Leact: Fusion der Kinect V2 und Leap Motion zur erweiterten Menüsteuerung in VR

Patrick Höling, Tobias Michel Latta

Zusammenfassung

In dieser Ausarbeitung wird eine Möglichkeit zur Fusion von KinectV2- und Leap Motion-Sensordaten erklärt um eine Menüsteuerung in der Virtuellen Realität (VR) zu implementieren. Das erstellte Framework nennt sich *Leact* und ist motiviert durch die Kombination aus hoher Auflösung der Fingererkennung der Leap Motion (LM) sowie dem großen Bewegungsradius der Kinect V2. Validiert wird das Konzept über die Implementierung einer kaskadierenden Menüführung und Manipulation von 3D-Objekten. Die Menüführung wird über den großen Bewegungsradius der Kinect V2 realisiert und die Selektion sowie Manipulation der Objekte findet mit der LM statt. Aufgrund der verbesserungswürdigen Verwertung der Kinect V2-Sensordaten, die zu fehlerhaften Interaktionen in rund zehn Prozent der Fälle führt, wird eine Optimierung durch eine Tiefpassfilterung der Sensordaten empfohlen. So werden nur eindeutig und zeitlich über einen gewissen Zeitraum bestehende Gesten erkannt.

1 Einleitung

Diese Ausarbeitung ist im Zusammenhang mit der Vorlesung *Human Computer Interaction* (HCI), gehalten von Prof. Dr. Birgit Wendholt im Wintersemester 2017/2018 an der Hochschule für Angewandte Wissenschaft Hamburg, entstanden. Während der Vorlesung werden unterschiedliche Implementierungen im Bereich HCI behandelt und in insgesamt drei Laborterminen getestet. Unter anderem handelt es sich dabei um das VR-Headset *HTC Vive*, sowie das Handtracking-System *Leap Motion*. Beides in Verbindung mit der Entwicklungsumgebung Unity. Ebenfalls evaluiert wurde das Ganzkörper-Tracking-System von Microsoft, genannt *Kinect V2*.

Das Ziel dieser Ausarbeitung ist die Kombination von großem Bewegungsraum der Kinect mit dem genauen Handtracking der Leap Motion um eine intuitive Bedienung mit den Händen vor dem Körper hochauflösend abzubilden sowie außerhalb des Leap Motion-FOV Gestensteuerung zu ermöglichen. Durch die Bedienung der VR mit dem eigenen Körper wird die Immersion nicht zuletzt durch die visuelle Repräsentation der eigenen Hände gefördert. Auch muss kein externes Eingabegerät verwendet werden, welches Körperaktionen nur durch Tastendruck aufnehmen kann. Der Nutzer kann intuitiv die Geste aus der Realität vollziehen und sie wird durch die Leap Motion aufgenommen sowie in der VR verarbeitet.

Diese Intuivität ist wichtig um VR als Medium zu etablieren. An mehreren technischen Produkten lässt sich die Relevanz von einfacher Bedienung erkennen. Beispielsweise ist der Erfolg des Apple iPhone der ersten Generation auch darüber zu erklären, dass es eine intuitive Bedienung geschaffen hat, welche es auch weniger technisch-versierten Nutzern ermöglichte komplexe Funktionen abzurufen und zu nutzen (Laugesen, 2010). So wird die Hemmschwelle zur Nutzung von Technik herabgesenkt, was wiederum bei den Entwicklern und Unternehmern Interesse weckt, in die gleiche Richtung zu investieren/produzieren. Ohne größere Erklärungen oder Schnittstellen (Werkzeuge) sollen Abläufe mit dem eigenen Körper durchgeführt werden, so wie es auch in der Realität hauptsächlich der Fall ist. Die Fehlerhäufigkeit bei einem Funktionsaufruf soll dabei auf maximal einen Fehlaufruf unter fünf Versuchen beschränkt werden.

Um den Lösungsansatz verständlich zu dokumentieren, wird dieser Aufsatz wie folgt gegliedert: Zuerst findet eine wissenschaftliche Einordnung statt, indem die Inhalte mit bestehenden Arbeiten verglichen werden. Darüber hinaus werden verwendete Frameworks und Beispiele zur jeweiligen Implementierung von Kinect V2 und Leap Motion näher erläutert. Anschließend gibt es einen kurzen Rundumschlag über den derzeitigen Stand der Technik, der einen kritischen Blick auf die genutzten Produkte wirft, bevor sie in den Zusammenhang zum Vorlesungsinhalt von *HCI WS17/18* gebracht werden. Diese Einordnung ist sinnvoll um die Sensorik tiefergehend zu erläutern. Der Hauptteil beschäftigt sich mit der Konzeptionierung und Umsetzung der Idee.

2 Realisierung

2.1 Wissenschaftliche Einordnung

In dem Paper *Multi-sensor data fusion for hand tracking using Kinect and Leap Motion* (Penelle, 2014) wird eine Verbindung zwischen dem hochauflösenden Handtracking der Leap Motion sowie dem großen Bewegungsbereich der Kinect V1 beschrieben. Auf mathematischer Ebene wird beschrieben, wie die Handgelenke bei beiden Sensoren pixel-genau übereinander zu legen sind, wie also eine Kalibrierung und Registrierung beider Komponenten erfolgt. Anwendungsfall ist dabei die Schmerztherapie bei medizinischen Patienten, welche einen Arm verloren haben. Setzt sich der Betroffene ein VR-Headset auf, sollen die Bewegungen des vorhandenen Arms auf das letzte vorhandene Gelenk auf der amputierten Seite gespiegelt werden, um den Eindruck zu erwecken, dass beide Arme vorhanden sind. So können einfache Tätigkeiten ausgeführt werden und die Illusion des rekonstruierten Armes reduziert den Phantomschmerz. Überschneidungen zu diesem Aufsatz finden sich bei der komplementären Nutzung von Leap Motion und einer Kinect-Kamera. Zwar wird hier eine neuere Kinect-Version verwendet, jedoch ist der grundsätzliche Hintergrund einen größeren Bewegungsraum im Vergleich zur LM zu schaffen ebenfalls gegeben. Ein weiterer Unterschied ist die Herangehensweise an die Thematik. Durch die mathematische Beschreibung der Kalibrierungs- und Registrierungs-Algorithmen kann die Methodik auch auf andere Realisierung des Sensortyps angewendet werden. Dabei rückt jedoch die konkrete Implementierung in einer Entwicklungsumgebung in den Hintergrund, was die Ausführbarkeit für den durchschnittlichen Benutzer beschränkt. In diesem Aufsatz wird eine andere Zielsetzung verfolgt. Es soll ein Framework geschaffen werden, welches Sensorfusion von Kinect V2 und Leap Motion betreibt um eine interaktive und intuitive Menüführung zu bewerkstelligen, welche leicht für eigene Projekte adaptiert und genutzt werden kann.

Interaktive Systeme (Band II) (Preim & Dachselt, 2015) liefert in Kapitel 8 einen Überblick über die Interaktion mit 3D-Objekten und empfiehlt beispielsweise eine Kreismenü-Platzierung an festen Weltkoordinaten. Kontextabhängige Positionierung ist ebenfalls beschrieben und wird in dieser Ausarbeitung verwendet. Das Kreismenü ist an das Sichtfeld gekoppelt um stets verfügbar zu sein und eine klare Trennung zwischen Menüführung und Objektmanipulation zu realisieren. Des Weiteren sind im dritten Teil des Buches Natural User Interfaces beschrieben, genauer genommen auch Gesten die mit den Händen oder Armen ausgeführt werden. Die *Zeigegesten* (Preim & Dachselt, S.491, 2017) referenzieren (abstrakte) Objekte in der Umgebung und werden ruhig ausgeführt. Kinder nutzen diese nonverbale Referenzierung aufgrund ihrer intuitiven Deutung durch die Erwachsenen. Um die Menüauswahl möglichst einfach und selbsterklärend zu gestalten, wird daher diese Art von Gesten verwendet. Die verwendeten Körperteile definieren darüber hinaus die Gesten als Hand- und Körpergesten (Preim & Dachselt, S.500-503, 2017).

2.2 Einordnung *HCI Wintersemester 2017/2018*

Nachdem nun Vergleiche zu anderen wissenschaftlichen Arbeiten gezogen wurden, findet eine Verbindung zu den Vorlesungsinhalten statt.

Die Leap Motion, welche 2013 auf dem Markt erschienen ist, kann prinzipiell in jeder Position die Finger vor sich tracken. Häufig wird sie daher auf dem Schreibtisch vor die Tastatur gelegt um die Bedienung mit herkömmlichen Peripheriegeräten durch Handtracking zu ergänzen (Abbildung 2). Im VR-Kontext wird sie hauptsächlich an dem Head Mounted Display (HMD) als Inside OutVariante fürs Handtracking verwendet (Abbildung 1).

Abbildung 1:Leap Motion an HTC Vive befestigt

Abbildung 2:Leap Motion in Desktop-Verwendung

Sie arbeitet auf Basis der Infrarot-Laufzeitmessung und gleicht diese Sensordaten mit einem Handmodell ab um nicht plausible Fingerstellungen direkt raus zu filtern. Der genaue Ablauf ist wie folgt: Ein Raster aus Infrarot-Lichtpunkten wird in das FOV gesendet und anschließend wieder aufgenommen. In Abhängigkeit der Entfernung verändert sich die Zeit, welche das Licht benötigt um wieder einfangen zu werden. Mit allen Punkten kann somit das Raster in ein dreidimensionales Bild umgewandelt werden, welches wiederum auf charakteristische Gelenke untersucht wird. Gibt es nun mehrere Übereinstimmungen mit den Gelenken eines beispielhaften Handmodells, wird es auf die detektierten Gelenkpunkte gelegt und eine digitale Repräsentation findet statt. Liegt nun die Hand außerhalb des Infrarot-Rasters, kann dementsprechend keine Detektion stattfinden. Die 27 Freiheitsgrade einer Hand werden nur selten (1 von 100) falsch digitalisiert. Hauptsächlich treten Fehler auf, wenn sich die Hand an den Rand des *Sichtbereichs*/*Field of View* (FOV) begibt oder in einer unvorhergesehenen Position steht, da dann keine plausiblen Übereinstimmungen zu dem Beispiel-Handmodell bestehen.

Die Kinect V2 ist eine Infrarot-Tiefenkamera, welche von Microsoft für das Ganzkörper-Tracking entwickelt wurde. Sie arbeitet nach dem Outside-In Verfahren, benötigt also einen festen Standpunkt um den Nutzer zu tracken. Mit 6 Freiheitsgraden wird die Position wie folgt ermittelt: Eine großflächige Infrarot-Lichtquelle wird von der Kinect ausgesendet. Der zu detektierende Nutzer reflektiert in Abhängigkeit seines Standpunktes und seiner Haltung gewisse Bereiche der Infrarotlichtwolke, die von der Infrarot-Kamera aufgenommen werden. Dabei ist es ebenfalls möglich, Infrarotlicht aufzunehmen, welches von dem detektierten Objekt selbst ausgesendet wird. Beispielsweise kann so parallel zur Entfernung des Nutzers von der Kinect auch die Temperatur des Nutzers gemessen werden. In der Praxis wird eine Entfernung von 0,5 m bis 2,5 m zur Kinect empfohlen. Außerdem liegt der Fokus auf einer

Ganzkörpererkennung, welche bis zu 25 Gelenke des Körpers trackt. Im Bereich der Hand sind nur der Daumen, der Zeigefinger sowie das Handgelenk erfassbar. Es treten bei Nutzung in mehr als zehn Prozent der Fälle falsche Erkennungen der Handstellung auf, was vermutlich an der Sensorverarbeitung liegt, welche ohne weiterführende Logik, die Position der Gelenke zu Gesten interpretiert. Nach der Entwicklung von Kinect V1 und V2 wird in Zukunft keine Weiterentwicklung dieses Produktes erfolgen (Warren, 2017).

Da der Fokus dieser Ausarbeitung auf den eben genannten Tracking-Systemen liegt, wird nachfolgend nur ein kleiner Überblick über den verwendeten VR-Aufbau gegeben. Als Anzeigemedium wird die HTC Vive in Verbindung mit der Entwicklungsumgebung Unity verwendet. Derzeitig verfügt das HMD über eine Auflösung von 2160 x 1200 Pixel und ein Blickfeld von 110°[1]. Unity hat sich als Entwicklungsumgebung für VR-Inhalte durchgesetzt. Es besitzt eine hohe Kompabilität zu unterschiedlichen VR-Headsets und ermöglicht eine unkomplizierte Implementierung von gewünschten Inhalten.

3 Konzept und Umsetzung

3.1 Interaktionskonzept

Um zu Interagieren werden zwei Bereiche definiert. Der feingranulare Interaktionsbereich (FGI) und der Grobselektionsbereich (GS).

Im FGI liegt der Fokus auf Interaktionen mit Objekten und der Umgebung mit den Händen, deshalb bedarf dieser Bereich hochauflösender Sensorik um die 27 Gelenke der Hand ausreichend zu digitalisieren. Der FGI ist immer im Sichtfeld des Users.

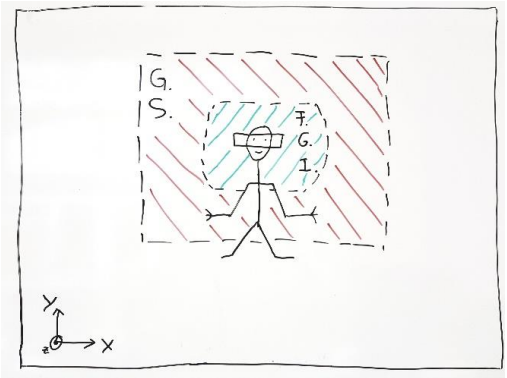

Abbildung 4:Abgrenzung GS/FGI

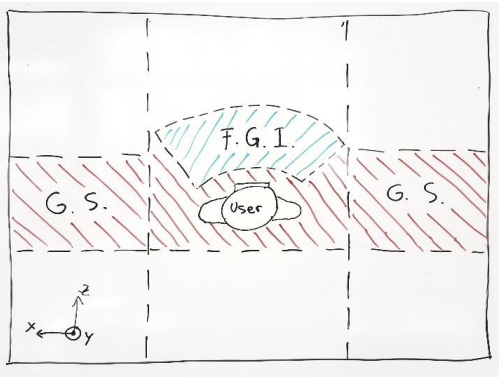

Abbildung 3:Interaktionsbereiche

[1] https://www.digitaltrends.com/virtual-reality/oculus-rift-vs-htc-vive/

Im Grobselektionsbereich (GS) werden Menüs bedient. Im GS soll der ganze Körper verwendet werden. Da dabei hauptsächlich die Gelenke der Extremitäten zu tracken sind, kann der Bereich mit niedriger Auflösung bei höherer Entfernung zu dem Sensor realisiert werden.

Es gibt ein Top-Level-Menü, welches weitere Untermenüs haben kann. Die Menüpunkte können weitere Untermenüs aufrufen oder ein Objekt festlegen welches in der virtuellen Umgebung manifestiert werden soll. Die Menüs werden im Blickfeldmittelpunkt des Users angezeigt und folgen dem Sichtfeld bei Bewegungen des Kopfes.

Durch einfaches Schließen der Hand im GS wird das Menü aufgerufen. Um im Menü zu navigieren, werden mit Rotationsbewegungen des Arms die Menüpunkte selektiert. Der Winkel der linken Hand zur linken Schulter bestimmt die Selektion des Menüpunktes.

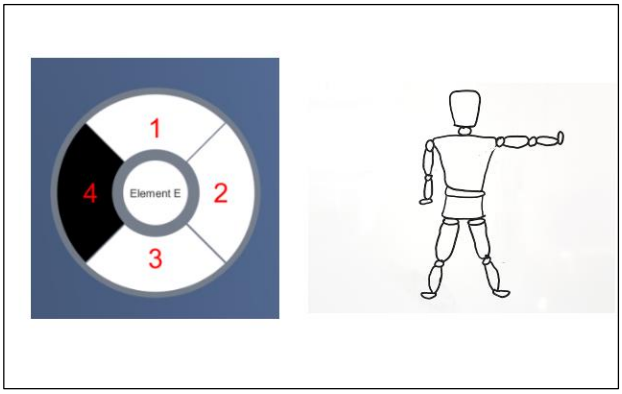

Abbildung 5:Selektion des linken Menüpunktes

Durch erneutes Schließen der Hand wird ein selektierter Menüpunkt aktiviert. Wenn der selektierte Menüpunkt ein Untermenü enthält, wird das Menü ausgeblendet und das neue Untermenü wird angezeigt. Um zum vorherigen Menü zurück zu kehren, ist das oberste Menüelement mit der „Zurück“-Funktion belegt. Das momentane Menü wird ausgeblendet und das vorherige Menü wieder sichtbar gemacht.

Um Menüs auszublenden, müssen die Hände den FGI betreten. Durch Verlassen des FGI und erneutes Schließen der linken Hand wird das zuletzt aufgerufene Menü erneut angezeigt.

Wenn ein Menüelement selektiert und aktiviert wurde, welches kein Untermenü enthält sondern ein Objekt, welches in der Welt manifestiert und manipuliert werden soll, so ist dieses Objekt bei Betreten der Hände des FGI, an der linken Hand verankert. Es kann von dort mit der rechten Hand entnommen und weiter interagiert werden.

3.2 Realisierung

Die Realisierung ist in Komponenten aufgeteilt.

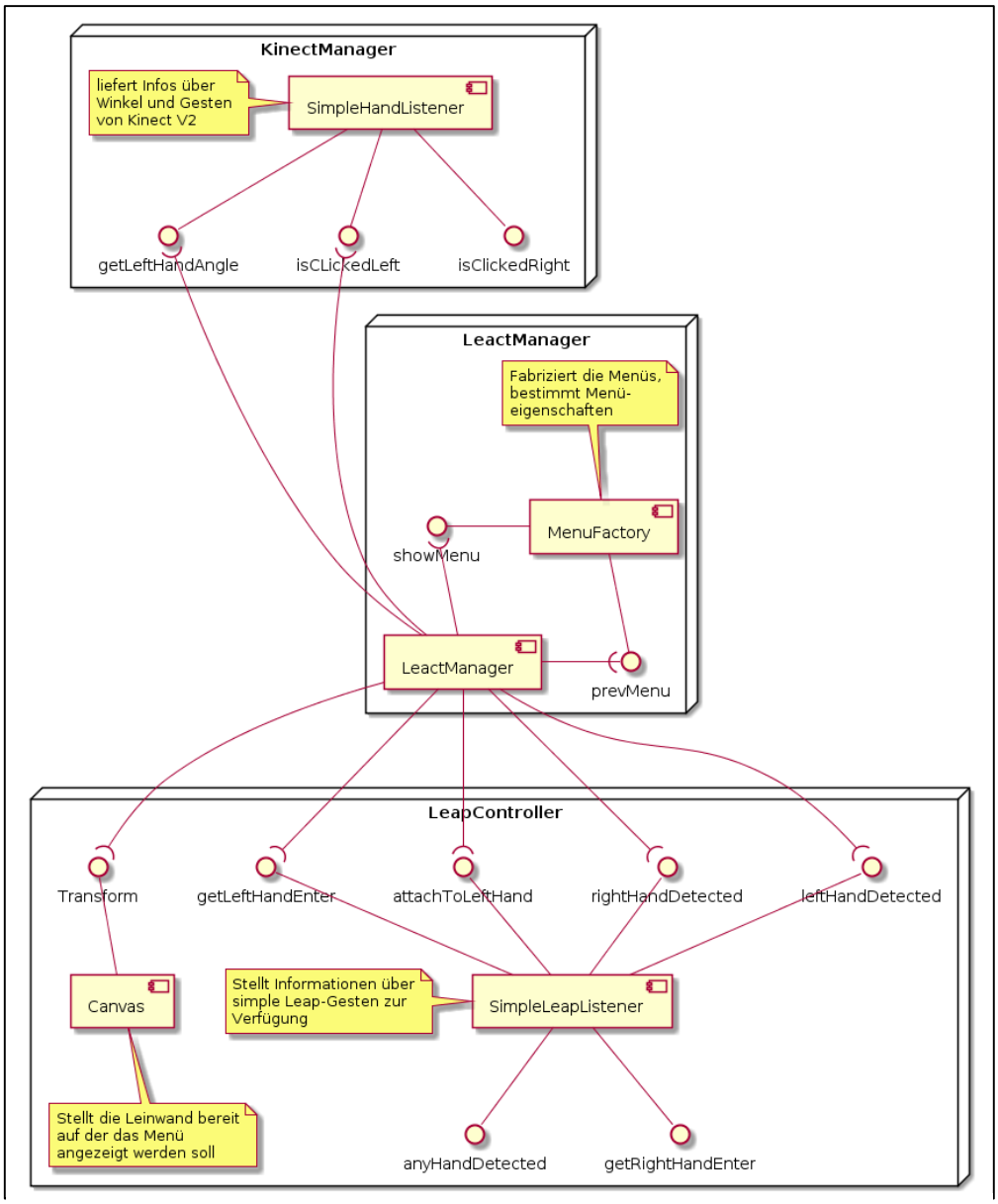

Abbildung 6: Übersicht der verwendeten Leact-Komponenten

3.2.1 Kinect Manager

Der KinectManager ist eine Komponente die es erlaubt Kinect V2 und VR gemeinsam zu verwenden. Die Komponente sorgt dafür dass der Körper des Users getrackt wird. Die Kinect V2 wird verwendet um im GS mit Menüs zu interagieren.

Das Skript SimpleHandListener (SHL) ist ein Bestandteil des KinectManager und musste erweitert werden. Es wird die Information über den Winkel von linker Hand zur linken Schulter benötigt. Diese Information ist eine Erweiterung die im Skript umgesetzt werden muss. Die linke Schulter sowie die linke Hand sind Gelenke, welche die Kinect V2 erkennt und somit die Bewegung trackt.

Das Schließen der linken Hand ist als Linksklick definiert. Das Skript muss um die Methode *isClickedLeft* erweitert werden, welche die Information über den Handzustand (geöffnet / geschlossen) bereitstellt.

Generell arbeitet das SHL-Skript mit den Rohdaten der Kamera und stellt vorgefertigte Funktionalitäten als Fassade bereit.

3.2.2 Leap Controller

Der LeapController ist ein Bestandteil des Orion-SDKs für LM. Er sorgt für das Tracking und Darstellen der Hände und Unterarme. Die Auflösung der LM ist höher gegenüber der Kinect, weshalb die LM für Objekt Manipulation im FGI-Bereich vorgesehen ist. Aus diesem Grund beschränkt sich auch der FGI Bereich auf das Field of View der LM.

Da der LeapController die Hauptkamera für VR enthält, muss die Kamera eine weitere Fläche zum Darstellen der Menüs (Canvas) enthalten. Die Fläche ist das Eltern-Element des Menüs damit diese sich immer im Sichtbereich des Users befinden.

Der LeapController wird um ein selbst implementiertes Skript *SimpleLeapListener* (SLL) ergänzt. Der SLL ist eine Fassade die Informationen bezüglich der LM Sensoren bereitstellt. Um den Zugriff auf diese Informationen zu gewähren, wird durch den Controller auf die aufbereiteten Sensordaten zugegriffen. Der Controller hat Daten der LM über vergangene Frames, aus denen festgestellt wird ob eine Hand den Sichtbereich der LM betritt. Dies ist durch einen Vergleich von dem vorherigen auf den aktuellen Frame realisiert, welcher ein plötzliches Erkennen einer oder beider Hände feststellt.

Die LM Beispiele liefern die Funktionalität einen Anker an der Hand zu definieren. Der SLL kann GameObjects, welche ein AnchorableBahaviour haben, an der linken Hand verankern.

3.2.3 Leact Manager

Der LeactManager ist eine selbst entwickelte Komponente. Er vereint die Daten der Kinect - und LM-Fassade, wodurch das Verhalten bei dem Übergang von FGI zu GS definiert wird. Des weiteren stellt er noch eine MenuFactory, die Menüs generiert und die Menü Historie verwaltet. Die Informationen der erkannten LM Gesten oder Kinect Gesten, werden durch Polling bei jedem Frame gesammelt.

Der LeactManager kann den Übergang von GS zu FGI detektieren, indem der SLL die Information über das Eintreten einer Hand in den FGI bereitstellt. Durch Betreten des FGI werden Modelle der Hand sichtbar, was das Manipulieren von Objekten mit der Hand möglich macht. Sichtbare Menüs werden ausgeblendet indem ihre zugehörigen GameObjects gelöscht werden. Sobald eine der beiden Hände im FGI ist, kann der GS nicht aktiviert werden.

Wenn vor Betreten des FGI ein Objekt in dem jeweiligen Menü gewählt wurde, kann es in der VR manifestiert werden. Dafür wird es bei Betreten des FGI an der linken Hand verankert. Die Fähigkeit Objekte an der Hand zu verankern wird von dem LM-Framework gestellt. Im Anwendungsfall-Kontext eines UML-Editors könnten diese Objekte Modellelemente sein, die in der Umgebung platziert und manipuliert werden sollen.

Wenn sich keine Hände im FGI befinden, kann mit einem einfachen Klick der linken Hand das zuletzt besuchte Menü aufgerufen werden, da sich die MenuFactory die Historie durch eine ID der zuletzt aufgerufenen Menüs merkt.

Diese eindeutigen IDs werden den jeweiligen Menüs und Objekten zugeordnet. Das Menüframework muss um diese Verwaltungsfunktionalität erweitert werden. Anhand der ID wird definiert welche Menüelemente ein Nachfolgemenü haben und wie viele Menüelemente der Nachfolger hat. Die Factory generiert anhand von Prefabs zur Laufzeit neue Instanzen und passt diese entsprechend an, was ein einfaches Anpassen der Menüstruktur ermöglicht.

Das Menüverhalten, also unterschiedliche Handling von Objekten und/oder Menüs, kann komplett in der MenuFactory konzentriert werden. Dies wird durch die ID realisiert, welche bei Objekten grundsätzlich anders ist als bei Menüs. So bleibt die Kontrolle der aufrufbaren Elemente bei der Factory da sie entweder ein neues Menü erzeugt oder ein Objekt an der Hand verankert.

3.3 Verwendete Bibliotheken / Frameworks

Radial Menu Framework[2]: Dieses kostenlose Framework wird im Unity-Assetstore angeboten. Es stellt leicht anpassbare Kreismenüs zur Verfügung, die sich durch kleine Änderungen in die VR integrieren lassen. Es wird mit einer Dokumentation ausgeliefert, was dabei hilft es für die Verwendung mit der MenuFactory anzupassen.

Kinect-v2 VR Examples von Rumen Filkov[3]: Diese Beispiele wurden von Rumen Filkov auf Anfrage bereitgestellt. Sie vereinen die Kinect-V2-SDK Beispiele mit VR und machen die Funktionalität der Kinect V2 in VR nutzbar. Durch die enthaltenen Beispiele und die Online Dokumentation ist die Benutzung und Handhabung gut zu verstehen und leicht zu lernen.

Leap Motion Orion[4]: Die Unity-Core Assets für Leap-Motion bieten Controller und Beispiele für die Leap Motion. Für die Interaktion im FGI und dem Verankern der eingefügten 3D-

[2] https://assetstore.unity.com/packages/tools/gui/radial-menu-framework-50601

[3] https://rfilkov.com/2016/05/07/kinect-v2-mobile-vr-examples/

[4] https://developer.leapmotion.com/unity/

Objekte an der Hand wurde sich an dem LM Beispiel *Dynamic UI* orientiert. Die Beispiele und die Verwendung von LM sind gut dokumentiert.

3.4 Usability und Bewertung

Da der Fokus dieser Ausarbeitung auf der technischen Umsetzung liegt, lassen sich Verbesserungen hauptsächlich im Bereich der Usability finden. Bespielsweise ist derzeitig eine einhändige Menüführung realisiert, welche jedoch aufgrund der einseitigen Armbelastung auf Dauer als anstrengend empfunden werden kann. Würde eine zweihändige Gestensteuerung implementiert werden, welche die Menüaufrufende Hand als dominant definiert, könnten sich die Körperhälften abwechseln. Dadurch würde nicht nur jeder Nutzer seine dominante Körperhälfte nutzen können, sondern auch die subdominante Hälfte für weitere, steuernde Gesten. Derzeitig ist beispielsweise die Geste zum Schließen des Menüs durch ein Hereinführen der linken Hand in den Bereich der LM realisiert. Würde man das Schließen auf eine Wischgeste der rechten Hand legen, könnte die Bedienung erweitert werden. Generell lässt sich diese Anwendung auf weitere Konzepte zur alternativen Bedienung prüfen.

Im technischen Bereich ist ein Hauptaugenmerk auf das Refactoring zu legen. Derzeitig bedingt die Umsetzung mehrere Abänderungen der APIs von LM und Kinect V2. Es wäre jedoch sehr gut, eine Implementierung alleine durch das Einfügen des selbst geschriebenen Packages zu realisieren. Dafür müssen, wie in Abbildung 6 zu sehen, die Fassaden-Methoden des SHL und SLL mit in den Leact-Manager gezogen werden um ein autonom funktionierendes Package zu ermöglichen. Auch kann man das Hinzufügen und Abändern der Menühierarchien vereinfachen, wenn eine Add-Methode im Leact-Manager implementiert wird, welche notwendige Einträge und Verknüpfungen erstellt um eine Einbindung des neuen Menüs zu bewerkstelligen.

Generell ist eine Erweiterung für mehrere, gleichzeitige Nutzer wünschenswert um die Einsatzmöglichkeiten des Frameworks noch umfangreicher zu machen. Die Verwertung von Sensorsignalen innerhalb der vorhandenen Frameworks kann ebenfalls überarbeitet werden um falsche Erkennung von Gesten und Handstellungen zu vermeiden.

4 Zusammenfassung / Ausblick

Während der Bearbeitung dieses Projekts, beziehungsweise dieser Ausarbeitung, wird ein großer Einblick in die VR-Programmierung geliefert. Dabei liegt der Erkenntnisgewinn eher bei der Arbeit mit Unity als beim Verfassen der C#-Skripte. Durch die C#-Entwicklungsumgebung *MonoDevelop*, welche bei Unity standardmäßig für die Bearbeitung der Skripte eingestellt ist, wird durch Autokorrektur- und Vervollständigung das Verfassen stark vereinfacht. Daher ist es leicht, die entwickelten Konzepte für die Programmlogik in Code umzuwandeln. Sehr viel interessanter ist die Arbeit mit dem *Szenen-Konzept* von Unity. Das Einfügen und wichtige verschachteln der Projekt-Strukturen über die *Hierarchy* stellt sich als recht intuitiv heraus. Eine gewisse Einarbeitungszeit benötigt es jedoch, die Umsetzung von angefügten Komponenten an ein Projekt-Objekt nach zu vollziehen. Da benutzte Frameworks meist Skripte verwenden, welche für eigene Implementierungen noch abgeändert werden, ist es teilweise nicht so leicht, das richtige *GameObject* auszuwählen, welches die Skript-Komponente beinhaltet. Generell erfolgt das Einfügen von bereits vordefinierten Projekt-Bauteilen, den sogenannten *Assets*, sehr intuitiv. Durch den offiziellen und integrierten *Asset-Store* können vorprogrammierte Bausteine mit gewissen Funktionalitäten oder Optiken mit wenigen Klicks in das eigene Projekt eingebunden werden.

Generell ist es eine gute Erfahrung zwei Sensoren erfolgreich zu implementieren und durch die Kombination beider Signale eine Erweiterung der bestehenden Funktionalitäten zu erreichen. In dieser Ausarbeitung gelingt es eine Ganzkörper-Gestensteuerung mit hochauflösendem Handtracking zu erweitern. Die Beschränkungen bei der Nutzbarkeit dieses Produktes sind auf die Ungenauigkeiten der Sensorsignale zurückzuführen. Die Daten der Infrarot-Kameras lassen sich nur mit entsprechender Vor- und Nachfilterung durch die mitgelieferten Frameworks verwenden. Dadurch ist man auf die eingestellten Parameter angewiesen und muss bei ungenügender Genauigkeit eigene Algorithmen zur Verbesserung der Signale aufbauen. In dem Rahmen dieses Projektes ist es leider nicht möglich, kann jedoch für zukünftige Weiterentwicklungen in Aussicht gestellt werden.

Grundsätzlich sollte bei einer Weiterentwicklung in Betracht gezogen werden, dass eine offizielle Weiterentwicklung von Leap Motion und Kinect ausgesetzt ist. Daher ist es eventuell nicht sinnvoll neue Ausarbeitungen auf Basis dieser Produkte umzusetzen, welche keinen Support mehr erfahren. Trotz alledem sind diese Sensoren in ihrem Anwendungsfeld noch die besten Implementierungen für die VR. Der Fokus sollte dennoch auf neuen, frischen Produkten liegen, welche zukünftig auf dem Markt erscheinen oder auch nicht direkt zur VR gehören wie beispielsweise die Microsoft HoloLens mit ihrer eingebauten Handgesten-Erkennung.

Literaturverzeichnis

Robertson, A (2016). *Leap Motion's revamped hand tracking is getting built straight into VR headset*. https://www.theverge.com/2016/2/17/11021214/leap-motion-hand-tracker-virtual-reality-orion-mobile-vr

Penelle, B. (2014). *Multi-Sensor Data Fusion for Hand Tracking using Kinect and Leap Motion*.Université Libre Brussels. Brüssel. https://dl.acm.org/citation.cfm?id=2620710

Laugesen, John & Yufei, Yuan (2010). *What factors contributed to the success of Apple's iPhone?*. McMaster University, DeGroote School of Business, Ontario

Preim, Bernhard & Dachselt, Raimund (2015). *Interaktive Systeme, Band 2: User Interface Engineering, Natural User Interfaces*, Berlin, Springer Vieweg Verlag, S 343-396

Warren, Tom (2017). *Microsoft kills of Kinect, stops manufacturing it*. https://www.theverge.com/2017/10/25/16542870/microsoft-kinect-dead-stop-manufacturing

Abbildungsverzeichnis