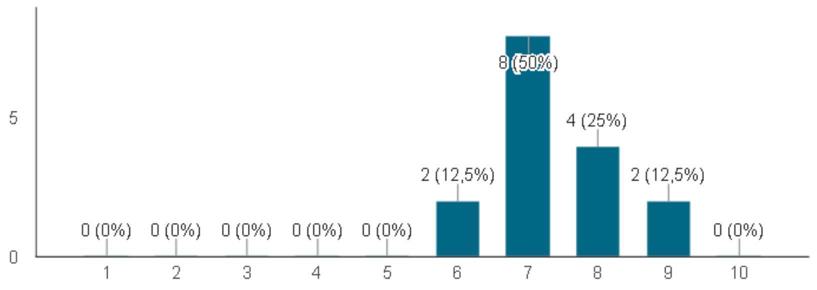


Figure A.20: How would you rate the entire system?