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

1.1 Motivation

Vera

1.2 Usage Context

Vera

1.3 Project Goal

Vera

2 State of the Art

Laura

In the following sections the 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

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

3.1 Hardware

Britta

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

@Britta: Vielleicht möchtest du das nur auf Englisch übersetzen?
Die *HTC Vive* ist ein Head-Mounted Display, welches von *HTC* in Kooperation mit *Valve* [19] produziert wird. Vorgestellt wurde dieses am 1. März 2015 im Vorfeld des *Mobile World Congress* [13].
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 [9]. 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

CGPC6	**Beschreibung**
Prozessor	Intel Core i7 6700 CPU @ $4 \times 3.4 - 4.0$ 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 [10].

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 [8] und wird von *Unity Technologies* [15] 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 [17].

Unity bietet eine sehr breite Plattformunterstützung [16] und erlaubt ebenso die Entwicklung für Head-Mounted Displays, wie etwa die *Oculus Rift* [18] oder auch die in diesem Projekt verwendete *HTC Vive* [18].

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 [12] 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 [14] 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 4.1.1), 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

4.1.1 Learningroom

4.1.2 Supermarket

4.2 Controller Menu

Anna

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 [11]. 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 4.1.1), 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 **??** PICTURE). 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.

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 PICTURE.

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 PICTURE. 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.

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 PICTURE 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.

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.

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

5 Evaluation

Britta

5.1 Tasks

Vera oder Anna

6 Project Management

Vera

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

Dieses Kapitel beschreibt die Planung und das Management des Projektes *MArC*. Es ist in die einzelnen Projektphasen gegliedert, welche jeweils die wichtigsten Punkte der entsprechenden Phasen enthalten.

6.1 Project Definition

Vera

6.1.1 ??

Vera

6.1.2 ??

Vera

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

Self-Reflexion

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

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

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

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

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

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

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

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

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

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

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

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