Toolchain für 3D-Objektmanipulation in VR-Trainingsumgebungen

Niklas Gerwens und Jonathan Becker
niklas.gerwens@haw-hamburg.de
jonathan.becker@haw-hamburg.de

Hochschule für Angewandte Wissenschaften Hamburg, Deutschland
http://www.besteseitederwelt.de/

Zusammenfassung. In dieser Ausarbeitung geht es um den Aufbau einer Toolchain zur Unterstützung von Entwicklern bei der Gestaltung von Objektmanipulation in VR-Trainingsszenarien. Hierfür werden zuerst notwendige Grundlagen erläutert und anschließend die zwei Komponenten der Toolchain vorgestellt. Die erste Komponente beinhaltet ein Framework zur uniformen Gestaltung von Interaktionen, die zweite eine Plattform zur Integration verschiedener Eingabegeräte. Abschließend wird eine Zusammenfassung über den aktuellen Zustand der Toolchain und ein Ausblick auf weiterführende Projekte gegeben.

Schlüsselwörter: Virtual Reality · Mensch-Computer-Interaktion · 3D-Interaktionsdesign · VR-Training · 3D-Objektmanipulation · Toolchain

1 Einleitung

Um in Virtual Reality den virtuellen Raum zu beeinflussen sind Interaktionen notwendig. Diese sind von der ersten Umsetzung bis zur heutigen Zeit vielfältig und zahlreich geworden. Sie bestimmen die Art, mit der das System beeinflusst und gesteuert wird. Nicht alle Möglichkeiten sind dabei für jedes Szenario geeignet. Im Bereich von Schulung und Training stehen die zu vermittelnden Konzepte im Mittelpunkt, für die es gilt die Interaktionen zu optimieren. Das Ziel der virtuellen Welt ist Möglichkeiten zum Ausprobieren und Probehandeln anzubieten, ohne dass vorher komplexe virtuelle Interaktionstechniken erlernt werden müssen. Wichtig sind die Übertragbarkeit in die reale Welt, der Komfort der Bedienung, die Erlernbarkeit und die Kosten. Somit ist der Ansatz des Ultimate Displays [1] nicht zielführend, da es weder um die vollständige Immersion noch um die perfekte Abbildung der Realität geht. Vielmehr steht der richtige Abstraktionsgrad im Fokus, um Nutzer beim Lernen zu unterstützen und die Kosten möglichst gering zu halten.
Für diese Problemstellung im Schulungskontext gibt es noch keine etablierten Vorgehensweisen und nur wenig Lösungsansätze. Deswegen wird in diesem Grundprojekt eine Toolchain mit zwei Komponenten entwickelt, um VR-Entwickler bei der Umsetzung von objektspezifischen Interaktionen innerhalb dieses Kontextes zu unterstützen.

Die erste Komponente beinhaltet die Umsetzung eines Frameworks zur Erstellung von uniformen Interaktionen. Die Interaktionen, die so ermöglicht werden, sollen sich orthogonal bedienen lassen und verhalten, um eine Übertragung zwischen den Interaktionen zu erlauben. Das Framework soll sich um zusätzliche Interaktionen erweitern lassen.
Die zweite Komponente beschäftigt sich mit der Frage wie Entwickler Hardware zur Interaktion mit Objekten einbinden können. Dazu wird eine Plattform aufgebaut, in der verschiedene Eingabegeräte integriert sind und die um weitere Eingabegeräte und Interaktionstechniken erweitert werden kann.

2 Grundlagen

In diesem Kapitel werden für das Verständnis der folgenden Kapitel notwendige Grundlagen erläutert. Hierfür wurde eine Literaturanalyse durchgeführt und die relevanten Inhalte zusammengefasst. Für eine Vertiefung der hier vorgestellten Recherche wird auf die einzelnen Quellen verwiesen.

2.1 Einleitung - VR / HCI

Das grundlegende Prinzip von Virtual Reality (VR) ist es einem Nutzer eine virtuelle Welt so zu präsentieren, dass dieser möglichst keinen Unterschied zur realen Welt bemerkt. Hierfür wird eine 3D-Welt simuliert und verschiedene Sinneseindrücke durch einen Computer generiert. Während erste Ansätze von VR bereits in den 1960er Jahren entwickelt wurden [1], ist das Thema durch die erhöhte Verfügbarkeit von preiswerter Hardware gerade in den letzten Jahren in den Fokus verschiedener Forschungsrichtungen gerückt, welche sich durch den Einsatz virtueller Umgebungen diverse Vorteile versprechen [8].
Die technische Umsetzung einer VR-Umgebung erfolgt durch ein VR-System. Ein VR-System ist ein Computersystem, welches drei Aufgaben erfüllt: die Erfassung von Informationen und Interaktionen des Nutzers, die Erzeugung von Reizen für den Nutzer sowie die Simulation einer virtuellen 3D-Welt. Hierfür werden spezielle Ein- und Ausgabegeräte genutzt, um verschiedene Interaktionen des Nutzers zu registrieren sowie die virtuelle Welt und das Feedback dieser auf entsprechende Interaktionen zu präsentieren. Zur Bestimmung des gewünschten Feedbacks wird zusätzlich zur Hardware noch ein passendes Verhaltensmodell der virtuellen Welt benötigt, um in der Weltsimulation sowohl Reaktionen der Welt auf die Aktionen des Nutzers als auch Änderungen in der virtuellen Welt, die von Aktionen des Nutzers unabhängig sind, zu generieren [8]. Eine graphische Darstellung der Komponenten eines VR-Systems und ihres Zusammenspiels findet sich in Abbildung 1.
Im Kontext dieses Gesamtsystems interagiert ein Nutzer mit der virtuellen Umgebung, indem er über Eingabegeräte mit dem Computersystem der virtuellen Umgebung kommuniziert. Dieser Ablauf ist abstrakt betrachtet eine Mensch-Computer Interaktion (MCI, engl. Human-Computer Interaction, HCI). Der Forschungsbereich der MCI beschäftigt sich mit dem Design, der Evaluierung und

der Realisierung interaktiver computerbasierter Systeme. Der Fokus liegt hierbei auf der benutzergerechten Gestaltung der Schnittstellen anhand von Erkenntnissen der Informatik, aber auch anderer Gebiete wie der Arbeitswissenschaft, der Psychologie und des Designs [7].
In den letzten Jahrzehnten haben sich hauptsächlich Graphische Benutzerschnittstellen z. B. WIMP-Schnittstellen (Windows, Icons, Menus, Pointing) als ein Paradigma der MCI durchgesetzt. Diese wurden ursprünglich für 2D-Systeme und Aufgaben innerhalb dieser z. B. der Dokumentenverwaltung entwickelt. Die Integration von WIMP-Schnittstellen in eine virtuelle 3D-Umgebung setzt voraus, dass einzelne 3D-Aktionen in mehrere 2D-Aktionen übersetzt werden, welches den Aufwand erheblich erhöht. Dies kann dennoch sinnvoll sein, um den Lernaufwand zu verringern, da Nutzer in der Regel schon weitreichende Kompetenzen mit WIMP-Schnittstellen erworben haben [8].
Im Gegensatz zu den klassischen Benutzerschnittstellen existieren für VR noch keine etablierten Vorgehensweisen und Richtlinien zur Entwicklung von benutzergerechten Schnittstellen. Deswegen ist man darauf angewiesen, für die jeweiligen Aufgaben und Anforderungen der Nutzer Lösungsmöglichkeiten prototypisch zu entwickeln und diese auf ihre Eignung hin zu evaluieren.

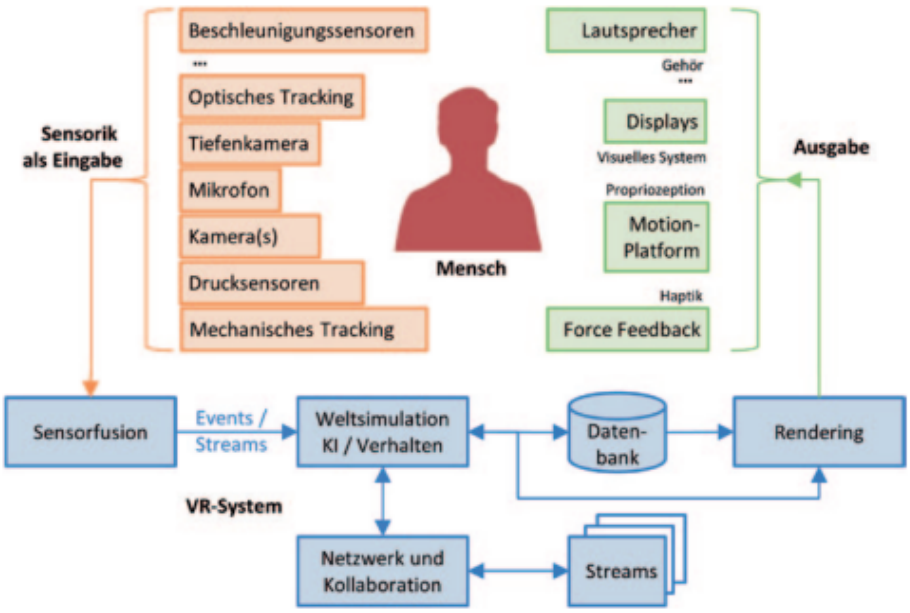

Abb. 1. Graphische Darstellung eines VR-Systems [8]

2.2 VR-Eingabegeräte

Für die technische Umsetzung von Interaktionstechniken in virtuellen Umgebungen wird für die Anforderungen des Anwendungskontextes geeignete Hardware benötigt. Hierbei sind vor allem der Grad der Natürlichkeit, die Präzision, die Anzahl der unterstützten Freiheitsgrade sowie der Lernaufwand des Nutzers Kriterien für die Wahl entsprechender Ein- und Ausgabegeräte. Während sich für die Navigation von 2D-Systemen Maus und Tastatur als Eingabegeräte durchgesetzt haben, gibt es für 3D-Umgebungen keine Standard Hardware. Demnach befindet sich eine Vielzahl an Geräten für die Bewegungssteuerung und die Selektion/Manipulation von Objekten in virtuellen Umgebungen im Umlauf. Da die Manipulation von Objekten im Fokus dieser Arbeit steht, werden aktuelle Kategorien an Eingabegeräten, welche für diese Aufgabe geeignet sind, im Folgenden spezifiziert.
Grundlegend lassen sich Eingabegeräte zur Manipulation von Objekten unterteilen in Zusätzliche Artefakte, Hand-Tracking sowie Hand-unabhängige Ansätze. Zusätzliche Artefakte sind Geräte, die vom Nutzer in der Hand gehalten werden und deren Position/Rotation durch eingebaute Sensoren oder ein externes Tracking System bestimmt werden. Anhand dieser Werte lässt sich eine virtuelle Hand mit sechs Freiheitsgraden abbilden. Des Weiteren können zusätzliche Interaktionen wie Greifaktionen und Skalierungen durch Knöpfe, Joysticks, Drucksensoren oder Touchpads umgesetzt werden. Ebenfalls ist es durch Zusätzliche Artefakte möglich dem Nutzer haptisches Feedback zu geben.
Zusätzliche Artefakte bieten demnach einige Optionen Nutzer-Eingaben zu erfassen, lösen jedoch nicht automatisch Probleme wie passende Abbildungen zwischen Eingabe und Transformation, Rotation sowie Skalierung eines Objekts, Objektmanipulation außerhalb der Reichweite oder fehlende Genauigkeit. Hierfür werden häufig zusätzliche Interaktionstechniken innerhalb des VR-Systems benötigt [15].
Hand-Tracking Ansätze dagegen nutzen meist optische Tracking-Verfahren, um Position und Rotation der Hand sowie einzelner Handelemente zu bestimmen und damit eine Hand-Rekonstruktion in der virtuellen Umgebung zu schaffen. Dies erlaubt dem Nutzer mehr Freiheit, da kein zusätzliches Gerät in der Hand gehalten werden muss. Weiterhin bietet die Möglichkeit einzelne Finger zu tracken ganz neue Interaktionsmöglichkeiten. Hierbei treten jedoch häufig Performance Probleme durch das erschwerte Tracking auf. Auch ist es für über die sechs Freiheitsgrade hinausgehende Interaktionen nötig Gestenerkennung oder andere aufbauende Ansätze zu implementieren [15].
Neben den Ansätzen, die sich mit dem Tracking der Hand oder gehaltener Objekte beschäftigen, existieren noch weitere Hand-unabhängige. So bietet auch Hardware zur Blickverfolgung oder zur Spracherkennung mit entsprechender Implementation die Möglichkeit Objekte in virtuellen Umgebungen zu manipulieren. Bei diesen gestaltet es sich jedoch schwierig Metaphern aus der realen Welt zu transferieren, wodurch ein erhöhter Lernaufwand unter Verlust von Natürlichkeit entsteht. Auch gilt es bei der Blickerkennung das Midas Touch Problem zu lösen [8]. Folglich ist es nicht sehr intuitiv sich bei der Objektmanipulation vollständig

auf eines der Verfahren zu verlassen. Eine Kombination mit anderen Ansätzen zur Ergänzung dieser um eine weitere Aktionsmöglichkeit bietet sich an [15].
Als visuelles Standardausgabegerät für virtuelle Umgebungen haben sich Head-mounted Displays (HMDMs) in den vergangenen Jahren etabliert. Diese sind durch stark gesunkene Preise mittlerweile weithin verfügbar und bieten durch ihre Hardwarespezifikationen ein immersives Erlebnis, welches sich auch positiv auf Lernerfolge auswirkt [14]. Neben HMDs kommen häufig Lautsprecher oder im HMD integrierte Headsets für auditives Feedback zum Einsatz.
Wenn es um hochspezialisierte Interaktionen geht, brauchen diese komplexeres Feedback als ein Standard-Controller bieten kann. In diesen Fällen sind spezialisierte Interfaces notwendig wie in "A Haptic Virtual Reality Training Tool for Paper Conservators" [5], wo ein "haptic Arm" genutzt wird, um das physikalische Feedback zu simulieren. Zu einem ähnlichem Ergebnis kommt "Performance Benefits of High-Fidelity Passive Haptic Feedback in Virtual Reality Training" [12]. Es wird ein Teil eines Golf-Simulators vorgestellt, in dem Schläger und Controller getestet wurden. Die ersten beiden Thesen der Ausarbeitung wurden bestätigt, dass sowohl die Performance als auch die subjektive Wahl der Nutzer für die spezialisierten Interfaces sprechen. Die dritte These, dass die Korrelation zwischen echter Welt und Interface besser sind als im Vergleich zum Controller konnte nicht bestätigt werden. Die Autoren stellen an dieser Stelle aber auch die These auf, dass dieses Ergebnis sich ändern kann, wenn die Simulation realistischer wird z.B. ein Avatar als Repräsentation des Nutzers enthält.

2.3 Interaktionen aufgeschlüsselt

Interaktion bestehen aus verschiedenen Komponenten. In [4] wird eine Taxonomie darüber aufgestellt, siehe Abbildung 2, die Interaktionen in drei Phasen teilt: Selection, Manipulation, Release. Jede dieser Phasen hat untergeordnete Kategorien, die häufige Features enthalten.
Die Phase der Selektion nutzt oft als Input und Feedback die Distanz zum zu Selektierenden Objekt. Die Selektion kann über unterschiedliche Techniken stattfinden, abhängig von den Interaktions Radien. Für entfernte Objekte gibt es Techniken wie WIM (World in Miniatur)[2]. Diese stellt die Umgebung als Miniaturmodell in Reichweite des Nutzers dar. In diesem können die Objekte selektiert werden. Den gleichen Zweck erfüllt die Go-Go Technik [3], die mit einem nicht linearen Mapping der Distanz des Controllers zum Nutzer arbeitet, um entfernte Objekte zu erreichen. Wenn es nicht notwendig ist entfernte Objekte auszuwählen, kann mit direktem Mapping gearbeitet werden. Dieses hat den Vorteil, dass es der Selektion in der realen Welt gleicht. Als Input für die vorgestellten Selektionen dient ein Eingabegerät, welches die Hand in mindestens drei Graden der Freiheit repräsentiert. Dieses Eingabegerät kann auch zusätzlich zum Auslösen der Selektion genutzt werden, wenn dieses Gesten erkennt. Mögliches Feedback ist die Vibration des Eingabegerätes oder der virtuellen Repräsentation von diesem.

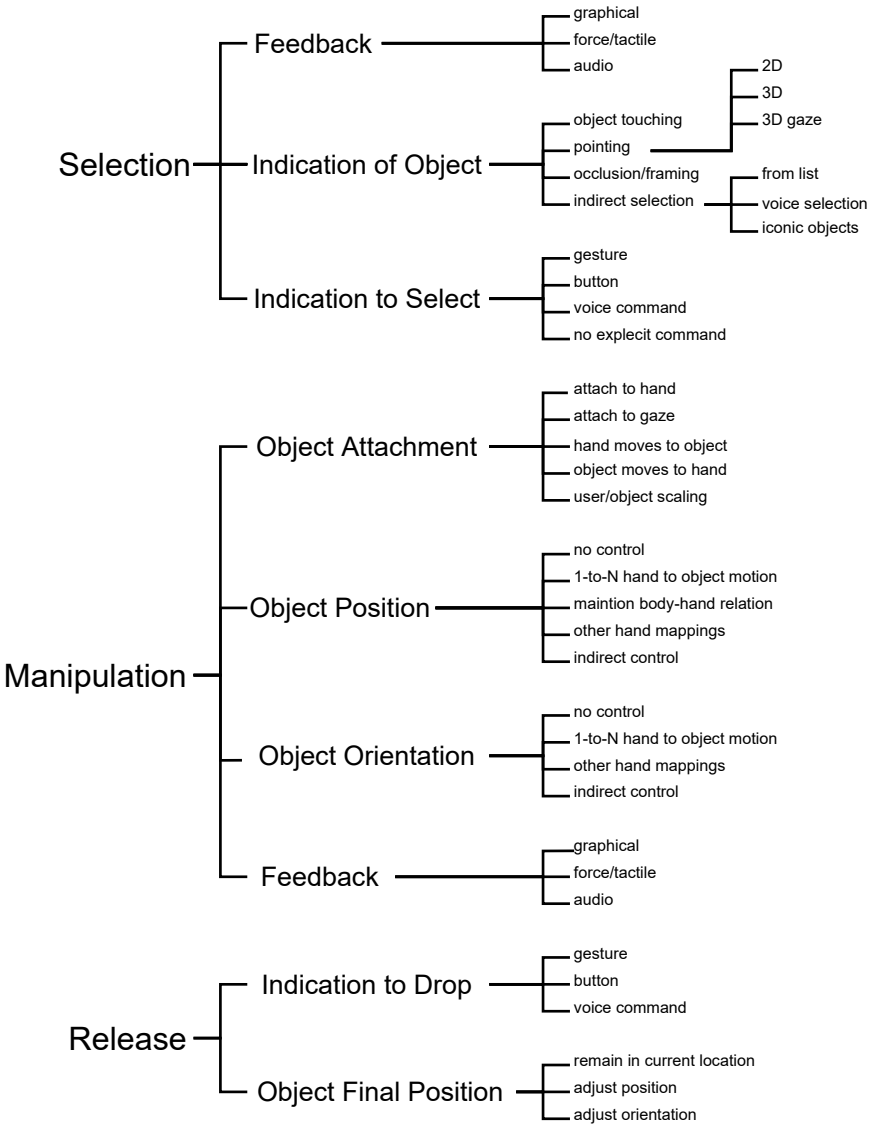

Abb. 2. Taxonomie von Interaktionstechniken nach Figure 1 in [4]

Andere Selektionstechniken wie Raycasting können auch den Blick des Nutzers nutzen, sie sind aber ohne zusätzlichen Input für das Auslösen der Selektion oft nicht nutzbar. Eine Möglichkeit dieses Problem zu lösen ist es die Interaktionen durch das Verweilen des Blickes auszulösen. Bei der Selektion der Techniken kommt es auch auf die benötigte Präzision an, die bei Techniken wie Raycasting mit der Entfernung abnimmt.

WIM hat keine Präzisionsverluste über Distanz und umgeht das Problem der direkten Verdeckung. Aufgrund der Miniaturisierung gibt es aber Probleme, wenn es viele kleine Interaktionsobjekte auf engem Raum gibt.
Viele der Selektionstechniken können direkt zur Manipulation übergehen, z.b. wenn ein Objekt gegriffen wird, um es dann zu bewegen, indem das Mapping des Eingabegerätes direkt auf das Objekt übertragen wird. In vielen Fällen ist dieses nicht möglich wie bei der Flashlight Selektion. Hier muss für alle Interaktionen, die mehr als Selektion benötigen, auf einen sekundären Input zurückgegriffen werden.
Viele Selektionstechniken nutzen ein 6dof Controller, der Gesten erkennen kann oder mit Knöpfen ausgestattet ist. Diese Art von Controller erlaubt es den Körper stärker in die VR einzubeziehen. So wird ein direktes Mapping von Rotation und Position für die Interaktion ermöglicht. Andere Techniken wie die Handlebar Technik erlauben zusätzlich zu der 6dof Manipulation auch noch das Skalieren, indem das Objekt zwischen den beiden Controllern auf der Stange gehalten wird. Während das Skalieren keine natürliche Interaktion ist, ist diese aber durch Tablets und Smartphones etabliert. Wenn das Mapping nicht verwandte Eigenschaften verknüpft, entstehen unnatürliche Techniken. Manche von diesen lassen sich für eine spezifische Anwendung etablieren, nicht ungleich den Shortcuts in Programmen. Wenn die notwendigen Interaktionen zu kompliziert werden, z.b. wenn komplexe Eigenschaften eines Objekts manipuliert werden müssen, kommen UIs (User Interfaces) zum Einsatz. Auch in diesem Fall wird meist ein 6dof Controller genutzt, da andere nur sehr eingeschränkte Interaktionen im 3D Raum zulassen. Das Feedback ist oft das Ziel der Manipulation und so von dem Ziel der Interaktion abhängig.
Das Loslassen eines Interaktionsobjekts wird durch das Beenden der Aktion zur Selektion oder eine erneute Aktion ausgelöst. Das primäre Feedback besteht darin, dass das Interaktionsobjekt nicht länger auf Input reagiert. Zusätzlich gibt es die Möglichkeit, dass eine weitere direkte Reaktion erfolgt und/oder das Objekt wieder seinen intrinsischen Eigenschaften folgt.

2.4 Frameworks: Möglichkeiten und Herausforderungen

Ein Framework für die Nutzung im Zusammenhang mit einer Gameengine zu erstellen unterliegt eigenen An- und Herausforderungen. Der erste Punkt sind die angestrebten Nutzergruppen, hier gibt es zwei extreme: Jene, die nur mit UIs arbeiten und die anderen, welche die Schnittstellen nutzen, um auf diesen ihre eigenen Mechanismen aufzubauen. Im Normalfall wird aber eine Mischung verwendet [6], da vorhandene Strukturen die Arbeit beschleunigen aber jedes Szenario seine Spezialitäten hat, die eine Erweiterung oder Anpassung des Frameworks erfordern.
Ein Vorteil für die Erstellung ist, dass viele Gameengines auf Erweiterungen ausgelegt sind. So sind solche Frameworks stark Plattformspezifisch, können aber die bereitgestellten Features der Gameengine nutzen. Grafische Oberflächen der Gameengine erlauben es das Framework anwenderfreundlich zu gestalten, da vertraute Oberflächen genutzt werden können.

Am Ende ist ein solches Framework Aufgabenspezifisch. Die Aufgaben, die ein Framework übernimmt, sollten modular gestaltet sein. Ein allumfassendes Framework macht es schwierig dieses mit anderen zu verbinden, da die Schnittstellen zu groß und unübersichtlich werden. Des Weiteren wird die Wiederverwendung in anderen Kombinationen und Projekten erschwert, ebenso die Einarbeitung wie auch das chery picking von Features.

2.5 SteamVR und Unity Physics

Um das System unabhängiger von Ein- und Ausgabegeräten zu gestalten, kann SteamVR verwendet werden. Grundlegend fungiert SteamVR als Adapter zwischen der spezifischen VR-Hardware und der Entwicklungsumgebung. Für den Zugriff auf die gekapselte Hardware bietet SteamVR verschiedene Plugins für gängige Entwicklungsumgebungen an.
Das SteamVR Plugin für Unity ermöglicht die Einbindung verschiedener HMDs und Controller durch eine Schnittstelle mit der sich die Geräte verbinden können und Entwickler deren Werte über Hardware unabhängige Befehle auslesen können. Des Weiteren lädt SteamVR automatisch Modelle für die jeweiligen Geräte in die virtuelle Umgebung und aktualisiert diese entsprechend der Bewegung im realen Raum. Neben der Einbindung von Hardware bietet SteamVR auch ein Interaktionssystem, welches vorgefertigte Skripte und Prefabs zur Gestaltung von Interaktionen mit Objekten in der virtuellen Welt enthält [17].
Dabei macht SteamVR Gebrauch von dem Unity internen Physik System. Dieses simuliert physikalische Kräfte innerhalb der virtuellen Welt. So wird sichergestellt, dass Objekte korrekt beschleunigen und auf Kollisionen, Schwerkraft und andere Kräfte reagieren. Wichtige Komponenten hierbei sind Collider und Rigidbodies. Collider definieren die Form des Objekts, welche zur Berechnung von physikalischer Kollision genutzt wird. Während ein Rigidbody ein Objekt in Unity überhaupt ermöglicht von physikalischen Kräften beeinflusst zu werden und Optionen zur Gewichtung dieser bereitstellt [18].

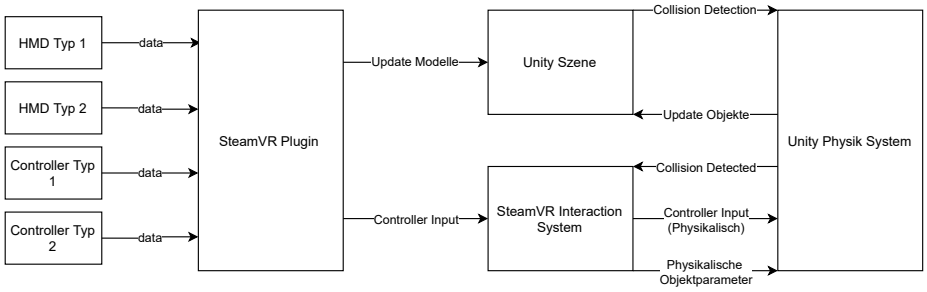

Abb. 3. Zusammenspiel von SteamVR und dem Unity Physik System

2.6 Training und Didaktik

Virtual Reality bietet viele Möglichkeiten z.B. eine erhöhte Präsenz im Geschehen, die dazu genutzt werden können Körper-Interaktionen im Lernkontext zu verankern [10]. Dieses ist besonders bei gefährlichen Szenarien oder schwer Nachzustellenden hilfreich. Besonders bei diesen ist der Platz und Skalierungsfaktor hervorstechend [13]. Auch nützlich ist es, dass es möglich ist digitale Szenarien zu pausieren, neu zu starten, zu sichern oder an beliebiger Stelle einzusteigen.
Limitiert wird der Nutzen durch die Replikationsgüte des Szenarios und die davon abhängige Übertragung in die reale Welt. Diese korreliert stark mit den Interaktionen, je höher die geforderte Präzision und je komplexer die verwendeten Werkzeuge, desto komplexer und aufwändiger wird die Erstellung eines funktionierenden Szenarios. Weiterhin wird der Einsatz von VR in Lernumgebungen erschwert durch den Mangel an etablierten Standards und die teilweise sehr kontextspezifischen Anforderungen der Szenarien.
Einige Beispielszenarien und erste Erkenntnisse zum Thema VR in Lernumgebungen finden sich in den Arbeiten von Bronsch [11] und Gavish et al. [9].

3 Interaktions Framework

Das Ziel dieses Kapitels ist ein Framework, für das einfache Erstellen und Benutzen von Interaktionen. Dieses Framework soll zur Erstellung von orthogonalen Interaktionen dienen. Die Features sollen dem Nutzer des Frameworks und dem Endbenutzer helfen. Hierfür gibt es in der Entwicklung folgende Ziele:

- Es muss einfach zu erweitern sein.
- Die einzelnen Elemente sollen Modular sein.
- Möglichst große Unabhängigkeit von Hardware.
- Die Interaktionen sollen möglichst einheitlich zu bedienen sein.

Die Entwicklung wird in die folgenden Kapitel aufgeteilt: Das erste Kapitel spezifiziert das Gesamtkonzept und stellt die Annahmen und Anforderungen auf. Im folgendem Kapitel werden Interaktionen identifiziert, zerlegt und kategorisiert. Das Folgende Kapitel bereitet den theoretischen Unterbau für die Implementation vor. Dazu wird die Vererbungsstruktur und die Schnittstellen vorbereitet. Im viertem Kapitel wird die Implementation von einigen Beispielklassen beschrieben und ihre Potenzialitäten beschrieben. Das letzte Kapitel fasst die Implementierung in einem Fazit zusammen und bietet einen Ausblick.

3.1 Ein Framework für Interaktionen in VR

Für ein Interaktions Framework gelten bestimmte Anforderungen. Gleichermaßen gilt es auch Voraussetzungen aufzustellen, um das Framework abzugrenzen.

Annahmen und Voraussetzungen Die verwendete Toolchain besteht aus der Gameengine Unity [1] zusammen mit dem SteamVR Plugin für Virtual Reality. Des Weiteren werden folgende Annahmen für den Input aufgestellt: Er besteht aus 6dof (Sechs Grade der Freiheit, drei Axen für Position und drei für Rotation), unabhängig davon ob diese virtuell sind oder direkt von der Hardware kommen. Diese Funktion wird bereits von dem SteamVR Plugin zur Verfügung gestellt. Die angenommenen Anwender des Frameworks sind zwischen UI-Nutzern und Programmierern, da beide Extreme den Nutzen einschränken. Für den Endbenutzer sollen mit der Standardimplementation keine besonderen Anforderungen gelten.

Bedingungen an das Framework stehen auf der anderen Seite. Es soll durch einen modularen Ansatz einfach zu erweitern sein. Des Weiterem soll eine größtmögliche Unabhängigkeit von der Hardware und Unity Version erreicht werden, um eine Weiter- und Wiederverwendung der Module zu ermöglichen. Performance ist für VR besonders wichtig, sowie die Verwendung einer Vielzahl von Interaktionen in einem Szenario. Deshalb gilt es den Footprint für Module und Interaktion klein zu halten.

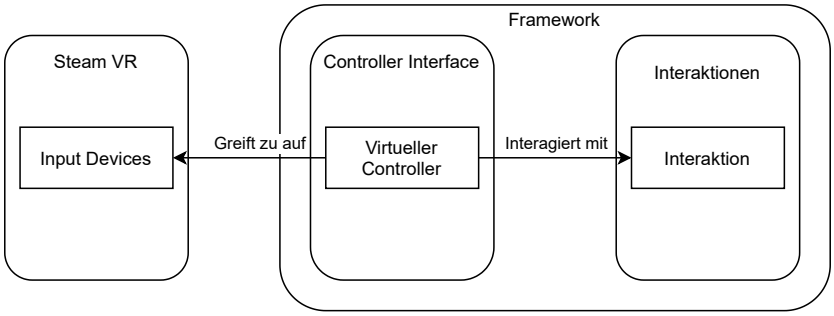

Abb. 4. Übergreifende Komponenten und ihr Zusammenspiel

Weitere Eigenschaften beschreiben die Nutzung: Die Interaktionen müssen einfach einzubinden sein und in der VR einfach zu bedienen. Dieses soll durch entsprechende UI-Elemente unterstützt werden. Die Interaktionen selber sollen dem Ziel der Arbeit nach orthogonal ohne Lernaufwand in der VR zu bedienen sein. Dieses setzt strukturierten Input und Feedback für die Interaktionen voraus. Hilfreich ist die Angleichung an verbreitetes und bereits bekanntes Verhalten.

[1] Für dieses Projekt wurde Version 2017.4 mit der Programmiersprache c# verwendet

Wie Läuft eine Interaktion ab Dieses wird hier dargestellt. Dazu werden folgende vier Interaktionen als Beispiele genutzt:

- Ein Ball
- Ein Lichtschalter
- Eine Tür, die von selber zufällt
- Eine Schraube

Für Interaktionen lassen sich minimale Ablaufzustände definieren, siehe Abbildung 5. Interaktionen befinden sich am Anfang in einem Idle Zustand. Viele Objekte verändern ihre Eigenschaften hier nicht, andere werden von Physik, der Ball, oder durch Skripte, die von alleine zufallende Tür, verändert. In diesem Zustand lassen sich Interaktionen aktivieren, der Moment, in dem sie mit dem Controller selektiert und ausgelöst werden. An dieser Stelle kann es bereits eine erste Reaktion geben, z.B. der Lichtschalter. Auch können initiale Parameter berechnet werden, um z.B. die Bewegung des Controllers zu verfolgen. Die Interaktion kann hier auch enden und wieder in den Idle Zustand übergehen, ein Lichtschalter bei dem das Licht beim Drücken zwischen an und aus wechselt. Wenn dieses nicht der Fall ist, geht sie in den Zustand Halten über, hier kann die Interaktion auf die 6dof des Controllers reagieren, der Ball und die Tür der Hand folgen und die Schraube auf Drehungen antworten. Das Ende der Interaktion wird mit dem Release erreicht, bei dem die Interaktion deaktiviert wird. An dieser Stelle kehrt die Interaktion wieder in den Idle Zustand zurück, vorher kann aber noch eine direkte Reaktion folgen, z.B. beim Lichtschalter wenn dieser wieder zurückspringt. Im Idle Zustand folgt das Interaktionsobjekt wieder seinen intrinsischen Eigenschaften, sei es das Fallen des Balls oder dass die Tür sich wieder schließt.

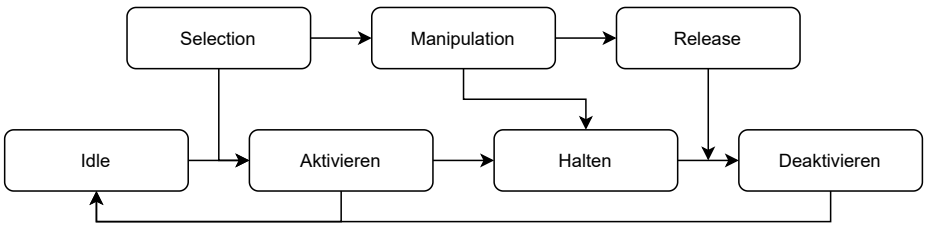

Abb. 5. Ablauf einer Interaktion nach [4] und die Zustände der Interaktion

Wenn die Verhaltensweisen konstant umgesetzt werden, hilft es dabei die Interaktionen gleichmäßig wirken zu lassen. Diese Orthogonalität hilft dem Endbenutzer von bekannten Interaktionen auf neue zu schließen.
Wenn Interaktionen sich an reale Vorbilder halten, trifft dieses besonders zu. Interaktionen wie schrauben, ziehen, drücken und hebeln lassen sich so nach dem Erlernen der Selektionsgeste ähnlich bedienen.

Feedback ist Teil von Interaktion Damit diese Muster erkannt werden können, ist ein ausreichendes Feedback notwendig, um die Interaktion zu bewerten. Ein Bespiel ist der Lichtschalter, wenn dieser mit jeder Interaktion den Zustand wechselt, ist die physische Änderung zu erkennen. Wenn dieser aber beim Loslassen immer wieder in seine Ausgangsposition zurückkehrt, ist dieses nicht mehr so deutlich. Beim Lichtschalter ist dieses im Normalfall kein Problem, da er das Licht als weiteres Feedback ein und ausschaltet. Dieses Feedback ist ein wichtiger Teil der Interaktion, um die Funktionsweise der Interaktion zu verdeutlichen.

Welche Interaktionen lassen sich abbilden Die Anzahl möglicher Interaktionen ist groß, denn es kann einerseits die reale Welt abgebildet werden und andererseits die fantastische Seite der virtuellen Welt. Nicht alles ist dabei umsetzbar ober möglich. Viele Interaktionen sind zu feinteilig, um diese mit den herkömmlichen Systemen umzusetzen, z.B. eine Tastatur, die ohne präzises Fingertracking und Feedback nicht gut zu bedienen ist. Ebenso sehr kleine Interaktionsobjekte stellen ein Problem da, wie ein Haufen Murmeln. Die Controller haben hier Probleme mit Verdeckung und Präzision, was diese Interaktionen unhandlich und ungenau macht. Wenn Interaktionen mit Kleinteilen notwendig sind, wird eine Abstraktion oder andere Technik benötigt.

3.2 Klassifizierung von verwandten Interaktionen

Es gibt viele unterschiedliche Interaktionsobjekte: einmal drücken, zweimal drücken, ziehen, in der Hand halten, schieben, schrauben, drehen, Hebel, Türen, und weitere. Diese lassen sich anhand unterschiedlicher Parameter in Gruppen teilen, aus denen sich Interaktionen zusammensetzen.

Die Bedienung von Interaktionen bestimmt die Art des benötigten Inputs. Schrauben und Drehknöpfe nutzen die Rotation des Controllers. Schalter nutzen die Aktivierung und Deaktivierung. Schieberegler und Hebel nutzen die Position des Controllers. Gehaltene Gegenstände benötigen sowohl Rotation als auch Position.

Das Feedback von Interaktionen ist stark vom Input abhängig und in externes Feedback, welches nicht in der Interaktion selbst enthalten ist, und internes Feedback aufgeteilt. Internes Feedback ist die Positionsänderung des Balls, des Lichtschalters, der Tür oder der Schraube während der Interaktion. Externes Feedback ist das An- und Ausschalten des Lichts sowie nach dem Loslassen die Positionsänderung des Balls oder der Tür.

Die Zustände der Interaktionen ergeben sich aus ihren intrinsischem Feedback, da dieses eine persistente Zustandsänderung ist. Der Ball hat sowohl Rotation als auch Position, da er frei zu bewegen ist. Ein Schalter hat lediglich An und Aus als Eigenschaften, sowie eine feste Position und Rotation für beide Zustände.

Eine Tür hat eine Rotationsachse, ihre äußeren Limits und eine Mittelposition, wenn diese in beide Richtungen aufgeht. Eine Schraube hat auch Rotation und Position. Schieben und Ziehen haben nur eine Positionskomponente, die aber entlang eines festgelegten Wegs verändert wird.

Reduktion auf die gemeinsamen Komponenten ist wichtig für die Modularität. Dieses erlaubt es bei der Vererbung minimale Klassen zu implementieren. Die Zustände eignen sich hierfür als Grundlage, da diese die Basis der Interaktionen sind. Dieses nutzt aus, dass Interaktionen, die sich ähnlich verhalten, oft ähnliche Daten benötigen.

Events: Beeinflussung der Umgebung Viele Interaktionen dienen zur Beeinflussung anderer Elemente, wie der Lichtschalter. Für jedes Zusammenspiel zwischen Objekten eine Überinteraktion zu programmieren, die Änderungen überwacht und entsprechend reagiert, ist nicht skalierungsfähig.
Um dieses Problem zu lösen werden Events eingesetzt, da sie zu den Interaktionen hinzugefügt werden können und von der Interaktion selber ausgelöst werden. Dies macht es einfacher andere Elemente wie Sound, Licht und Schlösser zu verbinden und zu Gesamtsystemen zu kombinieren.

3.3 Vielfältige Systeme und die Planung ihrer Umsetzung

Aus den Bestandteilen lassen sich die Anforderungen an die Implementierung der Interaktionen ableiten. Der wichtigste Teil des Frameworks ist die Konfiguration der Interaktionen und damit die Datenstruktur, welche diese Informationen speichert. Um die Interaktionen klein und eigenständig zu halten eignet sich ein Orchestrator ähnliches Pattern.

Die Konfiguration Mit den richtigen Möglichkeiten zur Konfiguration kann jede Interaktion für eine Vielzahl unterschiedlicher Objekte verwendet werden.
Ein Ball lässt sich vielleicht beliebig greifen, aber ein Werkzeug soll immer auf die gleiche Weise gegriffen werden. Auch ist es ein Unterschied, ob es möglich ist den Gegenstand in die andere Hand zu übergeben ohne diesen loszulassen, Abbildung 15.
Ein Lichtschalter hingegen kann seinen Zustand sowohl beim Drücken als auch beim Loslassen ändern: An, aus, Zustand wechseln oder keine Änderung. Wichtig ist auch, ob der Zustand zwei Mal in einer Interaktion geändert werden kann, Abbildung 14.
Ein Hebel ist vergleichbar mit einer Tür oder einem Steuerrad. Alle haben zwei äußere Limits und eventuell eine mittlere Ausgangsposition. Die Tür ist selbst schließend und ein Steuerrad lässt sich mehr als 360 Grad in beide Richtungen drehen, Abbildung 17. Wenn die Konfiguration geschickt aufgebaut ist lassen sich all diese Eigenschaften in einer Konfiguration unterbringen.

Die Datenstruktur der Schraube hat Ähnlichkeiten mit der des Hebels, die Rotation und die Limits sind bei beiden vorhanden, nur eine Positionsänderung gibt es beim Schrauben zusätzlich, Abbildung 16.
Die Zustände einer Interaktion sind wichtig, denn nicht immer soll jede Interaktion zur Verfügung stehen und Interaktionen die aktive Elemente haben sollen nicht permanent Updates benötigen. Hierzu kann jede Interaktion einen von fünf Zuständen annehmen:

- Disabeld: Die Interaktion lässt sich nicht selektieren oder ändern.
- Inactive: Die Interaktion wartet darauf gegriffen zu werden.
- Grabed: Die Interaktion ist gerade gegriffen und reagiert entsprechend.
- Running: Die Interaktion ist nicht gegriffen, hat aber eine im Moment aktive Komponente.
- Service: Die Interaktion reagiert nur auf manuellen Input und sendet keine Events.

Die UI in Unity zur Konfiguration von Skripten ist der Inspektor, der alle konfigurierbaren Felder des ausgewählten Objekts enthält. Es ist möglich diese Felder zu beschriften und mit Beschreibungen zu versehen. Zur weiteren Unterstützung ist es hilfreich Felder automatisch auf Vollständigkeit und Korrektheit zu prüfen. Diese Fehler sind sonst nur zur Laufzeit zu finden, ohne diese zuordnen zu können. Auf diese Art wird verhindert, dass mit Hilfe von Exeptions der Programmcode nach der Ursache durchsucht werden muss.
Ein weiterer Punkt der UI sind die ContextMenuItems, bei diesen handelt es sich um Methoden, die über die UI aufzurufen sind. Die Limitierung ist, dass diese keine Parameter enthalten können. Sie können aber bei der Konfiguration helfen, um Zustände herzustellen oder Input zu simulieren.

Die Datenstruktur ist in allen Interaktionsobjekten gleich aufgebaut und so angeordnet, dass sie von oben nach unten gelesen werden kann. Damit das Framework auch ohne lange Einarbeitung genutzt werden kann, ist defensives Programmieren und automatisches Prüfen auf ihre Korrektheit der Parameter notwendig.

Die Vererbung hat eine Voraussetzung, dass alle Interaktionsobjekte von der Unityspezifischen Klasse MonoBehavior erben müssen. In der Abbildung 6 ist die entsprechende Vererbungsstruktur zu sehen.

OBJWithState enthält die Verwaltung von Zuständen und Methoden, um diese von Interaktionsobjekten (IOJ) gebündelt abzufragen. Damit ermöglicht diese Klasse eine übergreifende Strukturierung und Kontrolle.

IIOJGrabable ist ein Interface, das zwei Methoden enthält, um Interaktion zu ermöglichen. Diese stellen die Schnittstelle zu den Controllern dar und erlauben die Selektion und Deselektion von Interaktionsobjekten.

```
public interface IIOJGrabable
{
    void Grab(ICControllerGrab controller);
    void Release(ICControllerGrab controller);
}
```

OBJWithNameConvention ist ein Interface, das erzwingt, dass die Methode bool OBJNameConventionIsFullfild() implementiert wird. So ist es möglich zu überprüfen, ob die Interaktionsobjekte einem vorgegebenen Namensschema folgen.

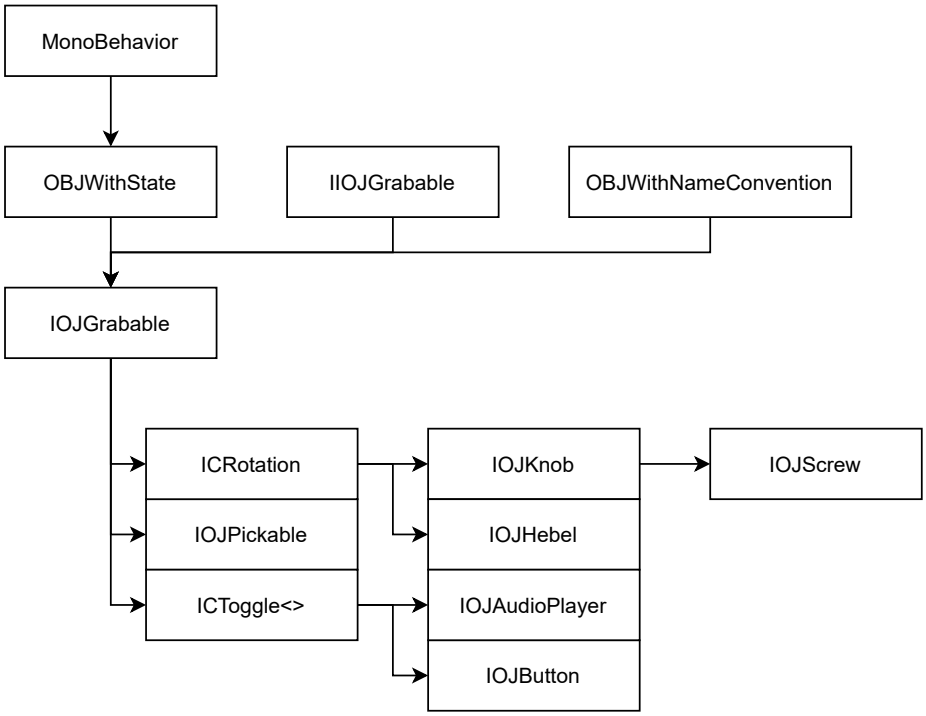

Abb. 6. Vererbung von Interaktionstechniken

Das Methodenschema dient zur Vermeidung von Problemen mit der Vererbung. Es erleichtert den Überblick darüber, welche Methoden geerbt, überschrieben oder neu sind. Wichtig dabei ist, dass wenn die "Update Methode" definiert ist, diese von Unity in jedem Frame ausgeführt wird. Reaktiv zu arbeiten erlaubt es diese Methode zu meiden. Eine Übersicht findet sich im Anhang A.

Vor und Nach Diese Aufteilung erlaubt es spezifische Interaktionen kompatibel mit den Vorhandenen zu implementieren. Je weiter unten in der Vererbung eine Interaktion sich befindet, desto mächtiger wird sie, aber auch anfälliger und komplexer. Auch führt dieser Zusammenhang dazu, dass die UI im Inspektor nach häufiger Vererbung überladen und unübersichtlich wird. Es muss also, unter Berücksichtigung der Nachteile von Funktionsdopplung und fehlender Übersicht, abgewogen werden, wann es sinnvoller ist weiter oben in der Vererbung einzusteigen.

3.4 Aufbau des Interaktionssystems

In der Umsetzung gibt es immer Abweichungen und nicht erfüllte Anforderungen. Im Folgendem werden die erwähnten Beispielklassen anhand dieser Kategorien evaluiert und vorgestellt. Dazu gehören: Erweiterbarkeit, Funktionalität, Events, Nutzung für Endbenutzer und Programmierer.

IOJGrabable ist die erste interaktive Klasse. Sie bietet Trigger für das Selektieren und Deselektieren eines Colliders ohne weitere Interaktion oder Reaktion. Sie ist die Grundlage für weitere Interaktionen. Sie enthält ein Feld, um den Letzten bei dieser Interaktion aktiv geworden Controller zu speichern und bei diesem einzustellen, ob es erlaubt ist die Interaktion direkt von einer in die andere Hand zu übergeben. Dieses Feld ist dann von Interesse, wenn ein unbeabsichtigtes Wechseln der Hand an einer Interaktion vermieden werden soll. Die Events erlauben die Überwachung einer Interaktion, auch wenn diese nicht aktiv ist, Abbildung 7, was für die Auswertung von Interaktionsketten wichtig ist.
Zur Konfiguration stehen entsprechend zwei Felder, für den Status und Allow Hand Swap, zur Verfügung, Abbildung 13.

ICRotation bietet selber keine interaktiven Elemente. Sie enthält aber Datenfelder, um die Rotation eines Objekts zu managen, Abbildung 17. Dazu gehören die Rotationsachse und die Limits für die Rotation sowie die Punkte, ab denen diese als erreicht gelten. Ebenso kann aktiviert werden, dass das Interaktionsobjekt mit einer eingestellten Geschwindigkeit in seine Ausgangsposition zurückkehrt. Zusammen mit automatischen Sanity Checks für die eingestellten Werte bietet diese Klasse so eine funktionale Grundlage für Rotationsobjekte.
Spezielle Events dieser Klasse sind für das Überschreiten der äußeren Schranken in beiden Richtungen vorgesehen. Für eine vollständige Interaktion fehlt die Erweiterung um die Berechnung wie die Rotation sich in Abhängigkeit vom Controller ändert und sich die Rotation auf das Objekt auswirkt.

IOJPickable ist für alle frei beweglichen Objekte, Abbildung 15. Sie bietet die Möglichkeit Objekte mit dem Controller zu greifen und dann frei zu bewegen. Zur Erweiterbarkeit gibt es noch keine Informationen. Die gebotene Funktionalität beinhaltet die relative Position zum Controller zu halten oder eine Vordefinierte

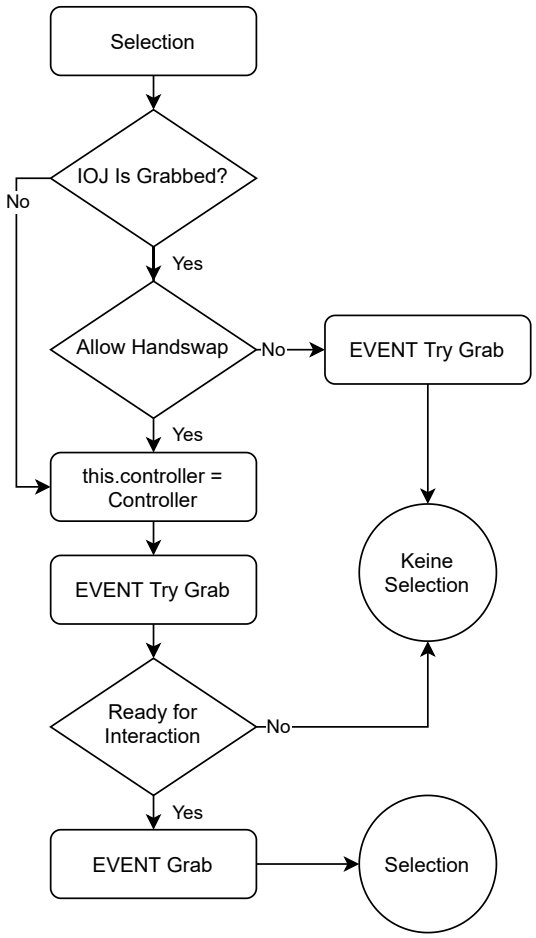

Abb. 7. Darstellung des Ablaufes nach der Selektion

einzunehmen. Diese Klasse besitzt keine weiteren Events über die Geerbten hinaus.

ICToggle<> ist eine Template Klasse, Abbildung 14. Sie bietet alle Funktionen, die für Objekte notwendig sind, die zwischen zwei Zuständen wechseln. Die Klasse ist typisiert und macht es möglich, dass die erbenden Klassen lediglich ein Interface für den Zustandswechsel implementieren müssen. Für die Interaktion lässt sich einstellen wie jeweils auf Selektion und Deselektion reagiert wird: aktivieren, deaktivieren, nichts, oder Zustand wechseln. Zusätzlich kann angegeben werden, ob pro Interaktionszyklus nur eine Zustandsänderung erlaubt ist.

3.5 Fazit

In diesem Abschnitt ist ein Framework für die Gestaltung von Interaktionen in VR-Umgebungen entstanden. Der aktuelle Zustand erlaubt es die implementierten Interaktionen zu verwenden. Die Erweiterbarkeit sowie die Funktionalität ist mit den Beispielklassen gezeigt. Die Event Struktur hat sich bis jetzt als gute Lösung herausgestellt, um Watchdogs zu vermeiden. Die Performance konnte bis jetzt noch nicht bewertet werden. Sollte diese ein Problem werden, kann dieses mit einer eigenen Lösung für die Update Schleife behoben werden. Es ist bereits bei den aktuellen Interaktionen zu sehen, dass die Konfiguration von Gizmos profitieren würde, um die Einstellungen zu visualisieren. In der VR haben die Interaktionen technisch funktioniert. Ebenso unterstützt das Framework den Einsatz in Schulungsszenarien. Das entstandene Framework lässt sich erweitern, um es zur Untersuchung von orthogonalen Interaktionen zu nutzen.

4 Hardware Plattform für VR-Eingabegeräte

Im vorherigen Kapitel wurde dargestellt wie unterschiedliche Interaktionen, die in einem VR-Trainings Szenario auftreten, klassifiziert und durch ein Framework abstrahiert werden können. Der folgende Abschnitt setzt sich mit dem Thema Gestaltung von VR-Interaktionen im Trainingskontext aus einer eher Hardware fokussierten Sicht auseinander. Genauer soll eine Plattform geschaffen werden, in der verschiedene Eingabegeräte zur Durchführung einer ausgewählten Beispielinteraktion in einem Testszenario eingebunden werden.

4.1 Randbedingungen

Für die Auswahl der Eingabegeräte und den Aufbau der Plattform sind mitunter die Randbedingungen relevant, welche deswegen in den folgenden Abschnitten kurz erläutert werden sollen.

CSTI Dieses Projekt wird in der Laborumgebung des Creative Space for Technical Innovations (CSTI) an der HAW Hamburg durchgeführt. Das CSTI ist ein interdisziplinäres Labor mit Fokus auf das Themengebiet Human-Computer Interaction und Smart Environments. Studierende aus verschiedenen Studienrichtungen erhalten hier Raum, Equipment und Betreuung, um ihre Ideen als Projekte umzusetzen. Dabei werden besonders Konzepte wie Agiles Prototyping und die Verbindung zur regionalen Wirtschaft gefördert.
Das Labor stellt den Studierenden für ihre Projekte eine Vielzahl von Hardware zur Verfügung. So befinden sich neben leistungsstarken Rechnern diverse Tracking Systeme, die größtenteils in einer Traverse untergebracht sind sowie verschiedene Ausgabegeräte z.B. VR/AR Brillen in den Räumlichkeiten des CSTI.

Entwicklungsumgebung Die VR-Anwendung des Projekts wird in Unity (2017.3) entwickelt. Als eine der populärsten Entwicklungsumgebungen für VR-Anwendungen bietet Unity Support und Schnittstellen für diverse Eingabe- und Ausgabegeräte. Des Weiteren lassen sich Erkenntnisse aus anderen CSTI Projekten transferieren, die größtenteils in Unity entwickelt werden.

Ausgabegerät Als Ausgabegerät für die VR-Anwendung dient die HTC Vive. Diese ist zum Zeitpunkt der Projektdurchführung das Standard HMD im CSTI. Durch die umfangreiche SteamVR Library lässt die Vive sich leicht in Unity einbinden und verschiedene Eingabegeräte gut koppeln.

4.2 Szenario

Für den Aufbau der Plattform wird ein exemplarisches Szenario benötigt. Im Fokus soll die Durchführung einer Interaktion mit verschiedenen Eingabegeräten stehen. Es gilt also nicht ganze Interaktionsfolgen oder verschiedene Interaktionen, die in einem Schulungskontext auftreten, abzubilden.
Als Beispielinteraktion wurde das Öffnen eines Schrankes gewählt. Diese Interaktion ist Teil vieler alltäglicher als auch Schulungsspezifischer Situationen und somit einerseits kontextrelevant und andererseits den meisten Nutzern vertraut. Weiterhin beinhaltet das Öffnen eines Schrankes sowohl die Aspekte der Selektion als auch der Manipulation, in denen Eingabegeräte unterschiedliche Merkmale aufweisen können.
Konkret wurde das Modell eines Industrieschrankes in einem virtuellen Nachbau des CSTI VR-Bereiches platziert. Die Türen des Schrankes wurden dann durch einen Mix aus Skripten und Komponenten der Unity Physik Engine modifiziert, um durch Eingabegeräte selektiert und manipuliert werden zu können. Die Umsetzung des letzten Schrittes erfolgte hierbei mehrfach für die jeweiligen Eingabegeräte.

4.3 Eingabegeräte

Bei der Auswahl der konkreten Eingabegeräte, welche in diesem Projekt verwendet werden, spielte vor allem die Verfügbarkeit der Hardware im Kontext des Labors und die Schwierigkeit der Integration in die Entwicklungsumgebung eine wichtige Rolle. Des Weiteren wurde darauf geachtet für alle in Kapitel 2.2 vorgestellten Kategorien mindestens ein Beispiel zu implementieren. In den folgenden Abschnitten werden sowohl technische Spezifikationen der Hardware als auch die Vorgehensweise zur Integration dieser in das Szenario erläutert.

4.4 Zusätzliche Artefakte

In diesem Projekt werden zwei verschiedene Hardware Lösungen für den auf zusätzliche Artefakte basierenden Ansatz zur Interaktionsgestaltung implementiert. Konkret werden die Controller der HTC Vive und der Valve Index in die Anwendung eingebunden. Beide Controller sind Teil des Vive Tracking Systems, welches bereits für das Tracking des HMDs genutzt wird, und lassen sich leicht durch die SteamVR Library in Unity integrieren.
Während beide Controller Typen innerhalb des Tracking Systems und der SteamVR Library grundsätzlich austauschbar sind, unterscheiden sie sich dennoch in ihrem Grundansatz. Dies ist hauptsächlich durch die zusätzliche Option der Vive Index Controller, die Hände des Nutzers zu tracken, bedingt. Im Folgenden werden beide Typen und ihre jeweiligen Hardware Spezifikationen genauer beschrieben.

HTC Vive Controller Die HTC Vive Controller bestimmen ihre Position und Ausrichtung im Raum sowie die Griffstärke des Nutzers anhand von 24 verbauter Sensoren. Außerdem besitzen sie ein Multifunktions-Trackpad, einen zweistufigen Trigger Knopf, Knöpfe für System- und Menüzugriff sowie eine Vibrationsfunktion im Griff für haptisches Feedback. Dabei sind die Controller kabellos und halten mit aufgeladenem Akku ungefähr sechs Stunden Betrieb aus.

Vive Index Controller Bei den Vive Index Controllern werden die Werte von 87 Sensoren genutzt, um Handposition, Fingerpositionen, Bewegung und Grifffestigkeit des Nutzers zu erkennen. Weiterhin besitzen die Controller drei Tasten, einen Trigger Knopf, einen Analogstick sowie eine Track-Taste mit Kraftsensor. Haptisches Feedback ist im Griff und in einigen der Tasten verbaut. Auch die Index Controller laufen kabellos jedoch sogar mit bis zu sieben Stunden durchgängiger Betriebszeit.

Die Integration beider Controller Typen in das Projekt erfolgt durch das Player Prefab aus der SteamVR Library. Das Player Prefab steuert einerseits die Ausrichtung und Position der Kamera, indem sie die Tracking Werte der HTC Vive auswertet und bestimmt andererseits automatisch welcher Controller Typ genutzt wird, um ein entsprechendes Modell des Controllers oder der Hand in die Szene

zu laden und folgend anhand der Tracking Daten zu aktualisieren. Die Modelle beider Controller sind in Abbildung 8 dargestellt.
Damit die Tür auf die Interaktion des Nutzers reagieren kann, wird diese durch das Interactable Skript und das Circular Drive Skript aus dem Interaction System der SteamVR Library ergänzt. Da das SteamVR Interaction System auf dem Unity Physik System aufbaut, benötigen Tür und Griff ebenfalls ein Collider und ein Rigidbody. Das Interaction System abstrahiert die Interaktion mit Objekten, um eine Controller unabhängige Implementation zu ermöglichen. Konkret werden verschiedene Trigger Events zur Selektion unterstützt. So kann ohne Anpassungen an den Skripten vorzunehmen sowohl das Drücken des Trigger Knopfes der HTC Vive Controller als auch das Schließen der Hand bei den Vive Index Controllern in Collision Reichweite der Tür bzw. des Türgriffs als Selektion interpretiert werden. Einmal selektiert folgt die Tür dem Controller oder der Hand des Nutzers bis die Trigger Bedingungen nicht mehr erfüllt sind. Dabei richet sich die Tür in ihrer Bewegung an die physikalischen Restriktionen, die einerseits im Circular Drive Skript definiert sind und andererseits sich aus dem Unity Physik System ableiten.
Um eine alternative Implementationsmöglichkeit zu untersuchen wurde in diesem Projekt des Weiteren ein zweiter Ansatz zum Öffnen der Tür mit zusätzlichen Artefakten umgesetzt. Dafür wurden beide Controller Typen mit dem Raycast Skript aus der SteamVR Library versehen. Durch das Raycast Skript ist der Nutzer in der Lage Objekte aus der Ferne mit dem Trigger Event des entsprechenden Controller Typs zu selektieren. Das Verhalten der Tür ist in einem selbst geschriebenen RaycastHandler Skript definiert, welches die Selektionsevents des Raycasts implementiert. Dabei prüft das Skript ob das ausgewählte Objekt als interagierbar markiert ist, welches für die Tür der Fall ist. Anschließend bewegt sich die Tür abhängig von einer Variable, welche zwischen zwei Modi unterscheidet. Im ersten Modus springt die Tür auf geschlossen oder geöffnet je nachdem welcher Zustand vorher gegeben war. Im zweiten Modus hingegen folgt die Tür der Neigung des Controllers oder der Hand, wobei eine Neigung nach links die Tür langsam bis zu einem Maximum von 90 Grad öffnet und eine Neigung nach rechts die Tür langsam schließt.

(a) HTC Vive Controller (b) Vive Index Controller

Abb. 8. Vergleich der Controller Modelle in VR

4.5 Optisches Tracking

Neben den Ansätzen, die zusätzliche Artefakte benötigen, wurde auch ein Ansatz zum optischen Tracken der Hände implementiert. Dafür wurde der Leap Motion Sensor als Eingabegerät genutzt. Die Leap Motion kann mehrere Hände in einem halbkugelförmigen Bereich mit einem Radius von einem Meter ausgehend von dem Sensor tracken. Hierbei wird für jede Hand die Position von 20 verschiedenen Finger Joints bestimmt. Zusätzlich dazu werden sieben weitere Joints der Hand bzw. der Arme berechnet und damit eine optionale Darstellung der Arme ermöglicht.
Zur Integration in eine VR-Umgebung lässt sich die Leap Motion mit einer Halterung an der Vorderseite der HTC Vive befestigen (siehe Abbildung 11). Eine Unity Integration des Leap Motion SDKs wird vom Hersteller angeboten. Diese bietet unter anderem ein Player Prefab, welches sowohl die HTC Vive Positionsdaten als auch die Tracking Daten der Leap Motion verarbeitet. Somit kann die Position der Hände abhängig von der Position des Spielers bestimmt werden. Für die Darstellung der Hände kann ein Standard Modell verwendet werden oder die Daten der Leap Motion auf ein selbstgeneriertes Modell mit entsprechenden Joints abgebildet werden. Das in diesem Projekt genutzte Modell ist in Abbildung 9 dargestellt.

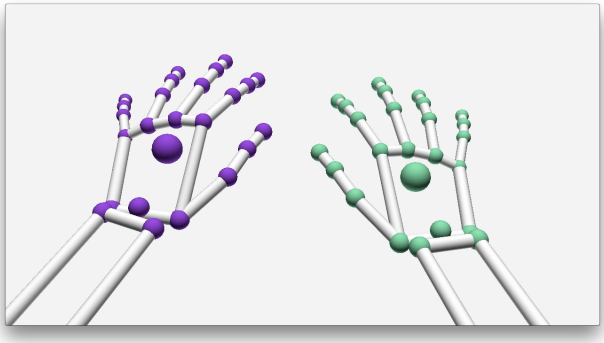

Abb. 9. Leap Hand Modell [19]

Um die Interaktion zwischen dem Hand Modell und der Schranktür zu ermöglichen wurden ähnlich wie bei den Artefakten zwei Ansätze implementiert. Der erste Ansatz nutzt Komponenten aus dem Leap Motion Interaction Engine Modul, welches in der Unity SDK Integration enthalten ist. Dabei werden Tür und Griff mit einem Rigidbody und dem InteractionBehaviour Skript versehen. Anschließend wird das Hand Modell durch ein physikalisches Modell ergänzt. Durch das Unity interne Physik System lässt sich nun die Tür mit der Hand bzw. den Fingern öffnen und schließen (vgl. Abbildung 10).

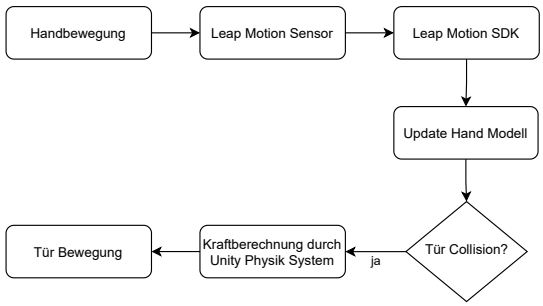

Abb. 10. Ablauf des Leap Motion Trackings

Bei dem zweiten Ansatz hingegen wird die Verwendung des Unity Physik Systems für die Manipulation der Tür vermieden. Dafür wird die Collision von Hand und Tür durch Collider auf den Objekten bestimmt. Tritt ein Collision Event ein wird über das Leap Motion SDK abgefragt, ob die Hand geschlossen ist oder nicht. Sollte die Hand geschlossen sein, wird ein Skript aktiviert, welches die Bewegung der Tür innerhalb ihrer Limits der Handbewegung anpasst bis die vorher beschriebenen Bedingungen nicht mehr erfüllt sind.

Abb. 11. Leap Motion an der HTC Vive befestigt [16]

4.6 Sprachsteuerung

Für die Umsetzung eines sprachbasierten Ansatzes zur Interaktion mit dem Schrank wird einerseits ein Eingabegerät zur Soundaufnahme (Mikrofon) und andererseits ein Verfahren zur Spracherkennung benötigt. Da die HTC Vive ein eingebautes Mikrofon besitzt, bietet es sich an dieses zu nutzen. Dabei ermöglicht Unity die Einbindung des Mikrofons in das Projekt ohne zusätzliche Schritte solange die Anwendung für PC-Systeme ausgelegt ist (PC-Standalone Build). Auch wenn für das Projekt nur wenige Ausdrücke erkannt und interpretiert werden müssen, ist die Implementation einer eigenen Spracherkennung eine sehr komplexe Aufgabe. Deswegen lohnt es sich an eine externe Lösung in das Projekt zu integrieren. Da die Anwendung auf einem Windows System läuft, kann Microsofts Spracherkennungs Software Cortana für diese Aufgabe verwendet werden. Cortana besitzt eine Unity interne Integration und läuft lokal ohne zusätzliche Netzwerkschnittstellen implementieren zu müssen wie es bei cloudbasierten Alternativen der Fall wäre.
Konkret wird zuerst ein Dictionary eine Zuordnung von Worten oder Phrasen zu Actions erstellt. Hierbei kann jede Sprache, die auf dem Windows System installiert ist, genutzt werden. Eine Action kann eine beliebige Methode sein, die im Unity Projekt definiert wurde. Für das Beispielszenario wurden die Befehle "Tür auf" und "Tür zu" mit entsprechenden Methoden verknüpft, welche die Schranktür in einer fließenden Bewegung öffnen oder schließen. Das gefüllte Dictionary wird dann einem KeywordRecognizer übergeben. Dieser ist während der Ausführung der Anwendung im Hintergrund aktiv und reagiert auf jede Registration eines Dictionary Eintrages, indem er die verbundene Action ausführt. Für einen reibungslosen Ablauf bietet es sich an Actions, welche länger als ein Frame aktiv sind, als Coroutine zu implementieren. Diese erlauben eine nahezu parallele Ausführung ohne den Unity internen Kontrollfluss negativ zu beeinflussen. Eine Zusammenfassung des Ablaufs findet sich in Abbildung 12.

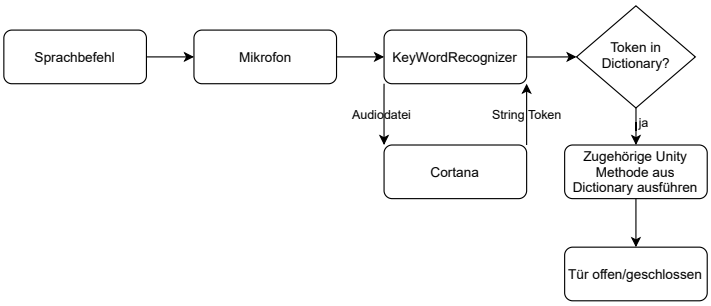

Abb. 12. Ablauf der Spracherkennung

Die Selektion welches Objekt auf die Sprachbefehle reagieren soll ist in diesem Projekt nicht notwendig, da nur ein Interaktionsobjekt mit den definierten Befehlen vorhanden ist. Dies gilt jedoch nicht für ein komplexeres Szenario, in dem z. B. mehrere Schränke mit gleichem Befehlset existieren. Bei diesem müsste der vorhandene Ansatz um eine Möglichkeit zur Selektion ergänzt werden.

4.7 Fazit

Das Ziel dieses Projektabschnitts war die Entwicklung einer Plattform, in der verschiedene Eingabegeräte zur Konstruktion von interaktiven VR-Elementen genutzt werden können. Im Zuge dessen wurde unter Berücksichtigung der Randbedingungen ein Testszenario entwickelt, in dem ein Nutzer einen Industrieschrank öffnet. Zur Durchführung dieses Beispielszenarios ist es erfolgreich gelungen Ansätze der drei Kategorien Zusätzliche Artefakte, Optisches Tracking sowie Sprachsteuerung (Hand-unabhängig) zu implementieren. Hierbei wurden neben den unterschiedlichen Eingabegeräten auch verschiedene Vorgehensweisen zur Integration, besonders in Bezug auf das Verhalten des Interaktionsobjekts, vorgestellt.
Eine Bewertung der Eingabegeräte und ihrer Einbindungen unter Berücksichtigung des Anwendungskontextes Training in VR ist nicht Teil dieses Projekts. Eine Evaluation in weiterführenden Projekten wird zeigen wie sinnvoll die gewählten Optionen sind. Dabei kann eine Erweiterung der vorhandenen Hardware für weitere Vergleichsoptionen in Erwägung gezogen werden. So bietet die Oculus Quest von Facebook ein in der Brille verbautes optisches Handtracking, welches ohne Controller oder extra Sensorik auskommt. Auch eine Kombination aus Ansätzen wie die Ergänzung der Sprachsteuerung um eine Selektionskomponente durch Controller basiertes Raycasting bildet eine Alternative zu den bereits implementierten Ansätzen.

5 Zusammenfassung und Ausblick

Innerhalb dieses Grundprojekts wurde eine Toolchain entwickelt, um Entwickler bei der Gestaltung von objektspezifischen Interaktionen in VR zu unterstützen. Dafür wurden zwei Komponenten verwirklicht. Mit der ersten Komponente wurde ein Framework entworfen, welches die Erstellung und Untersuchung von Inkteraktionen in VR vereinfacht. Mit der zweiten Komponente wurde eine Plattform zur Einbindung verschiedener Eingabegeräte für die Interaktion mit Objekten in der VR geschaffen. Für beide Komponenten wurden unabhängig voneinander das Konzept und die Implementation sowie mögliche Erweiterungen präsentiert, jedoch keine Bewertung durchgeführt.
Folglich bietet es sich an im Hauptprojekt die Toolchain zu evaluieren. Gerade die Eignung in Bezug auf den Anwendungskontext "Training in VR" gilt es zu erforschen. Denn wie bereits in der Einleitung erwähnt ergeben sich aus diesem Kontext spezielle Anforderungen an das VR-System. So ist das Ziel nicht die perfekte Immersion, sondern die Maximierung des Lernerfolgs des Nutzers und

der einfache Transfer des Gelernten in die reale Welt, während Kosten und Komplexität möglichst niedrig gehalten werden sollen.
Für eine Evaluation könnte es sich als nützlich erweisen die beiden Komponenten zusammenzuführen und ein umfassendes Szenario zu entwickeln, welches anschließend im Rahmen einer Nutzerstudie untersucht werden kann. Als Grundlage für eine solche Untersuchung müsste zunächst jedoch die Literaturanalyse um die Bereiche der Didaktik und der Usability erweitert werden. Auch die Umsetzung der formulierten Erweiterungen für die jeweiligen Komponenten könnte in Erwägung gezogen werden, wenn sich dadurch bessere Ergebnisse durch die Studie erwarten lassen.

Literaturverzeichnis

[1] Ivan E. Sutherland. "The Ultimate Display". In: *Proceedings of the IFIP Congress*. 1965, pp. 506–508.

[2] Richard Stoakley, Matthew J. Conway, and Randy Pausch. "Virtual Reality on a WIM: Interactive Worlds in Miniature". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '95. Denver, Colorado, USA: ACM Press/Addison-Wesley Publishing Co., 1995, pp. 265–272. ISBN: 0201847051. DOI: 10.1145/223904.223938.

[3] Ivan Poupyrev et al. "The Go-go Interaction Technique: Non-linear Mapping for Direct Manipulation in VR". In: *Proceedings of the 9th Annual ACM Symposium on User Interface Software and Technology*. UIST '96. Seattle, Washington, USA: ACM, 1996, pp. 79–80. ISBN: 0-89791-798-7. DOI: 10.1145/237091.237102.

[4] Doug A. Bowman, Donald B. Johnson, and Larry F. Hodges. In: *Proceedings of the ACM Symposium on Virtual Reality Software and Technology*. VRST '99. London, United Kingdom: Association for Computing Machinery, 1999, pp. 26–33. ISBN: 1581131410. DOI: 10.1145/323663.323667. URL: https://doi.org/10.1145/323663.323667.

[5] Angie Geary. "A Haptic Virtual Reality Training Tool for Paper Conservators". In: *ACM SIGGRAPH 2003 Sketches &Amp; Applications*. SIGGRAPH '03. San Diego, California: ACM, 2003, pp. 1–1. ISBN: 978-1-4503-7466-8. DOI: 10.1145/965400.965498. URL: http://doi.acm.org/10.1145/965400.965498.

[6] Zayd Hendricks, Gary Marsden, and Edwin Blake. "A Meta-Authoring Tool for Specifying Interactions in Virtual Reality Environments". In: *Proceedings of the 2nd International Conference on Computer Graphics, Virtual Reality, Visualisation and Interaction in Africa*. AFRIGRAPH '03. Cape Town, South Africa: Association for Computing Machinery, 2003, pp. 171–180. ISBN: 1581136439. DOI: 10.1145/602330.602362. URL: https://doi.org/10.1145/602330.602362.

[7] Andreas M. Heinecke. *Mensch-Computer-Interaktion*. Springer Berlin, 2012. URL: https://doi.org/10.1007/978-3-642-13507-1.

[8] R. Dörner et al. *Virtual und Augmented Reality (VR/AR)*. eXamen.press. Berlin: Springer Vieweg, 2013. ISBN: 978-3-642-28902-6. DOI: 10.1007/978-3-642-28903-3.

[9] Nirit Gavish et al. "Evaluating virtual reality and augmented reality training for industrial maintenance and assembly tasks". In: *Interactive Learning Environments* 23.6 (2015), pp. 778–798. DOI: 10.1080/10494820.2013.815221. URL: https://doi.org/10.1080/10494820.2013.815221.

[10] Alan Cheng, Lei Yang, and Erik Andersen. "Teaching Language and Culture with a Virtual Reality Game". In: *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. CHI '17. Denver, Colorado, USA: Association for Computing Machinery, 2017, pp. 541–549. ISBN: 9781450346559. DOI: 10.1145/3025453.3025857. URL: https://doi.org/10.1145/3025453.3025857.

[11] Johann Bronsch. “Einsatz von Virtual-Reality Techniken in Lernumgebungen.” MA thesis. Hamburg, Germany: Hochschule für Angewandte Wissenschaften Hamburg, Feb. 2018.

[12] Anton Franzluebbers and Kyle Johnsen. “Performance Benefits of High-Fidelity Passive Haptic Feedback in Virtual Reality Training”. In: *Proceedings of the Symposium on Spatial User Interaction*. SUI ’18. Berlin, Germany: Association for Computing Machinery, 2018, pp. 16–24. ISBN: 9781450357081. DOI: 10.1145/3267782.3267790. URL: https://doi.org/10.1145/3267782.3267790.

[13] Mehdi Hafsia, Eric Monacelli, and Hugo Martin. “Virtual Reality Simulator for Construction Workers”. In: *Proceedings of the Virtual Reality International Conference - Laval Virtual*. VRIC ’18. Laval, France: ACM, 2018, 11:1–11:7. ISBN: 978-1-4503-5381-6. DOI: 10.1145/3234253.3234298. URL: http://doi.acm.org/10.1145/3234253.3234298.

[14] S. Werrlich et al. “Comparing HMD-Based and Paper-Based Training”. In: *2018 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. 2018, pp. 134–142. DOI: 10.1109/ISMAR.2018.00046.

[15] D. Mendes et al. “A Survey on 3D Virtual Object Manipulation: From the Desktop to Immersive Virtual Environments”. In: *Computer Graphics Forum* 38.1 (2019), pp. 21–45. DOI: https://doi.org/10.1111/cgf.13390. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.13390. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.13390.

[16] *HTC Vive FAQ: What You Need to Know About Leap Motion + SteamVR*. https://blog.leapmotion.com/leap-motion-htc-vive-faq/. Zugriff: 09.03.2021.

[17] *SteamVR Unity Plugin*. https://valvesoftware.github.io/steamvr_unity_plugin/. Zugriff: 23.03.2021.

[18] *Unity Manual - Physics*. https://docs.unity3d.com/Manual/Physics.html. Zugriff: 23.03.2021.

[19] *VUO: Leap Motion hand skeleton points*. https://vuo.org/component/vuo.leap. Zugriff: 09.03.2021.

A Anhang

```
class IOJ

        #region Variabels
        #endregion Variabels
        override public void
            Grab(ICControllerGrab controller){}
        override public void
            Release(ICControllerGrab controller){}
        override public void Awake(){}
        override public void Start(){}
        public void Update(){}
        public void OnValidate(){}
        new public bool OBJWithNameConvention(){}
        public void ResetOBJ(){}
    #region Personal Methods
        private void CalculatePosition(){}
        private void CalculateRotation(){}
    #endregion Personal Methods
    #region ContextMenuItem
    #endregion ContextMenuItem

end class
```

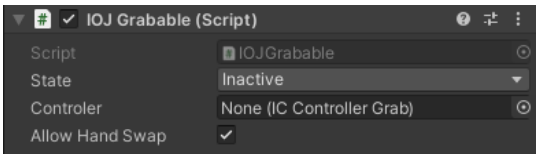

Abb. 13. Inspektor des IOJ Grabable

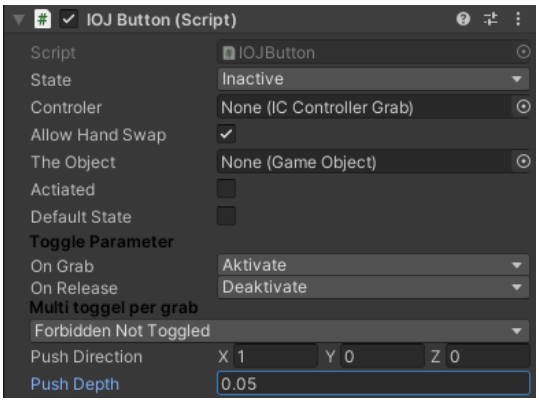

Abb. 14. Inspektor des IOJ Button

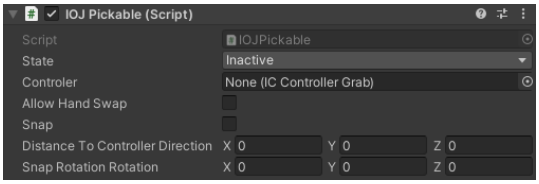

Abb. 15. Inspektor des IOJ Pickable

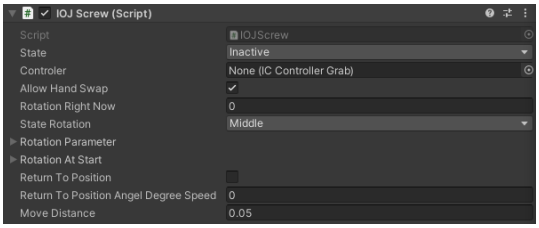

Abb. 16. Inspektor des IOJ Screw

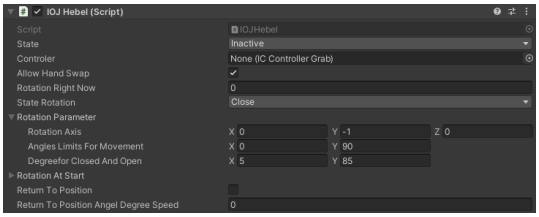

Abb. 17. Inspektor des IOJ Hebel