Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorarbeit

Lars Hendrik Porep

Skelettbasierte Modellierung und Animation von
2D Charakteren mittels Rigging

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Lars Hendrik Porep

Skelettbasierte Modellierung und Animation von 2D Charakteren mittels Rigging

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. Olaf Zukunft

Abgegeben am 21.12.2016

Lars Hendrik Porep

Thema der Arbeit

Skelettbasierte Modellierung und Animation von 2D Charakteren mittels Rigging

Stichworte

Computergrafik, Skelettbasierte Modellierung, Rigging, Skinning

Kurzzusammenfassung

Diese Arbeit beschäftigt sich mit der Deformation von 2D-Charaktergrafiken durch ein skelettbasiertes Modellierungsverfahren. Es wird ein Prototyp entwickelt und evaluiert, der ein automatisches Skinning-Verfahren implementiert.

Lars Hendrik Porep

Title of the paper

Skeleton-based modeling and animation of 2D-characters using rigging

Keywords

Computer graphics, skeleton-based modeling, rigging, skinning

Abstract

This bachelor thesis deals with the deformation of 2D character graphics using a skeleton-based method. A prototype is developed and evaluated which implements an automatic skinning process.

Inhaltsverzeichnis

1 Einführung

1.1 Motivation

Laut einem Sprichwort, sagt ein Bild mehr als tausend Worte. Bilder vermögen es oft schneller einen Sachverhalt oder auch eine Emotion zu vermitteln als es Wörter könnten. Für diesen Zweck werden in verschiedenen Bereichen Grafiken benötigt, z.B. für die Aufbereitung einer Präsentation oder für die Erstellung von Lernmaterial. Auch wenn durch das Internet eine riesige Anzahl von Bildern verfügbar ist, so gibt es Situationen, in denen das passende Bild einfach nicht gefunden wird. Sei es aus lizenzrechtlichen Gründen oder weil die schiere Anzahl an Bildern das gesuchte Ergebnis überlagert.

Im Bereich der Animationen ist es ein anderer limitierender Faktor. Der Aufwand für eine erstellte Grafik entspricht nur einem Bruchteil dessen, was für die Erstellung einer aus vielen Bildern bestehenden Animation benötigt wird. Das betrifft natürlich nicht nur große Animationsstudios, sondern auch die Entwicklung von Computerspielen in kleinen Studios.

Aus diesen Gründen erscheint dieser Bereich der unterstützen Grafikerzeugung äußerst interessant. Gerade wenn mithilfe von Algorithmen die Generierung auch von Anwendern durchführbar ist, die nur ein begrenztes künstlerisches Talent besitzen.

1.2 Zielsetzung

Wie aus der Motivation hervorgeht, ist es das Ziel dieser Arbeit, die einfache Erschaffung neuer zweidimensionaler Grafiken zu erproben. Der Fokus der Arbeit liegt dabei auf der Manipulation von Charaktergrafiken. Auf Basis eines Verfahrens zur skelettbasierten Modellierung soll ein System entwickelt werden, das es ermöglicht, eine bestehende Charaktergrafik mit einem darunterliegenden Skelett zu verbinden. Anschließende Bewegungen des Skeletts sollen auf die Grafik übertragen werden und diese dadurch deformieren. Zur Unterstützung des Anwenders, soll ein Verfahren, zur automatischen Verbindung von dem Skelett und der Grafik, ausgewählt und implementiert werden. Ziel ist es, außerdem die angewandten Verfahren zu bewerten und daraus eventuelle neue Ansätze für eine Weiterentwicklung abzuleiten.

1.3 Aufbau der Arbeit

Zu Beginn der Arbeit in Kapitel 2 werden einige Grundlagen und Begriffe aus dem Bereich der Computergrafik und der Animationstechnik erläutert, die für ein besseres Verständnis dieser Arbeit wichtig sind. Im Anschluss erfolgt im dritten Kapitel ein Blick auf den aktuellen Stand der Technik. Dort werden Verfahren mit ähnlichen Zielsetzungen zur Grafikerzeugung vorgestellt.

Im vierten Kapitel werden die Grundlagen für das zu entwickelnde System analysiert. Der Blick erfolgt zuerst aus der Anwenderperspektive. Daraus ableitend wird diese Sicht in Anforderungen für die technische Umsetzung transferiert.

Kapitel 5 beschreibt die Konzeption des Systems. Es werden die ersten Entwicklungsentscheidungen getroffen und Anforderungen beschrieben, wie das System aufzubauen ist. Außerdem erfolgt die Erläuterung und Ausarbeitung wichtiger Verfahren und Algorithmen für die Implementierung.

Im Anschluss wird im Kapitel 6 die erfolgte Implementierung vorgestellt. Nach grundlegenden Entscheidungen wird dafür die Umsetzung der Datenstrukturen und die Umsetzung der Verfahren beschrieben.

Im darauffolgenden Evaluationsteil, dem Kapitel 7, erfolgt eine Auswertung des erstellten Systems. Implementierung und Entwicklungsentscheidungen werden analysiert. Dabei werden die vorher gestellten Anforderungen mit dem entwickelten System abgeglichen, eine Performanceanalyse diskutiert und die durch die eingesetzten Verfahren erzielten Ergebnisse beleuchtet.

Abgeschlossen wird die Arbeit durch ein persönliches Fazit und einen Ausblick auf mögliche weitere Ansätze, die sich aus dieser Arbeit ergeben haben.

2 Grundlagen

In diesem Kapitel werden einige Begriffe aus dem Bereich der Computergrafik eingeführt. Zuerst wird mit Skelett und Knochen ein Blick auf Datenstrukturen der Animationstechnik geworfen. Es folgt die Beschreibung der Dreiecksnetz-Datenstruktur, die die Grundlage für viele 3D-Modelle bietet, zusammen mit der hierauf folgenden Beschreibung von Texturen. Rigging und Skinning beschreibt schließlich den Prozess, der für die Verknüpfung dieser Datenstrukturen angewandt wird.

2.1 Begriffserklärung

2.1.1 Dreiecksnetz

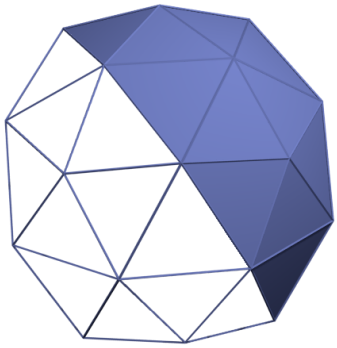

Abbildung 1: Ein Dreiecksnetz zur Hälfte mit transparenten Facetten dargestellt.

Zur Modellierung von Polyedern hat sich in verschiedenen Bereichen der 3D-Grafik das sogenannte Dreiecksnetz etabliert. Ein Dreiecksnetz ist die spezialisierte Form eines Polygonnetzes, mit der Besonderheit, dass das gesamte Drahtgittermodell aus Dreiecken besteht.

Formal betrachtet ist ein Dreiecksnetz eine Sammlung von Vertices, Kanten und Facetten. Ein Vertex beschreibt eine Position im Raum und kann weitere Informationen wie die

Farbe, die Normale (senkrecht auf der Oberfläche) oder Texturkoordinaten enthalten. Eine Kante ist eine Verbindung zwischen zwei Vertices. Als Facette wird die Fläche zwischen drei zusammenhängenden Kanten bezeichnet, die damit ein Dreieck bilden (vgl. [1]).

Nach der Graphentheorie ist ein Dreiecksnetz ein ungerichteter Graph ohne Mehrfachkanten. Für die Verarbeitung und Speicherung gibt es eine Vielzahl verschiedener Algorithmen und Datenstrukturen (vgl. [1]; [2]). Eine detailliertere Betrachtung dieser Strukturen ist für diesen Anwendungsfall allerdings nicht nötig und würde über den Rahmen dieser Arbeit hinausgehen. Aus dem gleichen Grund muss auf alternative Darstellungsformen für 3D-Modelle wie Viereckснetze, Splines oder NURBS in dieser Arbeit ebenfalls verzichtet werden.

2.1.2 Skelett und Knochen in der Animation

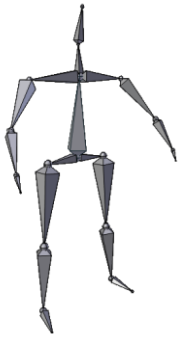

Abbildung 2: Skelett bestehend aus einzelnen Knochen, als Grundlage für die Animation eines 3D-Modells

Ein verbreitetes Verfahren zur Animation von 3D-Charakteren ist die skelettbasierte Animation. Grundlage dafür ist ein aus Knochen bestehendes Skelett, das auch Rig genannt wird (siehe [Abbildung 2]). Diese dient als abstrahierte Datenstruktur der Transformation dieser Modelle. Mehrere Knochen werden innerhalb einer hierarchischen Struktur zu einem Skelett zusammengefasst. Das Skelett ist ein zusammenhängender kreisfreier Graph bzw. ein Baum. Punkte, in denen zwei Knochen aufeinandertreffen, werden als Gelenk bzw. Joint bezeichnet.

Die Oberfläche von 3D-Modellen werden durch eine Datenstruktur wie das Dreiecksnetz repräsentiert. Den Knochen im Skelett werden Teile dieser Oberflächen zugeordnet. Nach der Zuordnung können die Positionen dieser Flächen über den Knochen beeinflusst werden.

Bei der Animation wird also der nicht sichtbare Knochen eines 3D-Modells bewegt. Durch die Bewegung dieses Knochens wird der Teil der Oberfläche beeinflusst, der mit diesem Knochen verbunden wurde. (vgl. [3])

2.1.3 Texturen und Texturkoordinaten

Eine Textur bezeichnet eine Grafik, die einem 3D-Modell mithilfe von Texturkoordinaten zugeordnet wird. Im Normalfall wird diese Textur aus einer Rastergrafik bzw. einer Bitmap erzeugt.

Ohne das eigentliche 3D-Modell durch zusätzliche Vertices feiner aufzulösen, d.h. den Detailgrad und damit die Anforderungen an Berechnungen zu erhöhen, ermöglichen Texturen eine detailreichere und realistischer wirkende Darstellung dieser Modelle. Dafür kann jedem Vertex eine sogenannte Texturkoordinate zugewiesen werden. Diese sogenannte uv-Koordinate beschreibt die Position innerhalb der hinterlegten Textur. Die für das 3D-Modell zu zeichnenden Farben der Facetten werden dann beim Rendering anhand der hinterlegten Texturkoordinate aus der Textur ausgelesen (vgl. [4]; [5]).

2.1.4 Rigging und Skinning

Das Verfahren, ein skelettbasiertes Modell für die Animation vorzubereiten, wird als Rigging bezeichnet. Dies umfasst zwei grundlegende Schritte. Zum einen das Erstellen des Skeletts und des Weiteren das sogenannten Skinning, das Erzeugen einer Beziehung zwischen der Oberfläche des 3D-Modells und den Knochen des Skeletts, (vgl. [6]; [7]).

Beim Skinning wird festgelegt, wie sich die Manipulation bestimmter Teile des Skeletts auf die Oberfläche auswirkt. Knochen des Skeletts werden mit Teilen des Dreiecksnetzes verknüpft. Diese Verbindung erfolgt in der Regel über Gewichte, die die Verbindung zwischen Knochen und dem jeweiligen Teil der Oberfläche unterschiedlich stark definiert.
Beim Skinning eines Charakters wird beispielsweise ein Handknochen in der Regel den Teil des 3D-Models beeinflussen, der die Hand darstellt aber keinen Einfluss auf die Rotation des Kopfes haben.

Die einzelnen Begriffe werden in der Literatur unterschiedlich abgegrenzt. So wird zum Beispiel das Texturieren eines 3D-Modells mithilfe einer Rastergrafik teilweise ebenfalls als Skinning bezeichnet (vgl. [4]). Im weiteren Verlauf wird sich diese Arbeit beim Skinning aber auf das Verfahren zur Verbindung von Knochen und Vertices beziehen.

3 Stand der Technik

Dieses Kapitel soll einen kurzen Überblick über einige Verfahren bieten, die das Ziel haben, den Aufwand für die Grafikerstellung zu reduzieren, indem sie aus bestehenden Grafiken oder Modellen neue Grafiken erzeugen.

Der gesamte Bereich in diesem Themenfeld ist sehr umfangreich, aus diesem Grund sollen hier einige ausgewählte Verfahren vorgestellt werden, die sich in einem ähnlichen thematischen Umfeld bewegen und versuchen, durch geschickte Manipulation zusätzliche bzw. mehrere Charaktergrafiken zu erzeugen.

3.1 Deformation von Grafiken und Dreiecksnetzen

Seit Jahrzehnten beschäftigen sich verschiedene Ansätze mit der Verformung von Dreiecksnetzen bzw. 2D-Grafiken.

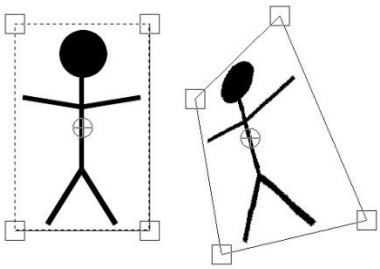

Abbildung 3: Frei-Form-Deformation an einem 2D-Objekt mit 4 Deformationspunkten. (Quelle: https://upload.wikimedia.org/wikipedia/de/a/a4/Freiform-Deformation.jpg - Lizenz: Gemeinfrei - Abgerufen am 08.12.2016)

Ein Verfahren ist die sogenannte Frei-Form Deformation [8], bei diesem Verfahren wird um das Objekt eine Box gespannt. Anschließend wird diese Box über Deformationspunkte verformt und die Transformation auf die Grafik bzw. das Gittermodell übertragen (siehe [Abbildung 3]).

Mit *Image Deformation Using Moving Least Squares* beschreiben Schaefer et al. in [9] einen Algorithmus, bei dem ein Dreiecksnetz mit Kontrollpunkten versehen wird. Die Manipulation dieser Punkte wirkt sich auf das zugrundeliegende Dreiecksnetz aus und verformt es entsprechend.

Nachteilig erscheint bei diesen Methoden, zumindest für die Manipulation von Charaktergrafiken, dass mit der Bedienung der Kontrollpunkte ein gewünschtes Ziel nicht ganz einfach erreicht wird und für den Anwender wenig intuitiv erscheint (vgl. [10]).

3.2 Keyframe Animation

Bei der Keyframe Animation oder auch Morph-Animation wird die Abfolge der Bilder für die Animation nicht anhand eines 3D Modells und vorhandener Manipulationspunkte definiert. Stattdessen werden vorab einzelne Teilschritte der Animationssequenz erstellt. Anschließend wird zwischen diesen sogenannten Keyframes interpoliert, um die Zwischenschritte der nicht definierten Frames zu errechnen. Dieses Verfahren wird in der 2D-Animation von zum Beispiel Comic-Animatoren verwendet, findet aber auch bei 3D-Modellen und deren Animation Anwendung.

Das Datenformat .md2 des Computerspiels „Quake2“ von „id Software, Inc.“ basiert auf dieser Technik und speichert für animierte 3D-Modelle alle benötigten Zustände einzelner Keyframes (vgl. [4], S. 782).

3.3 Skelettbasierte Animation

Animationen von Bewegungen und Verformungen bei 3D-Modellen erfolgt heutzutage häufig über ein Verfahren, das auf einem Skelett basiert.

Hierfür wird die Oberfläche eines 3D-Modells in Form eines Dreiecksnetzes erstellt. Zusätzlich wird ein Skelett erstellt und mit dem Modell verbunden. Wird ein Knochen bewegt, bewegt sich der damit verbundene Teil des 3D-Modells ebenfalls.

Wie diese Verbindung abgebildet wird, ist unterschiedlich, hat aber einen großen Einfluss auf die dadurch erzielten Ergebnisse. Lander beschreibt dazu ein Verfahren, bei dem ein Vertex mit mehreren Knochen, durch Gewichte, verbunden werden kann. Die neue Position ergibt sich aus der gewichteten Interpolation, nachdem die Bewegung der einflussnehmenden Knochen auf die Vertex-Position übertragen wurde (vgl. [11]; [12]).

Bei stark gewundenen Übergängen zwischen Gelenken, kann es durch die Interpolation der einzelnen Ergebnisse zu optischen Beeinträchtigungen des Modells kommen. Kavan et al. beschreiben ein Verfahren, bei dem der Interpolierungsvorgang nicht nur die berechneten, neuen Vertex-Positionen beachtet, sondern schafft auch einen Bereich um Gelenke herum,

um dort durch Interpolation der Knochenpositionen realistischere Übergänge zu erzeugen (vgl. [13]).

Andere Verfahren beschäftigen sich damit, die Verformung der Modell-Oberfläche besser abzubilden und erweitern aus diesem Grund das skelettbasierte Verfahren um zusätzliche Ebenen. Ein Beispiel dafür ist der Ansatz, auch den Einfluss von Muskeln zu simulieren. In Bereichen, wie der Mitte eines Knochens, kann dies das Ergebnis verbessern, indem sich die Oberfläche durch das Zusammenziehen der Muskeln aufwölbt, während das Skelett die Kontrolle über die Lage des Bereichs behält (vgl. [14]).

Gemeinsam haben diese skelettbasierten Verfahren, dass dem Anwender eine einfache Kontrolle über das zu verformende Modell gegeben werden soll, gleichzeitig soll aber auch die Qualität der Ergebnisse hochgehalten werden. So ist eine Abwägung zwischen den verschiedenen Anforderungen zu tätigen. Die Bedienbarkeit wird durch zusätzliche Ebenen oder Freiheiten erschwert und der Prozess, das Skelett mit dem Modell zu verbinden, wird ebenfalls aufwendiger. Um diese Komplexität wieder zu reduzieren, können durch Algorithmen Arbeitsschritte ganz oder teilweise übernommen werden. Diese müssen aber auch entwickelt und implementiert werden und gleichzeitig sind sie durch Hardwareanforderungen limitiert.

Insgesamt bietet die skelettbasierte Animation dem Anwender eine intuitive Möglichkeit mit dem Modell zu interagieren, dazu bietet es einen hohen Grad an Flexibilität bei der Kombination von verschiedenen Teil-Animationen.

Ist das Skelett zwischen unterschiedlichen Modellen austauschbar, dann ist es möglich, ein fertig animiertes Skelett mit seinen Bewegungen auf ein anderes Modell zu übertragen. Durch Verfahren wie Motion Capturing können zusätzlich realistische Bewegungen durch einen Menschen ausgeführt und vom Computer mit geeigneter Hardware abgetastet und verarbeitet werden. Diese Bewegungsabläufe können im Anschluss auf das Skelett und damit auf das 3D-Modell übertragen werden (vgl. [4]; [15]).

4 Analyse

Dieses Kapitel startet mit einer kurzen Zusammenfassung zur Motivation für die Verwendung eines skelettbasierenden Verfahrens. Im Anschluss wird das zu entwickelnde System kurz beschrieben und für ein besseres Verständnis noch mit einem exemplarischen Szenario ergänzend weiter vorgestellt. Darauf folgt die Beschreibung und Bestimmung von funktionalen und nicht funktionalen Anforderungen. Am Ende des Kapitels erfolgt eine Abgrenzung zu Bereichen, die in dieser Arbeit nicht behandelt werden können.

4.1 Motivation - Skelettbasierter Modellierung

Wie bereits im Stand der Technik dargestellt, gibt es unterschiedliche erforschte und angewandte Herangehensweisen an die Erzeugung neuer Grafiken. Eine Option wäre, wie bei Keyframe-Animationen, Bilder mit verschieden positionierten Charakteren als Ausgangspunkt zu verwenden und anschließend Zwischenschritte zu berechnen.

Alternativ wäre auch ein Deformationsverfahren wie die Frei-Form Deformation denkbar, bei dem die zu manipulierende Grafik in umgebende Körper gekapselt und durch Umrechnung der Transformation vom Körper auf die Grafik übertragen wird.

Für die Erzeugung neuer 2D-Charaktere auf Basis von möglichst wenig grafischer Vorarbeit erscheint das Verfahren der skelettbasierten Modellierung in diesem Kontext als vorteilhaft. Mit den Knochen ist eine Struktur gegeben, die auf natürlichen und damit verständlichen Prinzipien beruht (vgl. [10]). Auch ist die Einflussnahme der Knochen hauptsächlich auf geometrischer Nähe begründet. Dies wird als positiv gewertet, weil es aufgrund der Verständlichkeit und der Nähe zum realen Vorbild die Bedienung und die Umsetzung von Berechnungen und Algorithmen vereinfacht.

Bei 3D-Modellen haben sich skelettbasierte Animationsverfahren bereits bewährt und werden gerade bei Kreaturen und Charakteren umfangreich genutzt. Auch wenn diese Ansätze nicht eins zu eins für einen Transfer in den 2D-Bereich geeignet sind, so ergeben sich dennoch Parallelen, die auf eine erfolgreiche Umsetzung hindeuten.

Es ist auch zu erwähnen, dass heutige Grafikkarten und deren Schnittstellen wie OpenGL u.a. auf die Arbeit mit Dreiecksnetzen und damit verbundenen Transformationen ausgelegt sind. Damit scheint in diesem Umfeld auch ein hoher Performanceanspruch kein Hindernis zu sein.

4.2 Kurzbeschreibung des zu entwickelnden Prototypen

Im System wird ein Skelett mit einer zweidimensionalen Charaktergrafik verbunden. Durch anschließende Rotationen der Knochen, z.B. der Bewegung eines Armknochens, wird das verknüpfte Bild ebenfalls transformiert und deformiert: in diesem Beispiel der Bildabschnitt des darüber liegenden Arms.

Verschiedene Skelettpositionen werden vorgespeichert und können vom Anwender geladen werden. Das Laden von Skelettpositionen transformiert das Skelett und deformiert mit ihm das verbundene Bild. Der Anwender kann eigene Charaktergrafiken für eine Bearbeitung durch das System in dieses importieren. Es ist auch möglich, die dadurch erzeugten neuen Grafiken zu exportieren. Animationen können als Bilderserien exportiert werden, sie lassen sich aus Teilschritten einer Transformation zwischen zwei Skelettpositionen erzeugen.

4.3 Szenario eines ausgebauten Systems

Um ein besseres Verständnis für das Gesamtbild des Systems zu entwickeln, soll anhand eines Szenarios der erfolgreiche Durchlauf eines Anwenders durch ein vollständig entwickeltes System beschrieben werden.

Usage-Scenario:
Tom Adams möchte für eine Präsentation eine neue Grafik erstellen, auf der ein Charakter dem Betrachter zuwinkt.

1. Tom startet das System.
2. Das System präsentiert einen Dialog, der zur Auswahl einer Grafik auffordert.
3. Tom wählt eine Grafik aus.
4. Das System präsentiert einen Dialog, der zur Auswahl eines Skeletts auffordert.
5. Tom wählt ein Skelett.
6. Das System zeigt die vom Skelett überlagerte Charaktergrafik.
7. Tom passt die Knochenpositionen an, sodass sie über den korrespondierenden Teilen der Charaktergrafik liegen.
8. Tom lädt eine der vorgespeicherten Skelettposition.
9. Das System dreht den Charakter in die gewünschte Position.
10. Tom macht noch eine kleine Änderung an der Pose.

11. Tom exportiert die neue Grafik.
12. Tom beendet das System.

Aufgrund des Umfangs kann in dieser Arbeit das entwickelte System nicht vollständig, sondern nur prototypisch entwickelt werden. Im Folgenden sollen daher die Kerneigenschaften des Systems aus diesem Szenario abgeleitet und in User-Stories überführt werden.

4.4 Abgeleitete User-Stories

Folgende User-Stories wurden aus dem Szenario des vorherigen Kapitels extrahiert.

> *Als Anwender möchte ich meine Grafiken im System öffnen, um diese zu bearbeiten.*
>
> *Als Anwender möchte ich zu meiner Grafik ein vorgefertigtes Skelett auswählen und öffnen können.*
>
> *Als Anwender möchte ich meine Grafik mit einem Skelett einfach verbinden können.*
>
> *Als Anwender möchte ich vorgespeicherte Skelettpositionen laden können, um schnell eine neue Grafik zu erstellen.*
>
> *Als Anwender möchte ich das Skelett nach meinen Wünschen drehen können, um das individuell ansprechendste Ergebnis zu erzeugen.*
>
> *Als Anwender möchte ich eine erstellte Grafik exportieren können, um sie in einem anderen Programm weiter zu verwenden.*

Da in dieser Arbeit kein Dialog mit dem Anwender stattfinden kann, um daraus Akzeptanzkriterien abzuleiten, werden diese User-Stories in funktionale Anforderungen überführt, die später evaluiert werden können.

4.5 Anforderungen

„Anforderungen in Softwaresystemen werden oft in funktionale und nicht funktionale Anforderungen unterteilt“ [16].
Unter funktionalen Anforderungen versteht man, was ein System tun sollte. Dies können bestimmte Dienste oder Reaktionen sein. Sie beschreiben vor allem anwendungsspezifische Anforderungen.

Nicht funktionale Anforderungen beziehen sich eher darauf, wie gut das zu entwickelnde System arbeitet und definieren damit eher allgemeine Qualitätseigenschaften, die größere Bereiche des Systems betreffen (vgl. [16]).

4.5.1 Funktionale Anforderungen

Der Anwender soll:

- eine Rastergrafik im PNG-Dateiformat in die Anwendung importieren können.
- ein von der Anwendung vorgefertigtes Skelett laden und verwenden können.
- die Skelettposition an die Grafik anpassen können, indem einzelne Knochen verschoben werden.
- vorgespeicherte Skelettposen laden können um damit das verknüpfte Bild zu deformieren.
- Knochen einzeln rotieren können um damit die verbundene Grafik zu deformieren.
- ein fertig positioniertes Skelett automatisch mit der Grafik verbinden können.
- Animationen in Form einer Einzelbilderserie exportieren können.
- manipulierte Grafiken im PNG-Dateiformat exportieren können.

4.5.2 Nicht funktionale Anforderung

Alle Benutzerinteraktionen sollen dem Anwender spätestens nach 3 Sekunden ein Ergebnis liefern und vom System verarbeitet worden sein, damit die Benutzerfreundlichkeit nicht stark eingeschränkt wird. Zulässige Ausnahme für diese Zeitbeschränkung ist die Verarbeitung sehr feingegliederter Dreiecksnetze mit mehr als 10.000 Polygonen, da hiermit bereits eine sehr detaillierte Auflösung möglich ist. Bei höheren Anforderungen an den Detailgrad, ist dem Benutzer eine längere Reaktionszeit zuzumuten.

4.6 Abgrenzung

Grundsätzlich wird das System als Prototyp entwickelt. Der Fokus liegt primär auf der Umsetzung und anschließenden Bewertung der Funktionalitäten. Aufgrund der begrenzten Zeit soll das System sich nur auf die Modellierung von Zweibeinern konzentrieren. Gleichzeitig aber so aufgebaut werden, dass die bestehende Lösung mit geringem Aufwand auch auf andere Anatomien erweitert werden kann.
Auch wenn für die Umsetzung Skelette benötigt werden, kann sich die Arbeit aus Zeitgründen nicht mit der automatischen Generierung von Skeletten auseinandersetzen.

Der Fokus der Arbeit liegt auf der Manipulation von Charaktergrafiken, um damit möglichst einfach neue Bilder zu generieren. Animationen bestehen aus vielen, meist nur leicht veränderten Einzelbildern, die in einer schnellen Abfolge angezeigt werden. Der Erkenntnisgewinn bei der Erstellung neuer Einzelbilder ist somit auch im Animationsbereich anwendbar. Die Entwicklung eines Systems zur Steuerung, Konstruktion, Anzeige oder Bearbeitung von Bilderserien für die Animation muss aber aus Zeitgründen entfallen.

Die Benutzeroberfläche soll einen einfachen Einstieg für die Verwendung und Weiterentwicklung des Systems darstellen, erhebt aber keinen Anspruch auf ein ästhetisch ansprechendes Design oder hohe Benutzerfreundlichkeit.

5 Konzeption

In diesem Kapitel wird das zu entwickelnde System konzipiert. Dafür erfolgt zu Beginn die Auswahl eines Frameworks und anschließend die Beschreibung für den grundsätzlichen Aufbau des Systems inklusive einer Beschreibung zu den verwendeten Datenstrukturen. Außerdem werden grundlegende Verfahren und Algorithmen beschrieben, die in der Entwicklung umgesetzt werden sollen.

5.1 Auswahl eines Frameworks

Um den Hauptfokus der praktischen Arbeit auf die Entwicklung der Funktionalitäten des Systems zu legen, soll eine Umgebung gewählt werden, die hierfür möglichst viel Unterstützung bietet. Aus diesem Grund soll das System auf einem Framework bzw. einer Grafikengine aufbauen.

Das Java-Framework *CgResearch*, entwickelt und betreut durch Prof. Dr. Philipp Jenke, bietet eine flexible Zwischenschicht zwischen einem System und *Java Bindings for OpenGL (JOGL)*. Es ist dem Verfasser dieser Bachelorarbeit durch vorherige Entwicklungen im Rahmen einer Computergrafik-Vorlesung bereits bekannt.

Das Framework bietet bereits die Datenstruktur eines Szenegraphen. Eine Struktur, die Objekte und Eigenschaften von Objekten als gerichteten Graphen hierarchisch strukturiert. Außerdem wird diese Struktur in einen Renderinggraphen überführt und an die Rendering-Pipeline weitergegeben. So ermöglicht sie die schnelle Entwicklung sichtbarer Ergebnisse, indem eigene Objekte in den Szenegraphen eingefügt werden und damit in die darunterliegenden Bereiche des Systems weitergereicht und verarbeitet werden.

Ebenfalls vorteilhaft sind auch die bereits vorhandenen Klassen für Vektoren, Drahtgittermodelle und GUI-Elemente. Gleichzeitig ist das System ausreichend flexibel, um sich eventuell auftretenden Herausforderungen während der Entwicklung anzupassen. Nicht zuletzt, weil der gesamte Quellcode verfügbar ist und erweitert werden kann. Selbst ohne Erweiterung des Frameworks ist es einfach möglich, ein spezialisiertes Dreiecksnetz zu

erstellen, eigene Vertex- und Fragment-Shader hinzuzufügen und einzubinden oder eine eigene Datenstruktur in die Struktur des Frameworks zu integrieren.

Es gibt unzählige Frameworks und Grafikengines, die ebenfalls verwendet werden könnten. Gerade aus der Spieleentwicklung sind viele Alternativen verfügbar. Aufgrund der persönlichen Erfahrungen und dem Aufbau und der Flexibilität des Frameworks, erspart die Verwendung von *CgResearch* Einarbeitungszeit und weitere Vorarbeiten bei der Entwicklung des Systems. Aus diesem Grund fällt die Wahl auf die Verwendung dieses Frameworks.

Eine genauere Analyse alternativer Systeme würde über den Rahmen dieser Arbeit hinausgehen und muss aus diesem Grund an dieser Stelle entfallen. Aufgrund der raschen und ständigen Weiterentwicklung in diesem Bereich ist für jeden Anwendungsfall eine aktuelle Recherche, mit den persönlichen Anforderungen im Blick, angeraten.

Verbesserungen für *CgResearch*, die durch die Entwicklung der praktischen Arbeit entstehen und das Framework sinnvoll erweitern, sollen dort mit einfließen. Im Lauf dieser Arbeit wird aber nicht weiter konkret auf diese Bezüge eingegangen. Sie stellen vielmehr einen positiven Nebeneffekt der Arbeit dar.

5.2 Applikationsgrundlage

Das System soll den im Framework vorhandenen *JoglAppLauncher* nutzen. Der auf Java-SWING basierende GUI-Teil des Frameworks dient als Grundlage für das grafische Benutzerinterface des zu entwickelnden Systems.

Zentrales Element des Systems wird eine Editor-Klasse, die zum einen für die Initialisierung aller benötigten Komponenten des Frameworks und des Systems verwendet wird und gleichzeitig, als zentrale Schnittstelle für die einzelnen Funktionalitäten des Programms agiert.

5.3 Beschreibung der Datenstrukturen

In diesem Unterkapitel sollen die grundlegenden Datenstrukturen beschrieben werden, die für die Umsetzung des Systems benötigt werden. Soweit möglich und sinnvoll, soll der Szenengraph des Frameworks verwendet werden. Dafür ist es nötig, dass einige Klassenvererbungen beachtet werden (siehe [Abbildung 4]).

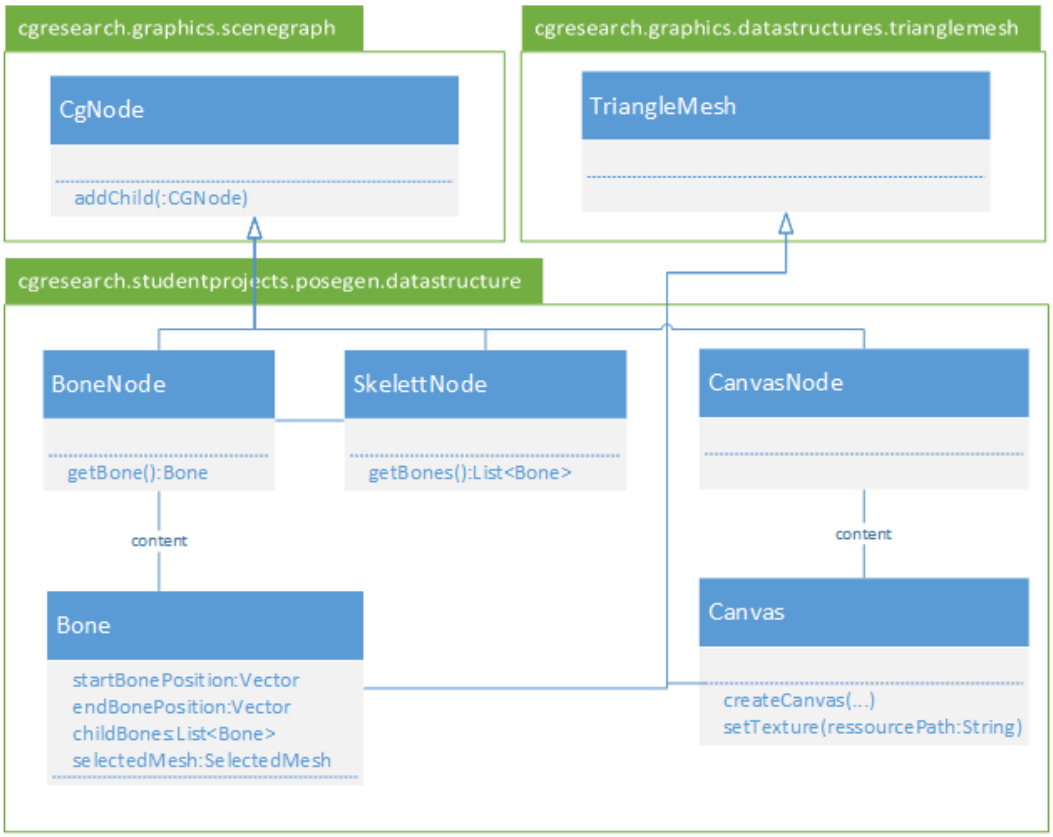

Abbildung 4 : Vereinfachte Klassenübersicht für Objekte zur Integration in den Szenegraphen

5.3.1 Knochen

Ein Knochen ist über eine einzigartige ID zu identifizieren. Er enthält Attribute für eine Start- und eine Endposition. Für die Bewegung eines Knochens wird außerdem eine Referenz auf alle am Ende „angehängten“ Knochen gespeichert. Im Fall einer Bewegung wird diesen sogenannten Kind-Knochen ebenfalls die Bewegung mitgeteilt, damit auch diese ihre Transformationen vornehmen. Jedem Knochen können beliebig viele Kind-Knochen zugeordnet werden, wobei jedem Kind-Knochen nur ein Eltern-Knochen zugeordnet ist. Das Feld zum Speichern der Referenz auf die Kind-Knochen kann leer sein.

Wird einem Knochen eine Rotation über einen Methodenaufruf mitgeteilt, initiiert er die Bewegung der an ihn gebundenen Dreiecke innerhalb der Canvas-Datenstruktur. Die an ihn gebundenen Dreiecke dieser Datenstruktur speichert jeder Knochen innerhalb einer SelectedMesh Datenstruktur. Zur optischen Darstellung erzeugt jeder Knochen ein Dreiecksnetz zwischen seinem Start- und Endpunkt.

5.3.2 Skelett

Das Skelett ist eine Sammlung von Knochen, da sich die Baumstruktur des Skeletts über die Beziehungen der Knochen untereinander ergibt, ist das Skelett selbst eine auf Knochen spezialisierte Liste.

Zusätzlich dient die Struktur als Schnittstelle, um einzelne Knochen zu rotieren oder gesamte Skelettpositionen zu laden. Beim Laden von Skelettpositionen ist darauf zu achten, dass die Rotationen streng hierarchisch ausgehend vom Root-Knoten hinunter zu den Blättern zu erfolgen hat. Damit wird sichergestellt, dass keine Rotation einen Einfluss auf eine bereits vorher durchgeführte Rotation ausübt.

5.3.3 Canvas / Leinwand

Als Canvas oder Leinwand wird die Struktur bezeichnet, die das verwendete Bild in der Benutzeroberfläche repräsentiert.

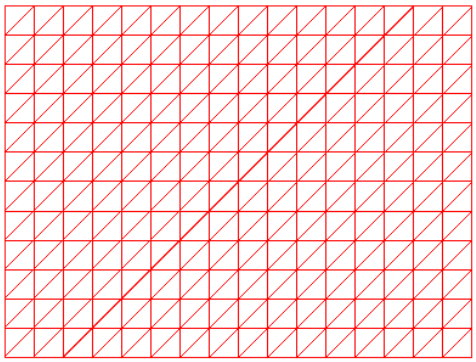

Abbildung 5: Ein aus 192 Dreiecken bestehendes planares Dreiecksnetz

Die Basis der Datenstruktur bildet ein viereckiges, planares Dreiecksnetz. Durch Verwendung von Texturkoordinaten und einer Rastergrafik als Textur wird mit der Leinwand die Grafik dargestellt.

Die Abmessung der Leinwand soll einstellbar sein, um sich an verschiedene Grafiken anzupassen. Bei der Manipulation der Grafik werden die einzelnen Eckpunkte der Dreiecke verschoben, damit sind die Anzahl der erzeugten Vertices die Basis für den möglichen Detailierungsgrad der Deformationen. Auf diese Weise hat diese Anzahl auch einen direkten Einfluss auf die Qualität der erzielten Ergebnisse und soll über Parameter konfigurierbar sein. Je nach Anwendungsfall ist damit eine Abwägung möglich, zwischen einem optisch möglichst ansprechenden Ergebnis und dem Berechnungsbedarf, der für die verschiedenen Operationen im System nötig ist. Es gilt, je mehr Dreiecke verarbeitet werden, desto mehr Zeit nehmen die Berechnungen in Anspruch. Aber je weniger Dreiecke vorhanden sind, desto eher kommt es zu unerwünschten Verzerrungen des Bildes oder zur Bildung von grafischen Artefakten.

5.3.4 Gewichtszuordnung und SelectedMesh

Für die Verbindung der Knochen zu der Leinwand, soll eine geeignete Datenstruktur angelegt werden. Sie soll sich bei der Umsetzung an den Anforderungen, der unter 5.4 beschriebenen Vorgehensweise eines automatischen Skinning-Prozesses orientieren.

Jedem Knochen soll außerdem eine direkte Referenz übergeben werden, eine Referenz auf die mit ihm verbundenen Vertices und einem dazu individuellen Gewicht. Diese werden in einer Datenstruktur *SelectedMesh* gespeichert und übergeben.

5.3.5 Skelettposition

Eine Skelettposition speichert die Winkel aller im Skelett enthaltenen Knochen. Gleichzeitig bietet sie Operationen für den Vergleich mit anderen Skelettpositionen. Die Differenz zwischen zwei Skelettpositionen wird definiert als die Differenz der Winkel zwischen den beiden Positionen. Somit wird für jeden Knochen angegeben, um wieviel Grad eine Drehung zu vollführen ist, um dadurch die zweite Skelettposition zu erreichen.

5.4 Verfahren: Skinning

5.4.1 Grundgedanke und Einordnung

Aus den funktionalen Anforderungen ergibt sich, dass das System dem Anwender eine Funktion zum automatischen Skinning anbieten soll. Ziel ist es, dem Anwender möglichst viel Arbeit bei der Erstellung neuer Grafiken abzunehmen. Alternative Verfahren, wie die manuelle Verbindung von Vertices und Knochen sind vorerst nicht vorgesehen.

Die Knochen sollen mit einzelnen Vertices der Leinwand verbunden werden. Würde die Zuordnung zu den Facetten erfolgen, müssten zusätzliche Grenzfälle beachtet und behandelt werden. Wie bei dem Fall, dass ein Knochen an zwei aneinandergrenzenden Facetten eine unterschiedliche Gewichtung ausübt. Hier wäre zu entscheiden, wie die Vertices behandelt werden, die an beide Facetten grenzen. Auch müsste ein Referenzpunkt für die Entfernungsmessung definiert und berechnet werden und nicht zuletzt würde damit der mögliche Detailierungsgrad auf die Größe einer Facette beschränkt. Der Skinning-Prozess soll aus diesem Grund mit einem Verfahren zur Knochen- / Vertex-Zuordnung erfolgen.

5.4.2 Skinning-Varianten

Die Variante, das Skinning manuell durch den Anwender durchführen zu lassen, entfällt aufgrund der genannten Anforderungen. Ziel ist es ein, Verfahren auszuwählen, das mit möglichst wenig Einflussnahme durch den Anwender ein möglichst ansprechendes Ergebnis liefert. Die Zuordnung von Knochen zu den Vertices soll auf Grundlage der Distanz zueinander erfolgen. Von den verschiedenen Optionen, welche Zuordnung sich aus der Distanzmessung ergibt, sollen zwei mögliche Verfahren vorgestellt werden.

Variante A – Binäre Zuordnung:
Jedem Vertex wird genau ein Knochen zugeordnet. Die Auswahl fällt auf den Knochen, der dem jeweiligen Vertex am nächsten ist.

Variante B – Gewichtete Zuordnung:
Jeder Vertex wird unterschiedlich stark an beliebig viele Knochen gebunden. Wie stark ein Vertex an einen Knochen gebunden wird, hängt von der Entfernung und einer Gewichtungsfunktion ab. Mehrere Knochen summieren sich auf einen Gesamteinfluss von 100% für jeden Vertex bzw. auf 0% für Vertices, die in gar keinem Knochen-Einflussbereich liegen.

Optional sind zusätzliche Erweiterungen möglich, z.B. dem Anwender die Möglichkeit der Einflussnahme auf den Prozess der Zuordnung zu geben. Denkbar wären nachträgliche Anpassungen der Gewichte oder vorherige Einstellung bestimmter Knochen, sodass diese einen vom Standard abweichenden Einfluss ausüben.

5.4.3 Variantenanalyse

Beide Varianten erscheinen in bestimmten Situationen vorteilhaft. Beim binären Verfahren geht der Einfluss auf einen Vertex nur von einem Knochen aus. Dass dieser Einfluss von einem Knochen ausgeübt wird, der auf den Charakter an dieser Position keinen Einfluss haben sollte, ist sehr unwahrscheinlich.

Dafür hat das gewichtete Verfahren Vorteile an Gelenken, also an Stellen an denen zwei Knochen zusammentreffen. Bei einem natürlichen Skelett bzw. bei Muskeln und Sehnen gibt es ebenfalls Bereiche, die von mehreren Strukturen beeinflusst werden. So könnte es auch bei dieser abstrahierten Herangehensweise von Vorteil sein, wenn auf einige Vertices durch mehrere Knochen Einfluss ausgeübt wird. Der Einfluss von Knochen sinkt mit der Entfernung, an deren Stelle treten Einflüsse durch andere Knochen und so sollten gerade in Grenzbereichen, wie einem Gelenk, weichere Übergänge entstehen.

Als Nachteil können aber Bereiche von Vertices entstehen, die einem bestimmten Knochen gar nicht zuzuordnen, sind aber aufgrund ihrer Nähe im ausgewerteten Raum durch diesen beeinflusst werden. Denkbar wäre das zum Beispiel bei eng zusammenstehenden Beinen

einer Charaktergrafik, einer Grafik, bei der der Knochen des linken Beines die Grafik am rechten Bein durch eine Bewegung verzerrt.

Fällt die Gewichtung bei solchen entfernt liegenden Knochen sehr gering aus, ist es möglich, dass die daraus resultierenden Ergebnisse so gering verfälscht werden, dass sie dem Betrachter nicht negativ auffallen. Es könnte aber genauso Bereiche geben, in denen dieser Fall Ergebnisse produziert, die nicht tolerierbar sind.

Eine genaue Beurteilung welches Verfahren hier bessere Ergebnisse liefert erscheint aus derzeitiger Sicht nicht möglich. Gleichzeitig ist das binäre Verfahren leicht als eine vereinfachte Variante des gewichteten Verfahrens umzusetzen. Aus diesem Grund sollen beide Varianten, die binäre und die gewichtete Zuordnung, implementiert und gegeneinander abgewogen werden. Bei der gewichteten Variante soll auf eine Gewichtungsfunktion zurückgegriffen werden, die gerade auf die oben genannten möglichen Probleme Rücksicht nimmt.

5.4.4 Distanzmessung – Punkt zu Strecke

Als Grundlage für die umzusetzenden Skinning-Verfahren wird eine Distanzmessung benötigt. Ein mögliches Verfahren zur Ermittlung der kürzesten Distanz zwischen einem Knochen und einem Vertex wird nun hier vorgestellt.

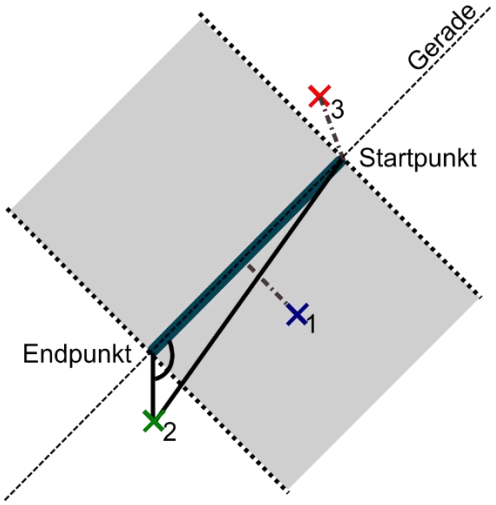

Abbildung 6: Bestimmung der Distanz zwischen einem Knochen und einem Punkt im Raum.

Drei Fälle sind bei der Ermittlung des Abstandes zu unterscheiden.

1. Fall: Die gesuchte Position liegt einem Punkt auf der Strecke *zwischen* Start- und Endpunkt am nächsten. (X1) (Der Punkt befindet sich irgendwo in dem grauen Bereich in [Abbildung 6].)
2. Fall: Die gesuchte Position liegt dem Endpunkt am nächsten (X2).
3. Fall: Die gesuchte Position liegt dem Startpunkt am nächsten (X3).

Zur Unterscheidung der Fälle werden die Winkel zwischen dem gesuchten Punkt und der Start- und Endposition des Knochens ermittelt.

Beträgt der Winkel am Startpunkt von der Geraden zum gesuchten Punkt >90° und <270°, handelt es sich um den 2.Fall.
Beträgt der Winkel am Endpunkt von der Geraden zum gesuchten Punkt >90° und <270°, handelt es sich um den 3. Fall.

Berechnungsvorschrift:

1. Bestimme X als Position des Vertex.
2. Erzeuge die Gerade, die durch den Start- und Endpunkt eines Knochens läuft.
3. Bestimme den Winkel A am Startpunkt zu X.
4. Bestimme den Winkel B am Endpunkt zu X.
5. Winkel A >90° und <270°
   → Der Abstand beträgt X zum Startpunkt.
6. Winkel B >90° und <270°
   → Der Abstand beträgt X zum Endpunkt.
7. Ansonsten
   → Der Abstand beträgt X zur Geraden. Auf der kürzesten Strecke, also dort wo der Verbindungsvektor senkrecht auf der Geraden steht.

5.4.5 Gewichtungsfunktion

Auf Basis der ermittelten Distanz zwischen jedem Knochen und jedem Vertex soll für die zweite Variante eine Gewichtung vorgenommen werden.

Durch eine geschickte Wahl der Funktion soll noch weiter verstärkt werden, dass die am nächsten liegenden Knochen den größten Einfluss auf die Vertices ausüben sollen. Aus diesem Grund soll eine Gaußsche Normalverteilung zur Berechnung der einzelnen Gewichte verwendet werden.

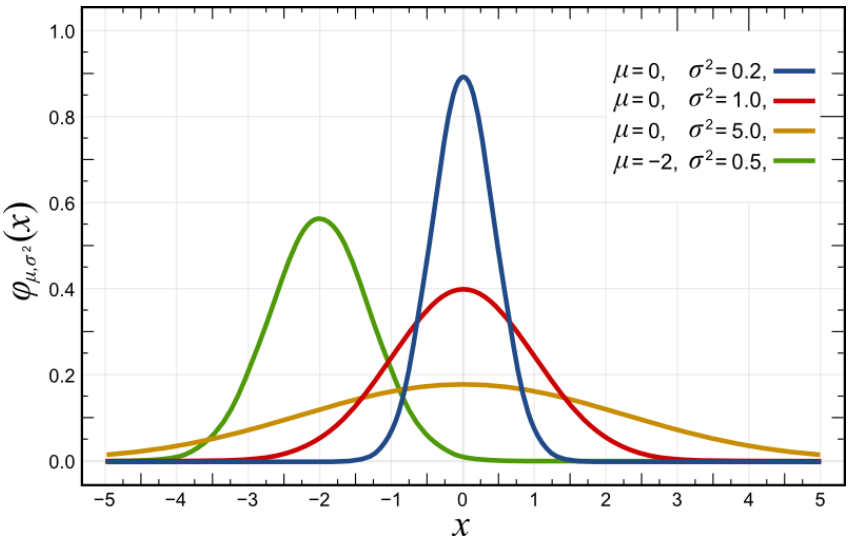

Abbildung 7: Unterschiedlich parametrisierte Gaußverteilungen im Vergleich. (Quelle: https://de.wikipedia.org/wiki/Datei:Normal_Distribution_PDF.svg – Lizenz: Gemeinfrei – Abgerufen am 06.12.2016)

Die Normalverteilung soll aus diesem Grund mit folgender Funktion implementiert werden:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

Dabei gilt:

x ist die Entfernung vom Vertex zum Knochen.

σ^2 ist die Varianz. Je geringer der Wert ausfällt desto stärker fallen die Punkte um den Erwartungswert herum ins Gewicht, aber umso stärker fällt dieser Bereich ab. Bei einer Normalverteilung beträgt dieser Wert 1. Eine Abweichung davon, um das Ergebnis zu optimieren, ist zulässig.

μ ist der Erwartungswert, um den herum sich der Wert zu beiden Seiten reduziert. Das bedeutet, dass der Funktionswert bei μ am größten ist. Dies soll genau dann der Fall sein, wenn der Abstand zwischen Knochen und Vertex 0 beträgt. Somit ist μ auf 0 zu setzen.

Verschiedene, unterschiedlich parametrisierte Gaußkurven sind zur Veranschaulichung in der [Abbildung 7] dargestellt.

Nach Ermittlung aller Gewichte für einen Vertex ist noch eine Normierung durchzuführen, sodass sich das summierte Gewicht aller Beziehungen auf insgesamt 100% addiert.
Andernfalls würden Grafikbereich stärker deformiert werden, sobald sich mehrere bewegende Knochen in der Nähe befinden. Für jeden Vertex ist das Gewicht also noch für jeden Knochen durch folgende Berechnung final einzustellen:

$$Gewichtsfaktor = \frac{Vertex\ Gewicht}{Gesamtgewicht}$$

Wobei das Vertex Gewicht, der von der Gaußverteilungsfunktion berechnete Wert für diesen Knochen ist und das Gesamtgewicht, die Summe aller Knochen-Vertex Gewichte auf diesem Vertex ist.

5.5 Verfahren: Winkelbestimmung Knochen

5.5.1 Motivation und Definition Knochen-Rotation

Für die Deformation der Grafik, sollen verschiedene Skelettpositionen geladen werden können. Verschiedene Skelettpositionen unterscheiden sich ausschließlich durch die Rotation ihrer einzelnen Knochen. Die Länge der Knochen ändert sich nicht bei Bewegungen. Um von einer Position zu einer anderen zu gelangen, muss also für jeden Knochen die Rotation der aktuellen Position mit der Rotation für die zu erreichende Position verglichen werden. Die hierbei ermittelte Differenz ergibt den Winkel, um den der Knochen an seiner Startposition gedreht werden muss, um in die neue Position zu gelangen.

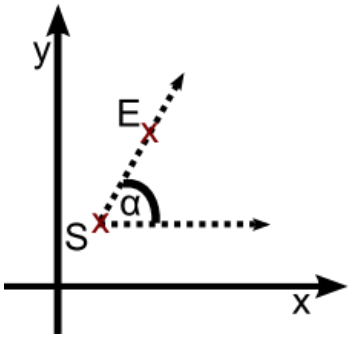

Abbildung 8: Bestimmung der Rotation eines Knochens. Wobei S der Start- und E der Endpunkt des Knochens ist.

Damit die Rotation für alle Knochen vergleichbar ist, wird diese wie folgt definiert: Die Rotation eines Knochens ist der folgende Winkel α (siehe [Abbildung 8]):

Winkelscheitel: Startposition des Knochens
Strahl 1: Beginnend in der Startposition, in positiver Richtung parallel zur X-Achse laufend
Strahl 2: Beginnend in der Startposition, in Richtung zum Endpunkt des Knochens

5.5.2 Ermittlung des Winkels

Zuerst werden mithilfe der Startposition ($\overrightarrow{S}$) und der Endposition ($\vec{E}$) des Knochens zwei Richtungsvektoren bestimmt.

Erster Vektor:

$$\overrightarrow{a} = (\overrightarrow{E} - \overrightarrow{S})$$

Zur Ermittlung des zweiten Vektors wird die Startposition positiv auf der X-Achse verschoben, um einen Punkt ($\overrightarrow{C}$) zu bestimmen.
Zweiter Vektor:

$$\overrightarrow{b} = (\overrightarrow{C} - \overrightarrow{S})$$

Durch Einsetzen in folgender Gleichung wird $\cos(\alpha)$ berechnet:

$$\cos(\alpha) = \frac{\overrightarrow{a} * \overrightarrow{b}}{|\vec{a}| * |\vec{b}|}$$

Das Ergebnis $\cos(\alpha)$ wird noch mit der Inversoperation arccos umgekehrt um den Winkel α zu erhalten. Zu beachten ist hierbei, dass der so ermittelte Winkel aufgrund der Umrechnung aus Kosinus zwischen 0° und 180° liegt.

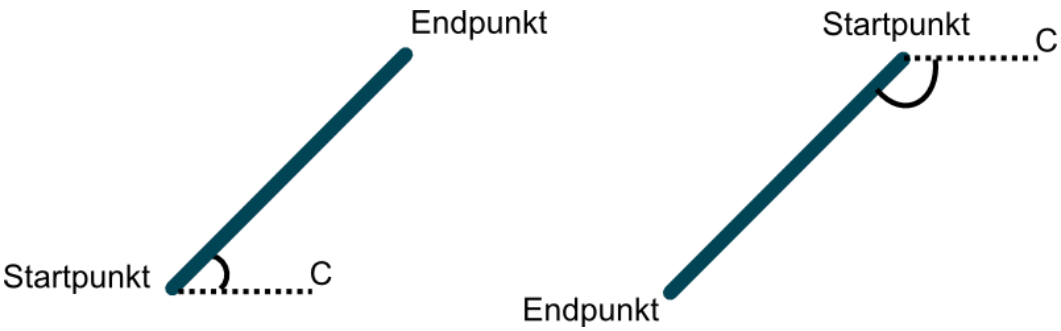

Abbildung 9 Knochenwinkel – Fallunterscheidung: Richtung des geöffneten Winkels.

Um zu ermitteln, in welche Richtung sich dieser Winkel öffnet, wird die Höhe vom Start- und Endpunkt genutzt. Liegt der Endpunkt höher als der Startpunkt, öffnet sich der Winkel nach oben. Liegt der Endpunkt niedriger als der Startpunkt, öffnet sich der Winkel zur anderen Seite. Er beträgt für eine Rotation somit z.B. nicht +90°, sondern -90° bzw. +270°

5.6 Verfahren: Mausinteraktion mit Dreiecksnetzen

5.6.1 Vorhandenes Picking-System

Für Interaktionen mit den Knochen und einzelner Teile des Dreiecknetzes muss das Framework erweitert werden. Die derzeitige Interaktionsmöglichkeit in dem Framework beschränkt sich auf das sogenannte Picking-System, bei dem an einer zentralen Stelle Kugeln innerhalb des Raumes platziert werden können. Anschließend können diese Kugeln ausgewählt und mit kombinierter Tastatur- und Mausinteraktion verschoben werden.

Dieses Picking-System erscheint für diese Anwendung zu statisch und kann vermutlich nur umständlich an die Anforderungen dieses Systems angepasst werden. Für die automatische Erzeugung von interaktiven Handlern zur Knochenpositionsmanipulation soll eine flexible Interaktionsmöglichkeit geschaffen werden.

5.6.2 Motivation

Aufgrund der Anforderungen an das System soll der Anwender mithilfe der Maus die Knochenpositionen verschieben und Knochen rotieren können. Da das vorhandene Picking-System des Frameworks, wie in Kapitel 5.6.1 beschrieben, hierfür nicht verwendet werden kann, soll eine eigene Implementierung erfolgen und hier beschrieben werden.

Für die Interaktion sind zwei Bereiche zu beachten, zum einen ein Anwender, der durch einen Mausklick mit der Szene interagieren möchte und das System, das aufgrund dieses Klicks ermitteln soll, welches Objekt angeklickt wurde.

Grundprinzip dieses Verfahrens ist, dass der Mausklick eines Anwenders einen Strahl in die Szene sendet und geprüft werden soll, ob der Strahl ein Objekt in Form eines Dreiecksnetzes schneidet. Mit einem Dreiecksnetz als Interaktionsobjekt, ist eine flexible Möglichkeit zur Interaktion geschaffen, da verschiedenste Objekte als Dreiecksnetz dargestellt werden können.

Um dies umzusetzen, sind mehrere Schritte erforderlich. Die Position eines Mausklicks muss aus den Bildschirmkoordinaten umgerechnet werden, um zu ermitteln, wo dieser Klick innerhalb der Szene erfolgt ist. Zusätzlich muss mit einem geeigneten Algorithmus ermittelt werden, ob und wenn ja welches Objekt in der Szene mit dem Mausklick getroffen wurde.

5.6.3 Benutzerinteraktion in die Szene

Auch wenn es sich bei der zu verarbeitenden Grafik um eine zweidimensionale Grafik handelt und sich die deformierenden Operationen auch nur auf zwei Achsen innerhalb des Koordinatensystems beziehen, soll das System innerhalb des dreidimensionalen Raumes des Frameworks arbeiten. Es wird nur die Z-Koordinate aller Objekte auf den gleichen Wert gesetzt.

Das Maus-Ereignis, das durch den Klick eines Anwenders erstellt wird, informiert nur darüber, wo auf dem Bildschirm der Klick erfolgt ist. Um zu ermitteln, wo dieser Klick innerhalb der 3D-Szene auf ein Objekt trifft, also welches Objekt der Anwender anklicken wollte, müssen die übergebenen Bildschirmkoordinaten noch umgerechnet werden.

Um den Ansatz für diese Umrechnung besser zu verstehen, ist eine kurze Erläuterung zum Ablauf des Renderings nötig:
Das Rendering in OpenGL erfolgt durch eine Rendering-Pipeline. Bei diesem Verfahren wird durch mehrere aufeinander folgende Schritte ein Bild aus den vorhandenen Objekten im Raum erzeugt. Ausgangspunkt ist hierfür eine virtuelle Kamera. Sie ist zu vergleichen mit einem Fotoapparat, der sich innerhalb des Raums befindet und ein Foto der Szene macht. Die virtuelle Kamera besteht aus einer Position, einem Referenzpunkt bzw. der Blickrichtung, einem Oben-Vektor und einem Öffnungswinkel. Zusätzlich werden noch eine Near- und eine Far-Clipping-Plane definiert - Objekte werden nur dargestellt, wenn sie sich zwischen diesen beiden Ebenen befinden (vgl. [17], S. 45). Wird ein Bild gerendert, so wird das Bild aus der Sicht der Kamera zusammengesetzt. Dafür findet zuerst eine Transformation statt, sodass die Kameraposition der Ursprung der Szene ist. Später findet dann eine Projektion auf die Near-Clipping-Plane statt, mit den Objekten, die aus der Position der virtuellen Kamera sichtbar sind (vgl. [18]).

Klickt ein Anwender also auf einen Bereich im gerenderten Bild, so zeigt er im übertragenen Sinne von der Position der virtuellen Kamera aus auf einen Bildpunkt. Dieser Bildpunkt ist das Ergebnis einer Projektion aus der Szene auf die Near-Clipping-Plane. Diesen Vorgang kann man auch als einen von der Kameraposition ausgehenden Strahl ausdrücken.

5.6.4 Baryzentrische Koordinaten

Mithilfe der baryzentrischen Koordinaten, kann ermittelt werden, ob ein gegebener Punkt innerhalb eines Dreiecks liegt.

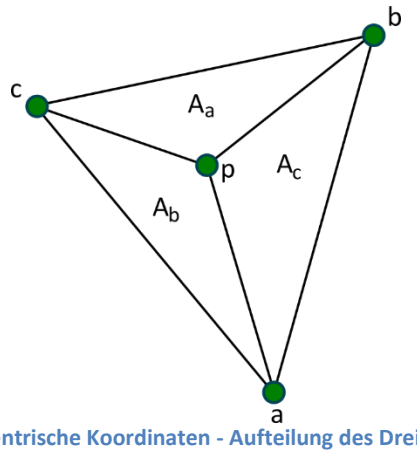

Abbildung 10: Baryzentrische Koordinaten - Aufteilung des Dreiecks in drei Flächen.

Der Punkt **p** innerhalb des Dreiecks (siehe [Abbildung 10]) lässt sich auch durch die Eckpunkte (a, b, c) des Dreiecks mit entsprechenden Gewichten darstellen.

$$p = \alpha\, a + \beta\, b + \gamma\, c$$

α, β, γ Sind hier die baryzentrischen Koordinaten von p

Zur Bestimmung dieser Koordinaten und damit der Gewichte, wird wie folgt vorgegangen:

Für jedes Teildreieck wird der Flächeninhalt berechnet (Aa, Ab, Ac).
Der Flächeninhalt (A) des Dreiecks zwischen den Punkten (a, b, c) wird berechnet.

Der Anteil der Flächen an der Gesamtfläche entspricht der jeweiligen baryzentrischen Koordinate. Es gilt also:

$$\alpha = \frac{Aa}{A} \qquad \beta = \frac{Ab}{A} \qquad \gamma = \frac{Ac}{A}$$

Anhand dieser baryzentrischen Koordinaten kann nun bestimmt werden, ob sich der Punkt p innerhalb des Dreiecks befindet. Liegt der Punkt innerhalb des Dreiecks, dann müssen sich die Gewichte auf 1 summieren. Außerdem liegen die einzelnen Koordinaten im abgeschlossenen Intervall zwischen 0 und 1.

Es gilt also: p liegt innerhalb des Dreiecks, wenn gilt:

$$1 = \alpha + \beta + \gamma$$
$$0 \leq \alpha, \beta, \gamma \leq 1$$

5.6.5 Schnitt Strahl - Dreieck

Ziel des Verfahrens ist es, zu ermitteln ob ein Objekt innerhalb der Szene angeklickt wurde. Dafür soll getestet werden, ob sich ein Dreieck eines interaktiven Objektes an dieser Position befindet. Um zu ermitteln, wo der Mausklickt erfolgt ist, muss ein Strahl erzeugt werden, der aus der Perspektive des Anwenders in die Szene läuft. Grundlage hierfür sind die Parameter der virtuellen Kamera. Ob ein bestimmter Punkt innerhalb eines Dreiecks liegt, lässt sich mithilfe der baryzentrischen Koordinaten berechnen.

Diese beiden Elemente sollen dazu genutzt werden, zu ermitteln, welches Objekt angeklickt wurde. Dafür soll berechnet werden, ob der Strahl des Mausklicks durch einen Punkt innerhalb eines Dreiecks von einem Objekt läuft.

Eine Möglichkeit für diese Berechnung ist der *Möller-Trumbore ray-triangle intersection Algorithmus*. Der Algorithmus baut auf dem Prinzip der baryzentrischen Koordinaten auf und ermittelt, ob ein gegebener Strahl durch ein gegebenes Dreieck läuft. Dafür wird eine Transformation konstruiert und auf den Ursprung des Strahls angewandt. Dieses liefert die Distanz zum Dreieck und die baryzentrischen Koordinaten (vgl. [19]).

Als weiterführende Quelle sei hier auf das Paper über diesen Algorithmus von Möller und Trumbore verwiesen ([19]). Neben einer detaillierteren Erläuterung enthält es auch den Quellcode für eine Implementierung. Eine alternative Implementierung ist unter: [20] verfügbar.

6 Implementierung

Nach der theoretischen Vorarbeit soll hier nun der umgesetzte Prototyp nach der Implementierung beschrieben werden. Dies beginnt mit einem kurzen Blick auf die Anpassungen der Benutzeroberfläche und geht dann detaillierter auf die Implementierung der Datenstrukturen ein. Es folgt eine Beschreibung der Umsetzung für die Interaktion des Benutzers mit Dreiecksnetzen und es wird die Implementierung der Skinning-Verfahren vorgestellt.

6.1 Entwicklungsumgebung

Wie in der Konzeption beschrieben, baut das System auf dem bereits bestehenden Framework *CgResearch* auf. Aus diesem Grund wird als Programmiersprache ebenfalls Java gewählt. Aufgrund vorhandener Erfahrung wurde als IDE *Eclipse Mars* verwendet und der Quellcode mit *Git* verwaltet.

6.2 GUI Erweiterungen

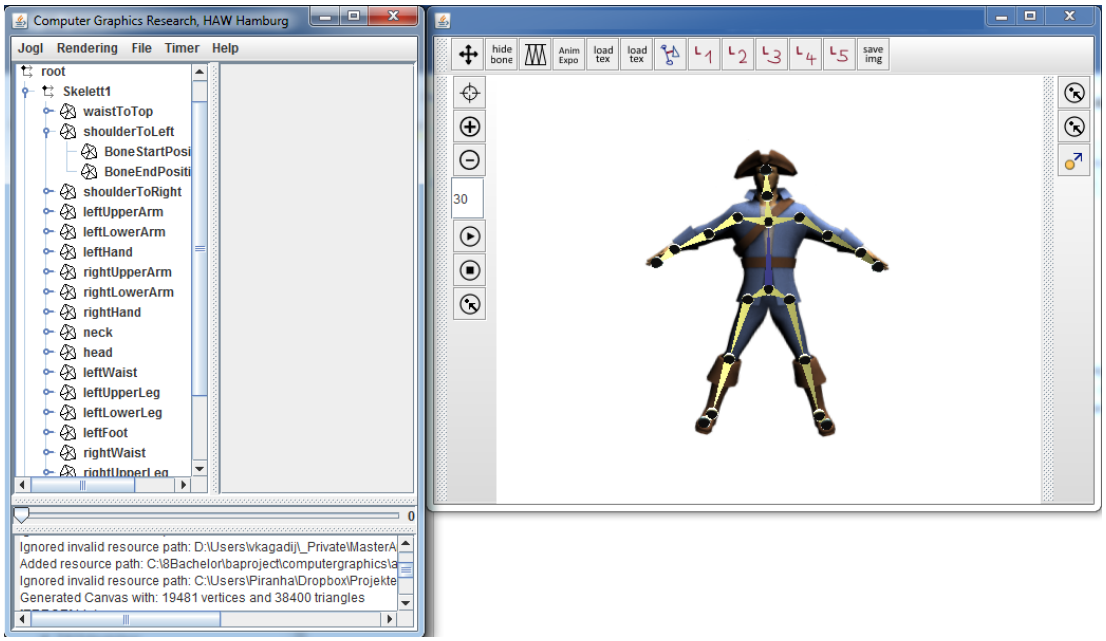

Abbildung 11: Grafische Oberfläche des implementierten Systems.
(Quelle: Die in der Szene enthaltene Charaktergrafik ist ein Rendering des folgenden Models: http://opengameart.org/content/pirate
Lizenz: CC-BY 3.0. Erstellt von Clint Bellanger - Abgerufen am 08.12.2016)

Der bereits vorhandene Aufbau der grafischen Oberfläche wurde größtenteils vom Framework übernommen. Erweitert wurde die Oberfläche für das System um zusätzliche Buttons, die in zwei Toolbars an der Oberseite und am rechten Rand zusammengefasst wurden (siehe [Abbildung 11]). Der Kameracontroller für die Steuerung des Sichtfeldes wurde angepasst, sodass keine versehentliche Rotation der Szene auftritt.

6.3 Datenstrukturen im Szenengraph

6.3.1 CgNode

Wie in Kapitel 5.3 beschrieben, sollten sich die Datenstrukturen in die bestehende Struktur des Szenegraphen eingliedern. Für diesen Zweck ist innerhalb von *CgResearch* die Klasse *CgNode* von großer Bedeutung. Aus diesen Knoten setzt sich der Szenengraph als Baum zusammen. Ein Knoten kann Inhalte, wie z.B. ein Dreiecksnetz enthalten, aber auch Operationen, wie z.B. eine Skalierung implementieren, die anschließend beim Rendering quasi vererbt wird und sich damit auch auf das Rendering der nachfolgenden Knoten auswirkt.

Für *Content* in *CgNodes* werden Rendering-Fabriken implementiert und registriert. Beim Rendering wird später dynamisch aus dem Szenengraph mithilfe der Rendering-Fabriken ein Renderinggraph erstellt.

6.3.2 Skelett und Knochen

Das Skelett ist als Erweiterung der CgNode-Klasse implementiert. In der derzeitigen Ausbaustufe des Systems wird bei der Initialisierung der Klasse ein Skelett bestehend aus aneinanderhängenden Knochen erzeugt. Diese Knochen können über Schnittstellen von außen erreicht werden. Zum Abrufen des aktuellen Status ist es möglich die aktuelle *Skelettposition* abzurufen. Das Skelett selber enthält keinen zu rendernden Inhalt, dieser entsteht erst durch die in ihm enthaltenen Knochen.

Es gibt zwei Klassen für den Knochen. Zur Eingliederung innerhalb des Szenegraphen, gibt es die sogenannte *BoneNode*-Implementierung. Der fachliche Teil der Datenstruktur wurde hingegen mit der *Bone*-Klasse umgesetzt. Die Knochen lassen sich über eine eindeutige ID identifizieren, was für spätere Verknüpfungen mit der Leinwand wichtig war. Visuell dargestellt werden die Knochen, indem sie eine Pyramide als Dreiecksnetz zwischen der Start- und der Endposition generieren.

Zum Verschieben der Knochenposition gibt es Steuerpunkte. Diese Steuerpunkte bestehen aus einem kleinen annähernd kugelförmigen Dreiecksnetz. Jeder Knochen generiert an seiner Startposition einen Kontrollpunkt zum Verschieben seiner Position. Handelt es sich bei dem Knochen um ein Blatt innerhalb der Baumstruktur des Skeletts, wird zusätzlich ein Kontrollpunkt an der Endposition des Knochens erstellt. Wird einer der Kontrollpunkte durch den Anwender bewegt, werden die verbundenen Knochen über ihre Positionsänderung informiert damit diese ihre Position entsprechend aktualisieren. An Gelenken, also den Verbindungen zwischen zwei Knochen, befindet sich nur ein Kontrollpunkt. Hiermit soll sichergestellt werden, dass die Verbindung zwischen zwei zusammenhängenden Knochen nicht getrennt werden kann. Eine Änderung an diesen Positionen führt zu einer Aktualisierung der Endposition des einen Knochens und zu der Aktualisierung der Startposition des anderen Knochens.

Skelettpositionen werden als Rotation ihrer einzelnen Knochen gespeichert und können mit anderen Skelettpositionen verglichen werden. Auf dieser Basis können Rotationsskelette erzeugt werden, sie speichern für jeden Knochen die nötige Drehung, um von einer zu einer anderen Position zu wechseln. Dies wird verwendet beim Laden einer gespeicherten Skelettposition, indem innerhalb des Skeletts ein Rotationsskelett erzeugt und verarbeitet wird. Zur Bestimmung der Winkel wurde das in 5.5 beschriebene Verfahren zur Ermittlung von Knochenrotationen angewandt. Alle Winkel werden im System, anders als in dem vorgestellten Verfahren, nicht in Gradmaß, sondern in Bogenmaß implementiert. Auch wird immer die kürzest mögliche Rotation ermittelt, wobei das Vorzeichen des Drehwinkels die Drehrichtung angibt.

6.3.3 Canvas / Leinwand

Analog zu den anderen Strukturen, die innerhalb des Szenen- und Renderinggraphen abgebildet werden sollen, besteht auch die Leinwand aus zwei getrennten Implementierungen. Sie besteht zum einen aus einer von CgNode abgeleiteten Klasse für die Eingliederung in den Szenengraphen. Für das Rendering existiert eine Content-Klasse, die mit *TriangleMesh* von der Klasse für Dreiecksnetze aus dem Framework erbt.

Wie in 5.3.3 beschrieben, sollte die Größe und Auflösung der Leinwand individuell anpassbar sein. In der Implementierung wurde das umgesetzt, indem bei der Erstellung Parameter hierfür übergeben werden können. Einstellbar ist die Gesamtbreite und -höhe, außerdem die Größe eines einzelnen Quadrats innerhalb der Leinwand. Ein Leinwand-Quadrat besteht immer aus zwei Dreiecken mit einer Trennung zwischen der oberen linken und unteren rechten Ecke. Anhand der Parameter werden dann durch zwei ineinander geschachtelte Schleifen die einzelnen Vertices an ihrer Ursprungsposition erstellt. Jeweils drei Vertices werden zu einem Dreieck verbunden. Da ein zusammenhängendes Dreiecksnetz erstellt werden sollte, musste noch drauf geachtet werden, dass an keiner Position zwei Vertices erstellt werden. Stattdessen ist abgesehen von den Ecken, jeder Vertex auch Bestandteil von mehreren Dreiecken.

Aus Performancegründen wurde die Prüfung, ob an einer Position bereits ein Vertex erstellt wurde, auf die letzten n-Vertices beschränkt. Wobei sich n hier aus dem Aufbau der Schleifen ergibt. Ein bereits vorhandener Vertex kann nur für ein anliegendes Dreieck erstellt worden sein, damit wurde es frühestens in der vorher erstellten Reihe von Leinwand-Quadraten erstellt.

Die Standardimplementierung für Dreiecksnetze verwendet bei der Erstellung der Vertices die Vertex Klasse aus dem *cgresearch.graphics.datastructures.trianglemesh-Package*. Diese Implementierung erlaubt allerdings keine nachträgliche Positionsmanipulation, deswegen wurde mit *VertexMutable* eine zusätzliche Klasse vom Typ *IVertex* eingeführt, bei der diese Operationen möglich sind.

Bei der Erstellung der Vertices für die Leinwand werden ebenfalls die Texturkoordinaten berechnet und in den Vertices gespeichert. Hierbei wird allerdings nicht berücksichtigt, welches Seitenverhältnis die später zu ladende Grafik hat, sondern es wird sich nur an dem Seitenverhältnis der Leinwand orientiert.

Es ist also darauf zu beachten, dass das Seitenverhältnis der Grafik mit dem übergebenen Seitenverhältnis der Leinwand übereinstimmt. Andernfalls wird die geladene Textur verzerrt dargestellt.

6.4 Maus – Objekt – Interaktionen

Mit der Maus ist es möglich, die oben beschriebenen Kontrollpunkte der Knochen zu verschieben, und damit die Position der Knochen zu verändern, außerdem können Knochen angeklickt werden, um diese zu drehen. Beides basiert auf dem in 5.6 beschriebenen Algorithmus zur Interaktion mit Dreiecksnetzen.

Der Algorithmus wurde in leicht abgewandelter Form implementiert. Die Tiefenprüfung entfällt. Trifft der Strahl, der aus einem Mausklick erzeugt wurde, mehrere Objekte innerhalb der Szene, dann ist das angeklickte Objekt das Objekt, das den kürzesten Abstand zur virtuellen Kamera hat. Da sich in diesem System die Tiefenposition der Objekte aber nicht unterscheidet, ist diese Verarbeitung nicht nötig. Aus diesem Grund werden die Entfernungsmessungen des Algorithmus nicht weiter ausgewertet, sondern verworfen.

Die aus mehreren Teilen bestehende Implementierung wird in der zentralen *Editor*-Klasse zusammengeführt. Die *TriangleMeshPicking*-Klasse implementiert den in 5.6.5 beschriebenen Algorithmus von Möller-Trumbore in der abgewandelten Form. Für die Erzeugung des Strahls aus der Mausposition wird die bereits implementierte Variante des Frameworks verwendet.

TriangleMeshPicking wird innerhalb der *Editor*-Klasse initialisiert und für den Empfang von Mausereignissen auf das GUI registriert. Dabei werden der Klasse die Referenzen auf die interaktionsfähigen Objekte übergeben, zusammen mit einem implementierten Interface, das bei einem erfolgreichen Klick auf das Objekt aufgerufen wird.

Auf diese Weise können die unterschiedlichen Verfahren bei einer Anwender-Interaktion in den Klassen individuell abgebildet werden.

6.5 Automatisches Skinning

Für das automatische Skinning wurden unter Punkt 5.4 zwei Verfahren beschrieben, die umgesetzt werden sollten. Die binäre und die gewichtete Variante der Knochen-Vertex-Zuordnung.

Dafür wurde eine Datenstruktur angelegt, die für einen Knochen eine Liste mit Vertices und jeweils einem individuellen Gewicht enthält. Wird über den Klick auf einen Button der Skinning-Prozess durch den Anwender gestartet, dann werden die Knochen und die Leinwand an eine weitere Klasse übergeben. Diese Klasse iteriert über alle Vertices der Leinwand. Mit dem in Punkt 5.4.4 beschriebenen Verfahren erfolgt dann die Distanzmessung zwischen allen Knochen und dem Vertex. Die so ermittelten Entfernungen werden in der oben genannten Datenstruktur zwischengespeichert.

Beim *gewichteten Verfahren* erfolgt nun die Berechnung der einzelnen Gewichte. Dafür wurde eine Methode implementiert, die alle Gewichte anhand der Gaußschen Normalverteilung zuordnet und anschließend alle Gewichte normiert, sodass ein Gesamtgewicht über alle Knochen an dem Vertex von 1,0 entsteht.

Für das *binäre Verfahren*, bei dem jeder Vertex nur von einem Knochen beeinflusst werden soll, wird aus der Liste eine Verbindung mit der geringsten Distanz, also dem größten Gewicht, behalten und alle anderen verworfen.

6.6 Utility – Vektormanipulation

Als zentrale Datenstruktur werden in vielen Bereichen des Systems Vektoren genutzt. Verwendet wurde dafür die *Vector*-Klasse aus dem *CgResearch*-Framework. Die Implementierung bietet bereits Methoden für die üblichen atomaren Operationen auf Vektoren. Dennoch werden bei der Umsetzung an unterschiedlichen Stellen weiterführende Operationen benötigt. Um diese Lücke zu schließen, wurde die Utility-Klasse *VectorHelper* erstellt. Sie bietet auf die Anforderungen des Systems zugeschnittene, spezielle Distanz-, Winkel- und Rotationsberechnungen.

7 Evaluation

In diesem Kapitel erfolgt eine Auswertung der Arbeit. Zu Beginn wird die Auswahl des Frameworks betrachtet. Im Anschluss werden die funktionalen und nicht funktionalen Anforderungen mit dem entwickelten Prototypen abgeglichen und eine Performance Analyse durchgeführt. Es erfolgt eine Auswertung des durchgeführten Skinning-Verfahrens und ein Ausblick auf den Umgang mit Tiefeninformationen im System.

7.1 Verwendung des CgResearch Frameworks

Insgesamt hat sich das Framework in seiner Anwendung bewährt, da es sich bei dem Framework um ein System handelt, das innerhalb des Studienbetriebs durch Prof. Dr. Jenke und Studierende der HAW-Hamburg von Grund auf neu entwickelt wurde, ist der Umfang des Frameworks im Vergleich zu anderen auf dem Markt erhältlichen Systemen derzeit noch als eher klein zu betrachten. In einigen Bereichen der Entwicklung führte dies dazu, dass Funktionalitäten vermisst wurden oder für den speziellen Anwendungsfall noch nicht ausgelegt waren. Gleichzeitig stellte sich diese Eigenschaft aber auch als ein großer Vorteil heraus. Das Framework fühlte sich nie als Last an, die die Entwicklung in irgendwelchen Bereichen einschränkt. Neue Anforderungen durch das System konnten schnell und unkompliziert in das Framework integriert werden und dank des offenen Quellcodes und des Aufbaus, war es jederzeit möglich, sich tiefer in die Arbeitsweise des Frameworks einzuarbeiten, um ein besseres Verständnis zu entwickeln.

7.2 Erfüllung der Anforderungen

Zu den in Kapitel 4.5 aufgestellten Anforderungen an das Softwaresystem wird im Folgenden eine Evaluation durchgeführt.

7.2.1 Funktionale Anforderungen

Die einzelnen Anforderungen wurden bei der Implementierung berücksichtigt und alle Anforderungen konnten umgesetzt werden. Dennoch ist festzustellen, dass einige Anforderungen zwar als formal erfüllt gelten können, den Autor in seiner Umsetzung trotzdem nicht ganz zufriedenstellen. Diese werden aber nun detailliert betrachtet.

Eine Rastergrafik im PNG-Dateiformat in die Anwendung importieren können
Die Anforderung wird vom System teilweise erfüllt. Nach einem Klick auf den Button zum Laden einer Textur wird über einen Dialog zur Auswahl einer JPG- oder PNG-Datei aufgefordert. Nach erfolgreicher Auswahl wird die Datei an die Leinwand übergeben und dort als neue Textur hinterlegt. Es ist für den Anwender jedoch nicht möglich, das 4:3 Seitenverhältnis der Leinwand anzupassen. Wird eine Grafik mit einem anderen Seitenverhältnis importiert, wird dieses verzerrt auf der Leinwand dargestellt.

Ein von der Anwendung vorgefertigtes Skelett laden und verwenden können
Die Anforderung wird vom System teilweise erfüllt. Es wird automatisch ein vorgefertigtes Skelett bestehend aus neunzehn Knochen geladen. Skelette können aber nicht über eine Datei importiert werden. Es ist auch nicht möglich Knochen zu entfernen oder hinzuzufügen.

Die Skelettposition an die Grafik anpassen können, indem einzelne Knochen verschoben werden.
Die Anforderung wird vom System erfüllt. Die einzelnen Start- und Endpositionen der Knochen können im Editiermodus verschoben werden, damit sind alle Anpassungen möglich. Aus Sicht der Benutzerfreundlichkeit wäre es wünschenswert, dieses noch etwas intuitiver zu gestalten z.B. indem ganze Knochen verschoben und auch rotiert werden können.

Vorgespeicherte Skelettposen laden können, um damit das verknüpfte Bild zu deformieren.
Die Anforderung wird vom System erfüllt. Es sind fünf Skelettpositionen vorgespeichert, die direkt über einzelne Buttons geladen werden können. Beim Laden einer Position wird das Skelett in die gespeicherte Position gedreht.

Knochen einzeln rotieren können, um damit die verbundene Grafik zu deformieren
Die Anforderung wird vom System erfüllt. Der Anwender kann einen Button aktivieren für Rotation im Uhrzeigersinn bzw. für Rotation gegen den Uhrzeigersinn. Anschließendes

Klicken auf einzelne Knochen, dreht diese um einen fest voreingestellten Wert von 3°. Ähnlich wie beim Verschieben der Knochenpositionen, könnte hier durch eine etwas intuitivere Interaktionsmöglichkeit die Benutzerfreundlichkeit noch erhöht werden.

Ein fertig positioniertes Skelett automatisch mit der Grafik verbinden können.
Die Anforderung wird vom System erfüllt. Durch Klicken auf den Skinning-Button, werden die Knochen automatisch mit der Grafik verbunden. Eine detailliertere Bewertung der verwendeten Verfahren findet in einem gesonderten Teil dieser Evaluation statt (siehe: Kapitel 7.3).

Animationen in Form einer Einzelbilderserie exportieren können
Die Anforderung wird vom System teilweise erfüllt. Für den Export von Grafiken wird eine bestehende Schnittstelle genutzt, die das aktuell gerenderte Bild speichert. Die Kontrolle über diese Schnittstelle ist aber nur eingeschränkt möglich. Aus diesem Grund wird dem Anwender für jeden Frame ein Dialog präsentiert, bei dem er einen Dateinamen eingeben muss. Dies erscheint zumindest aus der Perspektive von Benutzerfreundlichkeit nachteilig.

Manipulierte Grafiken im PNG-Dateiformat exportieren können.
Die Anforderung wird vom System erfüllt. Das derzeitig gerenderte Bild, kann nach Aktivieren der Exportfunktion über einen Dateidialog als PNG-Datei exportiert werden. Aufgrund der Hintergrundfarbe der gerenderten Szene, wird die Grafik aber ohne Transparenz also ohne Alphawerte gespeichert.

7.2.2 Nicht funktionale Anforderungen

Die nicht funktionale Anforderung an die Performance, dass keine Benutzerinteraktion länger als 3 Sekunden in Anspruch nehmen soll, wird in vollem Umfang erfüllt. Selbst bei Leinwänden mit mehr als 150.000 Polygonen können die Anforderungen auf dem Testsystem erfüllt werden, eine detaillierte Auswertung zu den performance-kritischsten Bereichen ist dem folgenden Kapitel zu entnehmen.

7.2.3 Performance Analyse

Die folgenden Performancetests wurden auf einem PC mit folgenden Spezifikationen durchgeführt:

Betriebssystem:	Windows 7 Professional – SP1 / 64bit
Prozessor:	Intel(R) Core(TM) i5-4590 CPU @ 3.30GHz
RAM:	8,00 GB
Grafikkarte:	HIS Radeon R9 280X / 3072MB

Merkbare Reaktionszeiten im System treten nur in zwei Fällen auf. Zum einen betrifft dies das Durchführen des automatischen Skinning-Prozesses und die initiale Erstellung der Leinwand. Das Importieren von Texturen findet in nicht messbarer, kurzer Zeit statt.

Tabelle 1: Skinning-Prozess mit unterschiedlich stark detaillierten Leinwänden. (benötigte Zeit in ms)

Durchlauf	153.600 Dreiecke	10.004 Dreiecke
1	1.787	169
2	1.701	128
3	1.747	125
4	1.715	121
5	1.709	132
6	1.755	108
7	1.761	109
8	2.201	110
9	1.893	137
10	1.787	109
Durchschnitt	**1.806**	**125**

Wie in [Tabelle 1] zu erkennen, erfolgt der Skinning-Prozess im System ausreichend schnell. Mit einer detaillierten Leinwand, bestehend aus 10.004 Dreiecken, werden durchschnittlich 125ms benötigt, um alle Vertex-Knochen Verbindungen und Gewichtungen durchzuführen. Selbst bei einem sehr fein detaillierten Dreiecksnetz, bestehend aus 153.600 Dreiecken, dauert der Prozess nur durchschnittlich 1806ms.

Tabelle 2: Leinwanderstellung. Linke Tabelle: Aktuelles System. Rechte Tabelle: System ohne last-n Optimierung. (benötigte Zeit in ms)

Durchlauf	153.600 Dreiecke	10.004 Dreiecke
1	981	34
2	970	30
3	982	31
4	969	30
5	1.054	30
6	988	32
7	972	35
8	971	33
9	986	31
10	984	30
Durchschnitt	**986**	**32**

Durchlauf	153.600 Dreiecke	10.004 Dreiecke
1	44.537	105
2	44.555	100
3	44.245	96
4	46.175	103
5	47.639	106
6	44.154	102
7	44.364	109
8	45.206	103
9	45.329	105
10	44.872	106
Durchschnitt	**45.108**	**104**

Die initiale Erstellung einer Leinwand, inklusive der Konstruktion des gesamten Dreiecksnetzes und der Erstellung aller Texturkoordinaten, benötigt durchschnittlich 32ms für eine Leinwand mit 10.004 Dreiecken [Tabelle 2]. Auch in diesem Fall bleibt die benötigte Zeit, selbst bei einer sehr fein detaillierten Leinwand aus 153.600 Dreiecken, mit durchschnittlich 986ms unterhalb der geforderten Antwortzeit von 3 Sekunden.

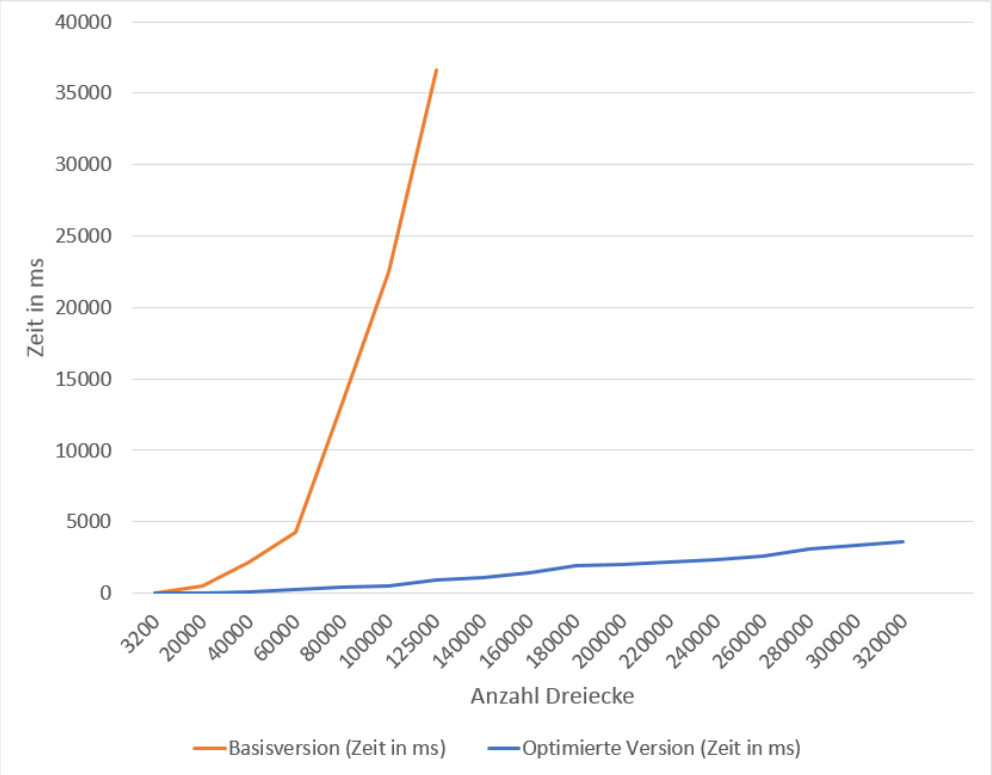

Abbildung 12: Erstellung verschieden detaillierter Leinwände. Vergleich Zeitbedarf. (Zeitbedarf in Millisekunden – Durchschnitt aus fünf Einzelmessungen. Basisversion ohne Optimierung nur bis 125.000 Dreiecke)

Die in 6.3.3 beschriebene Optimierung, nur die letzten n-Vertices nach bereits vorhandenen Vertices zu durchsuchen, bringt gerade bei sehr großen Leinwänden eine spürbare Geschwindigkeitsverbesserung, ablesbar in der Vergleichsmessung, die auf der rechten Seite unter [Tabelle 2] aufgeführt ist. Während sich die Zeit für eine Leinwand mit 10.004 Dreiecken ungefähr verdreifacht und damit trotzdem nur circa eine Zehntelsekunde benötigt, steigt der Zeitbedarf bei dem fein detaillierten Dreiecksnetz fast um den Faktor 46.

Die Differenz entsteht offensichtlich dadurch, dass beim Hinzufügen noch nicht vorhandener Vertices die gesamte Liste durchsucht wird. Dieser Effekt fällt bei immer mehr zu verarbeitenden Dreiecken auch immer stärker ins Gewicht. Beim optimierten Verfahren steigt die Anzahl der zu durchsuchenden Dreiecke nur linear um die Anzahl der zusätzlichen Dreiecke einer Spalte, während er bei dem nicht optimierten Verfahren quadratisch ansteigt. Dieser Komplexitätsunterschied zeigt sich auch in einer Vergleichsmessung, bei

der unterschiedlich stark detaillierte Leinwand-Instanzen erzeugt wurden (siehe [Abbildung 12]). Es zeigt sich der exponentielle Charakter der orangen Kurve, des *nicht-optimierten* Verfahrens. Dagegen steht die lineare Entwicklung bei dem *optimierten* Verfahren, dargestellt durch die blaue Kurve.

Noch performanter wäre aber eine Implementierung, die alle Vertices in einem Durchgang erzeugt und Referenzen in einer Datenstruktur, wie einem Array, zwischenspeichert. Bei der Facettenerzeugung könnte dann direkt über einen Index auf die Vertices zugegriffen werden und damit die Suche komplett entfallen.

7.3 Vergleich und Analyse der Skinning-Verfahren

Während der Implementierung wurden zwei Skinning-Verfahren getestet. Hier erfolgt nun eine Evaluation der erzeugten Resultate. Die gewichtete Variante hat sich trotz Schwächen insgesamt als vorteilhaft gezeigt und wird aus diesem Grund von dem System eingesetzt.

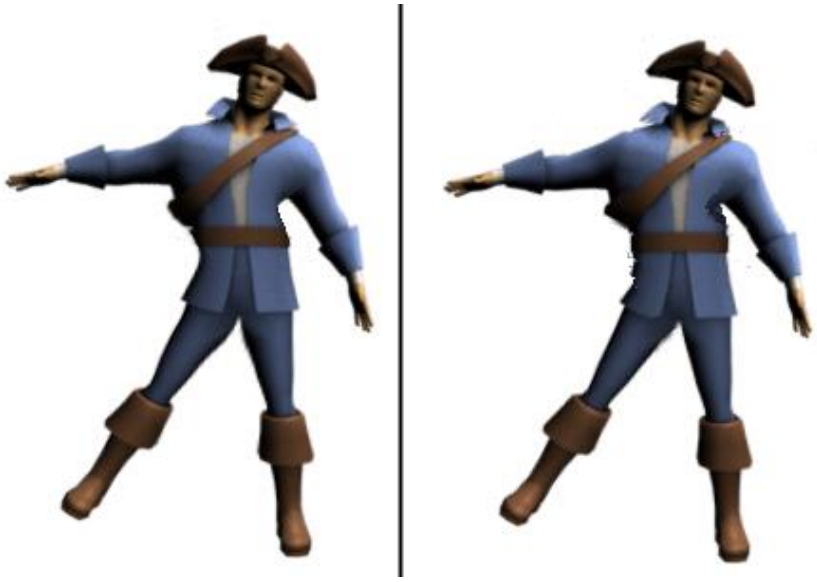

Abbildung 13: Vergleich binäres und gewichtetes Skinning.
Links: Gewichtete Zuordnung. Rechts: Binäre Zuordnung.
(Quelle: Die in der Szene enthaltene Charaktergrafik ist ein Rendering des folgenden Models: http://opengameart.org/content/pirate
Lizenz: CC-BY 3.0. Erstellt von Clint Bellanger - Abgerufen am 08.12.2016)

Der Performancevorteil des binären Verfahrens ist zu vernachlässigen, da auch das gewichtete Verfahren ausreichend schnell arbeitet. Bewertungskriterien sind aus diesem Grund die optischen Eigenschaften der erzeugten Grafiken.

Ein direkter Vergleich der beiden Verfahren ist in [Abbildung 13] zu sehen. Links das Ergebnis des gewichteten Verfahrens und rechts eine Deformation als Ergebnis des binären Verfahrens. Bei der binären Zuordnung kommt es verstärkt zur Bildung von grafischen Artefakten. Die Textur scheint gerade in der Nähe von Übergängen zwischen zwei Knochen zu zerreißen. Selbst kleine Drehungen wie beim rechten Bein des Charakters erzeugen diese auffälligen Übergänge (siehe [Abbildung 13]).

Auch wenn derartige Fehler erwartet wurden, so überrascht ihre starke Auffälligkeit. Der Grund dafür kann an dieser Stelle nur vermutet werden. Eventuell hängt es damit zusammen, dass die Kontur der Oberfläche an den Stellen der Fehler abrupt ihren Lauf ändert (mit einem Winkel von nahezu 90°). Dies wäre aber im Rahmen einer aufbauenden Arbeit zu klären und könnte Ansätze für messbare Qualitätskriterien oder neue Verfahren liefern.

Insgesamt erzeugt das dagegenstehende, gewichtete Verfahren bei allen Tests optisch ansprechendere Ergebnisse. Es erzeugt weichere, fast fließende Übergänge. Eine irgendwie geartete Gewichtung vorzunehmen erscheint als die bessere Herangehensweise.

Mit einer Gaußschen Normalverteilung scheint auch eine gute Basis-Gewichtungsfunktion gegeben, aufgrund der einstellbaren Parameter konnte nach einigen Versuchen ein gutes Gleichgewicht gefunden werden, um den Einflussbereich der Knochen hauptsächlich auf die ihr anliegenden Konturen zu beschränken und trotzdem an den Gelenken fließende Übergänge zu erzeugen.

Abbildung 14: Auffälligkeiten bei gewichteter Zuordnung
(Quelle: Die in der Szene enthaltene Charaktergrafik ist ein Rendering des folgenden Models: http://opengameart.org/content/pirate
Lizenz: CC-BY 3.0. Erstellt von Clint Bellanger - Abgerufen am 08.12.2016)

Dennoch gibt es Charakterposen, die auch in dem System mit der gewichteten Variante nicht gut umgesetzt werden können. Ein Problem tritt an den Gelenken auf, die Übergänge sind zwar weicher als bei der binären Variante, doch bei starken Drehungen wirken die Übergänge nicht mehr natürlich. Der nach unten gebogene Arm in der [Abbildung 14]

scheint sich wie Gummi zu verbiegen. Vom Betrachter erwartet, wird hier wohl eher das Einknicken und Reduzieren der Oberfläche an den Innenseiten der Gelenke.

Ein weiteres Problem ist beim anderen Arm in der [Abbildung 14] zu erkennen. Es entstehen Bereiche, in denen nicht zugehörige Knochen einen Einfluss auf die Grafik ausüben. Der erhobene Arm hat durch die Drehung ebenfalls einen Teil des Oberkörpers mit verzerrt. Hauptursache ist hier, dass der Knochen in der Mitte des Rumpfes eine ähnliche Entfernung zur Kontur des Oberkörpers hat wie der Oberarmknochen. Beide Knochen erhalten dadurch einen ähnlichen Einfluss auf den Bereich.

Mit dem derzeitigen Gewichtungsverfahren erhalten alle Knochen bei gleichen Entfernungen den gleichen Einfluss. Doch je nach Struktur des Skeletts und der Abmessung der Charaktergrafik gibt es Bereiche, die unterschiedlich weit nach außen beeinflusst werden sollten. Ein Knochen in der Mitte des Oberkörpers müsste einen weiteren Bereich der Grafik beeinflussen, als ein Knochen in der Mitte einer Hand. Die Alternative, das Skelett anzupassen, indem einfach mehrere Knochen innerhalb des Oberkörpers platziert werden, erscheint weniger sinnvoll, da die Knochen nicht nur zum Erzeugen der Gewichte gedacht sind, sondern als Abstraktion eine einfachere Kontrolle der umliegenden Grafik ermöglichen sollen. Zusätzliche Knochen würden die Kontrolle eher komplizieren, da für eine erwartete Bewegung nun mehrere Knochen gedreht werden müssten.

7.4 Tiefeninformation – Überdeckung einzelner Bereiche

Durch den Aufbau des Systems, hat der Anwender derzeit keine Kontrolle über die Tiefeninformationen der Charaktergrafik. Es ist als Anwender also nicht möglich zu entscheiden, welcher Teil der Grafik vor einem anderen Teil gezeichnet werden soll.

Abbildung 15: Tiefeninformation der Leinwand. Überlagerung von Grafikbereichen.
(Quelle: Die in der Szene enthaltene Charaktergrafik ist ein Rendering des folgenden Models: http://opengameart.org/content/pirate
Lizenz: CC-BY 3.0. Erstellt von Clint Bellanger - Abgerufen am 08.12.2016)

Innerhalb der Szene befinden sich alle Vertices der Leinwand auf der gleichen Tiefenebene. Überlagern sich Bereiche, wie in der [Abbildung 15] die beiden Arme und der Oberkörper, dann wird die Textur in der Reihenfolge der Vertices innerhalb des Dreiecksnetzes gezeichnet. Das bedeutet, dass die Textur der Vertices am Ende der Liste über bereits gezeichnete Bereiche gemalt wird. Wie an der [Abbildung 15] zu erkennen, führt das in dem System dazu, dass der rechte Arm hinter dem Körper verschwindet, während der linke Arm vor diesem erscheint. Das Ergebnis ist erstmal nicht als falsch oder unansehnlich zu bewerten, trotzdem deutet es auf einen Bereich hin, auf den ein Anwender eventuell Einfluss ausüben möchte.

8 Fazit und Ausblick

8.1 Fazit

Ziel dieser Arbeit war es ein System zu entwickeln, mit dem es einfach möglich ist, aus bestehenden Charaktergrafiken neue Grafiken zu erzeugen und dafür ein skelettbasiertes Verfahren zu verwenden. Mit dem entwickelten Prototypen wurde dieses Ziel erreicht. Auch konnte gezeigt werden, dass ein automatisches, gewichtetes Skinning-Verfahren für den Anwender eine zeitsparende Möglichkeit bietet, diesen Prozess zu unterstützen. Dennoch hat sich offenbart, dass die Qualität der damit erzeugten Grafiken nicht immer befriedigend ist. Dabei ist aufgefallen, dass es schwer ist, messbare Qualitätskriterien im Kontext der Grafikerzeugung aufzustellen, nicht zuletzt, weil ein optisch ansprechendes Ergebnis immer nur eine subjektive Einschätzung des Betrachters sein kann.

Trotz der beschriebenen Vorteile bei der Verwendung des Frameworks, wäre die Entwicklung in manchen Bereichen schneller vorangeschritten, wenn das System in einer auf zwei Dimensionen begrenzten Umgebung entwickelt worden wäre. Zusätzliche Möglichkeiten und Freiheiten, erzeugen manchmal auch zusätzliche Komplexität.

Insgesamt bin ich mit den Ergebnissen der Arbeit zufrieden, auch wenn sich im Verlauf immer mehr gezeigt hat, dass es noch viele Bereiche gibt, die optimiert oder erweitert werden können. Doch gerade vor dem Hintergrund erst einen möglichen Grundstein gelegt zu haben, ist es erfreulich zu sehen, wieviel schon mit diesem Ansatz erreicht werden konnte.

8.2 Ausblick

In dem Bereich der unterstützenden zweidimensionalen Grafikgenerierung ergeben sich einige verschiedene Anknüpfungspunkte. Einen großen Einfluss auf die Qualität der erzeugten Grafik hat der Skinning-Prozess, die Verbindung zwischen den Knochen und dem Bild. Dieser Prozess könnte auf unterschiedlichste Arten weiter ausgebaut werden. Denkbar

wäre ein System, bei dem der Anwender einzelnen Knochen unterschiedliche Stärken zuweist oder bereits zugeordnete Gewichte nachträglich bearbeiten kann. Interessanter hingegen erscheint die Optimierung des automatischen Verfahrens, wie z.B. eine Sonderfallbehandlung für Übergänge an Gelenken. Möglich wäre auch die Integration einer Konturerkennung oder Konturabstandsmessung, um einzelne Knochen automatisch auf die umliegenden Grafikbereiche anzupassen. Vielleicht sogar mit einem Fehlererkennungssystem, das ungewöhnliche Konturveränderungen (wie in 7.3 beschrieben) bei Bewegungen erkennt und behandelt.

Sobald sich Bildabschnitte eines Bildes überlagern, ist eine Tiefenordnung der einzelnen Bildbereiche gerade im zweidimensionalen Bereich von besonderer Bedeutung (siehe 7.4). Im Gegensatz zu animierten 3D-Objekten, fehlt hier die Tiefeninformation im zugrundeliegenden Modell.

Neben der Optimierung der erzeugten Grafiken, ergeben sich auch rund um das Skelett weitere Bereiche, die vertieft werden können. Eine Option wäre, dem Anwender mehr Einfluss zu gewähren und ihm zu ermöglichen, eigene Skelette zu erstellen oder zu bearbeiten. Mit Verfahren zur automatischen Skelettgenerierung ergeben sich weitere mögliche Themenfelder. Schließlich kann auch der Bereich der Skelettbewegung vertieft werden, z.B. mit einem Verfahren wie der inversen Kinematik.

9 Literaturverzeichnis

[1] B. Levy, P. Alliez, M. Pauly, L. Kobbelt und M. Botsch, Polygon Mesh Processing, Taylor & Francis, 2009.

[2] G. Teschl und S. Teschl, Mathematik für Informatiker - Band 1 Diskrete Mathematik und Lineare Algebra (3. Auflage), Springer-Verlag Berlin Heidelberg, 2008.

[3] N. Magnenat-Thalmann, R. Laperrière und D. Thalmann, „Joint-Dependent Local Deformations for Hand Animation and Object Grasping," 1988.

[4] P. J. Dobrovka, D. Mühlbacher und J. Brauer, Computerspiele - Design und Programmierung, mitp, 2003.

[5] „https://www.opengl.org/wiki/Texture," 14 11 2016. [Online]. Available: https://www.opengl.org/wiki/Texture. [Zugriff am 08 12 2016].

[6] B. H. Le und Z. Deng, „Robust and Accurate Skeletal Rigging from Mesh Sequenzes," ACM Trans Graph. 33, 2014.

[7] A. Ward, Game Character Development with Maya, New Riders, 2005.

[8] T. W. Sederberg und S. R. Parry, „Free-Form Deformation of Solid Geometric Models," Bd. 20, Nr. 4, 1986.

[9] S. Schaefer, T. McPhail und J. Warren, „Image Deformation Using Moving Least Squares," ACM, 2006.

[10] H.-B. Yan, S.-M. Hu, R. R. Martin and Y.-L. Yang, "Shape Deformation Using a Skeleton to Drive Simplex Transformations," vol. 14, no. 3, 2008.

[11] J. Lander, „Skin Them Bones: Game Programming for the Web Generation," *Game Developer Magazine,* Bd. 1, Nr. 5, pp. 10-18, 1998.

[12] J. Lander, „Over my dead, polygonal body," *Game Developer Magazine,* Bd. 17, Nr. 1, pp. 1-4, 1999.

[13] L. Kavan und J. Žára, „Real time skin deformation with bones blending," 2003.

[14] F. Scheepers, R. E. Parent, W. E. Carlson und S. F. May, „Anatomy-Based Modeling of the Human Musculature," in *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, 1997.

[15] B. Bodenheimer, C. Rose, S. Rosenthal und J. Pella, „The Process of Motion Capture: Dealing with the Data," *Computer Animation and Simulation'97,* pp. 3-18, 1997.

[16] I. Sommerville, Software Engineering 9., aktualisierte Auflage, Pearson, 2012.

[17] K. Pulli, T. Aarnio, V. Miettinen, K. Roimela und J. Vaarala, Mobile 3D Graphics: with OpenGL ES and M3G, Morgan Kaufmann, 2007.

[18] C. Hock-Chuan, „Computer Graphics Basic Theory," Nanyang Technological University, Singapore, July 2012. [Online]. Available: https://www.ntu.edu.sg/home/ehchua/programming/opengl/CG_BasicsTheory.html. [Zugriff am 12 Juli 2016].

[19] T. Möller und B. Trumbore, „Fast, Minimum Storage Ray/Triangle Intersection," 1997.

[20] „RayTriangleIntersection Java implementation," Sun Microsystems, Inc., 24 März 2007. [Online]. Available: https://github.com/sgothel/jogl-utils/blob/master/src/net/java/joglutils/msg/impl/RayTriangleIntersection.java. [Zugriff am 14 Dezember 2016].

A. Software Bedienungsanleitung

In diesem Kapitel wird die Bedienung des entwickelten Systems erklärt und auf einige Besonderheiten hingewiesen, die dabei zu beachten sind.

A.1. GUI Gesamtübersicht

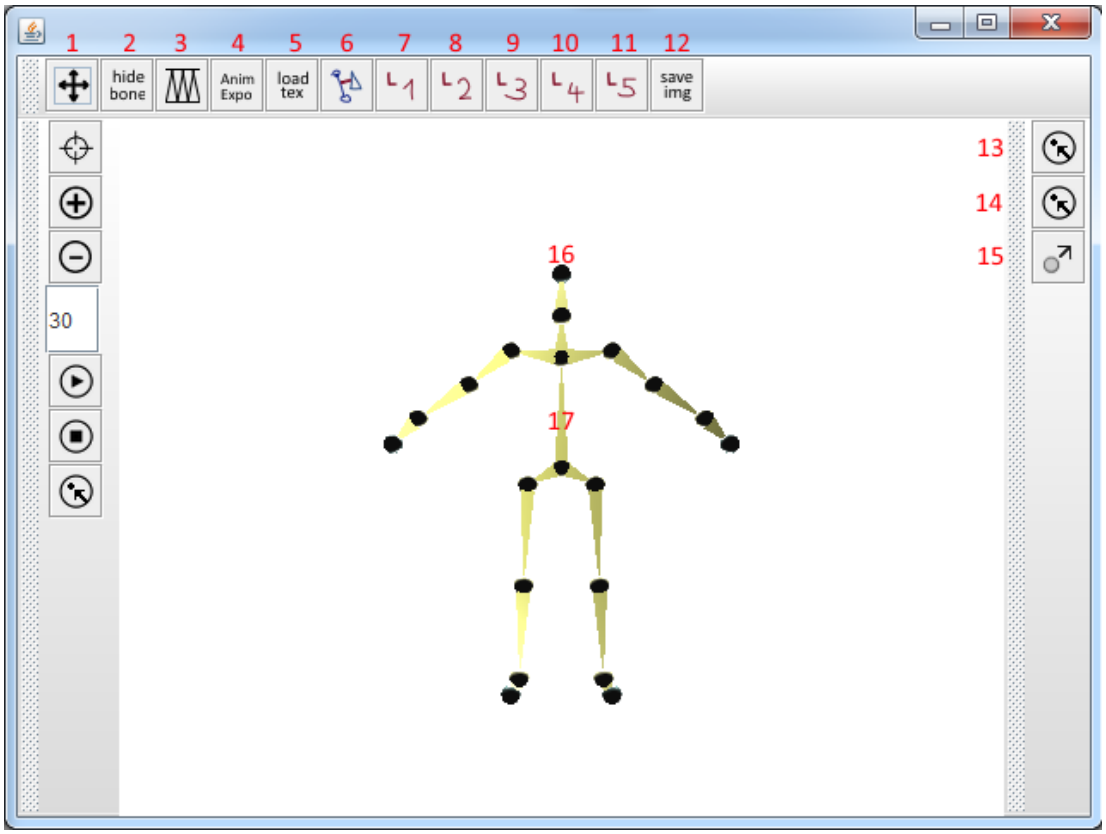

Abbildung 16: GUI Gesamtübersicht.

Die [Abbildung 16] zeigt die grafische Oberfläche des entwickelten Systems. Bei den folgenden Erklärungen der einzelnen Funktionen werden diese über die roten Nummern in dem Bild referenziert.

Obere Toolbar:

1. Kamera Mauskontrolle (Umschalter)
   Aktiviert die Mauskontrolle mit der die Kamera innerhalb der Szene bewegt wird.
2. Anzeige Knochen (Umschalter)
   Versteckt die Anzeige der Knochen innerhalb der Szene und damit in exportierten Bildern.
3. Anzeige Gitternetz (Umschalter)
   Aktiviert die Anzeige vom Dreiecksnetz der Leinwand
4. Animationsexport (Umschalter)
   Aktiviert den Animationsexport. Ist dieser Modus aktiviert, wird beim Laden einer Skelettposition die Drehung in einzelne Frames unterteilt und zum Export der einzelnen Bilder aufgefordert.
5. Textur importieren
   Erfragt über einen Dialog eine Grafikdatei. Diese wird anschließend importiert und auf die Leinwand gezeichnet.
6. Skinning-Automatik
   Aktiviert den automatischen Skinning-Prozess. Dieser verbindet die Knochen mit der umliegenden Grafik.
7. Skelettposition laden
   Dreht das Skelett in die vorgespeicherte Skelettposition und deformiert dabei die verbundene Grafik. Die Buttons 7 bis 11 laden die fünf unterschiedlichen, vorgespeicherten Skelettpositionen.
12. Grafik exportieren
    Exportiert das aktuell gerenderte Bild nach der Auswahl eines Pfades und Dateinamens.

Rechte Toolbar:

13. Knochen rotieren – Gegen den Uhrzeigersinn (Umschalter)
    Nach Aktivierung können Knochen und daran angehängte Bereiche der Leinwand gegen den Uhrzeigersinn gedreht werden. Jeder Mausklick auf einen Knochen, rotiert diesen dann um einen geringen Betrag.
14. Knochen rotieren – Im Uhrzeigersinn (Umschalter)
    Nach Aktivierung können Knochen und daran angehängte Bereiche der Leinwand im Uhrzeigersinn gedreht werden. Jeder Mausklick auf einen Knochen (17), rotiert diesen dann um einen geringen Betrag.
    Achtung: Sind die beiden Rotationsbuttons (13 und 14) aktiviert, heben sich die Rotationen gegenseitig auf.

15. Verschieben der Knochenpositionen (Umschalter)
    Nach Aktivierung können mit der Maus die Start- und Endpositionen der Knochen verschoben werden. Dabei wird der damit verknüpfte Bildbereich der Leinwand *nicht* mit verschoben.

Szene:

16. Kontrollpunkt der Knochenposition.
    Diese Punkte innerhalb der Szene können verschoben werden, sobald der Umschalter (15) für diese Funktion aktiviert wurde.
17. Knochen
    Repräsentation des Knochens.

A.2. Ausführung und Verwendung

Starten des Systems:
Zum Starten der Anwendung wird die main-Methode innerhalb der Editor Klasse im *Package cgresearch.studentprojects.posegen.editor* ausgeführt.

Alle Referenzen beziehen sich auf die Nummerierung innerhalb der Gesamtübersicht (siehe [Abbildung 16])

Grafik importieren und vorbereiten:
Zuerst wird eine Grafik importiert (Button 5). Danach müssen die Knochenpositionen an die Grafik angepasst werden. Dafür wird der Knochenpositions-Editiermodus (Button 15) aktiviert. Anschließend werden die Kontrollpunkte (16) so über die Grafik gezogen, dass die Knochen mittig über den korrespondierenden Bereichen der geladenen Grafik liegen. Sind alle Knochen in die gewünschte Position verschoben, dann wird das Skelett mit der Grafik verbunden, indem das automatische Skinning-Verfahren aktiviert wird (Button 6).

Skelettposition verändern:
Wurde das Skelett und die Grafik entsprechend der vorherigen Beschreibung vorbereitet, können nun Deformationen an der Grafik vorgenommen werden.
Dies geschieht entweder durch das Laden einer vorgespeicherten Skelettposition (Button 7 bis 11) oder durch das manuelle drehen von Knochen. Für die manuelle Anpassung wird entweder Button 13 **oder** Button 14 aktiviert. Danach wird auf den zu drehenden Knochen geklickt – jeder Klick dreht den anvisierten Knochen um einen kleinen Betrag.

Exportieren von Grafiken:
Zum Export einer Grafik wird der Button 12 angeklickt. Daraufhin erscheint ein Dialog indem ein Pfad und Dateiname bestimmt wird. Im Anschluss wird die Grafik an der gewünschten Position erzeugt.

Exportieren von Animationen:
Zum Exportieren einer Animationssequenz wird zuerst der Exportmodus aktiviert (Button 4). Im Anschluss muss eine vorgespeicherte Skelettposition geladen werden (Button 7 bis 11). Das System erfragt nun für jeden Frame einen Pfad und Dateinamen.

Versicherung über Selbstständigkeit

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

Hamburg, den _____ _____