Technology
Arts Sciences
TH Köln

Project Documentation

Interaction Lab

written by

Vera Brockmeyer	(Matrikelnr. 11077082)
Anna Bolder	(Matrikelnr. 11083451)
Britta Boerner	(Matrikelnr. 11070843)
Laura Anger	(Matrikelnr. 11086356)

Interactive Systems in SS 2017

Supervisor:
Prof. Dr. Stefan Michael Grünvogel
Institute for Media- and Phototechnology

Inhaltsverzeichnis

1 Introduction

Vera

The main idea of Virtual Reality (VR) is to ensure a totally immersive user experience. A first important milestone was the development of the first Head-Mounted-Display (HMD) in 1966 by Sutherland and Goertz. This HMD offers the possibility to walk around in each virtual scene and to look at the scene from every view point. Despite this possibility, a totally immersive experience asks for user interactions with virtual objects like selecting, grabbing, manipulating, movement and indirect controlling via widgets, gestures and voice input in virtual scenes. Especially, a realistic grabbing and positioning of these objects is required which should come close to human motion sequences and fits to the human cognition and experiences. These kind of immersive interaction methods should include further requirements as well. First, a high precision rate is unavoidable and must be guaranteed. Second, the method should not be tiring for the users. Finally, the methods should be easy to learn and offer an ergonomic usage.

The number of VR devices and applications increases heavily during this decade and the relevance of consumer and business VR applications rises steadily. Several suppliers currently offer various methods and devices for grabbing interactions. The most common are the *Oculus*-HMD, *HTC Vive*-HMD, *HTC Vive*-Controller, data gloves and motion capturing systems for hand-tracking like the *LeapMotion*-Controller. Most devices offer a system with two hand controllers at this state. These hand controllers enable the use of virtual menus and offer a robust hand tracking. The latter is urgently needed to implement common grabbing interactions. Due to this rising amount of VR devices, it is vital for the development and improvement of grabbing methods to evaluate their usability and performance in an adequate environment as well as to compare them with the state of art methods.

In the following sections, we describe and discuss an interactive VR system to compare different controller-based grabbing methods to evaluate their usability, error rate and time consume. This VR system runs with the *HTC Vive*-HMD and two *HTC Vive*-Controllers. Thus, all offered grabbing methods are controller based which could be learned in a special learning room and evaluated in a predefined VR supermarket with all required measurement tools.

1.1 Motivation

Vera

Currently, there exist no interaction laboratory that compares the different grabbing interaction methods. Similar laboratories [14][19][**?**] have been developed but they do not allow the user to compare different kind of grabbing methods. In case of [**?**] and [19], it provides only some VR applications or devices to experience the methods in different scenarios. Another [14] laboratory evaluates natural grabbing methods without a credible evaluation.

This circumstance claims a virtual laboratory, where different interaction methods could be compared, demonstrated, or tested in adequate virtual test scenes. Furthermore, all tests and compares should be based on scientific standards to allow factual and useful results. Thus, this laboratory should provide all required measurement tools to start credible and scientific studies of pre-implemented grabbing methods. This helps to standardise the evaluations and to increase their comparability of common studies. Another achievement is that the user friendliness of interaction methods which are not tiring and do not destroy the immersion could be improved by researcher.

Another aspect of those user-friendly methods is the increasing usability of VR applications which will yield to a higher consumer preference of devices with their implementation. Therefore, the profit of VR device suppliers will be squeezed and the relevance of those products will expand worldwide.

In an educational context, it could be used for demonstrations during lectures. This will give the lecturer an effective tool at hand to explain the importance of usability as well as the advantages and disadvantages of each grabbing method. Another aspect of this laboratory, is to give students a tool for the technical realisation of interaction studies in virtual (or augmented) reality environments.

1.2 Project Goal

Vera

Hence, a virtual interface laboratory with an environment to test and compare grabbing methods should be developed. Furthermore, it should offer a possibility to develop new ones as well and it offer a use for teaching purposes. Thus, one task is to develop sophisticated test scenes for testing the interaction methods. These scenes should implement test exercises in different difficulty levels and represent typical and well-known environments like shops. All relevant parameters for the evaluation of the methods should be measured automatically and saved in an output comma separated value file. The latter must be easily imported in common statistic tools. Additional, the laboratory should offer digital questionnaires to evaluate the usability of each provided method as well as one that enquire the system relevant properties like motion sickness, immersion, and latency.

2 State of the Art

Laura

In the following sections the fundamentals and scientific knowledge of all parts of the *Interaction Lab* will be described. Publications of similar projects are described in section 2.1. Furthermore, an overview of methods of (grabbing) interactions (compare section 2.2), as well as their evaluation methods (compare section 2.3) are given.

2.1 VR Labor

Anna

In the development process of the grabbing interactions there were also different evaluations of the differences between the interactions. There are a few papers and other articles. In the following three different works will be described shortly.
In the year 1999 Pouryrev and Ichikawa categorized and evaluated different grabbing interactions in their paper [17]. In their project they used hand tracking to measure the position of the real hand and to visualise a virtual representation of the hand. The participants than had to select different simple test objects, like cubes, spheres and cylinders. Also a positioning task were supplied. A few years later Lee et al. evaluated different raycast grabbing methods [13]. Lee at al. used a 3D mouse for tracking positions and orientation of the hand. With this mouse the participants selected spheres on various and random positions. For this the participants were ask not to change the position of their head.
In the last year Eriksson published his master thesis, which is relative similar to this project [8]. Erikkson used the Oculus Rift with the Oculus Touch to provide the tracking and visualisation. In this project the participants also had to complete tasks in a virtual shop. The tasks had similar requirements related to the selection but also refer to manipulation and translation of the objects.

2.2 VR Grabbing-Interactions

Laura

To create a immersive virtual environment it is, inter alia, necessary to make use of adequate grabbing methods [2]. As mixed reality was not taken into consideration for implementing the *Interaction Lab*, all objects, the user can interact with, are completely virtual.
The interaction methods provided by the *Interaction Lab* have no haptical feedback [1], which would allow a conclusion about the surface quality of the gripped objects. Furthermore, they only use the hardware described in section 3.1.2 for the execution of the interactions.
Many of the common methods of interaction, which are available in the *Interaction*

Lab (compare section 4.3) are explained in the paper by Bowman and Hodges [2]. In their explanations they also mentioned the *Go-Go* technique, which presents a conceivable extension of the laboratory.

2.3 VR Grabbing-Interactions Evaluation

Britta

3 Materials

Britta

The following section will give an overview over all hard- and software that was used in this project.

3.1 Hardware

Britta

The hardware consists of a head-mounted display, the HTC Vive and a computer.

3.1.1 Computer

Britta

@Britta: Vielleicht möchtest du die Tabelle einfach übernehmen und nur die Daten des PCs ändern?

Die Hard- und Software-Voraussetzungen für die Ausführung der *Unity*-Anwendung in Verbindung mit der *HTC Vive*, welche in Tabelle 2 aufgelistet sind, werden von dem verwendeten Computer übertroffen.

3.1.2 HTC Vive

Britta

The *HTC Vive* is a head-mounted display which is being produced by *HTC* in co-operation with *Valve* [24]. It was introduced on the 1st of March 2015 at the *Mobile World Congress* [16].

Dieist ein Head-Mounted Display, welches von *HTC* in Kooperation mit *Valve* [24] produziert wird. Vorgestellt wurde dieses am 1. März 2015 im Vorfeld des *Mobile World Congress* [16].

CGPC6	**Beschreibung**
Prozessor	Intel Core i7 6700 CPU @ $4 \times 3.4 - 4.0\,\text{GHz}$
Arbeitsspeicher	16 GB
Grafikkarte	NVIDIA GeForce GTX 980
Betriebssystem	Windows 10 Education 64 bit
Schnittstellen	2× USB 3.0, 5× USB 2.0, 1× HDMI

Tabelle 1: Übersicht der technischen Daten des Computers für die *Unity*-Simulation.

HTC Vive	Systemvoraussetzungen
Prozessor	mindestens Intel Core i5-4590 oder AMD FX 8350
Grafikkarte	mindestens NVIDIA GeForce™ GTX 1060 oder AMD Radeon™ RX 480
Arbeitsspeicher	mindestens 4 GB
Videoausgang	1× HDMI 1.4-Anschluss oder DisplayPort 1.2
USB	1× USB 2.0-Anschluss
Betriebssystem	Windows 7 SP1, Windows 8.1 oder Windows 10

Tabelle 2: *HTC Vive* Systemvoraussetzungen [11].

Die Auflösung des Displays beträgt insgesamt 2160×1200 Pixel, was 1080×1200 Pixeln pro Auge enstpricht. Die Brille bietet ein Sichtfeld von bis zu 110° bei einer Bildwiederholrate von 90 Hz [10]. Alle technischen Systemvoraussetzungen können in Tabelle 2 eingesehen werden.
Zur Positionsbestimmung im Raum wird die Lighthouse-Technologie [3] von *Valve* genutzt. Zusätzlich sind neben einem Gyroskop auch ein Beschleunigungssensor und ein Laser-Positionsmesser verbaut. Mittels proprietärer Hand-Controller wird bei der *HTC Vive* eine Interaktion mit virtuellen Objekten ermöglicht.

3.2 Software

Britta

3.2.1 Unity

Britta

@Britta: Vielleicht möchtest du das nur auf Englisch übersetzen?

Unity ist eine sogenannte Spiel-Engine, also eine Entwicklungs- und Laufzeitumgebung, die speziell auf die Entwicklung von 3D-Spielen ausgelegt ist. Die Software wurde am 6. Juni 2005 veröffentlicht [9] und wird von *Unity Technologies* [20] entwickelt und vertrieben. In der Spieleentwicklung ist *Unity* weit verbreitet, so werden beispielsweise 34 % der kostenfreien Top-1000-Spiele im mobilen Sektor mit *Unity* entwickelt [22].

Unity bietet eine sehr breite Plattformunterstützung [21] und erlaubt ebenso die Entwicklung für Head-Mounted Displays, wie etwa die *Oculus Rift* [23] oder auch die in diesem Projekt verwendete *HTC Vive* [23].

3.2.2 Visual Studio 2015

Britta

@Britta: Vielleicht möchtest du das nur auf Englisch übersetzen?

Micosoft Visual Studio 2015 ist eine verbreitete integrierte Entwicklungsumgebung (IDE), welche unter anderem die Programmiersprachen Visual Basic, Visual C#, und Visual C++ unterstützt. Mit Hilfe dieser IDE kann ein Entwickler Win32/ Win64 Anwendungen sowie Web-Applikationen und Webservices [15] programmieren und anschließend kompilieren. Für *MArC* wurde mit der Version 14.0.25123.00 Update 2 gearbeitet.

3.2.3 Steam VR

Britta

@Britta: Vielleicht möchtest du das nur auf Englisch übersetzen?

Steam VR [18] ist die Schnittstelle zwischen der *HTC Vive* und *Unity*. Um das HMD nutzen zu können, musss *Steam VR* auf dem Computer installiert sein. Für den Nutzer ist ein kleines GUI Element auf dem Monitor sichtbar, welches den Status der Geräte der *Vive* darstellt. Hierdurch werden Fehlermeldungen kommuniziert, Kalibrierungen durchgeführt und eine Kommunikation mit dem HMD bereitgestellt, so dass das Gerät im Fall der Fälle neu gestartet werden kann.
Innerhalb von *Unity* stellt *Steam* ein Plugin zur Verfügung, welches direkt in Szenen in *Unity* eingebettet werden kann. Der Entwickler ist also in der Lage, eine vorhandene *Unity*-Szene um die VR Möglichkeit bequem per Drag-and-drop-Technik zu erweitern.
Das bereitgestellte *Unity*-Prefab beinhaltet alle notwendigen Elemente um mit der Hardware kommunizieren zu können. Dabei wird eine Positionsbestimmung ebenso wie ein Kamera Rig für die stereoskopische Bildwiedergabe bereitgestellt, wie auch die Controllereingabe und Weiterverwendung der Daten möglich gemacht.

4 System

Laura

The actual application can be divided into to type of VR rooms. First of all there is a learning room (compare section **??**), where the user can get in touch with the different interaction methods and afterwards different tasks will be presented to him in a VR supermarket scenario (compare section 4.1.2). In the learning room the user will be supported in his learning process by a selfteaching system (compare section 4.4), which can get switched on and off, when he is in the supermarket. The user can make this setting among all other settings in a Menu (compare section 4.2), which is controllable with the *HTC Vive*-controller.

4.1 VR Labor

Anna - first rough version

In this section the different rooms of the VR Labor will be described.
There are two different kinds of rooms. First there is a simple room to get familiar with the system and the methods, which were implemented. This room will be specified in the section 4.1.1.
The second room is created to look like a small supermarket. Here the user has to solve different little tasks. This room will be specified in section 4.1.2.

4.1.1 Learning room

The learning room is the first room where the user will experience the system. On one side the user will get familiar with the virtual experience and the HTC Vive system (HMD and controller). On the other side the user will learn the different interactions.

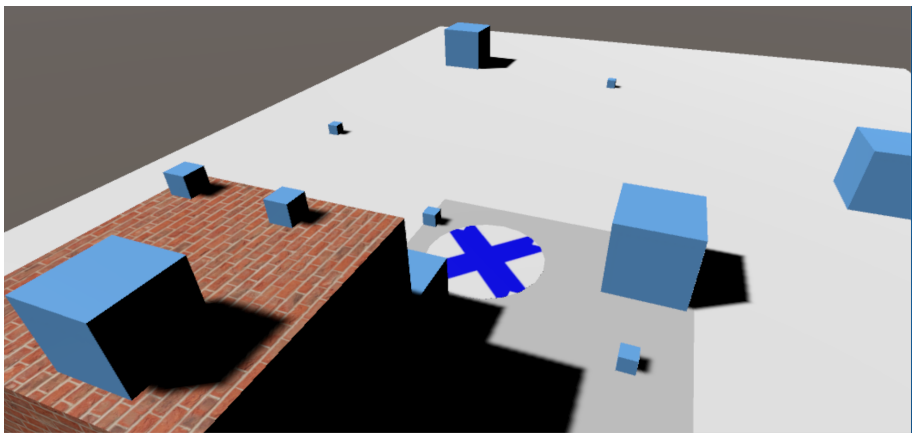

Abbildung 1: Overview of the learning room.

The room is designed very clean and simple. In different sizes and distances simple

objects, in this case cubes, are distributed randomly, shown in Figure 1. The virtual room is slightly bigger than usable room in reality. So the user is forced to user interactions, which are designed for bigger distances, to move some objects. Due to different colors of the floor the user know how big the real room is approximately. Also the HTC Vive system offers a coloured wire, which is shown in the HMD, if the user gets close to the border of the calibrated area.
The user should already know the different interactions as well as the menu settings, when he/she enters the supermarket. So the menu and interactions are similar in both rooms. Also the labelling of the target object and the reaction of the target area to the target object will be established. This will be described in section **??**.
To get to know all settings and methods the user will be led to the entire learning room by a selfteaching system. This selfteaching will be described in section 4.4.

4.1.2 Supermarket

The second room is a small supermarket.

Abbildung 2: Overview of the supermarket.

Like it is shown in figure 2 in this supermarket are different sized objects in various distances. All objects are things that could be found in a real object, from fruits to milk. This objects were used from the asset store. **CITE?**
In the supermarket is the size of the real room also shown by the color of the floor. The supermarket is bigger than the real room. So interactions for far distance have to be used according to the tasks.

Abbildung 3: Closer view of the shelves.

In some tasks the participant has to pick objects which are hard to pick. On one side they could be at a lower position, on the other side they could have to move other objects before they reach the labelled object. To implement this into a supermarket naturally shelves, like in figure 3, are appropriate for a supermarket. This shelves were build with different scaled cubes.

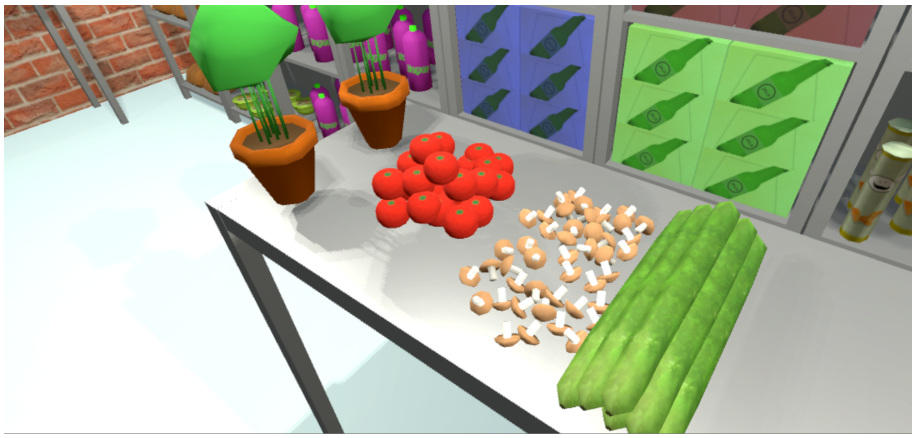

Abbildung 4: Closer view of the table.

In the real supermarket objects are not always in shelves. Also some objects where placed on tables as one bulk, for example fruits and vegetables. This is implied with two tables in this supermarket. The closer look to one of this tables is shown in figure 4.
Depending on the tasks different target areas can appear in the supermarket. In this situations different objects will be blend out. Also the target object of every tasks will be labelled. The first target area and target object are shown in figure 2.
The structure of the labelling of the target object are specified in section 4.4. The different tasks as well as the reaction of the target area to the target object will be described in section **??**.

4.2 Controller Menu

Anna - first rough version

The controller menu is the main menu of the system to change settings without leaving the virtual environment. Within this menu interaction methods can be changed and settings like snapping for the interactions can be enabled. In the learning room the user is able to turn the selfteaching informations on and off. In the same way the user can decide if the tasks will be shown or not in the supermarket. In the supermarket is also the option available to start the task again. With the reset button the scene will be reloaded.

For the measurement, see section ?, the user has to start and stop every tasks. The start button will be the first menu button in every scene. Only if this button is pressed the user can interact with the environment and will be led to the next menu. When the user presses the stop button the switching area will appear and the corresponding box collider will be enabled. If the user reaches this area the next scene will be loaded.

The structure of the menu is built on the Plug-in "VRTK"(Virtual Reality Toolkit). This is a free packet with different scripts and prefabs for interaction in the virtual reality. From this plug-in the radial menu is used for the controller menu. This creates menu buttons over the touchpad of the controller of the Vive. By scrolling over the touchpad the button can be selected. The selection will be shown by highlighting the button. If this button should be executed, the touchpad has to be pressed when the button is selected. The number of buttons per menu can be changed within the Unity editor. Each button can get a different icon. Through a link to a function of a script this function can be executed if the button is pressed. In this project there is a menu script in which all functions for the menu are collected. Due to this script different settings can be done.

4.3 Interaction Methods

Laura

Of course there were various different interaction methods required to make the *Interaction Lab* suitable for the testing described and evaluated in section 5. Also all interaction methods are implemented to realise the grabbing of virtual objects, there can be separated in the two categories, described in the next two paragraphs:

Close Range (CR) Interactions: The CR can be interpreted as a synonym for the natural interaction radius of the person. Due to this definition it is excluded that those interactions can be used outside an area, which the person can not reach with his arm, or to be more precise: with the controller in his hand. In other words: the CR combines all interactions which can be used to pick up objects in the direct reach of the user.

Far Range (FR) Interactions: Due to a limitation of the range of motion in VR applications, it is common to have grabbing interactions, which allow the users to grab objects which are normally seen as out of their reach [12]. Interaction methods allowing such an acting are called FR interactions.

Whereas the CR interactions differ mainly in the accuracy of the selection of an object while grabbing it, the FR methods differ in their usability. All characteristics of the various interaction methods can be traced in their descriptions (compare sections 4.3.1 - 4.3.6).

For a better understanding it should be mentioned, that all interaction methods can be controlled with the *HTC Vive*-controller. Even there are plenty of different possibilities to grab an object, all methods have in common, that the grabbing is caused by pressing the trigger on the *HTC Vive*-controller. The releasing of the object is than triggered by letting it go. Whenever there is a divergent Usability necessary, it is described in the respective section (compare 4.3.5).

Due to an easier integration into the learning room (compare section **??**), as well as the actual supermarket scenes (compare section 4.1.2) all methods using a ray (compare sections 4.3.5, 4.3.4 and 4.3.3) are summed up in one script called *AllRaycastMethods.cs*. All other methods have their on script, where the grabbing and releasing is implemented. Also the *Raycast Head Mounted Display*-method, described in section 4.3.6, is using a ray, it is not included into the script mentioned above. This is caused by remaining problems during the implementation of this method, which lead to an unfinished work. Further explanations on why this method is not available in the *Interaction Lab* can be found in the according section.

The two interaction methods, described in sections 4.3.1 or rather 4.3.2, can be used with snapping or without it. This technique is used to reassign the position and orientation of a grabbed object in hand. By assuming that the middle of the ring of the *HTC Vive*-controller is the new center of the grabbed object, the actual grab could appear more realistic to the user.

In the application all objects, which can be grabbed are tagged as moveable.

In the following sections all available interaction methods of the *Interaction Lab* are presented. To guarantee a better overview they are sorted by their interaction range.

4.3.1 Close Range: Touch Grab

Laura

When this interaction method is selected, the user can make use of the *HTC Vive*-controller to pick up objects directly by pulling the trigger. To release the object the trigger needs to be released as well. An object can be grabbed, whenever it is tagged as moveable and collides with the *HTC Vive*-controller. This collision is detected by giving the object a collider, which fits its form best [4][6] and applying a *BoxCollider* to the controller (compare figure 5). In the script *TouchGrab.cs* in which

the interaction method is implemented there will be checked frequently, whether there is a overlap of the collider of the controller with the collider of a moveable object or not. Whenever they collide, the respective object is coloured green to show the user, that he could grab it by pulling the trigger.

Abbildung 5: Grabbing a virtual Object by using *Touch Grab*.

4.3.2 Close Range: Proximity Grab

Laura

When it comes to the CR interactions the *Proximity Grab* is by far the most inexact selection. Grabbing and releasing and object are realised by using the trigger of the *HTC Vive*-controller.
The functionality is provided by the script *ProximityGrab.cs*. The basic idea is that the object can be grabbed, whenever the object triggers the *BoxCollider* [4] placed at the end of the *HTC Vive*-controller. A more detailed description can be found in section 4.3.1 In contrast to the *Touch Grab* described in section 4.3.1 this *BoxCollider* is bigger than the actual size of the controller. To show the user which object collides with the controller and can therefore be grabbed the respective object is coloured green, as shown in figure 6. The small gap between the actual grabbed object and the controller shows the difference between the interaction method shown in figure 5.

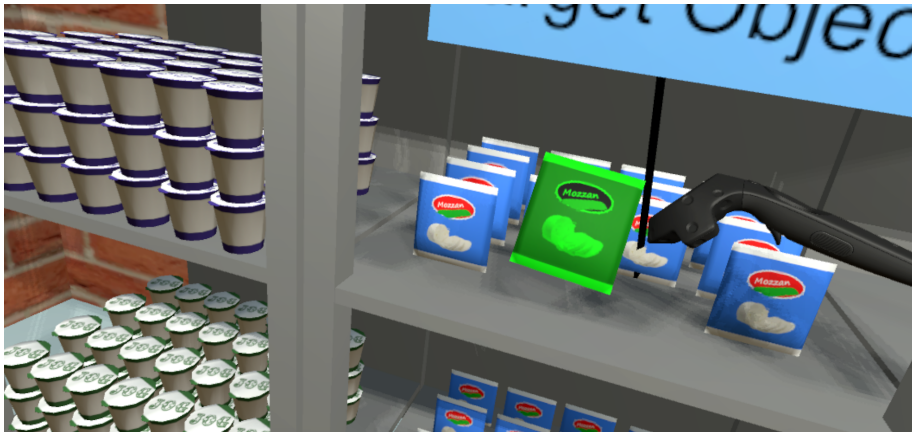

Abbildung 6: Grabbing a virtual Object by using *Proximity Grab*.

4.3.3 Close Range: Wand Grab

Laura

In contrast to the interaction method described in 4.3.2 this method can be used to grab very tiny objects. Thereby it is not needed that the target object is very isolated from other objects. To give the user such an high grade of accuracy a stick is added to the controller like shown on figure 7. The user can grab an virtual object he touches with the *HTC Vive*-controller by pulling the trigger. To place the object on the target area he simply release the trigger after he moved the object to its destination.
The implementation can be found in *AllRaycastMethods.cs*. The wand consists of two elements: a ray [5] and a cube. The cube is only for the visualisation and has a fixed size in all three dimensions. The collision detection, which is necessary for the actual grabbing, is done with the ray. That means, that an object can be grabbed, if the ray which has the same dimensions like the cube, touches this specific object. The cube will then turn from black to green to show the user, that there is an object, which can be grabbed.

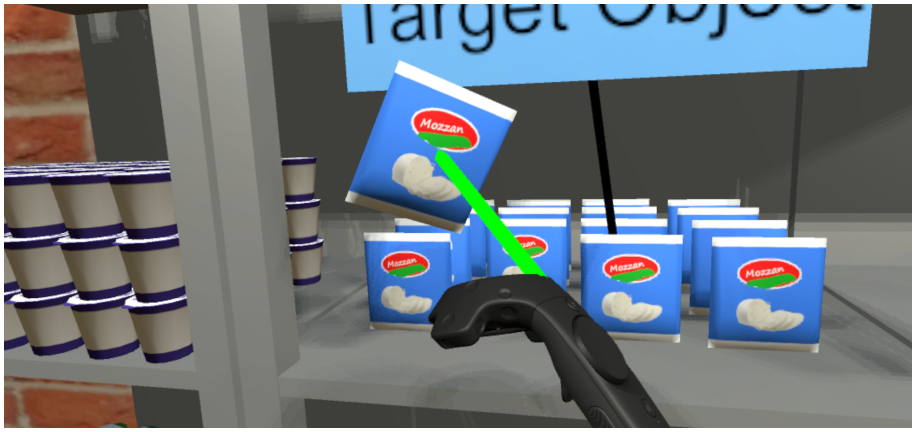

Abbildung 7: Grabbing a virtual Object by using *Wand Grab*.

4.3.4 Far Range: Raycast

Laura

By using the *Raycast* method, the user can grab virtual objects, which are further away, as well as objects in his CR. As shown in figure 8 a ray is coming out of the *HTC Vive*-controller pointing away from the user. At the end of the ray is a small sphere, which turns green, if it collides with an moveable object. Whenever the ray hits an objects, like for example the floor or a product in the supermarket (compare section 4.1.2), the ray is shortened to the distance between the controller and the respective object.
The implementation can be found in the *AllRaycastMethods.cs* script. As already explained in section 4.3.3 a cube and a ray are combined to reach the intended functionality. In contrast to the *Wand Grab*, the ray and the visible cube have no fixed length and there is a sphere added to the end of the cube.

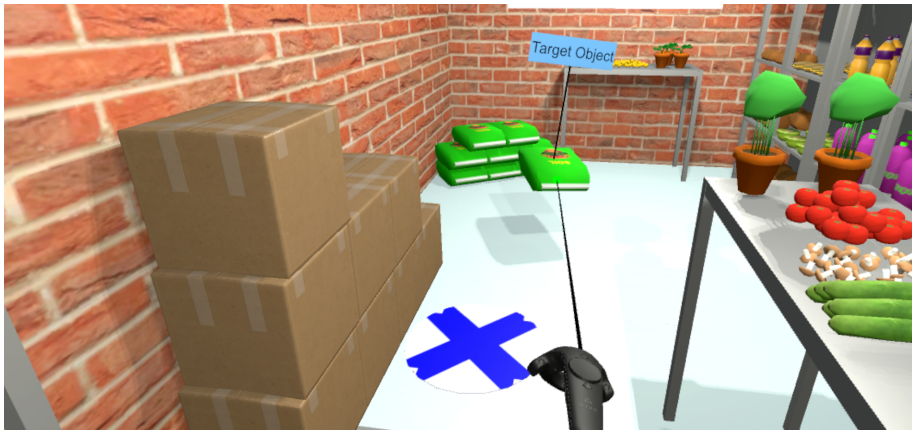

Abbildung 8: Grabbing a virtual Object by using *Raycast*.

4.3.5 Far Range: Extendable Ray

Laura

The actual ray is build as described in section 4.3.4. The ray [5] is complemented by a cube with a sphere at the end, to make it visible for the user. In contrast to the normal *Raycast* method, the length of the ray is set to a start value of 3 meters. The user can shorten and lengthen the ray by pressing the touchpad of the *HTC Vive*-controller in the lower or rather upper area. This subtracts or adds a constant value to the length of the visible ray. What remains is the behaviour of the sphere, which turns green, whenever a moveable virtual object is brushed. The *Extendable Ray* is one of the three interactions methods (compare sections 4.3.3 and 4.3.4), which are combined in the *AllRaycastMethods.cs* script. This interaction method is shown on figure 4.3.5.

Abbildung 9: Grabbing a virtual Object by using *Extendable Ray*.

4.3.6 Far Range: Raycast Head Mounted Display

Laura

It was planned to realise an interaction method, where there is a ray coming out of the HMD, which can be used similar to the method described in section 4.3.4. Due to the low accuracy of this method, it did not become a part of the *Interaction Lab*. There was a try to parent [7] the position of the HMD to the starting point of the ray. It turned out, that the parenting is not successful, when it comes to the position and rotation of the HMD. Even this method was effective for adding a ray to the *HTC Vive*-controller, there is a irregular shift when you try to implement it with the HMD. To proof that an easy example (compare source code 1) was observed. In this example a simple cube should be rendered at the front of the HMD. In reality this cube was rendered in a position, which can be seen as random.

```
1 cube.transform.SetParent(HMDEye.transform);
```

Source Code 1: Test on the parenting of *HTC Vive*-HMD and an virtual object.

All effort on this method, can be found in the script *RaycastingMethodeHMD.cs*.

4.4 Self-Teaching

Anna - first rough version, Probleme mit reinschreiben oder gesondertes kapitel?

To introduce the user to the system there will be a teaching in the learning room. When the program will be started informations will be shown next to the controller. This informations led the user step by step to the system. He/She will get to know how the interaction methods can be changed and what settings can be made. This informations will just be shown in the learning room. The position of the information area changes depending on which button is important for the interaction.

All informations and steps are saved in an external CSV-file. Due to an additional script (Name?) this file will be read line after line and be saved into different lists. In each line are three main informations. First there is the text which should be shown in this step. Second is the Button ID, which defines on which button the information should be fixed. Third and last is the hight of the information area. This hight changes depend on the amount of informations.

In the selfteaching script is a counter implemented. This counter controls which line of the file will be displayed. This counter will be increased or set from different scripts depending on an action. This increasing is still a problem if the user does not follow the selfteaching exactly. For example, if the user grabs a object twice and the selfteaching plans to grab just once, the selfteaching will go on and the user misses information.

The task are shown in the supermarket in the same way.

The information canvas is also a prefab of the "VRTK"plug-in. Here they are called controller tips. This creates a area next to the controller and a line from this area to the selected button of the controller. This area is parented to the controller and moves according to it. Different settings can be changed within the Unity Editor or by script.

The "VRTK"plug-in has also a prefab for object tips. This is also a canvas in which information can be display but this can be fixed on any object you like. With this canvas the target objects of the tasks are labelled in the supermarket.

5 Evaluation

Britta

5.1 Tasks

Vera oder Anna

6 Project Management

Vera

This chapter describes the project planning and management of the *Interaction Lab*. It is divided into the different project phases. Each division includes all important facts of its project period.

6.1 Project Definition

Vera

This section describes the results of the project definition phase in detail. This includes a problem analysis, a list of objectives and requirements, a solution concept as well as a workability analysis.

6.1.1 Problem Analysis

Vera

The demand for Virtual Reality (VR) devices and applications increased heavily since the first consumer devices like *HTC Vive* and *Oculus Rift* were released during last years. One main difficulty of the current development of VR-applications is the lack of standardisation of the Software Development Kit (SDK) and interfaces. The most acknowledged suppliers *HTC* and *Oculus* do not work together or force standards for VR application development. Thus, all applications are system related and incompatible with other devices. Accordingly, each device offers different opportunities of interaction methods. These methods could be divided in the acknowledged categories selecting, grabbing, manipulating, movement and indirect controlling via widgets, gestures and voice input. Several suppliers currently offer different devices for interaction. And with focus on the grabbing and positioning methods, the most common are the *Oculus*-HMD, *HTC Vive*-HMD, *HTC Vive*-Controller, data gloves and motion capturing systems for hand-tracking like the *LeapMotion*-Controller.

As mentioned in section 1.1, there exist currently no interaction laboratory which compares the different interaction methods in a scientific and credible way. Hence, the development of a virtual laboratory is highly requested to compare and test different interaction methods in adequate test environments. Thus, user friendly interaction methods which nearly full-fill usability requirements could be improved by researcher which yields to a higher demand of VR devices and application. That will squeeze the profit of VR device suppliers which include those user friendly interaction methods.

6.1.2 Usage Context

Vera

Hence, the required laboratory has mainly two usage contexts. First, it could be used to run scientific studies in VR research or development. Second, it could demonstrate and exemplify the differences of grabbing interactions in education proposes or support the students to develop and test grabbing methods on their own during lectures.

6.1.3 Objective and Requirements

Vera

At least two scenes should be realised to provide a laboratory which allows to run scientific and reliable study as well as is useful for the education of students. In the first scene, the user will be able to learn the offered grabbing methods. Therefore, this room is will provide simple cubes of various sizes which are in different distances from the users. Every cube is moveable and could be placed at every place. Each user is forced to follow the introductions of a self teaching before every offered method could be tested independently. The current user can only begin with the actual study after every method is trained to ensure equal preconditions.

The second room will be modelled after a supermarket because this model offers various options of grabbing and positioning tasks. In this room, the participant will get different tasks which will differ by complexity, distance of grabbing and size of the objects. The user will be able to change the options of grabbing independently but not choose the current method. An optional extension of the project will be another type of task where the user decides which type of method is preferred for this task.

The grabbing methods can be categorised into close range and far range and include the grabbing, rotating, and positioning of an object. Possible types of close range methods, are the actual touching of a movable object to select it or by holding the controller in the proximity of it but without touching it. Another more precise option is the selection with a thin wand in front of the controller. This of collection of methods that includes close human cognition methods as well as less or very accurate ones. The far range interaction will have different options as well. One will be a ray that shoot out of the controller, another one will extend a ray from the head and the third one will extend the arm in the pointed direction. This means the user will be able to point at an object with the controller or to look in the direction of it.

The system offers two measurements and the related saving of the different parameters. First, the duration time is measured for every performed task to compare and validate the performance of the different interaction methods. Second, every single grabbing try of a task will be counted and saved to get a conclusion about the learn-ability, accuracy, and performance.

Furthermore, there will be a questionnaire designed to give the users of *Interaction Lab* an usability evaluation tool at hand. This questionnaire will test parameters as tiring, learnability, self-descriptiveness and fulfilling expectations.

6.1.4 Solution Concept

Vera

An interaction laboratory for grabbing and positioning interactions at close or far range will be developed in *Unity*. It includes two test rooms e.g. scenes, where the first is a learning room, in which the users can get familiar with the interaction methods. The second room is designed as a supermarket. This environment was chosen because it offers various possibilities of exercises under changing difficulties like grabbing small mushrooms, fetching distantly placed tins or putting goods on provided target areas. The exercises are offered in form of a tasks that tells the participant what goods have to be grabbed and repositioned. These various tasks are predefined and cover all difficulties that a type of grabbing method could have. They are displayed on tables which are connected to the controller and could be shown or hide in the controller menu.

All rooms are implemented in Unity and the VR components are controlled by the same framework. Further, the *HTC Vive*-HMD and the corresponding controllers are used to run the interactions, imaging and orientation in the environment. It is planned to realise at least six interaction methods of grabbing and positioning. Additional, the complete framework should be compatible with new test scenes and other interaction categories.

The system offers a measurement of the accuracy as well. A time measuring of duration and an error rate for every performed task is planned. Each measuring of every room is automatically saved in an output file which could be easily imported in common statistic tools. Furthermore, there will be a questionnaire designed to give the users a usability and simulator sickness evaluation tool at hand. This usability questionnaire will test parameters as tiring, learn-ability, self-descriptiveness and fulfilling expectations of each method. Whereas the simulator sickness evaluation asks for motion sickness and other system properties of the complete system. All questionnaires are acknowledged and pre-tested questionnaire which fits the requirement of VR systems and application. The results of each questionnaire will be saved in an output file as well.

6.1.5 Workability Analysis

Vera

There are several risks according to the concept in section 6.1.4. First, the measurements could be implemented incompletely or inaccurately. This can be avoided by a thorough testing before the final release with some external test persons. The tasks could be incomprehensible for them as well which should be prove as well. The

system integration future extensions could cause trouble. Therefore, the systems architecture should be designed wisely and consequently to avoid incompatibilities. Another risk of the implementation is that they might be more costly and complex as recommended but this is widely acknowledged. After the implementation is finished the interaction method performance or validation could be too expensive which results in a higher latency. These circumstances must be observed during the implementation and testing. Due to the high workload of the testing, the time slot for it and the trouble shooting might be underestimated. Another time risk is that there is limited access to the facilities and VR laboratory because of the huge number of running project at the current time.

Nevertheless, the concept is feasible and the project goals could be reached during the time schedule because all the risk seems to manageable and could be observed during the scheduled testing.

The demand of the students project are satisfied and a financial profitability check is not necessaries due to the fact that the facilities of the university can be used and no further purchases are affordable.

6.1.6 Project Organisation

Vera

The project manager is Vera Brockmeyer who mainly should manage the appointments and facilities as well as to communicate to the outside. The latter is done via email or in a meeting with the concerned persons. Another task is to create and maintain the project plans that includes to keep the overview of the complete project progress and to ensure the milestones. The current state should be hand out weekly to the team in form of an email or a team meeting.

All other team members have their own responsibilities. Anna is head of the scene building which includes the definition of the general scene design, research, and to inform the project manager about current problems and timing. The latter two points concern each head of a section. The other section is split into the close and far range interactions. The head of close range interaction is Britta Boerner and the other is Laura Anger. Both manage the implementation of their section.

The formal an informal non-verbal communication in the team is done via email with the subject *VR Interface Lab* and a *Google Calendar* is exclusively maintained by the project manager where all team appointments are intercalated. This calendar shows the availability of all team members and the VR laboratory, too. More complex problems or team decisions are made in the weekly team meeting with stringently required appearance. Due to the requirements and availability of the team members, the meeting is held via *Skype* or in personal.

All files belonging to the project are organized in a cloud folder of *Google Drive* or in two *GitHub* repositories. The first one is for all *LaTex* files and the second manage the complete framework. Whereas the cloud folder contains presentation files, graphics and images, To-Do-List, papers and more.

The required facility is one of the VR laboratory of the faculty which should have a minimum size of 15qm and be located in the university building. These laboratories have a complete *HTC Vive* system and a compatible computer (section **??**).

6.2 Project Planning

Vera

6.2.1 ??

Vera

6.2.2 ??

Vera

6.3 Project Execution

Vera

6.3.1 ??

Vera

6.3.2 ??

Vera

6.4 Project Completion

Vera

6.4.1 ??

Vera

6.4.2 ??

Vera

7 Reflexion

Vera

8 Conclusion

Vera

9 Self-Assessment

???

Laura:Hier müssen wir uns am besten eine gemeinsame Struktur überlegen, oder? Vielleicht beschreibt jeder was er gemacht und dann eine kurze Selbstrefelktion?

9.1 Anna Bolder

Anna

9.2 Vera Brockmeyer

Vera

9.3 Britta Boerner

Britta

9.4 Laura Anger

Laura

Working Ours: 97

Written Sections: 2, 2.2, 4 and 4.3 (subsections included)

Responsibilities:

Besides mandatory tasks, like for example doing research on specific problems and helping with the project management, I was the head of far range interactions. It should be obvious that I put my main effort into the various interaction methods of the *Interaction Lab*. As it turned out it was not wise to separate between the interaction methods into two sections and letting two people (B. Boerner and me) work on each part. This is caused by the overlap of the different interactions. For example the *Wand Grab* (compare section 4.3.3), which is a CR interaction is very similar to the FR interactions *Raycast* (compare section 4.3.4) and *Extendable Raycast* (compare section 4.3.5).
To cut a long story short, I implemented the FR as well as the CR interactions together with B. Boerner. A brief description of all completed, as well as one unfinished, methods can be found in section 4.3.
Together with Anna Bolder it I integrated the implemented interactions into the actual scenes (compare section 4.1) of the *Interaction Lab*. We also took care of the visualisation of the tasks for the user. Therefore we created a display on the wall of the supermarket as well as next to the controller. On both of the displays the corresponding tasks are presented for the user.
I was also involved in creating the project video and all images for this documentation.

Self-Reflexion:

All in all I felt like the project run smoothly. Of course I was confronted with unknown terrain, as I never had implemented interactions of any kind. Maybe this was why I first started to create one script per interaction method. In the end of the project I noticed, that both Raycast methods and the *Wand Grab* work after the same rules, so I combined them in one script. If I had done that earlier I could have avoided a lot of trouble for me and Anna Bolder as we needed to integrate them in to all scenes, separately first.

Literatur

[1] R. J. Adams and B. Hannaford. Stable haptic interaction with virtual environments. *IEEE Transactions on Robotics and Automation*, 15(3):465–474, Jun 1999.

[2] Doug A. Bowman and Larry F. Hodges. An evaluation of techniques for grabbing and manipulating remote objects in immersive virtual environments. In *Proceedings of the 1997 Symposium on Interactive 3D Graphics*, I3D '97, pages 35–ff., New York, NY, USA, 1997. ACM.

[3] Doc-Ok.org. Lighthouse tracking examined. http://doc-ok.org/?p=1478. Aufgerufen: 30. März 2017.

[4] Unity Documentation. Boxcollider. https://docs.unity3d.com/ScriptReference/BoxCollider.html. Aufgerufen: 12. July 2017.

[5] Unity Documentation. Ray. https://docs.unity3d.com/ScriptReference/Ray.html. Aufgerufen: 16. July 2017.

[6] Unity Documentation. Spherecollider. https://docs.unity3d.com/ScriptReference/SphereCollider.html. Aufgerufen: 16. July 2017.

[7] Unity Documentation. Transform.setparent. https://docs.unity3d.com/ScriptReference/Transform.SetParent.html. Aufgerufen: 16. July 2017.

[8] Mikael Eriksson. Reaching out to grasp in virtual reality: A qualitative usability evaluation of interaction techniques for selection and manipulation in a vr game, 2016.

[9] John Haas. *A History of the Unity Game Engine*. PhD thesis, Worcester Polytechnic Institute.

[10] HTC. Htc vive. https://www.vive.com/. Aufgerufen: 30. November 2016.

[11] HTC. HTC Vive – Für Vive geeignete Computer. https://www.vive.com/de/ready/. Aufgerufen: 18. März 2017.

[12] Jason Jerald. *The VR Book: Human-Centered Design for Virtual Reality*. Association for Computing Machinery & Morgan und Claypool, New York, NY, USA, 2016.

[13] Sangyoon Lee, Jinseok Seo, Gerard J Kim, and Chan-Mo Park. Evaluation of pointing techniques for ray casting selection in virtual environments. In *Third international conference on virtual reality and its application in industry*, volume 4756, pages 38–44, 2003.

[14] Jonathan Lin and Jürgen P Schulze. Towards naturally grabbing and moving objects in vr. *Electronic Imaging*, 2016(4):1–6, 2016.

[15] Microsoft. Introducing Visual Studio. https://msdn.microsoft.com/en-us/library/fx6bk1f4(v=vs.90).aspx. Aufgerufen: 18. März 2017.

[16] Mobile World Congress. Mobile World Congress. https://www.mobileworldcongress.com/. Aufgerufen: 30. November 2016.

[17] IVAN POUPYREV and TADAO ICHIKAWA. Manipulating objects in virtual worlds: Categorization and empirical evaluation of interaction techniques. *Journal of Visual Languages & Computing*, 10(1):19 – 35, 1999.

[18] Steam. Steam VR Internetauftritt. http://store.steampowered.com/steamvr?l=german. Aufgerufen: 26. März 2017.

[19] Technische Hochschule Chemnitz. Human-Machine Interaction Lab. https://www.tu-chemnitz.de/mb/ArbeitsWiss/forschung/labore/human_machine_interaction_lab. Visted: 28. March 2017.

[20] Unity Technologies. Unity. https://unity3d.com/de. Aufgerufen: 8. März 2017.

[21] Unity Technologies. Unity – Multiplatform. https://unity3d.com/unity/multiplatform. Aufgerufen: 14. März 2017.

[22] Unity Technologies. Unity – Public Relations. https://unity3d.com/public-relations. Aufgerufen: 14. März 2017.

[23] Unity Technologies. Unity – VR Overview. https://unity3d.com/de/learn/tutorials/topics/virtual-reality/vr-overview. Aufgerufen: 14. März 2017.

[24] Valve. Valve Software. http://www.valvesoftware.com/. Aufgerufen: 30. November 2016.