HAW HAMBURG

Bachelorarbeit

David Godglück

Blended Reality: Ein Hardware-Skelettbaukasten zur Mesh-Animation

Fakultät Technik und Informatik
Department Informatik

Faculty of Computer Science and Engineering
Department Computer Science

David Godglück

Blended Reality: Ein Hardware-Skelettbaukasten zur Mesh-Animation

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung
im Studiengang Bachelor of Science Angewandte Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Philipp Jenke
Zweitgutachter: Prof. Dr. Stefan Sarstedt

Eingereicht am: 7. Dezember 2018

David Godglück

Thema der Arbeit

Blended Reality: Ein Hardware-Skelettbaukasten zur Mesh-Animation

Stichworte

Eingabegerät, Baukasten, Rigging, Skinning, Blended Reality, skelett-basierte Animation

Kurzzusammenfassung

Das Gestalten von Bewegungen in virtuellen Welten ist ein zeitaufwändiger Prozess. Dabei ist diese Aufgabe nicht nur auf Animationsfilme beschränkt, sondern ist auch bei der Erstellung von computergenerierten Effekten und in Videospielen zu finden. Um den Einstieg für Laien und den Aufwand für Animationskünstler zu verringern, wird für den Anwendungsfall der skelett-basierten Bewegungsanimation im Rahmen dieser Bachelorarbeit ein geeignetes Eingabegerät entwickelt werden.

David Godglück

Title of Thesis

Blended Reality: A Modular Input Device for Mesh-Animation

Keywords

Construction Kit, Input Device, Rigging, Skinning, Blended Reality, Skeletal-based Animation

Abstract

Character animation is a time consuming activity. This does not only affect computer generated imagery in modern films, but also animated films and video games. To lower the barrier for beginners in these fields of work and to speed up the processese for animation artists, a modular input device for skeletal-based animation will be developed as the major part of this bachelor thesis.

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einführung

1.1 Problemstellung

Ein Bereich, viele Anwendungen: Heutzutage werden in mehreren Bereichen der Computergrafik verschiedene Techniken der Mesh-Animation angewandt. Dabei geht es um die dynamische Anpassung von Dreiecksnetzen durch äußere Einflüsse. Ohne diese Technik wären moderne Computerspiele, Animationsfilme und computergenerierte visuelle Effekt (eng. **C**omputer **G**enerated **I**magery, CGI) nicht zu realisieren.

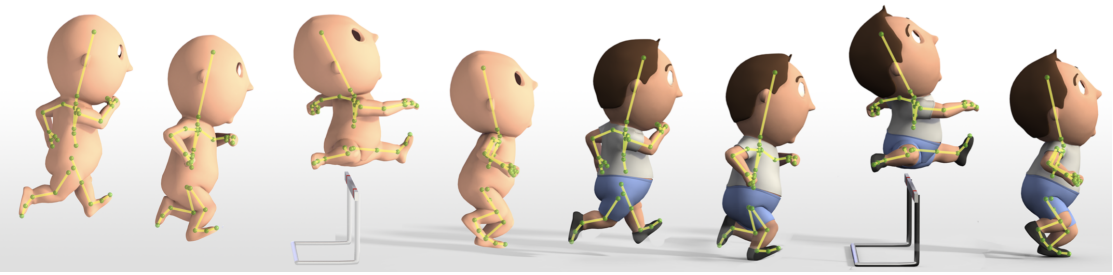

Abbildung 1.1: Bildfolge eines animierten Hürdenlaufs. Links: Mit einfarbigen Dreiecksnetz und dem hervorgehobenen Skelett; Rechts: Farbig und mit Ergänzungen am Model.

Quelle: [LH05]

Meist basiert die Transformation des Dreiecksnetzes auf einem Skelett. Wie in der Realität folgt dabei beispielsweise die Haut eines Oberschenkels dem Oberschenkelknochen. Das virtuelle Skelett hat dabei nicht die Aufgabe ein realistisches Skelett zu bilden, sondern sollte darauf ausgelegt sein, Bewegungen möglichst realistisch wirken zu lassen. In Abbildung 1.1 lässt sich gut erkennen, wie die einzelnen Extremitäten und Teile der Figur der Position und Orientierung der Knochen folgen.

Damit nicht für jedes generierte Bild bei einem Animationsfilm eine Figurenpose modelliert werden muss, werden nur die sogenannten Schlüsselbilder (eng.: Keyframes) festge-

legt. Für die Zwischenbilder wird dann bei diesem Verfahren eine Zwischenpose aus den Schlüsselposen generiert. Die Modellierung dieser Schlüsselposen bildet den Kern dieser Arbeit.

Der Aufwand diese Posen per Hand zu erstellen ist, unabhängig von der unter Umständen hohen Anzahl, sehr groß. Mit dieser Bachelorarbeit soll ein geeignetes Eingabegerät entwickelt werden, welches die Arbeit für Animationskünstler massiv erleichtert. Die Idee hierfür basiert auf der Arbeit des **I**nteractive **G**eometry **L**ab (IGL) der ETH Zürich. Die Autoren der Veröffentlichung haben sich genau diesem Problem gewidmet und geeignete Hardware entwickelt [GMP+16] [JPG+14].

1.2 Ziel der Arbeit

Diese Bachelorarbeit orientiert sich an der Arbeit des IGL der ETH Zürich. Da die Arbeit des IGL nicht vollständig, aber voll funktionsfähig ist, wird diese als Basis zur Entwicklung einer simpleren Hardware genutzt. Die Konzeption und die Umsetzung eines Hardware Baukastens, und vor allem das Gestalten von Bewegungen in virtuellen Welten ist zeitaufwendig. Daraus resultiert ein weiteres Ziel dieser Arbeit: Einen einfachen Einstieg für Laien schaffen. Denn der Zugang zu dieser Thematik und der Nachbau der IGL-Hardware ist kompliziert – vor allem, wenn kein Fachwissen und keine Erfahrungen im Bereich der Elektrotechnik vorhanden sind. Aufgrund dessen wird in dieser Facharbeit für den Anwendungsfall der skelett-basierten Bewegungsanimation ein geeignetes Eingabegerät entwickelt. Dieses Eingabegerät soll die besonderen Anforderungen erfüllen und zeitgleich möglichst kostengünstig sein.

Ein weiteres Ziel ist es, dass die aus dem theoretischen Teil dieser Bachelorarbeit gewonnen Erkenntnisse und Ergebnisse zur Entwicklung beitragen. Hierbei soll es möglich sein mit Hilfe von modularen Bauteilen verschiedene Skelettaufbauten nachzubilden. Hinzukommend wird eine entsprechende Software entwickelt, die mit der konzipierten Hardware kommuniziert und diese als Skelett visualisiert - einfache Mesh-Animationen sollen so in Echtzeit ermöglicht werden.

1.3 Aufbau und Gliederung

Für die Bearbeitung der vorliegenden Bachelorarbeit *Blended Reality: Ein Hardware-Skelettbaukasten zur Mesh-Animation* wurde zunächst ein theoretischer Rahmen gesetzt, welcher nach der Einarbeitung durch einen praktischen Teil komplementiert wurde. Insgesamt ist diese Bachelorarbeit in sieben Kapitel unterteilt. Das **erste Kapitel** führt an die Thematik dieser wissenschaftlichen Arbeit heran, hier wird sowohl auf die *Problemstellung*, das *Ziel der Arbeit* sowie den *Aufbau* und die *Gliederung* eingegangen.

Im **zweiten Kapitel** *Grundlagen* folgen Begriffserklärungen, hier werden relevanten Grundbegriffe erklärt, die für das Verständnis dieser Bachelorarbeit wichtig sind. Es werden unter anderem Begriffe wie *Hallsensor*, *Dreiecksnetze* und *Szenengraph* vermittelt.

Das **dritte Kapitel** *Stand der Technik* beinhaltet eine genauere Beschreibung der aktuellen Entwicklung der Teilbereiche dieser Arbeit. Darüber hinaus wird die Hardware des IGL analysiert.

Im **vierten Kapitel** folgt die Konzeption der Hardware und Software. Als erstes wird auf die Problematik der bereits bestehenden Hardware eingegangen und im zweiten Schritt die Lösung dargelegt. Ergänzend hierzu wird auf die Erweiterung und Entwicklung der Skeletterstellung und die damit verbundenene Fehlerkorrekturen eingegangen.

Das **fünfte Kapitel** beschreibt die Umsetzung, welche auf den Ergebnissen des vierten Kapitels, den theoretischen Grundlagen sowie eigener Überlegungen basieren. Diese bilden die Grundlage für die Weiterentwicklung der Hardware, ohne Hallsensoren.

Im **sechsten Kapitel** wird eine Bewertung und Gegenüberstellung vorgenommen – die Evaluation dieser Bachelorarbeit.

Das **siebte Kapitel** bildet den inhaltlichen *Abschluss* dieser Arbeit. Es beinhaltet das Fazit und den Ausblick.

Abschließend folgt das Literaturverzeichnis.

2 Grundlagen

2.1 Hallsensor

Mithilfe eine Hallsensors lässt sich im Allgemeinen ein Magnetfeld messen. Hierfür wird der Hall-Effekt genutzt. Dieser besagt, das eine elektrische Spannung auftritt, wenn sich ein stromdurchflossener Leiter in einem stationären Magnetfeld befindet. Ein weit verbreiteter Anwendungsfall dieser Sensoren, in Verbindung mit Permanentmagneten, ist die Messung von Winkeln oder Winkelveränderungen zum Magnetfeld im Raum.

2.2 Dreiecksnetze

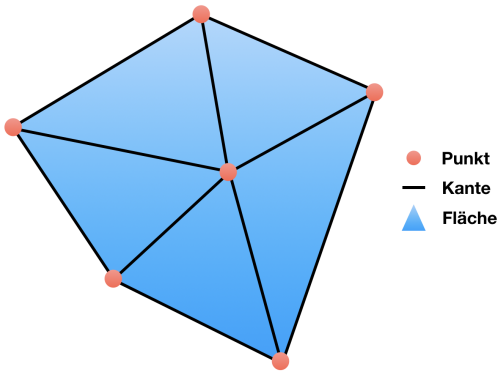

Abbildung 2.1: Ein einfaches Dreiecksnetz.
Quelle: Eigene Darstellung

Dreiecksnetze bestehen aus Punkten (genauer Vertices, Sg. Vertex), Kanten und Flächen. Es gibt unterschiedliche Datenstrukturen, die, je nach Anwendungsfall, diese drei Elemente oder weitere Strukturen verwenden. Eine Art eine einzelne Dreiecksfläche zu

definieren, ist z.B die Angabe der drei anliegenden Eckpunkte oder Kanten. Viele dieser Dreiecksflächen ergeben ein größeres Gebilde, ein sogenanntes Netz.

2.3 Szenengraph

Szenengraphen haben meist den Zweck, Objekte und Gruppen einer Szene in der Computergrafik logisch und/oder räumlich anzuordnen und zu trennen. Diese verwaltende Datenstruktur ermöglicht den Einsatz von unterschiedlichen Verfahren, z.B. zur Beschleunigung von Render-Vorgängen oder zur leichteren Organisation/Verwaltung von Objektgruppen.

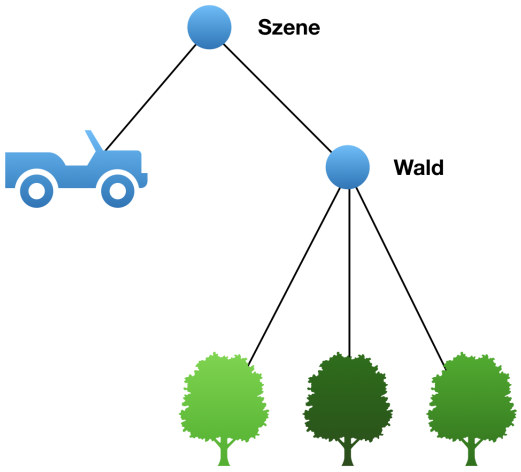

Abbildung 2.2: Ein einfacher Szenengraph
Quelle: Eigene Darstellung

Ein kleiner Szenengraph ist in Abbildung 2.2 zu sehen. Angenommen man würde gerne den kompletten Wald in der virtuellen Welt verschieben, dann müsste man in diesem Fall nicht jeden einzelnen Baum verschieben, sondern könnte den Szenengraphen um einen Transformations-Knoten ergänzen. Zwischen dem Szenen- und dem Waldknoten platziert, würde er alle untergeordneten Kindknoten beeinflussen. Transformationen umfassen die Skalierung, Translation und Rotation.

2.4 Skelett

Für die Animation von Figuren in der Computergrafik wird oft ein skelett-basiertes Verfahren genutzt. Wie bei Wirbeltieren besteht ein Skelett (eng. Skeleton) aus Knochen (eng. Bones). Sie werden, wie ein Szenengraph, in einer hierarchischen Struktur verwaltet. Kindknochen, also diejenigen Knochen, die mit dem Ende eines Elternknochen verbunden sind, werden unter ihnen im Graph platziert. Knoten im Graphen können als Gelenke (eng. Joints) bezeichnet werden. Ein fertiges Skelett wird auch Rig genannt.

2.5 Rigging

Rigging bezeichnet Verfahren, bei denen ein Rig aus Knochen und Gelenken konstruiert wird. Es wird festgelegt wie die Teile eines Dreiecknetzes bewegt werden.

2.6 Skinning

Beim Skinning wird das erstellte Skelett aus einem Rigging-Verfahren mit einem Dreiecksnetz gekoppelt. Unter dem Begriff zusammengefasst werden auch Methoden wie zum Beispiel optimierte Rotationszentren [LH05] oder Volumenerhaltung, um auftretende Artefakte und Schönheitsfehler im dargestellten Dreiecksnetz zu beseitigen.

3 Stand der Technik

3.1 Die Hardware des IGL der ETH Zürich

Wie bereits erwähnt basiert diese Arbeit auf der Entwicklung des IGL. Viele Unterlagen zur Hardware stehen auf Github[1] zur Verfügung. Darunter befinden sich 3D-Modelle für die Gehäuse, Schaltpläne für alle verbauten Platinen, die Firmware der verwendeten Mikrocontroller, Treiber zur Ansteuerung sowie weitere Software. Die Problematik: Es sind keine weiteren Erläuterungen oder eine Anleitung für einen Nachbau verfügbar. Im Rahmen der Arbeit [GMP+16] wurden im Wesentlichen drei miteinander koppelbare Elemente entwickelt. Zum einen Gelenk-Elemente mithilfe dessen die Orientierung zwischen zwei Knochen gemessen werden kann. In Abbildung 3.1 sind diese in rot, blau und grün eingefärbt - sie sind jedoch identisch.

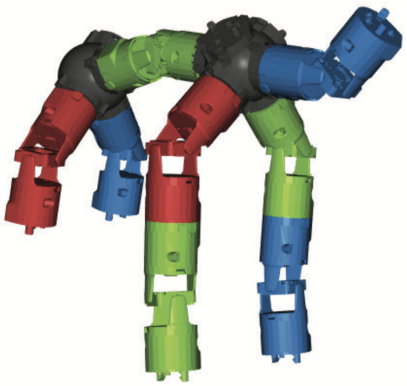

Abbildung 3.1: Hardware-Elemente des IGL als 3D-Modell.
Quelle: [GMP+16]

Die schwarzen Bauteile in Abbildung 3.1 werden in [GMP+16] als Splitter-Elemente bezeichnet. Zum besseren Verständnis werden diese im weiteren Verlauf der Arbeit Kupplungs-

[1] https://github.com/oliglauser/atamid

Element genannt. Sie ermöglichen das Koppeln von mehreren Gelenk-Elementen miteinander. Zwei dieser Gelenke lassen sich auch direkt miteinander verbinden, wie in Abb. 3.1 zu sehen. Kupplungselemente gibt es in verschiedenen Ausführungen, sie unterscheiden sich in der Anzahl der möglichen Verbindungen von Gelenk-Elemente und der physikalischen Ausrichtung dieser Kupplungen. Die Elektronik in den Kupplungs-Elementen ist relativ simpel: Ein Datenkanalmultiplexer sorgt für die Weiterleitung der Daten der angeschlossenen Elemente. Auf der Visualisierung nicht vorhanden sind die Bone- oder Knochen-Elemente. Diese sind hohle Röhren ohne Elektronik und dienen in [GMP+16] nur dazu, die physikalische Gestalt des Skelettmodells proportional an die der virtuellen Abbildung anzunähern. Kabel verbinden im Inneren lediglich die Steckverbindungen an den Enden. In der Ausarbeitung und den Beispielvideos wurde diese Art von Baustein nicht erwähnt. Der Vollständigkeit halber sei auch noch ein weiteres Element erwähnt, es handelt sich um das Netzteil, welches auch das Interface zum Computer bereitstellt. Für dieses Bauteil sind ebenfalls Gehäuseteile als 3D-Modell vorhanden.

Am interessantesten sind die Gelenk-Elemente. Jedes von ihnen besteht aus 10 Gehäuseteilen, die ineinander gesteckt und mit kleinen Schrauben fixiert werden. Die Abmessungen belaufen sich in der Länge auf 7,6cm und im Durchmesser auf maximal 2,4cm. Außerdem werden sechs runde Platinen mit einem Durchmesser von unter 2cm verbaut (siehe Abbildung 3.2).

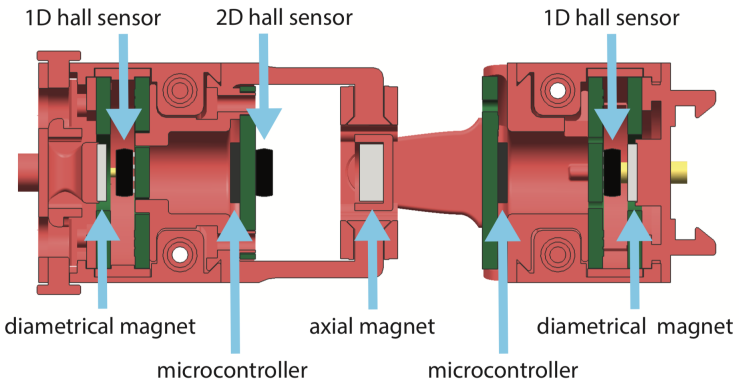

Abbildung 3.2: Querschnitt eines Gelenk-Elements des IGL
Quelle: [GMP+16]

Mithilfe von drei Hallsensoren und drei Magneten werden die Parameter des Gelenks ermittelt. Dafür notwendig sind zwei 1D-Hallsensoren an den Enden des Elements, zur Erfassung der Rotationen der Achsen, und ein 2D-Hallsensor, um den Knickwinkel des

Gelenks zu ermitteln. Die erfassten Daten werden auf den zwei inneren Platinen mit Mikrocontrollern verarbeitet, gekennzeichnet in Abbildung 3.2. Jeweils ein 1D-Sensor ist mit einem Mikrocontroller verbunden, der 2D-Sensor nur mit dem (im Querschnitt) linken. Die Mikrocontroller-Platinen sind miteinander über Kabel verbunden und jeweils mit den drei Kontakten an den Kupplungsenden.

Um eine unendliche Drehbewegung zu ermöglichen werden Schleifringe mit drei Kontakten verbaut. In Abbildung 3.2 sind diese nicht gekennzeichnet, es handelt sich aber um die beiden äußeren Platinenpaare. Die Arbeit des IGL basiert auf einer zwei Jahre älteren Veröffentlichung [JPG+14], ebenfalls größtenteils von der ETH Zürich und teilweise den gleichen Autoren. Eine wesentliche Verbesserung der Hardware ist die zusätzliche Freiheit in der Rotation des Elements, Tests haben dabei gezeigt, dass sich die nötige Zeit für das Erreichen einer bestimmten Pose deutlich reduziert. [GMP+16]

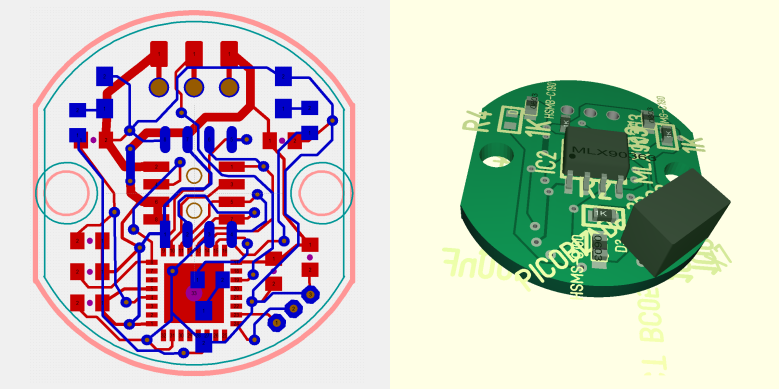

Abbildung 3.3: Visualisierung vom Mikrocontroller-Board wie es im Gelenk-Element verwendet wird, links als Schaltplan mit den beidseitigen Leiterbahnen in rot und blau, rechts als 3D-Modell mit Bauteilen bestückt.
Quelle: Eigene Darstellung

Abbildung 3.3 zeigt die wichtigste Hardwarekomponente des Gelenk-Elements. Zwei dieser Platinen sind verbaut, einmal mit 2D-Hallsensor bestückt und einmal ohne. Zu sehen sind die vielen filigranen Leiterbahnen auf Vorder- und Rückseite der Leiterplatte bei einem Durchmesser von etwa 2cm und viele kleine Einzelbauteile, wie Widerstände, Kondensatoren und LEDs. Die Hardware sollte nachgebaut werden, für einen Eigenbau mithilfe von Lochrasterplatinen ist dieses Bauteil zu winzig, beziehungsweise der Platz im Inneren der Gehäusebauteile nicht ausreichend. Von Nachbauten durch Ätzen der Platinen mit beispielsweise der Direkt-Toner-Methode wurde mir abgeraten. Darüber hinaus ist die Stückzahl der Platinen problematisch. Der Arbeitsaufwand wäre schlichtweg zu

hoch.

Die Herstellung der Leiterplatten kann aber auch von externen Dienstleistern durchgeführt werden. Jedoch sind bei der benötigten, relativ kleinen, Stückzahl und den unterschiedlichen Layouts die Kosten sehr hoch. Die Preise variieren stark. Je nach Dienstleister und Layout und bewegen sich in einem Bereich von 5-30 Euro pro Stück. Für ein Gelenk-Element müssten mit Kosten in Höhe von etwa 40 Euro für die Platinen gerechnet werden. Dazu kommen noch die Kosten für die Bauteile, die noch selbst aufgelötet werden müssten. Sie bewegen sich im Bereich von etwa 20 Euro für ein Gelenk.

Desweiteren fehlen bei dieser Kostenabschätzung noch die Schleifringe. Pläne hierfür sind in den veröffentlichten Unterlagen nicht zu finden. Einer der Autoren von [GMP+16] stellte, nach einem kurzen Mail-Austausch, neben den Datenblättern für die verwendeten Permanentmagnete auch die Baupläne für die Schleifringe zur Verfügung. Die speziellen Schleifringe für ihre Hardware wurden bei einem chinesischen Zulieferer in Auftrag gegeben und gefertigt. Die Schätzungen für ein Schleifring-Paar liegen bei etwa 15 Euro, damit belaufen sich die Gesamtkosten für ein Gelenk auf mindestens 80 Euro.

Die Rekonstruktion des Aufbaus der Hardware hat viel Arbeitszeit in Anspruch genommen. Viele Einzelheiten der Arbeit mussten hergeleitet werden oder wurden erst durch den Nachrichtenaustausch bestätigt. Eine Bauanleitung mit Anmerkungen zum Nachbau war geplant, wurde jedoch nie erstellt. Wegen der relativ schlechten Aussichten auf einen problemlosen Ablauf des softwareseitigen Teils des Nachbaus und der hohen Kosten wird die Hardware für diese Bachelorarbeit neu entwickelt.

Die Idee physikalische Objekte für die Erstellung von Bewegungen in virtuellen Welten zu nutzen ist nicht neu. In den letzten Jahren wurden viele verschiedenen Ideen verfolgt und getestet. Von Motion-Capturing-Techniken zur Erfassung von Puppen auf ein virtuelles Abbild [FGDJ08] über die Übertragung von Bewegungen von einem virtuellen Objekt auf ein anderes [Gle98] bis zu Modellen des virtuellen Ebenbildes [YSC+].

Darüber hinaus existieren auch noch weitaus aufwendigere Motion-Tracking-Techniken, wie sie beispielsweise bei der Produktion von Filmen eingesetzt werden. Hierbei werden die Bewegungen von Schauspielern aufgezeichnet und für visuelle Effekte weiter verarbeitet.

3.2 Rigging

Das erste Verfahren zur skelett-basierten Animation wurde bereits 1988 veröffentlicht [MTLT88]. Sogut wie jede moderne Animations-Software unterstützt die Erstellung von Skeletten und verschiedene Optimierungsverfahren. Bei dem Prozess der Erstellung oder Generierung eines Skeletts geht es nicht darum, ein möglichst realitätsnahes Skelett nachzubilden, sondern ein vorzugsweise optimales Grundgerüst für die Deformation eines Dreiecksnetzes zu erstellen.

Der Ansatz der skelett-basierten Animation verändert sich dabei nicht grundlegend. Allerdings entstehen neue Methoden, um die Prozesse des Riggings zu beschleunigen oder die sichtbaren Ergebnisse zu verbessern. Beispielsweise Verfahren zur automatischen Generierung eines Skeletts für unterschiedliche Dreiecksnetze [BTST12] oder Ansätze mit elastischen Körpern für Simulationen [CBC+05], die die Animationsarbeiten für viele Anwendungen erheblich erleichtern.

3.3 Skinning

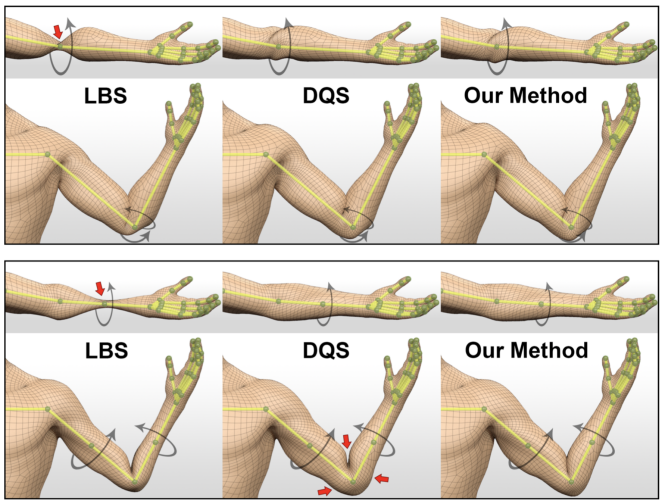

Abbildung 3.4: Vergleich von Korrekturverfahren von kritische Stellen eines deformierten Dreiecksnetzes.

Quelle: [LH05]

Skinning ist der größte und wichtigste Bereich der skelett-basierten Animation. Bei der Deformation von Dreickecksnetzen treten an kritischen Stellen Artefakte auf, wie beispielsweise an einem Ellenbogen.

In Abbildung 3.4 werden sind die Ergebnisse unterschiedlicher Korrekturverfahren verglichen. Zu sehen ist, dass bei den älteren Verfahren **L**iner **B**lend **S**kinning (LBS) und **D**ual **Q**uaternion **S**kinng (DQS) in einigen Szenarien trotzdem ungewollte Deformationen eintreten. LBS nutzt für die Positionsänderung eines Vertex im Netz nicht nur einen Knochen, sondern mehrere umliegende, gewichtete Knochen. Der abgebildete Fehlerfall wird Candy-Wrapper-Artifact genannt und tritt auf, wenn Knochen zu stark rotiert werden. Eine Lösung dies auszugleichen wäre die Anpassung des Skeletts durch das Hinzufügen von Knochen im Rigging-Schritt. Beim DQS-Verfahren tritt dieses Artefakt nicht mehr auf, dafür jedoch Ausbeulungen (siehe Abbildung 3.4: Mit roten Pfeilen markiert).

Die entwickelte Methode aus [LH05] (rechts in Abbildung 3.4) generiert in diesen beiden Fällen eine sehr realitätsnahe Deformation. Dieses Verfahren basiert auf der Optimierung der Rotationszentren der Knochen. Wie das beispielhaft genannte Verfahren gibt es andere Lösungsansätze um diese Artefaktbildung zu minimieren. Das Ziel: Die Generierung von noch realistischeren Animationen. Siehe hierzu auch andere interessante Verfahren aus [LCF00], [WPP13], [CBC+05] und [VBG+13].

4 Konzeption

4.1 Hardware

Zwei Aspekte rücken innerhalb dieses Kapitels zunehmend in den Fokus: Zum einen der Eins-zu-eins-Nachbau der IGL-Hardware, aufgrund der problematischen Umsetzung und zum anderen die daraus resultierende Entwicklung eigener Hardware auf Basis des Interactive Geometry Labs.

4.1.1 Die IGL-Hardware und ihre Problematik

Damit die Hardware des IGL erfolgreich nachgebaut werden kann, müssen fortgeschrittene Kenntnisse im Bereich der Elektrotechnik vorhanden sein. Eine Nachbau wäre jedoch durch die Anfertigung der Platinen von externen Dienstleistern möglich, allerdings ist dies und die Beschaffung der Schaltkreisbausteine sehr kostenintensiv. Darüber hinaus gestaltet sich eine Herstellung in Eigenleistung als problematisch, da die Platinen sehr klein (<2cm im Durchmesser) sind. Aufgrund dessen können die Lochplatinen weder gelötet noch die Schaltkreise geätzt werden.

4.1.2 Die Lösung: Hardware mit anderen Anforderungen

Die angerissene Problematik des Hardware-Nachbaus, im Unterkapitel 4.1.1, wird durch die Entwicklung eigener Hardware kompensiert. Hierbei wird die angefertigte Hardwarelösung keine Funktionalität einbüßen.

Anforderung an die Hardware

Die Hardware muss in der Lage sein die Orientierung von zwei Achsen, im Skelett als Knochen, zu erfassen. Anders gesagt: Der Winkel eines Gelenkes zwischen zwei Knochen

muss gemessen werden, dabei sollte mindestens der Knickwinkel zur Achsenorientierung des Knochens orthogonal dazu oder in x-,y- und z-Richtung sein.

Umsetzung auf Basis des IGL

Zur Erfassung der Parameter eines Knick-Elements nutzt die Hardware des IGL drei Hallsensoren. Die drei Sensoren setzten sich dabei aus zwei 1D-Hallsensoren für die Rotation der Verbindungen zu den Knochen und einem für den eigentlichen Winkel zusammen. Da diese Umsetzung jedoch Komplex und ein direkter Nachbau durch Standartkomponenten und begrenztem Raum nicht möglich ist wird innerhalb dieser Facharbeit eine Problemlösung auf Wissensbasis des IGL erarbeitet.

Anforderungen

Durch die Probleme beim Nachbau der Hardware des IGL ergeben sich folgende Anforderungen an die zu entwickelnde Hardware:

- Nachbaubar ohne Fachwissen: Das Hauptproblem der IGL-Hardware ist der schwierige Nachbau, deshalb soll sich die Neuentwicklung leicht reproduzieren sein.
- Kostengünstig: Die IGL-Hardware ist nicht günstig nachzubauen, die neue Hardware soll erschwingbarer werden.
- Erhalt der Funktionalität: Die Hauptfunktionalität der Hardware darf nicht verloren gehen und soll weiterhin die Modellierung von Schlüsselposen ermöglichen.

Hardware-Komponenten

Eine Erfassung der Orientierung mithilfe von Gyroskop-Sensoren wäre relativ simpel umzusetzen. Für die Erfassung der Parameter eines Gelenks müssten die Orientierungen der anliegenden Knochen erfasst werden. Aus diesen Informationen lassen sich, sofern überhaupt benötigt, die Winkel des Gelenks bestimmen, wenn die Anordnung der Knochen zueinander bekannt ist.

Um die Arbeit mit dem Baukasten komfortabler zu gestalten, soll jedes Knochen-Element ein Mikrocontroller-Board und einen Sensor erhalten. Dadurch müssen keine Kabel zwischen den Elementen verlegt werden und die Bewegungsfreiheit bleibt erhalten. Die Sensordaten sollen kabellos übertragen werden.

Genutzt werden sollen Arduino- oder besser die kostengünstigeren ESP32-Mikrocontroller-Boards, da diese bereits mit WLAN und Bluetooth ausgestattet sind. Verschiedene Boards und Gyroskop-Sensoren sind im CSTI-Labor (**C**reative **S**pace for **T**echnical **I**nnovations) der HAW Hamburg bereits vorhanden und können für diese Arbeit ausprobiert und genutzt werden.

Die 3D-Modelle für Knochen-, Gelenk- und Verbindungselemente des IGL sollen weiter verwendet werden, da sie einen modularen Aufbau von unterschiedlichen Skelettmodellen problemlos ermöglichen. Um die Modelle mithilfe eines 3D-Druckers herstellen zu können müssen sie jedoch angepasst werden.

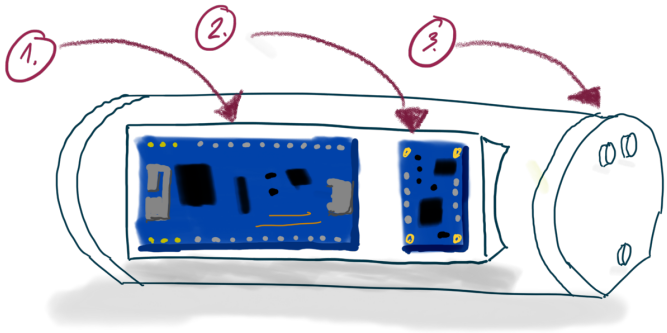

Abbildung 4.1: Skizze für Knochen-Element; 1.) Mikrocontrollerboard, 2.) Gyroskop-Sensor, 3.) 3D-gedrucktes Gehäuse

Quelle: Eigene Darstellung

Für die Datenübermittlung ist die Verwendung von HTTP-Servern über eine kabellose Netzwerkverbindung auf den Boards geplant. Die Rigging/Skinning-Software fragt die Daten per HTTP-Anfrage ab. Die Funktionalität wurde bereits getestet und hat sich als ausreichend zuverlässig und schnell erwiesen.

Die Energieversorgung der Boards könnte im weiteren Verlauf der Arbeit durch Akkus realisiert werden, dadurch wären die Knochenelemente komplett kabellos nutzbar. Das Ziel ist es, die Elemente ohne Kabelverbindungen zueinander und zum Client zu gestalten. Dies wäre eine komfortable Weiterentwicklung der Hardware des Interactive Geometry Labs.

4.2 Rigging

Für eine Visualisierung wird grundsätzlich Software benötigt. Die Computer-Grafik-Gruppe der HAW Hamburg bietet hierfür ein geeignetes Framework[1] zur Nutzung an, dieses ist in Java geschrieben und basiert auf der OpenGL-API. Vorhanden sind bereits Strukturen zur Erstellung eines Szenengraphen, verschiedene Transformations-Knoten und Interaktionsmöglichkeiten.

Für das Skelett wird eine geeignete Datenstruktur benötigt, es besteht aus Knochen und Gelenken. Ein Knochen erfordert einen Startpunkt, wie bei einem realen Skelett ist dies das Ende eines Nachbarknochens. Für die Abstraktion eines Skeletts bietet sich also einen hierarchische Struktur an. Das komplette Skelett hat als Einstiegspunkt einen Startknochen der nur einen Startvektor beinhaltet und keinen Elternknochen als Startpunkt. Alle anderen Knochen besitzen einen Elternknochen, dessen Endpunkt gleichzeitig ihr eigener Startpunkt im Koordinatensystem ist. Die Endpunkte sollen durch die Länge und Orientierung der einzelnen Knochen berechnet werden. So wird eine eventuelle mehrfache Datenverwaltung vermieden.

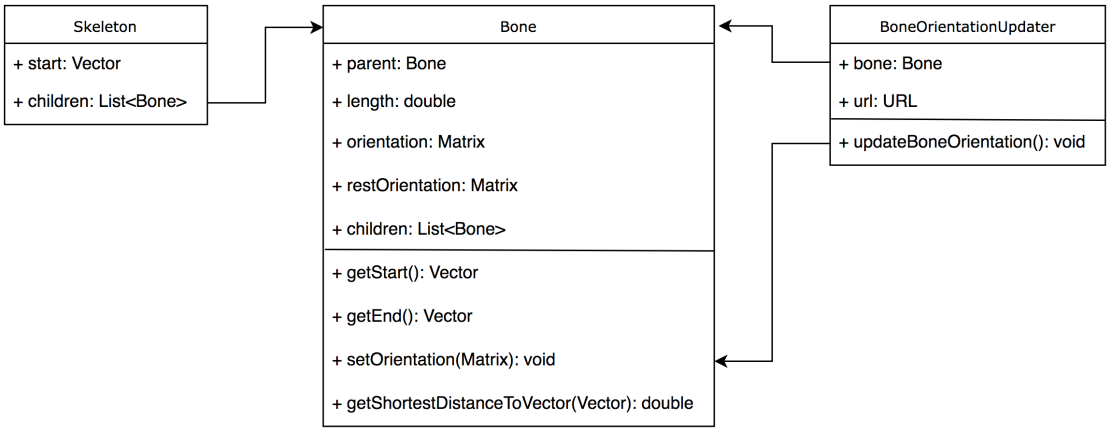

Abbildung 4.2: Benötigte Klassen für die Abstraktion eines Skeletts als UML-Diagramm
Quelle: Eigene Darstellung

Neben der aktuellen Orientierung, die durch eine BoneOrientationUpdater-Entität und dem Hardware-Knochen in Echtzeit aktualisiert werden soll, wird für jeden Knochen zusätzlich eine Ruhe-Orientierung benötigt. Für das Skinning-Verfahren muss das Skelett in einer klar definierten Pose mit dem Dreiecksnetz verbunden werden. Diese Pose setzt

[1] https://gitlab.informatik.haw-hamburg.de/vr_ar_lab/cg_vr_ar

sich aus den einzelnen Orientierungen der Knochen zusammen. Darüber hinaus verfügt jeder Knochen über eine Liste von seinen Kindknochen, also denjenigen Knochen, die mit seinem Ende verbunden sind.

Wie in Abbildung 4.2 zu sehen ist, soll ein Skelett-Objekt aus einem Startvektor bestehen und einer Liste von Knochen. Alle Knochen in dieser Liste beginnen im Startvektor und enden jeweils in einem Endpunkt. Diese Endpunkte ergeben sich aus der Länge und Orientierungs-Matrix des jeweiligen Knochen. Jeder einzelne Knochen kann wiederum beliebig viele Kindknochen besitzen, die in seinem Endpunkt starten.

4.3 Skinning

Für das Skinning wird neben dem Skelett ein Dreiecksnetz benötigt. Das verwendetet Framework beinhaltet schon Datenstrukturen und Abläufe zur Darstellung, Manipulation und Generierung von Dreiecksnetzen. Neben der Erstellung per Hand können auch Daten im offenen Wavefront OBJ-Dateiformat[2] eingelesen werden.

Wenn nun ein geeignetes Dreiecksnetz und Skelett vorhanden sind, kann das Skinning durchgeführt werden. Für eine spätere Animation muss jeder Vertex des Dreiecksnetzes seinem dichtesten Knochen folgen. Dieser Knochen muss für jeden Vertex bestimmt und in einer geeigneten Datenstruktur gespeichert werden. Hierfür soll eine Map genutzt werden, diese Datenstruktur bietet sich an, da sie nur einmal erstellt und danach nur noch gelesen und nicht mehr verändert wird.

Der Ablauf für die Erstellung dieser Daten sieht wie folgt aus:

Data: Dreiecksnetz, Skelett
Result: Map für Vertex und Knochen
forall *Vertices im Dreicksnetz* **do**
  forall *Knochen im Skelett* **do**
    kleinsten Abstand vom Knochen und Position des Vertex berechnen;
  end
  Vertex und dichtesten Knochen der Map hinzufügen;
end

Algorithm 1: Algorithmus für die Zuweisung von Vertices und Knochen

[2] http://www.martinreddy.net/gfx/3d/OBJ.spec

Damit dieser Vorgang richtig funktioniert, müssen das Dreiecksnetz und Skelett richtig zueinander positioniert sein. Hierzu sollte das Skelett komplett im Inneren des Dreiecksnetzes befinden, natürlich auch bei Gliedmaßen wie Armen und Beinen. Mit, im Framework bereits vorhandenen, Translationen und Rotationen lassen sich die beiden Teile zueinander bewegen. Für die Zuweisung wird die Ruhe-Pose des Skeletts verwendet.

Wenn jeder Vertex des Netzes einem Knochen zugewiesen ist, ist eine dynamische Veränderung des Netzes möglich. In jedem Verarbeitungsschritt wird die Position eines jeden Vertex verändert. Die Veränderung basiert auf der momentanen Orientierung des dichtesten Knochen aus dem Ruhezustand.

5 Umsetzung

5.1 Hardware

Die Entscheidung für die Hardware fiel auf das Mikrocontroller-Board ESP32-Pico-Kit-V4[1] weil es sehr klein, gut ausgestattet und leistungsstark ist. Durch die geringe Größe lässt sich das Board gut an einem Knochen-Element befestigen. Ebenso kann es mithilfe der Arduino-Entwicklungsumgebung einfach programmiert werden. Als Sensor wurde der Adafruit BNO055[2] gewählt. Im Gegensatz zu anderen 9DOF-Sensoren (DOF = Degrees of Freedom), kann er seine Orientierung im Raum durch die ermittelten Daten aus Beschleunigungssensor, Gyroskop und Magnetometer errechnen und per API zur Verfügung stellen. Genau diese 3D-Ausrichtung wird für die entwickelte Anwendung benötigt.

Der Aufbau der Software für die Mikrokontroller-Boards ist simpel: Ein HTTP-Server wartet auf Anfragen und beantwortet diese mit den Orientierungsdaten des BNO055-Sensors. Durch das integrierte WLAN-Modul des ESP-Boards kann dies über eine kabellose Netzwerkverbindung geschehen. Da sich die Koordinatensysteme der Sensoren und das Koordinatensystem der Rigging-/Skinning-Anwendung unterscheiden, müssen die Sensordaten transformiert werden. Diese Transformation findet in der Anwendungssoftware statt.

Die 3D-Modelle des Baukastens des IGL wurden für ein **F**used **D**eposition **M**odeling (FDM) Druckverfahren angepasst. Hierfür wurden Hilfsstege entfernt und einzelne Bauteile separiert. Ebenfalls wurde zur Befestigung des ESP-Boards auf dem Knochen-Element dieses um eine ebene Fläche ergänzt.

[1] https://docs.espressif.com/projects/esp-idf/en/latest/get-started/get-started-pico-kit.html
[2] https://cdn-learn.adafruit.com/downloads/pdf/adafruit-bno055-absolute-orientation-sensor.pdf

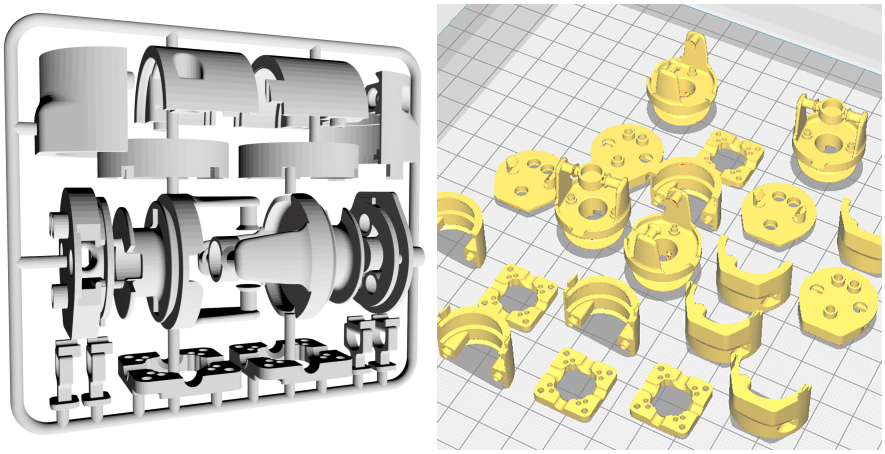

Abbildung 5.1: Link: Das 3D-Modell des IGL für ein Gelenk-Element; Rechts: Die für den Druck vorbereiteten 3D-Objekte

Quelle: Eigene Darstellung

5.2 Rigging

Für das Rigging wird das in der Konzeption erwähnte Framework genutzt und den Anforderungen entsprechend erweitert. Die Klassen Skeleton, Bone und BoneOrientationUpdater wurden hinzugefügt. Im Framework ist eine Struktur für Skelette bereits vorhanden, sie verhält sich jedoch teilweise etwas anders als die entwickelte. Der Hauptunterschied liegt darin, dass Kindknochen nicht nur die Endpostion des Elternknochens verwenden, sonder auch ihre momentane Orientierung. Das bedeutet, dass bei einer Änderung der Orientierung des Elterknochens sich auch die Orientierung des Kindknochen entsprechend verändert. Das Gelenk zwischen den Knochen bleibt starr. Dieses Verhalten macht Sinn, ist jedoch mit der entwickelten Hardware nicht nötig, da von ihr die absolute Orientierung von jedem Knochen übermittelt wird.

Ein weiterer Unterschied: Die Skelett-Klasse erbt von der Knochen-Klasse. Ein Skelett-Objekt ist dabei ein Knochen mit der Länge 0, daraus folgt: Der Start- ist auch gleichzeitig der Endpunkt des Knochen und andere Parameter haben darauf keinen Einfluss. Die vorhandene Datenstruktur wurde übernommen und leicht angepasst um mit der Hardware kompatibel zu sein.

Jeder Knochen bekommt einen BoneOrientationUpdater, der die Daten des Knochen-Elements kontinuierlich abfragt. Dabei laufen diese Abfragen im Hintergrund als eigener Thread. Dies hat den Vorteil, dass die Visualisierung nicht auf die Beantwortung von

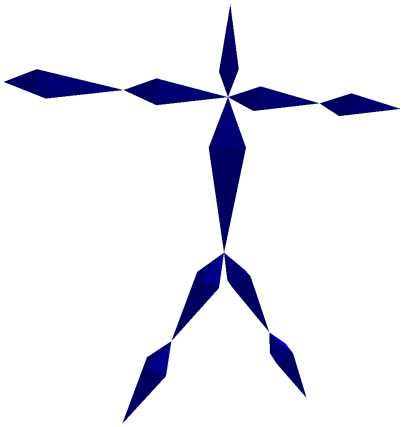

Abbildung 5.2: Skelett aus 10 Knochen im HAW-Framework
Quelle: Eigene Darstellung

Anfragen oder Zeitüberschreitungen bei Fehlern warten muss. Dafür benötigt ein Objekt dieser Klasse nur eine URL, bei der mithilfe des HTTP die Sensordaten übermittelt werden und einen Knochen, dessen Transformations-Matrix angepasst wird. Die Frequenz der Aktualisierung kann beliebig angepasst werden. Da in diesem Fall nur ein Thread schreibend auf das Bone-Objekt zugreift und der Thread der Visualisierung nur lesend, spielen die Probleme der Nebenläufigkeit eine untergeordnete Rolle.

Da es sich bei den Daten der Hardware um die absoluten Orientierungen handelt, dreht sich das virtuelle Skelett mit, wenn man das Hardware-Skelett dreht. Als Ankerpunkt dient hierbei der angegebene Punkt bei der Initialisierung des Skeletts. Um diesem Verhalten entgegen zu wirken, kann vor dem Skelett-Knoten im Szenengraph ein Translations-Knoten hinzugefügt werden. Dabei stammt die Transformations-Matrix von einem beliebigen Knochen, der damit zum neuen Ankerpunkt des Skeletts wird.

Wie bereits angemerkt wird eine Grundpose des Skeletts für das weiter Skinning benötigt. Da es sehr mühselig ist diese per Hand, mit der Methode "Trial and Error"(deu. Versuch und Irrtum), zu erstellen, bietet es sich an hierfür die Hardware zu verwenden. Zur Erstellung der Skelettpose reicht es im ersten Schritt, der Skelett-Initialisierung, die Längen und die allgemeine Anordnung der Knochen anzugeben. Darüber hinaus werden noch die URLs für die Datenabfrage benötigt. Mithilfe der Hardware kann man dann die Ruhe-Pose modellieren.

Damit die Ruhe-Pose für ein bestimmtes Dreiecksnetz nicht jedes mal neu modelliert wer-

den muss, sollen diese exportiert werden. Für diese Funktion wird die GSON-Bibliothek[3] verwendet. Sie kann beliebige Java-Objekte Serialisieren und Deserialisieren und diese als JSON-Datei abspeichern. Dabei gibt es jedoch folgendes Problem: Die verwendete Datenabstraktion besitzt durch die vorgehaltenen Eltern- und Kindknochen eine Kreis-Referenzierung (eng. circular reference). Bei der Serialisierung eines Knochen wird sein Elternknochen mit serialisiert, der wiederum seine Kindknochen, mindestens also auch der ausgehende Startknochen, serialisiert. Dieser Zyklus würde unendlich fortgesetzt werden. Eine Möglichkeit diese Endlosschleife zu umgehen wäre eine Veränderung der Datenstruktur, dies hätte jedoch zur Folge, dass viele Programmabläufe geändert werden müssten, darunter auch Teile des Renderings. Für diesen Anwendungsfall reicht es aus, die Referenzen auf die Elternknoten nicht mit zu serialisieren, dafür muss bei einem späteren Import der generierten JSON-Datei diese Beziehung wiederhergestellt werden. Hierfür ist eine einfache rekursive Funktion ausreichend, beginnend beim Startknochen wird für alle Kindknochen der Elternknochen festgelegt. Dieses Vorgehen wird solange für alle Kinder und Kindeskinder wiederholt, bis diese selbst keine Kinder mehr besitzen.

5.3 Skinning

Das Skinning Konzept ließ sich ohne größere Probleme umsetzen. Für eine homogene Sofware-Struktur und ein geordnetes Erscheinungsbild fehlt noch ein übergeordneter Szenengraph-Knoten, der den Dreiecksnetz-Knoten und den Skelett-Knoten samt den dazugehörigen Transformations-Knoten und Objekten beinhaltet.

In Abbildung 5.3 ist der Aufbau des Szenengraphen ab einem Skinning-Knoten zu sehen. Dieser besitzt zwei Kindknoten, ein Dreiecksnetz-Knoten und einen Skelett-Knoten. Zwischen diesen beiden Verbindungen können sich ein oder mehrere Transformations-Knoten befinden um beispielsweise Verschiebungen oder Rotationen zwischen Skelett und Netz auszugleichen. Ein Skelett-Knoten generiert bei der Konstruktion einer Entität aus einem Skelett-Objekt für jeden einzelnen Knochen einen Knochen-Knoten. Eigentlich handelt es sich dabei um Dreiecksnetz-Knoten, mit einem Netz, welches einen Knochen repräsentiert. Damit jedoch Abbildung 5.3 nicht falsch interpretiert wird, wurden dort die internen Dreiecksnetz-Knoten weggelassen. Für jeden Knochen-Knoten wird ebenfalls ein Transformations-Knoten generiert, der den Knochen an die richtige Stelle in der Szene verschiebt, skaliert und/oder rotiert.

[3] https://google.github.io/gson/apidocs/com/google/gson/Gson.html

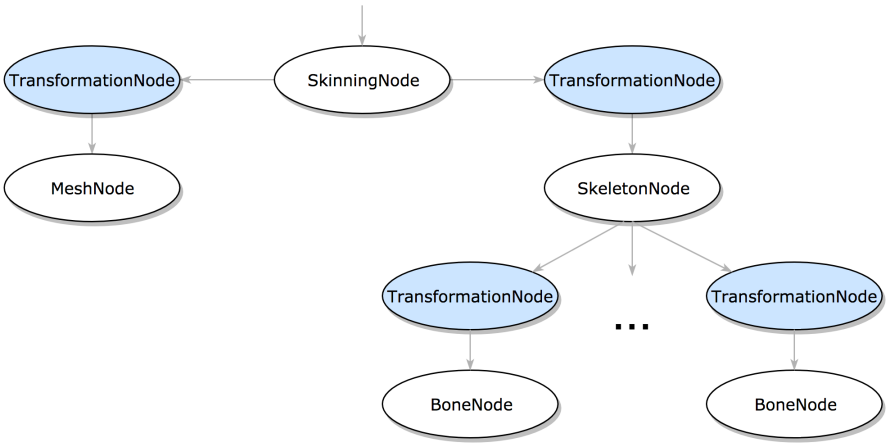

Abbildung 5.3: Aufbau eines Skinning-Knotens im Szenengraphen mit teilweise automatisch generierten Kindknoten

Quelle: Eigene Darstellung

Der Dreiecksnetz-Knoten besitzt ein Dreiecksnetz-Objekt und ermöglicht die Visualisierung. Die Dreiecksnetze lassen sich aus OBJ-Dateien einlesen und generieren. Dies klappt jedoch nicht mit allen Beispieldaten des Frameworks. Der Grund: Der verwendete Parser erwartet ein bestimmtes Format, damit er richtig arbeiten kann. Bei dem Versuch frei verfügbare Dateien einzulesen, scheitert das Einlesen ebenfalls. Mithilfe der 3D-Grafiksuite Blender können die Dateien zum fehlerfreien Parsen vorbereitet werden. Dafür muss die OBJ-Datei zunächst importiert und dann wieder mit der Option *Triangulate Faces* als OBJ exportiert werden. Dadurch werden Vielecke (eng. Polygons) des 3D-Modells in mehrere Dreiecksflächen zerlegt. Diese Modifikation der Quelldatei ermöglicht ein fehlerfreies Einlesen durch den Parser.

Ein weiterer interessanter Teil der Implementierung ist die Berechnung der dichtesten Entfernung zwischen einem Knochen und einem beliebigen Punkt. Dabei ist der Punkt im normalen Programmablauf die Position eines Vertex des Dreiecksnetzes. In der Analytischen Geometrie gibt es einfache Methoden um den Abstand zwischen einer Geraden und einem Punkt $\vec{a}$ zu berechnen. Der Knochen muss hierfür in eine Gerade umgewandelt werden. Dafür wird die Zweipunktform in der Vektordarstellung verwendet:

$$\vec{x} = \vec{p} + t \cdot (\vec{q} - \vec{p}) \quad \text{für} \quad t \in \mathbb{R}$$

Dabei ist der Stützvektor der Geraden $\vec{p}$ die Startposition des Knochen und der Ortsvektor $\vec{q}$ die Endposition. Über die Laufende-Punkt-Methode wird dann der Punkt auf der

Geraden ermittelt, der dem Punkt $\vec{a}$ am nächsten ist. Der Abstand ist am geringsten, wenn die Gerade zwischen dem laufenden Punkt und $\vec{a}$ orthogonal zum Richtungsvektor $(\vec{q}-\vec{p})$ der Geraden ist. Dies ist genau dann der Fall, wenn das Skalarprodukt der beiden Vektoren 0 ergibt.

$$(\vec{q}-\vec{p}) \circ (\vec{x}-\vec{a}) \quad = \quad 0$$

Aus dieser Gleichung lässt sich der Laufparameter t berechnen, mit dem sich wiederum, durch Einsetzen in die Geradengleichung, der Punkt berechnen lässt, der Punkt $\vec{a}$ am nächsten ist. Dann lässt sich mithilfe des Vektors zwischen den Punkten der Abstand errechnen.

Weil Knochen jedoch nicht unendlich lang sind, muss noch eine Einschränkung des Laufparameters t stattfinden. Durch die Wahl des Stütz- und Richtungsvektors der Geradengleichung kann am Laufparameter t leicht erkannt werden, ob der dichteste Punkt der Geraden auf dem Knochenabschnitt liegt.

$$wenn \quad t<0 \quad dann \quad t=0 \qquad und \qquad wenn \quad t>1 \quad dann \quad t=1$$

Dadurch werden Abstandsberechnungen in den Extremfällen vom Start- und Endpunkt berechnet, je nachdem welcher Punkt dichter liegt.

6 Evaluation

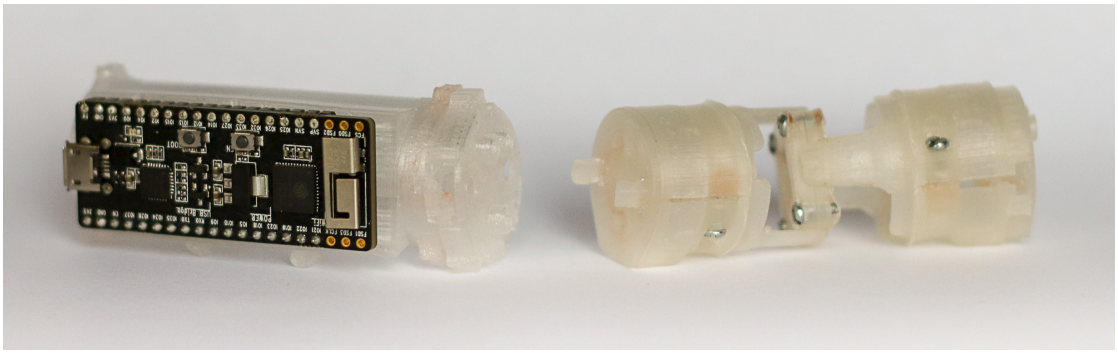

Abbildung 6.1: Foto von Knochen- und Gelenk-Element
Quelle: Eigene Darstellung

6.1 Hardware

Die entwickelte Hardware bietet einige Vorteile gegenüber der des IGL. Zum einen ist sie durch die verwendeten Standardkomponenten einfach und kostengünstig nachzubauen. Zum anderen etwas vielseitiger in den Anwendungsmöglichkeiten.

Anforderungen an die Hardware aus Kapitel 4.1.2:

- Nachbaubar ohne Fachwissen
- Kostengünstig
- Erhalt der Funktionalität

Nachbau ohne Fachwissen

Die Hardware besteht aus nur zwei unterschiedlichen Komponenten: Einem Mikrocontroller-Board und dem 9DOF-Sensor, zur Erfassung der Orientierung. Pro Knochen-Element

wird jeweils ein Paar dieser Komponenten verwendet, dabei dient das Mikrocontroller-Board lediglich als Schnittstelle zwischen Anwendung und Sensor. Dieser simple Aufbau und die Verwendung von Standardkomponenten sorgen für einen sehr einfachen Nachbau. Die Hürde für den Einstieg in die skelett-basierte Animation ist dadurch sehr niedrig und ist auch für Interessierte ohne Vorkenntnisse im Bereich der Informatik oder Elektrotechnik möglich. Der erste Punkt der Anforderungsliste ist also erfüllt.

Kostengünstig

Die verwendeten Hardware-Komponenten sind günstiger als die Anfertigung der Teile bei einem externen Dienstleister. Die Kosten für ein Gelenk-Element der IGL-Hardware liegen schätzungsweise bei 80 Euro. Im Gegensatz dazu belaufen sich die Kosten für die entwickelte Hardware auf lediglich 20 Euro pro Knochen-Element. Bei beiden Varianten müssen Gehäuseteile hergestellt werden. Mit dem verwendeten FDM-3D-Druck belaufen sich die reinen Materialkosten auf ungefähr 6 Euro pro Element-Gehäuse. Bei einer höheren Stückzahl verändern sich diese Summen jedoch zugunsten der Hardware des IGL und andere Druckverfahren werden kosteneffizienter. Da der Fokus bei der Entwicklung auf niedrigen Stückzahlen liegt, wie dies zum Beispiel bei Privatpersonen der Fall ist, ist die zweite Anforderung ebenfalls erfüllt. Bei Instituten oder Produktionsfirmen sind die Kosten meist nicht entscheidend oder es werden ohnehin andere Animations-Verfahren verwendet.

Erhalt der Funktionalität

Mithilfe des gewählten Sensors wird die dritte gesetzte Anforderung erfüllt, technisch gesehen ist die Hardware des IGLs jedoch überlegen. Durch die kabelgebundene Übertragung der Daten und die verwendeten Hallsensoren kann die Geometrie des Skeletts schneller und genauer erfasst werden. Für die Erstellung von Animations-Posen ist die Latenz einer kabellosen Übertragung jedoch zweitrangig und außerdem ausreichend gering. Ebenso sind die erfassten Daten des 9DOF-Sensors genau genug um diese verwenden zu können. Die Hardware des IGLs ist außerdem in der Lage, die Struktur der Skeletts eigenständig zu erfassen. Mithilfe von Mikrocontrollern in den Kupplungselementen kann die Anordnung der Gelenke ermittelt werden. Die Anwendungssoftware ist somit in der Lage, das Skelett, so wie es mit den Elementen gebaut wurde, ohne Konfiguration darzustellen. Diese Funktion ist bei der ESP-Hardware nicht vorhanden. Sie ließe sich beispielsweise durch Kabel zwischen den einzelnen Knochen-Mikrocontroller-Boards realisieren. Der Nachteil: Eine Einschränkung der Bewegungsfreiheit der Gelenke/Kupplungen und ein größerer Aufwand beim Bau eines Skeletts. Wegen der eigens gestellten Anforderung,

die entwickelte Lösung so simpel wie möglich zu gestalten, wird die Struktur des Skeletts in einer Konfigurationsdatei festgelegt. In dieser sind ebenfalls die Knochenlängen festgehalten, die bei der Hardware des IGLs auch konfiguriert werden müssen.

Grundlegend unterscheiden sich die Hardware-Varianten bei der Datenerfassung. Die Variante des IGLs ermittelt die Parameter eines Gelenks mithilfe der verbauten Hallsensoren in den Gelenk-Elementen. Die neu entwickelte Variante misst die Orientierung der Knochen durch, auf den Knochen-Elemente angebrachte, 9DOF-Sensoren. Die Hardware des IGLs arbeitet bei der Datenerfassung so gesehen lokal, da dort die Orientierung zwischen zwei Knochen gemessen wird. Bei der entwickelten Hardware wird die Orientierung der Knochen im Raum global und nicht zueinander gemessen. Aus beiden Datensätzen lässt sich jedoch die Geometrie des Skeletts ermitteln, dies geschieht bei beiden Varianten in der Anwendung.

Um eine komplett kabellose Hardware zu erhalten, müsste lediglich noch die Stromversorgung auf Akkus umgestellt werden. Zum Beispiel ließen diese sich gut im Inneren der Knochen-Elemente unterbringen. Dadurch wäre jedoch eine weitere Umgestaltung der 3D-Druck-Teile nötig.

Durch die Vermeidung von Kabeln zwischen den Knochenelementen soll der Bau und ein eventueller Nachbau von anderen so einfach wie möglich gehalten werden. Dieser erfüllte Aspekt kann die entstandenen Nachteile ausgleichen und die entwickelte Hardware erfüllt dabei die Funktionen der Hardware des IGL ähnlich gut.

Durch das FDM-Druckverfahren bedingt, sind die 3D-Teile nicht so präzise geworden, wie die des IGL. Der benötigte Widerstand, damit sich die Teile nicht durch ihr Eigengewicht bewegen, ist jedoch vorhanden und lässt sich bei Bedarf durch die vorhandenen Schrauben anpassen.

Abschließend lässt sich festhalten, dass die entwickelte Hardware-Variante für die festgelegten Anforderungen durch einen möglichen Nutzerkreis die bessere Alternative zur Hardware des Interactive Geometry Labs ist. Beide Varianten bieten Vor- und Nachteile in unterschiedlichen Anwendungsszenarien, je nachdem wie die Anforderungen ausfallen, müssen die Hardware-Varianten erneut bewertet werden.

6.2 Rigging

Um die Hardware zu entwickeln und zu testen wurde ein bereits vorhandenes Framework verwendet und den Anforderung entsprechend modifiziert und erweitert. Mithilfe der Anwendung lassen sich die beliebige Hardware-Skelette komfortabel und mit wenig Konfigurationsaufwand darstellen. Skelett-basierte Animationsverfahren sind weit verbreitet und in den meisten umfangreichen 3D-Programmen und -Engines vorhanden. Für einen produktiven Einsatz muss die entwickelte Hardware in diesen Anwendungen nutzbar sein. Da es jedoch keine standardisierte Schnittstelle für diese Art von Eingabegerät gibt, müsste eine entsprechende Erweiterung für jedes Programm einzeln entwickelt werden.

Zum Entwickeln und Testen der Hardware reicht an dieser Stelle das weiterentwickelte Framework aus. Von einer Entwicklung eines zusätzlich Add-ons, für beispielsweise Blender, ist Rahmen in dieser Arbeit abzusehen, da dies den Arbeitsschwerpunkt zu sehr verlagern und den Arbeitsaufwand zu sehr erhöhen würde.

6.3 Skinning

Die entwickelte Erweiterung des Grafik-Frameworks der HAW Hamburg ermöglicht einen leichten Einstieg in diesen Bereich der Computergrafik. Wegen der Größe dieses Bereichs ist es weniger sinnvoll fortgeschrittene Fehlerkorrekturen zu Implementieren, sie sind ebenfalls in umfangreicheren 3D-Programmen besser aufgehoben. Für die einfache Visualisierung der Hardware und zu Demonstrationszwecken von Dreiecksnetz-Manipulationen ist die Erweiterung gut geeignet. Bei den Fehlerkorrekturen gibt es eine stetig wachsende Anzahl von Ansätze, wie beispielsweise die Veröffentlichungen [JT05], [RF14] und [LH05] der letzten Jahre zeigen.

7 Abschluss

7.1 Fazit

Während der Entstehung dieser Bachelorarbeit hat sich herausgestellt, dass die Nutzung der bereits vorhandenen Hardware, vom Interactive Geometry Lab der ETH Zürich unzureichend dokumentiert ist. Aufgrund dieser Problematik wurde innerhalb dieser Forschungsarbeit eine Lösung entwickelt.

Das Ziel dieser Arbeit bestand darin einen Hardware-Baukasten zu entwickelten, welcher als Eingabegerät für die skelett-basierte Mesh-Animation genutzt werden kann. Die besondere Herausforderung hierbei: Das Ergebnis sollte die Arbeit in diesem Bereich vereinfachen.

Die Basis für die Entwicklung ist die Arbeit des IGL. Es hat sich herausgestellt, dass viele wichtige Details durch fehlende Dokumentation verloren gegangen sind – obwohl die Autoren bei dieser Arbeit einen Nachbau vorgesehen haben. Aufgrund dieser Problematik wird ein Nachbau für einen Außenstehender erschwert. Die Lösung: Ein ebenfalls modulares Eingabegerät, welches ein simples Nachbauen ermöglicht.

Während der Analyse, Konzeption und Entwicklung der eigenen Hardware, wurden weitere wichtige Details erkannt. Unter anderem wurde deutlich, wie entscheidend eine Zielgruppenanalyse für die Umsetzung eines Projektes ist. Hier sollte sich in jeder Hinsicht die Frage gestellt werden: Welches Produkt - im Falle dieser Bachelorarbeit: welches Eingabegerät – erfüllt die eigenen Anforderungen am besten. Darüber hinaus wurde ersichtlich, dass sich Prioritäten im Laufe der Arbeit verlagern und Teilbereiche eines Projekts stets beachtet werden müssen, damit sie nicht untergehen.

7.2 Ausblick

Wie bereits erwähnt fand die Entwicklung eines Hardware-Baukasten statt und ist nun abgeschlossen. Bei dieser Arbeit wurde ein modulares Eingabegerät entwickelt, welches verfügbare und günstige Standardkomponenten verwendet. Darüber hinaus lassen sich die Komponenten auch für andere Anwendungsfälle nutzen oder sind eventuell schon vorhanden.

Diese Arbeit bietet das Potenzial als Einstieg für Animations-Interessierte oder fortgeschrittene Animationskünstler genutzt zu werden. Es hat sich herausgestellt, dass der Nachbau der IGL-Hardware für diese Anwendergruppe, mit Hürden verbunden ist, die die Eigenentwicklung nicht besitzt.

Literaturverzeichnis

[BTST12] Gaurav Bharaj, Thorsten Thormählen, Hans-Peter Seidel, and Christian Theobalt. Automatically rigging multi-component characters. Technical report, MPI Informatik, 2012.

[CBC+05] Steve Capell, Matthew Burkhart, Brian Curless, Tom Duchamp, and Zoran Popovic. Physically based rigging for deformable characters. Technical report, University of Washington, 2005.

[FGDJ08] Tien-Chieng Feng, Prabath Gunawardane, James Davis, and Bolan Jiang. Motion capture data retrieval using an artist's doll. Technical report, University of California, California State University, 2008.

[Gle98] Michael Gleicher. Retargetting motion to new characters. Technical report, Autodesk Vision Technology Center, 1998.

[GMP+16] Oliver Glauser, Wan-Chun Ma, Daniele Panozzo, Alec Jacobsen, Otmar Hilliges, and Olga Sorkine-Hornung. Rig animation with a tangible and modular input device. Technical report, ETH Zürich, New York University, Columbia University, 2016.

[JPG+14] Alec Jacobson, Daniele Panozzo, Oliver Glauser, Cédric Pradalier, Otmar Hilliges, and Olga Sorkine-Hornung. Tangible and modular input device for character articulation. Technical report, ETH Zürich, GeorgiaTech Lorraine - CNRS UMI 2958, 2014.

[JT05] Doug L. James and Christopher D. Twigg. Skinning mesh animations. Technical report, Carnegie Mellon University, 2005.

[LCF00] J. P. Lewis, Matt Cordner, and Nickson Fong. Pose space deformation: A unified approach to shape interpolation and skeleton-driven deformation. Technical report, Centropolis, 2000.

[LH05] Binh Huy Le and Jessica K. Hodgins. Real-time skeletal skinning with optimized centers of rotation. Technical report, Disney Research, 2005.

[MTLT88] Nadia Magnenat-Thalmann, Richard Laperrier, and Daniel Thalmann. Joint-dependent local deformations for hand animation and object grasping. Technical report, Université de Montréal, 1988.

[RF14] Nadine Abu Rumman and Marco Fratarcangeli. Position based skinning of skeleton-driven deformable characters. Technical report, Sapienza University of Rome, 2014.

[VBG+13] Rodolphe Vaillant, Loic Barthe, Gael Guennebaud, Marie-Paule Cani, Damien Rohmer, Brian Wyvill, Olivier Gourmel, and Mathias Paulin. Implicit skinning: Real-time skin deformation with contact modeling. Technical report, Université de Toulouse, University of Victoria, Inria, Univerity of Bath, 2013.

[WPP13] Robert Y. Wang, Kari Pulli, and Jovan Popovic. Real-time enveloping with rotational regression. Technical report, Massachusetts Institute of Technology, Nokia Research Center, 2013.

[YSC+] Wataru Yoshizaki, Yuta Sugiura, Albert C Chiou, Sunao Hashimoto, Masahiko Inami, Takeo Igarashi, Yoshiaki Akazawa, Katsuaki Kawachi, Satoshi Kagami, and Masaaki Mochimaru. An actuated physical puppet as an input device for controlling a digital manikin. Technical report, NAIST, DHRC AIST, Keio university, JST ERATO, The University of Tokyo.

Erklärung zur selbstständigen Bearbeitung einer Abschlussarbeit

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Ort	Datum	Unterschrift im Original