Universität des Saarlandes
Naturwissenschaftlich-Technische Fakultät I
Fachrichtung Informatik
Master-Studiengang Informatik

Masterarbeit

Werkzeuge zur IK-basierten Gestenannotation mit Hilfe eines 3D-Skeletts

vorgelegt von

Nguyen Chi Hoang Quan

am 3. August 2009

angefertigt unter der Leitung von
Prof. Dr. Dr. h.c. mult. Wolfgang Wahlster

betreut von
Dr. Michael Kipp

begutachtet von

Prof. Dr. Dr. h.c. mult. Wolfgang Wahlster
Dr. Michael Kipp

Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig verfasst und alle verwendeten Quellen angegeben habe.

Saarbrücken, den 3. August 2009

Einverständniserklärung

Hiermit erkläre ich mich damit einverstanden, dass meine Arbeit in den Bestand der Bibliothek der Fachrichtung Informatik aufgenommen wird.

Saarbrücken, den 3. August 2009

Zusammenfassung

Das Ziel dieser Arbeit ist die Umsetzung einer Erweiterung für das Annotationstool ANVIL zur intuitiven und präzisen Gestenannotation. Bisherige Annotationswerkzeuge bieten nur grobe Approximationen der räumlichen Beschreibung von Gesten. Präzisere Beschreibungen sind mit einem hohem Aufwand verbunden. Daher soll diese Arbeit trotz einer höheren Präzision bei der Gestenannotation den Aufwand in vertretbarem Rahmen halten. Dies wird über ein 3D-Skelett realisiert. Es kann mit Hilfe der intuitiven inversen Kinematik aber auch über Forward Kinematics manipuliert werden. Die Annotation einiger Schlüsselposen ist ausreichend, da die Zwischenposen mit Hilfe des Interpolationsalgorithmus SLERP automatisch berechnet werden. So kann das Skelett zur Kontrolle der Annotation animiert werden. Die erstellte Klassenhierarchie erlaubt den Austausch und die Erweiterung von Interpolations- und Manipulationsalgorithmen. Mit Hilfe einer in ANVIL integrierten, intuitiven grafischen Benutzerschnittstelle sollen Annotationszeit und -aufwand reduziert werden. Mit der Evaluation konnte die Intuitivität dieser Applikation sowohl durch die subjektive Einschätzung der Probanden als auch durch die objektive Messung ihrer Annotationsleistung bestätigt werden. Der direkte Vergleich mit dem Annotationsaufwand eines Annotationsschemas zeigt, dass sich die Annotationszeit bei skelettbasierter Annotation nicht deutlich von der des Annotationsschemas unterscheidet. Dies belegt, dass das Ziel eines Interfaces zur intuitiven und präzisen Posenannotation mit vertretbarem Aufwand realisiert werden konnte.

Danksagung

An dieser Stelle möchte ich mich bei allen bedanken, die mich bei der Erstellung dieser Arbeit unterstützt haben oder diese erst ermöglicht haben. Zunächst möchte ich mich bei Herrn Professor Wahlster herzlich für die Ausgabe und Begutachtung dieser Arbeit bedanken, ohne die diese Arbeit nicht entstanden wäre.

Größter Dank gebührt an dieser Stelle zwei Menschen, die einen besonders großen Anteil am Gelingen dieser Arbeit hatten: Michael Kipp, meinem Betreuer, und meiner Freundin Johanna. Michael Kipp möchte ich ganz herzlich für die Aufnahme in sein EMBOTS-Team danken und dafür, dass ich dort während meiner Zeit als Hiwi und Masterstudent eine hervorragende Betreuung erfahren durfte. Zutiefst dankbar bin ich auch für seine immer motivierende und konstruktive Art der Kritik sowie für die wertvolle Unterstützung bei der Erstellung meines GALA-Beitrages.

Johanna danke ich von ganzem Herzen für die Unterstützung in der ganzen Zeit. Besonderes möchte ich ihr dafür danken, dass sie diese Arbeit gewissenhaft und (unzählige Male) korrekturgelesen hat (so dass sie wahrscheinlich selbst schon zur Expertin für skelettbasierte Annotation geworden ist). Auch ihre wertvollen Tipps und Vorschläge waren für die Evaluation meiner Arbeit eine große Hilfe.

Außerdem möchte ich mich bei Christian „Ross“, Sandro „Dori“, Lloyd, Johanna, David, Benni, Pascal, Ivan und Caro bedanken, die sich (auch sehr kurzfristig) als Versuchspersonen zur Verfügung gestellt haben, um diese Arbeit zu evaluieren. Alexis Heloir danke ich für seine hilfreichen Tipps zur Erstellung von Skelett-Topologien und zur Arbeit mit 3D-Programmen. Lloyd und Dori danke ich für die hilfreichen und fruchtbaren Diskussionen, wenn ich mal wieder „den Wald vor lauter Bäumen“ nicht sah.

Zum Schluss möchte ich mich bei meinen Geschwistern und meinen Eltern für die jahrelange Unterstützung während des Studiums bedanken. Ich danke auch Barbara und Alfred ganz herzlich für ihre Unterstützung in den letzten Jahren.

Und allen, die ich hier vergessen haben sollte!

Inhaltsverzeichnis

Abbildungsverzeichnis

1. Einleitung

Nonverbale Kommunikation spielt heute eine entscheidende Rolle. In seinem Buch *Die Macht des Bildes* schildert Siegfried Frey ihren Einfluss auf Kultur und Politik und die Entwicklung in diesem Forschungsbereich [Frey 1999]. Der Begriff der *nonverbalen Kommunikation* wurde erstmals 1956 von Ruesch und Kees verwendet und beschreibt die Kommunikation auf nicht sprachlicher Ebene, wie z.B. Gestik, Mimik oder Körperhaltung [Ruesch und Kees 1956].

Heute ist die Erforschung nonverbalen Verhaltens bei zwischenmenschlicher Kommunikation Thema vieler Forschungsgruppen u.a. bei Psychologen, Soziologen und Linguisten [Harrigan u. a. 2005]. Zu diesem Zweck müssen häufig Daten gesammelt, ausgewertet und untersucht werden[1]. Die erfassten Daten können beispielsweise Formen oder Aspekte der Proxemik[2], Kinesik[3] oder des Blickverhaltens bei der Kommunikation sein. So gibt die Erforschung des proxemischen Verhalten Aufschluss über das menschliche Raumverhalten. Denn je nach nach Situation und Grad der Vertrautheit zu dem Gesprächspartner lässt der Mensch unterschiedliche räumliche Nähe zu. Zum Beispiel wird einer vertrauten Person während eines Gesprächs, ein engerer Abstand gestattet, als einer fremden Person. Durch die Untersuchung des Blickverhaltens während eines Gespräches, versuchen Forscher herauszufinden, wie und welche Informationen über diesen Kanal übermittelt werden. So kann der erste Blickkontakt entscheidend darüber sein, ob ein Gespräch zustande kommt oder nicht [Cary 1978]. In der Kinesik wird unter anderem die Gestik und deren Einsatz bei zwischenmenschlicher Kommunikation und deren Zusammenspiel mit der Sprache untersucht. Solche Erkenntnisse können unter anderem Forscher im Bereich der künstlichen Intelligenz helfen, virtuelle Agenten, die bei Mensch-Maschine-Interaktionen eingesetzt werden, menschlicher zu gestalten [Rist u. a. 2003]. Dazu müssen diese Agenten auch in der Lage sein, Mimik und Gesten zu reproduzieren. Wie solche Gesten nach menschlichen Vorbildern für virtuelle Charaktere generiert werden können, zeigen Kipp u. a. [2007b]. Eine Methode, um die benötigten Daten zu erhalten, ist die manuelle Annotation und Analyse von Video- und Audiodaten. Zur Unterstützung dieser aufwändigen, manuellen Arbeit wurden daher zahlreiche computergestützte Annotationswerkzeuge entwickelt, um diesen Prozess zu beschleunigen und den Aufwand zu reduzieren [Bigbee u. a. 2001; Rohlfing u. a. 2006].

[1] In seinem Buch *The new Handbook of Methods in nonverbal Behavior Research* stellen Harrigan et al. grundlegende Methoden zur Erfassung von nonverbalem Verhalten vor [Harrigan u. a. 2005].

[2] „[I]n der Proxemik werden unterschiedliche Zonen des persönlichen Raumes abgegrenzt und die Bedeutung der räumlichen Distanz zwischen den Kommunikationspartnern für die Kommunikation untersucht." Definition aus [Bro 2009]; Stichwort: nonverbale Kommunikation.

[3] „...Die Kinesik behandelt kommunikationsrelevante Bewegungen des Gesichtes, der Augen und der Glieder,..." Definition aus [Bro 2009]; Stichwort: nonverbale Kommunikation.

Auch das Annotationstool ANVIL (siehe Abschnitt 2.1) wird zur Annotation von Gesten verwendet [Kipp 2001]. Neff et al. verwenden ANVIL beispielsweise um Gesten mit einem speziellen Annotationsschema zu annotieren [Neff u. a. 2008]. Dieses Schema sieht vor, dass jede Geste in die folgenden vier Phasen unterteilt werden kann: *preparation*, *hold*, *stroke* und *hold*. Preparation beschreibt die Phase, in der die Hände zur stroke-Position bewegt werden. Hold sind Pausen, die optional vor oder nach einem stroke auftauchen können. Stroke ist die wichtigste Phase, da sie die bedeutungstragende Phase einer Geste ist. Bis auf stroke sind dabei alle Phasen optional. Gesten können dann wiederum zu größeren Einheiten, den *Units*, zusammengefasst werden. Diese Unterteilungen können mit Hilfe von ANVIL geschehen. Dazu müssen Start- und Endzeit einer Phase definiert und zugeordnet werden. Dadurch erhält man eine Zerlegung der Geste mit Zeitinformationen, wie Dauer sowie Start- und Endzeitpunkt, allerdings ohne räumliche Information. Eine Möglichkeit diese räumlichen Informationen zu erfassen, ist eine Erweiterung dieses Annotationsschemas durch räumliche Beschreibungen. Mit diesem erweiterten Schema könnte dann die räumliche Orientierung der an einer Geste beteiligten Körperteile beschrieben werden. Beispielsweise kann die Position der Hände relativ zum Körper angegeben werden (weit vom Körper, nach am Körper usw.). Diese Art der Beschreibung kann jedoch nur eine grobe Approximation der tatsächlichen Lage darstellen. Aus diesem Grund soll diese Arbeit ANVIL eine präzisere Gestenannotation ermöglichen, die darüber hinaus auch intuitiv zu bedienen ist. Diese Arbeit bietet als Lösung eine skelettbasierte Gestenannotation an. Die zentrale Idee ist also, dass in ANVIL ein 3D-Skelett bereitgestellt werden soll, mit dessen Hilfe Posen direkt nachgebildet werden können. Über dieses Skelett können die Annotationen dann um die fehlenden räumlichen Informationen ergänzt werden. Durch Interpolation und Animation sollen dann aus den einzelnen Key-Posen die zu annotierenden Gesten entstehen. Auf diese Weise, kann zu jedem Zeitpunkt einer Geste, die exakte Position der Gelenke ausgelesen werden.

1.1. Motivation

Virtuelle Agenten werden heutzutage in vielen Bereichen wie Computerspielen, Kinofilmen und zur Interaktion zwischen Mensch und Maschine eingesetzt. Besonders in multimodalen Dialogsystemen hat man die Notwendigkeit von virtuellen Charakteren als weiteren Kommunikationskanal erkannt [Cassell 1989; Wahlster 2006]. Denn virtuelle Charaktere bieten Computersystemen die Möglichkeit, auch auf nonverbaler Ebene mit dem Benutzer zu kommunizieren. Dies geschieht, indem sie menschliches Verhalten, wie Gesten, Mimik und bestimmte Körperhaltungen nachahmen. Denn über diese (nonverbalen) Kanäle können zusätzliche Informationen übertragen werden. Beispielsweise können Gesten redundant oder komplementär zu dem Gesprochenen sein oder auch als Mittel benutzt werden, um den Gesprächsfluss zu steuern [Bickmore und Cassell 2005]. Bevor Agenten dieses Verhalten reproduzieren können, müssen zunächst Daten über menschliche Bewegungen gesammelt und ana-

lysiert werden. Grundlage dieser Analyse können Videoaufzeichnungen von realen Menschen, manuell animierten Modellen oder Motion-Capture-Daten sein.

Daten die durch Motion-Capture gewonnen werden, sind sehr präzise. Jedoch benötigt dieses Verfahren eine spezielle Ausrüstung und Arbeitsumgebung. Aufgrund ihres Signalcharakters muss man diese Daten einer aufwändigen manuellen Nachbearbeitung unterziehen. Bei der traditionellen Animation kann man auf zwei Arten animieren. Entweder mit der *Straight-Ahead Action* oder der *Pose-to-Pose Action (Key-Frame-Animation)* [Lasseter 1987]. Bei der Straight-Ahead-Methode wird eine Sequenz von Bildern, beginnend beim ersten, Bild für Bild vom Zeichner erstellt. Bei dieser Methode behält der Zeichner zu jeder Zeit die Kontrolle über den Animationsverlauf. Bei der Pose-to-Pose-Methode hingegen wird die Szene im Voraus geplant und mit den so genannten Key-Posen (Key-Frames) skizziert. Die „In-betweens“ oder Zwischenbilder werden dann später, eventuell auch von einem anderen Zeichner, hinzugefügt. Key-Frame-Animation ist heute die gängigste Art der computergestützten Animation, bei der der Computer die Berechnung der Zwischenbilder übernimmt. Das Ergebnis ist jedoch stark vom Interpolationsalgorithmus abhängig. Um das beste Ergebnis bei geringem Zeitaufwand zu erhalten, wird deshalb oft eine Kombination aus Motion-Capture und der traditionellen Animation verwendet. Dabei liefert die Motion-Capture die Rohdaten, die dann auf ein 3D-Modell übertragen und anschließend manuell korrigiert werden.

Beide Verfahren dienen der Animation und enthalten keine Information darüber, welche semantische Bedeutung die ausgeführte Bewegung hat oder wie sie aufgebaut ist. Diese Informationen sind jedoch wichtig, um Bewegungen zu untersuchen oder zu analysieren. Diese Informationen sind auch notwendig, um Bibliotheken von Gesten zu erstellen, mit denen man virtuellen Agenten verschiedene Ausdrucksstile menschlicher Sprecher verleiht [Kipp u. a. 2007b].

Eine Möglichkeit Bewegungsdaten mit semantischer Bedeutung zu erhalten, ist die Annotation mit Hilfe von Annotationswerkzeugen und Annotationsschemata. Jedoch bietet die Gestenannotation mit solchen Werkzeugen und Schemata nur eine grobe Approximation der Originalbewegung, d.h. keine exakten Informationen über die Position oder Orientierung der Gelenke oder Hände. So klassifiziert das Annotationsschema von Kipp u. a. [2007a] nach Bewegung und Position. Die Gesten werden dabei in die Bewegungsphasen untergliedert (Vorbereitungsphase, Haltephase usw.) und die Positionen approximiert (weit vom Oberkörper entfernt, normal entfernt usw.). Weitere Kodierungsschemata sind das Berner System [Siegfried u. a. 1981], das ebenfalls zur Beschreibung menschlicher Bewegungen entwickelt wurde und *FACS* (Facial Action Coding System) [Ekman und Friesen 1978] bzw. *EmFACS* (Emotion Facial Action Coding System) zur Kodierung von menschlicher Mimik und Emotionen. Jedoch ist der Kodieraufwand mit solchen Systemen relativ hoch [Altorfer u. a. 1997]. Die Annotation von einer Minute Video mit ANVIL und dem Annotationsschema von Neff u. a. [2008] dauert beispielsweise 90 Minuten. Ähnlich lange dauert die Annotation von Mimik mit FACS durch einen gut eingeübten Kodierer. Nach Cohn u. a. [2005] dauert die FACS-Kodierung von einer Minute Material etwa 100 Minuten, abhängig von der Ausprägung und Komplexität der Mimik.

Um präzise Bewegungsdaten mit semantischer Bedeutung zu erhalten, schlagen wir ein Werkzeug vor, dass die Vorteile beider Ansätze vereint: die Präzision von 3D-Animationstools und die semantische Beschreibung mit Hilfe von Annotationswerkzeugen und -schemata. Der Aufwand bei der Benutzung des Werkzeugs sollte jedoch in vertretbarem Rahmen bleiben.

1.2. Aufgabenstellung und Ziele

Das Ziel dieser Arbeit ist die Entwicklung einer Umgebung zur intuitiven und präzisen Annotation von Gesten mit Hilfe eines 3D-Skeletts. Dabei sollten folgende Anforderung erfüllt werden:

- **Intuitives, skelettbasiertes 3D-Interface:** Durch ein intuitives Interface soll der Benutzer schnell in der Lage sein, Gesten zu annotieren und zu modifizieren. Dazu sollen Funktionen zur Verfügung gestellt werden, die eine optimale Umgebung schaffen, um Posen von einem menschlichen Sprecher nachbilden zu können. Dazu gehört die Steuerung des Skeletts mit Hilfe von inverser Kinematik und Möglichkeiten, die Ansichten auf das Skelett und die 3D-Szene zu verändern. Beispielsweise soll zu jeder Pose das entsprechende Bild des Sprechers hinter dem Skelett positioniert werden, damit der Benutzer eine direkte Vorlage in der Szene erhält.
- **Animation mit Hilfe von Interpolation:** Zur Kontrolle und zum Vergleich der Annotationen mit der Originalgeste soll das Skelett animiert werden können. Dabei sollen nur Key-Posen erstellt werden, um den Aufwand gering zu halten. Die Zwischenposen sollen dann zur Laufzeit interpoliert werden.
- **Wiederverwendbarkeit der Daten:** Damit die erstellten Posen auch in 3D Programmen wie Maya verwendet werden können, soll eine Exportmöglichkeit bereitgestellt werden. Als Dateiformat soll COLLADA verwendet werden.
- **Einbindung in ANVIL:** Diese Arbeit soll ein fester Bestandteil von ANVIL (siehe Abschnitt 2.1) sein. Dazu müssen ANVIL und die Spezifikationsdatei angepasst werden.
- **Evaluation:** Um die Intuitivität der Applikation in Bezug auf Bedienbarkeit und Annotation von Gesten zu untersuchen, soll im Anschluss eine Evaluation durchgeführt werden.

1.3. Gliederung der Arbeit

Diese Arbeit soll die Entwicklung und Umsetzung der oben genannten Aufgabenstellung dokumentieren. Kapitel 2 beschreibt den Kontext der Arbeit: das Annotationswerkzeug ANVIL. Anschließend werden weitere Annotationswerkzeuge und relevante Tools vorgestellt. Das anschließende Kapitel 3 legt die Grundlagen zur Umsetzung

der gestellten Aufgaben. Die Implementierung dieser Arbeit wurde in zwei Kapitel aufgeteilt. Kapitel 4 zeigt, wie die Aufgabenstellung des intuitiven Interfaces gelöst wurde, während Kapitel 5 die Umsetzung der restlichen Aufgaben und die Einbindung in ANVIL erläutert. Die Ergebnisse der Evaluation und deren Diskussion finden sich in Kapitel 6. Abgeschlossen wird diese Arbeit mit Kapitel 7: der Konklusion und dem Ausblick auf folgende Arbeiten.

2. Annotation menschlichen Verhaltens

Die Erforschung nonverbalen Verhaltens ist Grundlage vieler Forschungsdisziplinen. So beschäftigen sich neben Psychologen, Anthropologen oder Soziologen auch Psychotherapeuten und Linguisten mit der Erforschung nonverbalen Verhaltens von Menschen [Harrigan u. a. 2005]. Deren Forschungsergebnisse können dann beispielsweise in der Informatik genutzt werden, um multimodale Dialogsysteme mit virtuellen Charakteren mit nonverbalem Verhalten zu erweitern [Cassell 1989]. Denn zwischenmenschliche Kommunikation basiert neben der Sprache auch auf nonverbalem Verhalten, wie Gestik, Mimik oder der Körperhaltung [Wahlster 2006].

Um den Aufwand hierbei zu reduzieren, wurden zahlreiche Annotationswerkzeuge geschrieben, die die Annotationsarbeit unterstützen sollen [Bigbee u. a. 2001; Rohlfing u. a. 2006]. Neben der Synchronisierung und Steuerung der Quelldateien (Video- oder Audiodateien) geben sie die Möglichkeit die Annotationen direkt im Programm zu tätigen. Diese Annotationen können dann anschließend in verschiedene Dateiformate exportiert werden, um sie entweder zu analysieren oder mit entsprechender Software statistisch auszuwerten (z.B. SPSS[1]). Einige dieser Annotationswerkzeuge sollen im Folgenden vorgestellt werden. Im Anschluss an diese Werkzeuge finden sich weitere Applikationen. Diese Applikationen beschäftigen sich beispielsweise mit der Organisation von Annotationen und sind nicht direkt mit den Anntotationstools ANVIL oder ELAN vergleichbar, aber in diesem Zusammenhang dennoch erwähnenswert.

Da ANVIL Grundlage dieser Arbeit ist, soll der Schwerpunkt dieses Abschnittes auch auf diesem Tool liegen. Der betreffende Abschnitt soll jedoch keine erschöpfende Anleitung zum Umgang mit ANVIL sein. Eine genaue Anleitung bietet Kipp [2003] oder das integrierte Online-Manual.

2.1. Das Annotationswerkzeug ANVIL

ANVIL[2] (Annotation of Video and Language) ist ein für Forschungszwecke kostenloses Annotationstool zur Annotation von multimodalen Dialogen in Video- oder Audiodateien [Kipp 2001]. Es wurde in Java implementiert und läuft sowohl unter Windows und Linux als auch unter Mac. ANVIL wird stetig weiterentwickelt und liegt mittlerweile in Version 5.0 vor.

[1] http://www.spss.com

[2] Informationen zum Download von ANVIL findet man unter http://www.anvil-software.de

2.1.1. Grundkonzept

Wie die meisten Annotationstools bietet auch ANVIL dem Benutzer die Möglichkeit auf verschiedenen Spuren (*Tracks*) Annotationen (*Elemente*), wie zum Beispiel Wörter, Gesten oder Dialog-Akte, hinzuzufügen. Dabei unterscheidet ANVIL zwischen *Primary*- und *Secondary*-Tracks. Primary-Tracks beinhalten einzelne Elemente, die jeweils einen Anfangs- und einen Endzeitpunkt definieren. Definieren sich die Elemente eines Tracks jedoch über die Start- und Endzeitpunkte von Elementen anderer Tracks, so handelt es sich dann um *Secondary*-Tracks. Bei referenzierten Tracks muss es sich nicht zwingend um Primary-Tracks handeln, d.h. Secondary-Tracks können wiederum auf Secondary-Tracks verweisen.

Jedes Element kann mehrere Attribut-Wert-Paare enthalten. Die verwendeten *Attribute* besitzen einen eindeutigen Typ. Mit diesem Typ werden die möglichen Werte festlegt und können beispielsweise *Strings*, *Booleanwerte*, selbstdefinierte *ValueSets* oder *MultiLinks* sein. Mit ValueSets erlaubt ANVIL die Definition einer Menge von Strings, die allen Attributen als Typ zur Verfügung stehen. *MultiLinks* ermöglichen die Verknüpfung von einzelnen Elementen eines Tracks mit anderen Elementen. Seit Version 4.7 können mit den Attributtypen *Points* und *TimestampedPoints* genaue Punkte oder bestimmte Bereiche auf dem Video annotiert werden [Kipp 2008]. Dabei kann der Benutzer die gewünschten Bereiche direkt auf dem Videofenster markieren, so dass diese dann als gelbe Kreisflächen auf dem Videofenster erscheinen.

Die von ANVIL erzeugten Annotationen können dann im `XML`-Format abgespeichert oder mit Hilfe der Export-Funktion in verschiedene Formate exportiert werden. Darunter auch Formate, die von Statistikprogrammen wie SPSS zur weiteren Auswertung eingelesen werden können.

2.1.2. Spezifikationsdatei

Vor der eigentlichen Annotation muss der Benutzer eine geeignete Spezifikationsdatei für ANVIL schreiben. In dieser Datei werden neben den grafischen Einstellungen auch die Dokumentation und die Definitionen der verwendeten Tracks und Attribute beschrieben. Hier legt der Benutzer fest, welche Attribute die späteren Elemente eines Tracks beinhalten und von welchem Typ diese Attribute sein sollen.

Zur besseren Übersicht oder zur logischen Organisation von Tracks können diese in *Groups* zusammengefasst werden. Groups dienen nur zu diesem Zweck und beinhalten selbst keinerlei Annotationsdaten. Listing 2.1 zeigt einen Ausschnitt einer Spezifikationsdatei. Dieser Ausschnitt spezifiziert zwei Primary-Tracks (*visible* und *Biff-stroke*), die unter einer Group (*gesture*) zusammengefasst werden. *Biff-stroke* enthält vier Attribute (*type*, *hand*, *repetitions* und *trajectory*). Attribute *type* ist vom Typ *gestType*, der als ValueSet unter <head> definiert wurde. Es ist auch möglich ein so genanntes *anonymes ValueSet*, wie bei Attribute *hand* zu definieren. Solche anonymen ValueSets sind dann nur für dieses spezielle Attribut sichtbar.

```
<annotation-spec>
  <head>
    <valuetype-def>
      . . .
      <valueset name="gestType">
          <value-el color="red">stroke</value-el>
          <value-el color="orange">touch-self</value-el>
          <value-el color="light blue">touch-other</value-el>
          <value-el color="green">touch-object</value-el>
        </valueset>
      . . .
    </valuetype-def>
    </head>
    <body>
    . . .
    <group name="gesture">
      <track-spec name="visible" type="primary"/>
      <track-spec name="Biff-stroke" type="primary">
        <attribute name="type" valuetype="gestType"/>
        <attribute name="hand" display="true">
          <value-el>LH</value-el>
          <value-el>RH</value-el>
          <value-el>2H</value-el>
        </attribute>
        <attribute name="repetitions" valuetype="Number(1,10)"/>
        <attribute name="trajectory" valuetype="TimestampedPoints"/>
      </track-spec>
    </group>
    . . .
    </body>
</annotation-spec>
```

Listing 2.1: Ausschnitt einer Anvil Spezifikationsdatei, die zwei Tracks definiert und diese in einer Group zusammenfasst.

2.1.3. Annotation

Die eigentliche Annotation geschieht bei ANVIL im *Annotation-Board*. Um neue Elemente hinzuzufügen, muss der Benutzer nur einen Start- und Endzeitpunkt festlegen. Ein Startzeitpunkt wird definiert, indem die *Recordline* an die gewünschte Stelle verschoben und dieser als Startpunkt definiert wird. Analog dazu wird der Endpunkt über die *Playbackline* festgelegt. Nach Definition der Zeitpunkte erscheint automatisch ein neues Fenster, in dem die definierten Attribute für den Track spezifiziert werden können. Abbildung 2.1 zeigt die Benutzeroberfläche von ANVIL.

Abbildung 2.1.: ANVIL-Benutzeroberfläche.

2.2. Weitere Annotationswerkzeuge

Neben ANVIL existieren auch andere Tools, die sich mit der Annotation von Video- und Audiodateien beschäftigen. Im Folgenden sollen einige dieser Applikationen vorgestellt werden[3].

2.2.1. ELAN

ELAN (EUDICO Linguistic Annotator) wurde am Max Planck Institut für Psycholinguistik in Nijmegen (Niederlanden) entwickelt und erlaubt ebenfalls die Annota-

[3]Bei Interesse findet sich in Rohlfing u. a. [2006] ein direkter Vergleich einiger Annotationstools aus dem Jahre 2006 (u.a. mit ANVIL, ELAN, EXMARaLDA und TASX).

tion von Video- und Audiodaten. Ursprünglich wurde ELAN zur Analyse von normaler Sprache und Gebärdensprache, sowie Gestik entwickelt (Abbildung 2.2 zeigt die Benutzeroberfläche von ELAN). Jedoch kann ELAN für jede Annotation, Analyse oder Dokumentation von Mediadaten verwendet werden [Brugman und Russel 2004; Wittenburg u. a. 2006]. Mit ELAN kann der Benutzer, wie bei ANVIL, unbegrenzte Annotationen zur Mediadatei hinzufügen. Annotationen können dabei ganze Sätze, einzelne Wörter, Kommentare oder nur Beschreibungen zu der zu bearbeitenden Mediadatei sein. Analog zu den Tracks bei ANVIL werden diese Annotationen bei ELAN in so genannten *Tiers* organisiert. Tiers können dabei auch hierarchisch miteinander verbunden sein und auf Mediadateien oder andere Annotationen verweisen. Äquivalent zur Spezifikationsdatei bei ANVIL, wird das Grundgerüst einer Transkription in eine ELAN-*Templatedatei* exportiert. Sie kann dann als Grundgerüst neuer Transkriptionen dienen und enthält die Definitionen der Tiers, der Typen usw. Diese Templatedatei wird im Gegensatz zu ANVIL direkt im Datenfile gespeichert. Dies kann bei mehreren Dateien zu Inkonsistenzen führen. Ein weiteres Feature von ELAN ist die Möglichkeit der Unterteilung von Annotationen in kleinere Segmente, was das inverse Konzept zu ANVILs Secondary-Tracks ist.

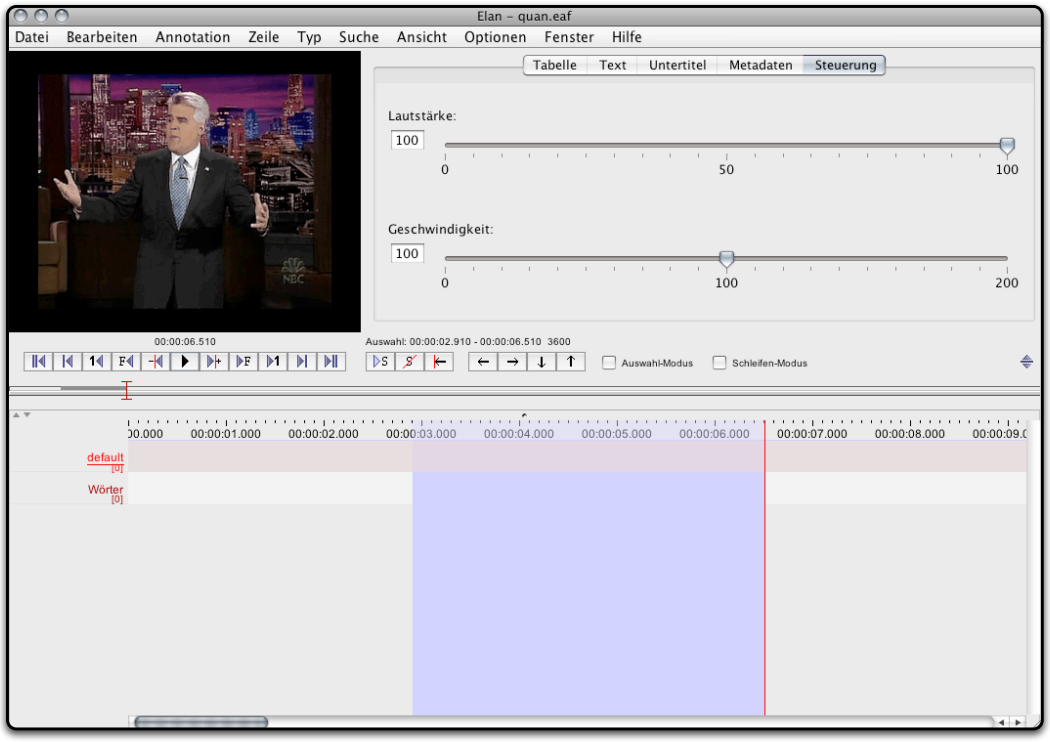


Abbildung 2.2.: ELAN-Benutzeroberfläche.

2.2.2. EXMARaLDA

EXMARaLDA (Extensible Markup Language for Discourse Annotation) von der Universität Hamburg ist ein Werkzeug aus dem Forschungsgebiet der Konversationsanalyse. EXMARaLDA besteht aus dem Datenmodell, dem Datenformat und Software-Werkzeugen zum computergestützten Erstellen und Analysieren von Korpora gesprochener Sprache [Schmidt 2005; Schmidt und Wörner 2005].

Äquivalent zu ELAN bezeichnet EXMARaLDA seine Spuren als *Tiers* und die Annotationen als *Events*. Über *Categories* wird festgelegt, was in den einzelnen Events annotiert wird. Mit *Types* werden die Tiers klassifiziert. Es existieren drei verschiedene Types:

1. **t (transcription)**: Erlaubt die Annotation von verbalem Verhalten.
2. **d (description)**: Definiert Tiers zur Annotation von nonverbalem Verhalten eine Sprechers.
3. **a (annotation)**: Speichert zusätzliche analystische Informationen zu Tiers des Typs t.

Zum Erstellen von Transkriptionen dient unter EXMARaLDA der Partitur-Editor, während der Corpus-Manager die Erstellung und Verwaltung von Korpora erledigt (Abbildung 2.3 zeigt den Partitur-Editor von EXMARaLDA).

Zur Auswertung solcher Korpora für gesprächsanalytische Zwecke bietet EXMARaLDA ebenfalls Suchwerkzeuge an. Auch in EXMARaLDA können digitale Audio- oder Videodateien in die Transkription eingebunden werden. Über den Partitur-Editor können dann Zeitpunkte aus der Aufnahme, den korrespondierenden Stellen in der Transkription zugewiesen werden. Damit kann man über die Transkriptionen automatisch immer zum dazugehörigen Abschnitt in der Aufnahme gelangen und diesen abspielen. Diese Zuordnung ist im Unterschied zu den meisten anderen Transkriptionswerkzeugen (insbesondere PRAAT, TASX und ELAN, s.o.) aber optional. Generell sind für das Arbeiten mit dem Partitur-Editor digitale Aufnahmen nicht zwingend notwendig. So muss auch die Synchronisierung nicht für die komplette Transkription durchgeführt werden.

2.2.3. TASX

Die TASX-Umgebung (Time Aligned Signal Data Exchange Format) der Universität Bielefeld wurde im Rahmen des LeaP (Learning Prosody) Projektes entwickelt. Im Gegensatz zu ANVIL, ELAN und EXMARaLDA gibt es für TASX keine aktive Entwicklung mehr. Auch die offizielle Seite[4] ist nicht mehr zu erreichen.

Die TASX-Umgebung unterteilt sich in die beiden Komponenten TASX-Format und TASX-Annotator. Das TASX-Format basiert auf XML und wurde speziell für die Annotation zeitlich geordneter, multimodaler Sprachdaten entwickelt

[4] http://tasxforce.lili.uni-bielefeld.de/ [Stand: 18. Juli 2009]

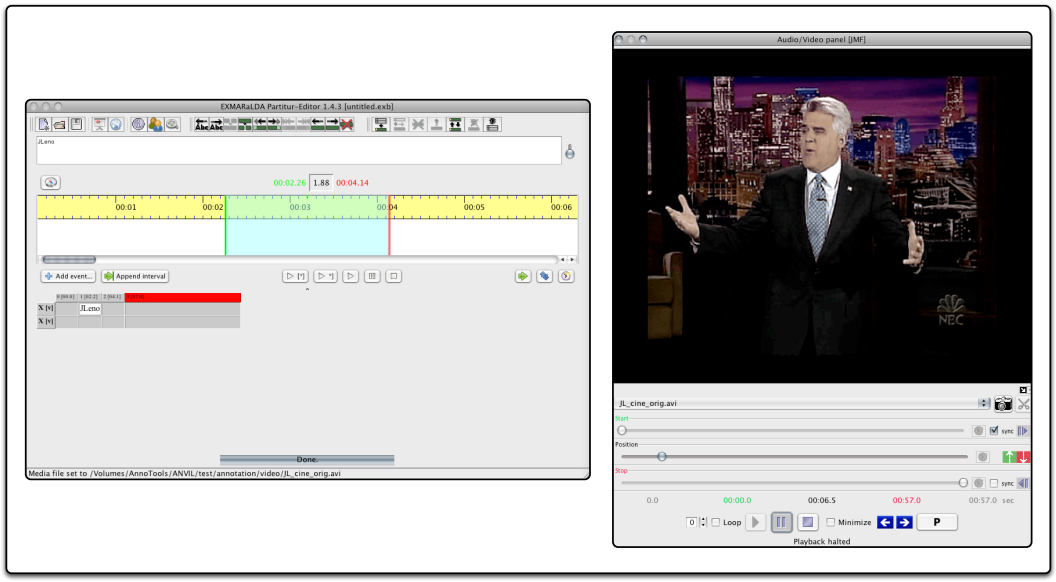


Abbildung 2.3.: EXMARaLDA: Partitur-Editor.

[Gut und Milde 2003; Milde und Gut 2001]. Der TASX-Annotator wird benötigt, um den TASX-Korpus zu erstellen. Der TASX-Annotator erlaubt die gleichzeitige Annotation von mehreren Video- und Audiodateien [Pitsch u. a. 2003].

TASX bezeichnet ANVIL-Tracks als *Layer* und die darin enthaltenen Annotationen als *Events* (vgl. *Element* bei ANVIL), wobei jedes Event beliebige Textinformationen enthalten kann. Über das XML-Attribut *e-id* können die Events identifiziert werden. Die XML-Attribute *start* und *end* legen das zeitliche Intervall fest. Diese Informationen werden in einer Aufnahme, einer *Session* gespeichert, und können optional Metadaten enthalten.

2.3. Weitere relevante Werkzeuge

Folgender Abschnitt stellt Werkzeuge vor, die entweder keine Annotationstools (Praat, ANNIS2) im Sinne von ANVIL sind oder deren Schwerpunkt nicht auf Annotation (Advene) liegt.

2.3.1. Praat

Praat[5] ist ein phonetisches Analyseprogramm, das von Boersma und Weenink an der Universität in Amsterdam entwickelt wird und ist hauptsächlich eine Software zur akustischen Analyse von Sprachsignalen [Boersma 2001]. Das Tool ist Open-Source und kann kostenlos heruntergeladen werden. Die Relevanz von Praat liegt in den zu

[5] www.praat.org

ANVIL komplementären Funktionen im Bereich der Audioverarbeitung. ANVIL gibt daher die Möglichkeit Praat-Dateien zu importieren.

Interessant ist Praat besonders für Forschende, die sich mit Phonetik und Phonologie beschäftigen, da es unter anderem die Lautstärke und den Tonhöhenverlauf visuell darstellen kann und viele Analyse-Algorithmen zur Verfügung stellt. Mittlerweile wird Praat auch für das Aussprachetraining beim Erlernen von Fremdsprachen benutzt[6], da es die Aussprache eines Wortes, die Satzbetonung und die unterschiedlichen Phoneme visualisiert darstellen kann.

2.3.2. Advene

Advene[7] (Annotate Digital Video, Exchange on the NEt) ist ein Projekt der Claude Bernard Universität in Lyon, mit dem es möglich ist, Video- und Audiodateien zu annotieren und sie als so genannte Packages von den Video- und Audiodateien entkoppelt bereit zu stellen [Auber und Prié 2005]. Unter Advene hat man die Möglichkeit, entweder die Annotationen unter Advene zu tätigen oder externe Annotationsdateien anderer Werkzeuge zu importieren. Dabei unterstützt Advene unter anderem auch ANVIL und ELAN. Die Stärke von Advene liegt in den zahlreichen Visualisierungsmöglichkeiten dieser Annotationen, wie Hypervideos oder dynamisch erzeugte XHTML-Dokumenten[8].

2.3.3. ANNIS2

ANNIS2 (ANNotation of Information Structure) ist eine webbasierte Applikation zur Suche und Visualisierung von Annotationen linguistischer Korpora [Dipper u. a. 2004; Götze und Dipper 2006]. ANNIS2 wurde entwickelt, um die verschiedenen Daten und Annotationen der 13 Forschungsgruppen des Sonderforschungsbereichs der Potsdam Universität und der Humboldt-Universität Berlin in einer Datenbank zu verwalten.

Über Importer können Annotationen verschiedener Annotationstools (wie EXMARaLDA, MMX2[9], RSTTool[10] usw.) importiert werden. Dafür werden diese Daten in das PAULA-Format (Potsdam interchange format for linguistic annotation)[11] konvertiert und anschließend in die Datenbank eingefügt. PAULA dient als Zwischenformat zur einheitlichen Darstellung von Annotationen verschiedener Annotationstools. Neben der Speicherung und Verwaltung dieser Daten, bietet ANNIS2 auch geeignete Suchfunktionen und die grafische Aufbereitung sowie Visualisierung. Über einen geeigneten Importer für ANVIL-Dateien, könnte ANNIS2 als eine Erweiterung für ANVIL angesehen werden. Damit könnten auch Forscher, die mit ANVIL arbeiten,

[6] http://www.praatlanguagelab.com
[7] http://liris.cnrs.fr/advene/index.html
[8] http://www.w3.org/TR/xhtml1/
[9] http://mmax.eml-research.de
[10] http://www.wagsoft.com/RSTTool/
[11] http://www.sfb632.uni-potsdam.de/~d1/paula/doc/

ihre Daten für linguistische Forschungszwecke aufbereiten und visualisieren sowie die Suchfunktionen von ANNIS2 nutzen.

Fazit: Mit allen in diesem Kapitel vorgestellten Annotationstools können Gesten annotiert werden. Keines dieser Tools bietet jedoch die Möglichkeit einer präzisen und intuitiven Beschreibung von Gesten. Auch unter den relevanten Werkzeugen findet sich kein Tool, dass diese Lücke schließen könnte. Aus diesem Grund halte ich folgende Lösung für sinnvoll: die Annotation und Animation von Gesten mit Hilfe eines 3D-Skeletts.

3. Skelettbasierte Repräsentation und Animation

Echtzeit-Animation von menschenähnlichen Charakteren ist ein wichtiges Spezialgebiet der Computer-Grafik. Unter Animation versteht man das Bewegen von Objekten, die sich von sich aus nicht selbst bewegen können [Parent 2001]. Dies geschieht, indem der Animationskünstler dem Objekt direkt oder indirekt mitteilt, wie sich das Objekt durch den Raum oder die Zeit bewegen soll.

Wie in der Motivation beschrieben, können Animationen über zwei Methoden erfolgen: der Straight-Ahead-Methode oder der Key-Frame-Methode. Bei der ersten Methode beginnt der Animationskünstler beim ersten Bild und zeichnet danach Bild für Bild, bis er die Endpose oder das Ende der Sequenz erreicht hat, ohne die Szene vorgeplant zu haben. Animationen, die genaues Timing erfordern, setzen hingegen auf die Key-Frame-Technik. Da wird der Ablauf mit Hilfe von Schlüsselposen skizziert, so dass bestimmte Zeitpunkte fixiert sind. Zur vollständigen Animation werden die Zwischenposen dann in einem späteren Prozess hinzugefügt. Bei heutigen computergestützten Animationen wird die Key-Frame-Animation verwendet, bei der der Computer die Erstellung der „In-betweens“ übernimmt. Dabei können je nach verwendeter Interpolationsmethode und Komplexität der Objekte, die interpolierten Bewegungsrichtungen verkehrt sein oder es kann zu Überlagerungen von Objekten kommen [Hodgins u. a. 1999].

Neben den beiden Animationstechniken gibt es noch das *Motion-Capture-Verfahren* [Hodgins u. a. 1999]. Bei diesem Verfahren werden spezielle Marker, so genannte *Tracker*, eingesetzt um Bewegungen aufzuzeichnen, die dann später auf entsprechende 3D-Modelle übertragen werden. Da diese Tracker jedoch auf speziellen Anzügen oder der Haut angebracht werden, ergeben sich Fehler in den erfassten Daten. Zusätzlich müssen die Größenverhältnisse zwischen dem 3D-Modell und dem aufgezeichneten Objekt übereinstimmen, da es ansonsten zu erheblichen Fehlern kommen kann. Legt ein Schauspieler beispielsweise seinen Arm auf einen Tisch, kann die ungleiche Größe zwischen 3D-Modell und Schauspieler dazu führen, dass sich der Arm des Modells in der Luft befindet oder durch den Tisch hindurchgeht. Zusätzlich ist das Ergebnis von den verwendeten Markern abhängig. Magnetische Marker können Störungen in den Daten erzeugen, während optische Systeme, bei denen passive oder aktive Marker von Kameras erfasst werden, sehr teuer sind. Zudem kann es bei optischen Systemen zu Datenverlusten kommen, wenn die Maker verdeckt werden, so dass sie von den Kameras nicht mehr erfasst werden können [Kitagawa und Windsor 2008].

Auch wenn die Ergebnisse manuell nachkorrigiert werden müssen, kommen Motion-Capture und die Key-Frame-Animation in vielen Bereichen und Applika-

tionen zum Einsatz, deren Einsatzfelder von der Medizin über Sport zu Videospielen und Filmproduktionen reichen [Savoye und Meyer 2008].

Dieses Kapitel soll die Grundlagen vorstellen, die für eine Animation von menschlichen 3D-Skeletten benötigt werden. Neben Java3D, der verwendeten Bibliothek zur Erstellung von 3D-Szenen, werden auch Aspekte der skelettbasierten Animation erläutert. Zu Letzterem gehören neben einem hierarchisch aufgebauten Skelett auch Techniken wie die inverse Kinematik, die die Manipulation eines Skeletts ermöglichen. Zur Animation kann der Animationskünstler mit Hilfe der inversen Kinematik Posen mit dem Skelett erstellen, während die „In-betweens“ über Interpolationstechniken automatisch berechnet werden (siehe Abschnitt 3.3.2). Im letzten Abschnitt dieses Kapitels wird COLLADA vorgestellt, ein auf XML-basierendes Dateiformat zum Austausch von Daten zwischen 3D-Modellierungsprogrammen und 3D-Systemen.

3.1. Programmierung in Java3D

Dieser Abschnitt soll nur einen Überblick über *Java3D* und dessen grundlegende Konzepte geben, um die in Kapitel 5 erläuterte Implementierung zu verstehen. Zum vollen Verständnis von Java3D oder einer Anleitung zum Programmieren mit Java3D wird an dieser Stelle auf die hervorragenden Tutorials von Sun[1] verwiesen.

Java3D ist eine Klassenbibliothek, die eine Standarderweiterung der *Java2-Plattform* darstellt. Somit können Java-Entwickler diese Bibliothek einfach in ihren vorhandenen Code integrieren und müssen dafür keine neue Programmiersprache erlernen oder ihre Plattform ändern. Mit Hilfe von Java3D können sie dreidimensionale Szenen aufbauen und Animationen, Interaktionen oder Audioeffekte realisieren. Vorgestellt wurde diese Bibliothek erstmals 1998 auf der Konferenz *SIGGRAPH 98* [Zeppenfeld 2004]. Seitdem wurde sie weiter aktualisiert und erweitert. Mittlerweile liegt sie in der Version *1.5.1* vor. Die aktuelle Version ist für die gängigsten Betriebssysteme wie Microsoft Windows, Mac OS X oder Linux erhältlich[2].

Java3D verfolgt beim Aufbau von dreidimensionalen Szenen den Ansatz des *Szenegraphen* (siehe Abschnitt 3.1). Dabei werden die einzelnen erzeugten Objekte in einer Baumstruktur zusammengefasst [Zeppenfeld 2004]. Im Szenegraph werden Geometrie, Beleuchtung, Platzierung, Orientierung und Erscheinung der Grafik- und Audioobjekte festgelegt.

Szenegraph

Die Wurzel des *Szenegraphen* wird durch das Universum (*VirtualUniverse*) repräsentiert und enthält die gesamte Szene. Ein Szenegraph kann auch als ein gerichteter azyklischer Graph angesehen werden. Zwischen den Knoten des Graphen herrscht eine *Eltern-Kind-Beziehung*, d.h. ausgehend vom Wurzelknoten des Szenegraphen kann jeder Knoten eine beliebige Anzahl von Kindknoten, jedoch nur genau einen

[1] http://java.sun.com/developer/onlineTraining/java3d/index.html
[2] http://java.sun.com/javase/technologies/desktop/java3d/

Vorgängerknoten besitzen. Enthält eine Szene mehrere Objekte, so können diese zu einer inhaltlichen Gruppe (*BranchGroup*) und/oder zu einer Transformationsgruppe (*TransformGroup*) zusammengefasst werden. Während die inhaltliche Gruppe nur Informationen über die Gruppierung der visuellen Elemente einer Szene zusammenfasst, stellt die Transformationsgruppe eine feste Gruppierung der Teilobjekte zu einem Gesamtobjekt dar. Das bedeutet, dass Transformationen nicht mehr auf den einzelnen Objekten ausgeführt werden, sondern das die gewünschte Transformation immer auf das zusammengesetzte Objekt angewendet wird. Somit stellt die TransformGroup die wichtigste Komponente eines Szenegraphen dar, da nur sie die Informationen über die Veränderungen eines Objektes (Translation, Rotation und Skalierung) und der dazugehörigen Kinder beinhaltet. Einzelne Orte der Szene werden in so genannte *Locales* aufgeteilt. Diese beschreiben einzelne Teile der Szene. Für jeden einzelnen Ort teilt sich der Graph in zwei Zweige: den Inhaltszweig (*content branch*), der alle Objekte und deren Eigenschaften enthält, und den Darstellungszweig (*view branch*), der für die Darstellung der Szene verantwortlich ist. Formen (*Shape*) sind die einzelnen Objekte der Szene und stellen Blattknoten (*Leaf*) in der Baumstruktur dar. Über die Komponente (*NodeComponent*) können die Geometrie (*Geometry*) oder das Aussehen (*Appeareance*) des Objekts bestimmt werden. Abbildung 3.1 zeigt ein Beispiel für einen Szenegraphen.

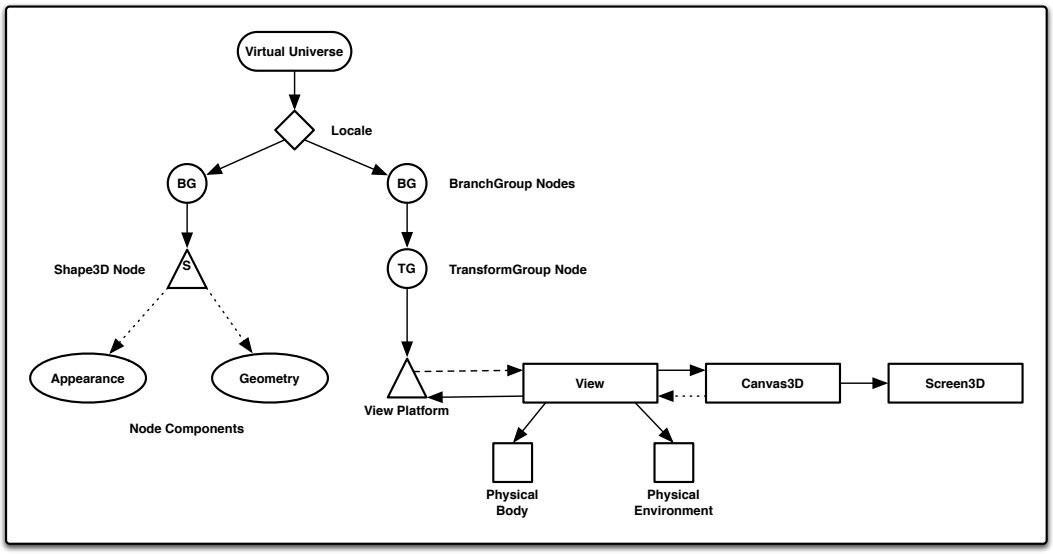


Abbildung 3.1.: Die Abbildung beschreibt einen Java3D-Szenegraphen.

TransformGroup

Objekte einer Szene können zu *TransformGroups* zusammengefasst werden, die geometrische Informationen wie Translationen und Rotationen beinhalten. Diese Transformationen werden in einer Transformationsmatrix, der *Transform3D*, gespeichert.

Sie repräsentiert die Transformationen (Translation, Rotation und Skalierung), die auf die 3D-Geometrien der Szene angewendet werden. Wird eine Transformation auf eine TransformGroup ausgeführt, dann wird sie nicht nur auf ein einzelnes Objekt der Gruppe, sondern auf alle Objekte dieser Gruppe ausgeführt. Dadurch wird die Manipulation von großen zusammengesetzten Objekten stark vereinfacht, da die Angabe der Transformation für die Unterobjekte entfällt. Es reicht aus, nur die oberste TransformGroup zu verändern, um die Teilobjekte mitzubewegen. Daher werden TransformGroups zur festen Gruppierung von Teilobjekten zu einem hierarchisch strukturierten Gesamtobjekt verwendet. Diese Struktur wird von Applikationen verwendet, um eine hierarchische Kontrolle zu realisieren. Beispielsweise kann ein Arm in drei Teile unterteilt werden: Oberarm, Unterarm und Hand. Jede dieser Komponenten wird in einer eigenen TransformGroup abgebildet, um auf ihnen unabhängige Transformationen durchführen zu können. So soll es zum Beispiel möglich sein, nur die Hand zu rotieren, ohne die Ausrichtung des Ober- und Unterarms zu verändern. Diese drei TransformGroups werden dann in einer obersten TransformGroup, dem Arm, erneut zusammengefasst. Führt man nun einen Translation auf die TransformGroup Arm durch, so wird der gesamte Arm, d.h. auch alle anderen Unterobjekte mitbewegt (Abbildung 3.2).

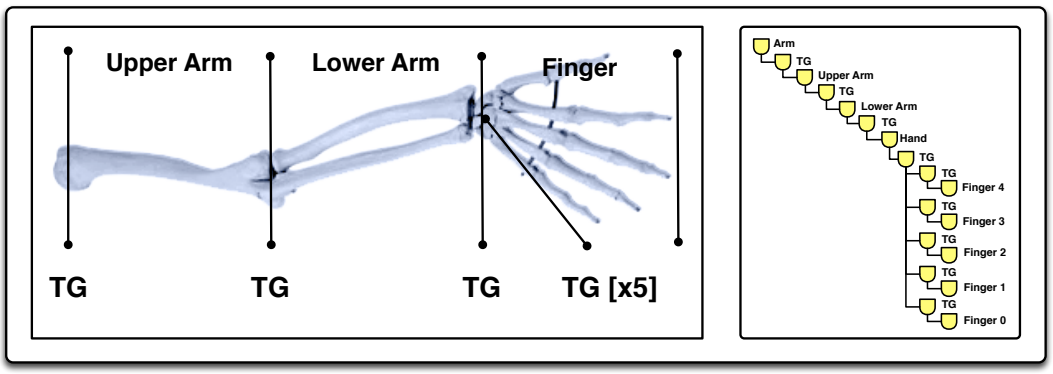


Abbildung 3.2.: Links ist die Unterteilung eines Arms in die drei Hauptteile: Oberarm, Unterarm und Hand. Die Hand enthält wiederum 5 Finger. Auf dem rechten Bild ist die Umsetzung mit Java3D und den `TransformGroups` für die einzelnen Komponenten zu sehen.

BranchGroup

Instanzen von *BranchGroup* stellen einen Untergraphen der Szene dar und sind die einzigen zulässigen Kinder eines *Locale*-Knotens. BranchGroups können mehrere Kinder haben, die entweder Gruppen oder Blattknoten sind. In BranchGroups können mehrere Objekte der Szene, die inhaltlich zusammengehören, zusammengefasst werden. BranchGroup-Objekte können zwei Zustände haben: *live* oder *compiled*. Durch das Hinzufügen zu einem *Locale* wechselt die BranchGroup und deren

Unterknoten in den *live*-Zustand, d.h. sie sind bereit zum Rendern. Im live-Zustand können dann nur noch Operationen durchgeführt werden, die vorher über die Methode `setCapability(···)` freigegeben wurden. Beim Kompilieren einer BranchGroup wird der Knoten selbst und seine Unterknoten in eine für das Rendern effiziente Form gebracht. Diesen Zustand bezeichnet man als *compiled*-Zustand. Allerdings sind Veränderungen dann nur noch eingeschränkt möglich. Beispielsweise können Geometrieinformationen nicht mehr verändert werden. Daher ist es empfehlenswert BranchGroups zu kompilieren, bevor sie in den *live*-Zustand gebracht werden.

3.2. Aufbau und Manipulation eines Skeletts

Nach einer kurzen Vorstellung von *H-Anim*, einem Standard zur Beschreibung von 3D-Figuren, beschreibt dieser Abschnitt die Umsetzung eines Skeletts in Java3D. Dabei wird auch erläutert, wie das benötigte Skelett in Java3D modelliert wurde und warum die hierarchische Struktur, die auch H-Anim vorgibt, besonders bei der Umsetzung mit Java3D hilfreich und sinnvoll ist. Der zweite Teil beschäftigt sich mit den Möglichkeiten der Skelettmanipulation über die direkte und inverse Kinematik (IK).

3.2.1. Aufbau

Ein Skelett wird nach Magnenat-Thalmann u. a. [2008] definiert, über eine hierarchisch angeordnete Menge von Gelenken, die ein oder mehrere Freiheitsgarde (Degrees of freedom, DOF) haben können. Die Anzahl und Auswahl der verwendeten Gelenke richten sich nach Einsatz des Skeletts. So werden bei medizinischer Anwendung eventuell alle Wirbelknochen der Wirbelsäule benötigt, während einfache Computerspiele schon mit deutlich weniger Gelenken auskommen können. Der Grund, warum oft nur die benötigten Gelenke implementiert werden, liegt in der steigenden Komplexität der Posenberechnung mit IK-Verfahren bei zunehmender Gelenkanzahl. Über die Menge der Ausrichtungen jedes einzelnen Gelenks wird eine Pose des Skeletts eindeutig definiert, d.h. dass jede Pose über eine Menge von Gelenkausrichtungen bzw. Gelenkwinkel eindeutig definiert und somit reproduziert werden kann.

Zur einheitlichen Bezeichnung der Gelenke und Knochen eines Skeletts sowie dessen Aufbau, wurde der *H-Anim (Humanoid animation)*-Standard entwickelt. Im Folgenden soll dieser Standard vorgestellt werden, da dieser auch zur Beschreibung des in der Arbeit erstellten Skeletts verwendet wurde. Denn um Skelette aus externen Beschreibungen in einer Anwendung benutzen zu können, müssen die einzelnen Gelenke der entsprechenden Repräsentation im Programm zugewiesen werden. Dies kann entweder manuell oder automatisiert geschehen. Damit diese Zuweisungen automatisch ablaufen können, müssen die Bezeichnungen übereinstimmen. Mit H-Anim soll eine einheitliche Bezeichnung der Gelenke und Knochen sichergestellt werden. In dieser Arbeit betrifft dies nur Teile des Arms. Des Weiteren werden kinematische Ketten vorgestellt, die ein wichtiger Bestandteil der inversen Kinematik sind.

3.2.1.1. H-Anim

Nach *H-Anim*[3], einem internationalen Standard zur abstrakten Repräsentation von menschlichen 3D-Figuren, werden menschenähnliche Avatare hierarchisch strukturiert [WEB3D Consortium]. Dadurch kann man die Abhängigkeiten innerhalb eines Skeletts direkt ersehen. So ist aus der hierarchischen Struktur direkt erkennbar, dass beispielsweise die Hand mit dem Handgelenk verbunden ist, das Handgelenk wiederum mit dem Unterarm und dieser mit dem Ellbogen. Dieser Standard wurde entwickelt, um den Austausch zwischen 3D-Modellierungs- und Animationsprogrammen verschiedener Hersteller zu ermöglichen (Abbildung 3.3 zeigt die hierarchische Struktur eines ausspezifizierten menschlichen Skeletts nach H-*Anim*).

Ein weiterer Vorteil von H-Anim ist die einfache progammiertechnische Umsetzung mit dem Szenegraphenkonzept, da die hierarchische Struktur über diese Spezifikation vorgegeben ist. In Abschnitt 5.1 wird beschrieben, wie diese hierarchische Struktur mit Hilfe des Szenegraphenkonzepts von Java3D umgesetzt werden kann. Da der Schwerpunkt dieser Arbeit auf Gesten liegt, wurde kein komplett spezifiziertes Skelett verwendet. So wurden unter anderem der größte Teil der Wirbelsäule, die Finger- und Fußknochen nicht berücksichtigt. Da die Knochen der Wirbelsäule nicht verwendet werden und somit keine automatische Zuweisung erfolgt, wurden sie unter *chest* und *waist* zusammengefasst. Diese sind zwar keine offiziellen Bezeichnungen von H-Anim, aber eingängiger als die von H-Anim vorgeschlagenen Bezeichnungen. Sollten diese Teile in zukünftigen Arbeiten einbezogen werden, so sollten auch dort die Bezeichnungen von H-Anim verwendet werden. Die verwendeten Bezeichnung sind in Abbildung 5.3 herauszulesen.

3.2.1.2. Kinematische Kette

Kinematische Ketten werden bei der inversen Kinematik (siehe Abschnitt 3.2.2.2) benötigt. Die wichtigsten Teile einer kinematischen Kette sind dabei die *Wurzel* (Basis) und der *End-Effektor*. Die *Wurzel (Basis)* ist das erste Gelenk einer kinematischen Kette. An der Wurzel werden die verschiedenen Glieder über Gelenke hierarchisch angehängt. Sie muss nicht zwingend mit der Wurzel des gesamten Skeletts übereinstimmen. Bewegt man die Wurzel dieser kinematischen Kette, so werden aufgrund der hierarchischen Struktur alle Elemente dieser Kette mitbewegt. Der *End-Effektor* ist das letzte Objekt einer Kette und befindet sich am Ende des letzten Gliedes. Der End-Effektor dient dabei als Kontrollobjekt der gesamten Kette. Dieses Element spielt besonders bei der inversen Kinematik (siehe Abschnitt 3.2.2.2 eine wichtige Rolle. Bewegt man nämlich den End-Effektor, so richten sich die Gelenke der dazugehörigen kinematischen Kette dementsprechend neu aus. Damit wird die Animation und das Erstellen von Posen deutlich vereinfacht, da nur noch ein Objekt verändert werden muss, anstelle von mehreren Gelenken.

[3] http://www.h-anim.org

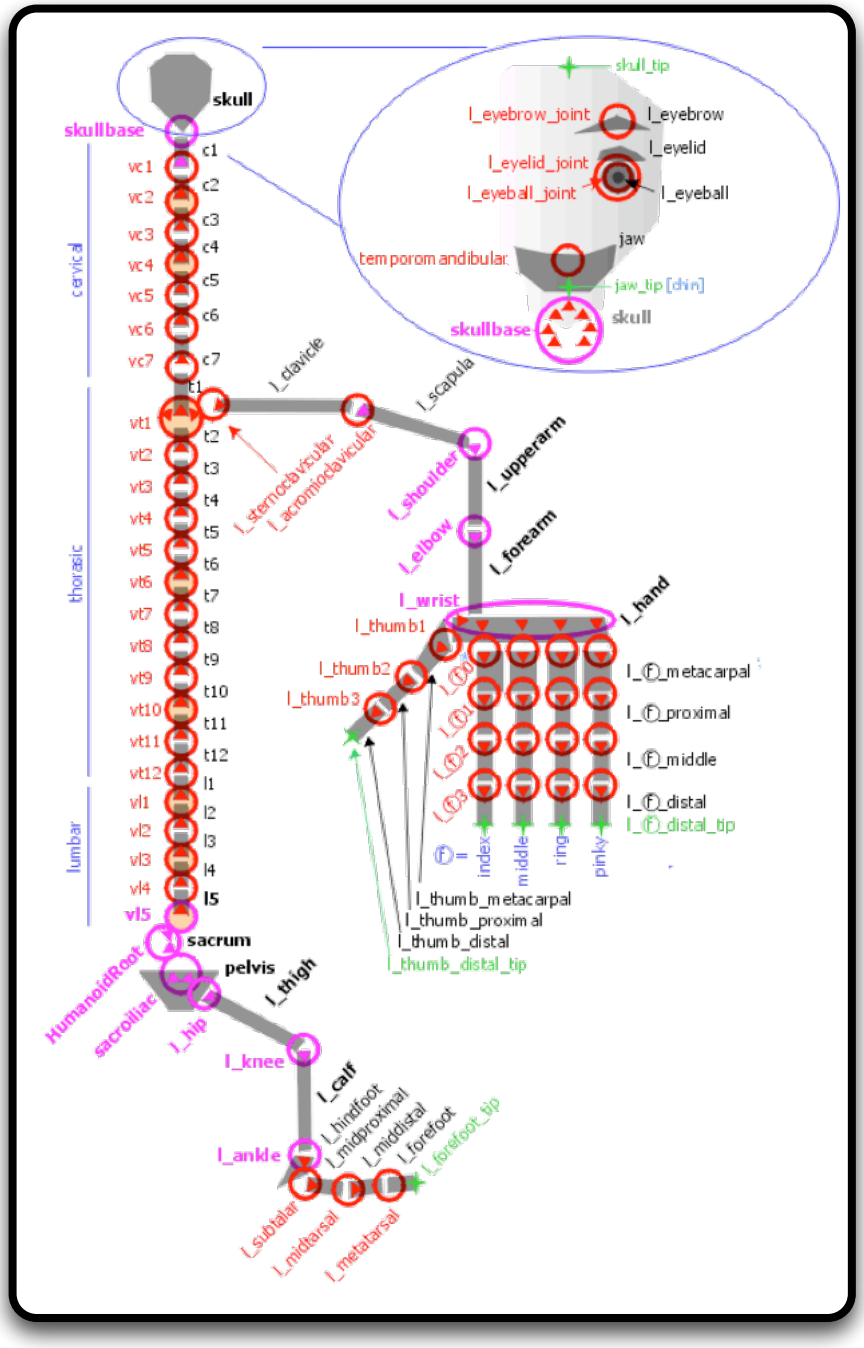


Abbildung 3.3.: Hierarchische Struktur eines Skeletts nach *H-Anim* (Entnommen aus [WEB3D Consortium]).

3.2.2. Manipulation

Folgender Abschnitt beschäftigt sich mit den zwei gängigen Methoden zur kinematischen Manipulation eines Skeletts: die *direkte Kinematik (engl. Forward Kinematics oder FK)* und die *inverse Kinematik (engl. Inverse Kinematics oder IK)*. Grundlagen dieser Methoden ist das in Abschnitt 3.2.1 beschriebene hierarchische Skelettmodell, welches zusätzlich noch aus verschiedenen kinematischen Ketten besteht [Breiner 2005]. Das Konzept der kinematischen Kette wurde in Abschnitt 3.2.1.2 erläutert. Neben der kinematischen Methode gibt es noch die dynamische Methode, die im Gegensatz zur kinematischen Manipulation die physikalischen Gesetze berücksichtigt [Welman 1993]. Diese Arbeit betrachtet jedoch nur den kinematischen Ansatz.

3.2.2.1. Forward Kinematics (FK)

Eine Standardmethode zur Manipulation von Skeletten ist die so genannte *FK (Forward Kinematics)* [Maestri 2006]. Mathematisch lässt sich das FK-Problem folgendermaßen beschreiben:

Seien $\Theta_1, ..., \Theta_n$ die Transformationsmatrizen mit den Winkelstellungen der einzelnen Gelenke und Vektor $\overrightarrow{x}$ die Position und Orientierung des End-Effektors, dann lässt sich FK als Konkatenation der Transformationsmatrizen beschreiben und hat folgende Form:

$$f(\Theta_1, ..., \Theta_n) = \overrightarrow{x} \tag{3.1}$$

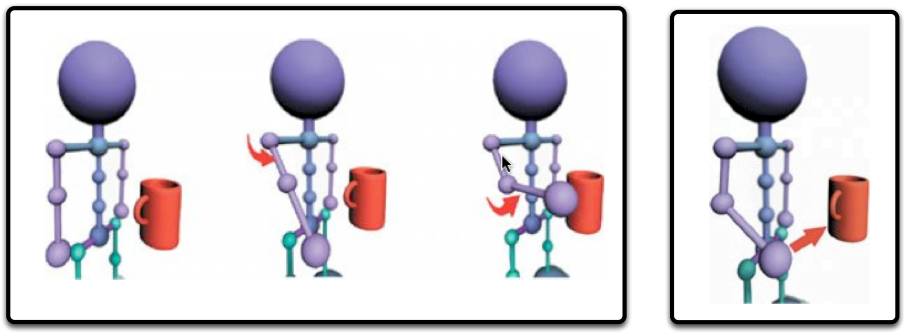
Dies bedeutet, dass eine Pose durch Rotationen der betroffenen Gelenke erreicht wird. Über die Winkelstellungen der Gelenke ist damit jede Pose eindeutig definiert. Für jede neue Pose müssen diese Schritte jedoch erneut durchlaufen werden. Abbildung 3.4(a) illustriert dies an einem Beispiel. Soll das Skelett die Pose einnehmen, so dass die Hand an der Tasse ist, müssen das Schulter-, Ellbogen und Handgelenk rotiert werden.

3.2.2.2. Inverse Kinematics (IK)

Im Gegensatz zur FK, bei der man die gewünschte Pose über Rotationen der einzelnen Gelenke erreicht, soll die Pose bei der *IK (Inverse Kinematics)* über die Angabe der Zielposition des End-Effektors erreicht werden. Dazu müssen die Ausrichtungen der betroffenen Gelenke berechnet werden. Dazu muss die inverse Gleichung von (3.1) gelöst werden:

$$(\Theta_1, ..., \Theta_n) = f^{-1}(\overrightarrow{x}) \tag{3.2}$$

Problematisch für das Lösen des IK-Problems, ist die *Nicht-Linearität* von f und die *Nicht-Eindeutigkeit* der Lösungen für die Gleichung (3.2). Die Lösung für Gleichung (3.1) hingegen ist immer eindeutig. Abbildung 3.4(b) veranschaulicht das Problem der inversen Kinematik.



(a) (b)

Abbildung 3.4.: Die linke Abbildung illustriert das FK-Problem: Um mit Hilfe von FK die Hand an der Tasse zu positionieren, werden alle benötigten Gelenke nacheinander bewegt. In diesem Falle wird zuerst die Schulter und danach der Ellbogen rotiert, bis die Hand das Ziel (die Tasse) erreicht hat. Bei der rechten Abbildung wird die Pose definiert, indem der End-Effektor (die Hand) verschoben wird. In diesem Falle soll die Hand ebenfalls an der Tasse sein. Dazu müssen die Gelenkausrichtungen für Schulter und Ellbogen berechnet werden (Abbildung aus Maestri [2006]).

3.2.2.3. Übersicht über IK-Verfahren

Im Allgemeinen werden IK-Algorithmen in numerische und analytische Verfahren unterteilt [Tolani u. a. 2000]. Analytische Methoden werden als *vollständig* bezeichnet, weil sie alle möglichen Lösungen finden. Weiterhin können analytische Methoden in solche geschlossener Form oder in Methoden unterteilt werden, die auf den algebraischen Eliminationsverfahren beruhen. Bei der ersten Methode können die Lösungen der Gleichungen als eine Menge von Gleichungen *geschlossener Form* angegeben werden. Dieses Verfahren ist jedoch nur für Systeme mit maximal sechs Freiheitsgraden möglich, die zusätzliche Eigenschaften erfüllen müssen. Bei algebraischen Eliminationsverfahren werden die Lösungen als Polynomialgleichung mit mehreren Variablen ausgedrückt. Beispiele für analytische Methoden wären die *Jacobian Transpose Method* [Welman 1993] oder der analytische Algorithmus von Tolani u. a. [2000] für kinematische Ketten mit sieben Freiheitsgraden.

Im Gegensatz dazu nähern sich numerische Verfahren wie CCD der Lösung iterativ an (siehe Abschnitt 3.2.2.4). Generell werden analytische Verfahren numerischen Verfahren vorgezogen, da sie alle möglichen Lösungen beinhalten, zudem schneller zu berechnen und die Ergebnisse verlässlicher sind. Bei zu komplexen Systemen greift man jedoch auf numerische Verfahren zurück, da diese auch beliebig lange kinematische Ketten lösen können. Dafür wird die langsamere, iterative Vorgehensweise in Kauf genommen.

Wie in Abschnitt 3.2.2.2 erläutert, existieren bei IK-Verfahren keine eindeutigen

Lösungen. Bei Verwendung mit menschlichen Skeletten bedeutet dies, dass neben der gewünschten Pose auch unerwünschte oder anatomisch inkorrekte Lösungen auftreten können (siehe Abbildung 3.5). In diesen Fällen muss der Animationskünstler manuell nachkorrigieren. Eine Möglichkeit, um unnatürliche Posen zu vermeiden, wäre die Begrenzung der Beweglichkeit von Gelenken durch so genannte *Joint-Limitations*. Dazu schränkt man die Beweglichkeit der Gelenke bezüglich der Rotationsachsen und -winkel ein. So kann beispielsweise das Knie- oder das Ellbogengelenk nur in einer Achse und bis zu einem bestimmten Grad rotiert werden, während das Schultergelenk in allen drei Achsen gedreht werden kann [Liu und Prakash 2003].

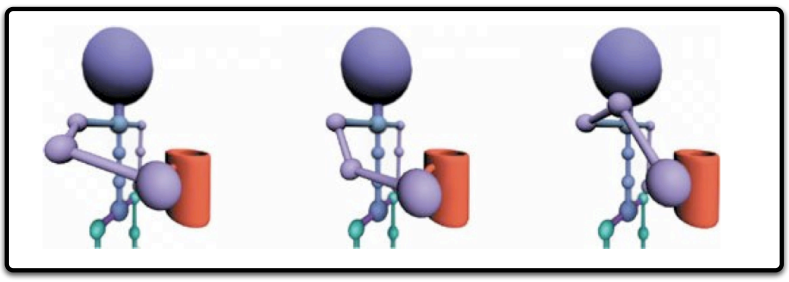


Abbildung 3.5.: Die Problematik bei der *Inverse Kinematics* ist die Nicht-Eindeutigkeit der Lösungen. Einige Lösungen sind akzeptabel (linkes und mittleres Bild) und andere können unerwünscht sein, da sie der menschlichen Anatomie nach nicht natürlich sind (rechtes Bild) (Abbildung entnommen aus Maestri [2006]).

3.2.2.4. Cyclic Coordinate Descent (CCD)

CCD (Cyclic Coordinate Descent) ist ein iterativer Alorithmus, der versucht, die Orientierung und Positionsabweichung zu minimieren. Dabei wird in jedem Iterationsschritt immer nur ein Gelenk verändert [Welman 1993]. Ein Vorteil von CCD ist, dass jede hierarchisch aufgebaute kinematische Kette, die nur einen End-Effektor besitzt, mit CCD verändert werden kann. Ausserdem ist sie einfach zu implementieren und schnell. Der in dieser Arbeit verwendete Algorithmus wurde von Lander [1998] vorgestellt: Ausgehend vom letzten Glied der kinematischen Kette läuft der Algorithmus iterativ bis zur Wurzel. Der Algorithmus ist beendet, wenn der End-Effektor das gewünschte Ziel erreicht hat oder ein bestimmter Abstand vom End-Effektor zum Zielpunkt unterschritten ist. Andernfalls beginnt der Algorithmus erneut beim letzten Glied. Im Einzelnen sehen diese Schritte folgendermaßen aus (zum Vergleich dazu Abbildung 3.6 oder CCD als Pseudo-Code in Listing 3.1.):

Schritt 1 Ausgehend von der Ausgangsposition werden die Vektoren $\overrightarrow{RE}$ und $\overrightarrow{RD}$ berechnet, wobei R der Ursprung des aktuellen Gliedes, E die Position des End-Effektors und D das gewünschte Ziel ist (Pseudo-Code Zeile 4-6).

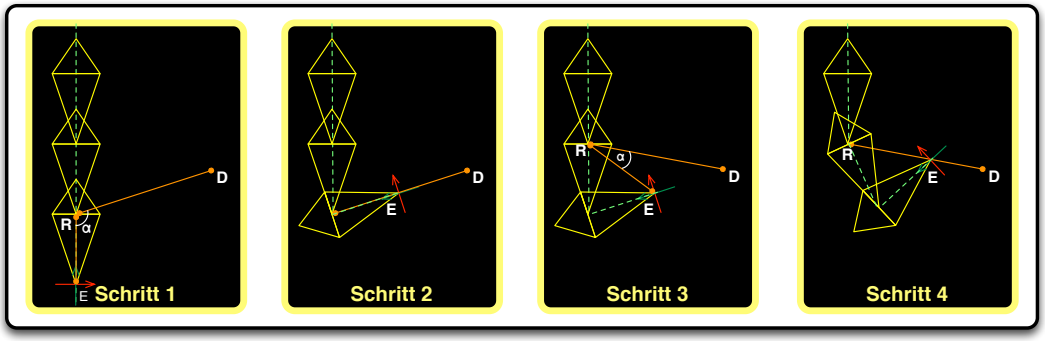


Abbildung 3.6.: Illustration von CCD (Schritt 1-4).

```
1  While (AbstandEffektor zu Zielpunkt > threshold
2      and AnzahlWiederholung < maxAnzahlWiederholung)
3  {
4    Wähle Gelenk R aus
5    Berechne Vektor RD (Gelenk zu Ziel)
6    Berechne Vektor RE (Gelenk zu End-Effektor)
7    Berechne Winkel Rotationswinkel rot zwischen (RD und RE)
8    Berechne Rotationsachse RotA
9    Rotiere Gelenk um Winkel rot um Achse RotA
10   If R ist Wurzel then beginne bei letzten Glied
11     Else wähle nächst höhere Gelenk der kinematischen Kette
12 }
13 End while
```

Listing 3.1: CCD Pseudo-Code

Schritt 2 Anhand der Vektoren aus *Schritt 1* wird der Winkel α mit Hilfe des Skalarproduktes berechnet. α beschreibt den Winkel um den $\overrightarrow{RE}$ gedreht werden muss, damit er in dieselbe Richtung zeigt wie $\overrightarrow{RD}$ (Pseudo-Code Zeile 7).

Schritt 3 Die Rotationsachse, um die das aktuelle Gelenk rotiert werden muss, berechnet sich aus dem Kreuzprodukt der Vektoren $\overrightarrow{RE}$ und $\overrightarrow{RD}$ (Pseudo-Code Zeile 8).

Schritt 4 Nachdem der Rotationswinkel α und die Rotationsachse bekannt sind, wird das Gelenk rotiert (Pseudo-Code Zeile 9).

Iteration Nach jeder Rotation nähert sich der End-Effektor dem Zielpunkt. Sollte der Zielpunkt noch nicht erreicht worden sein, beginnt der Algorithmus beim nächsthöheren Glied der Kette (Pseudo-Code Zeile 10).

Der hier vorgestellte Algorithmus vernachlässigt die Orientierung des End-Effektors (Stellung der Hand) und die Joint-Limitations. Bei einfachen Gelenken sind die Joint-Limitations leicht zu realisieren. In diesen Fällen wird vor der Rotation getestet, ob der gegebene Winkel für das betroffene Gelenk zulässig ist (Schritt 4). Bei negativem Test muss dementsprechend reagiert werden. Beispielsweise könnte in diesem Fall auf die Veränderung des Gelenkes verzichtet werden und das Gelenk wird übersprungen oder es wird nur um den maximal erlaubten Winkel verändert. Jedoch wird diese Testroutine schon bei einem menschlichen Arm sehr aufwändig und komplex, da die Beweglichkeit des Schultergelenkes von der Position des Handgelenks abhängt (siehe Abbildung 3.7).

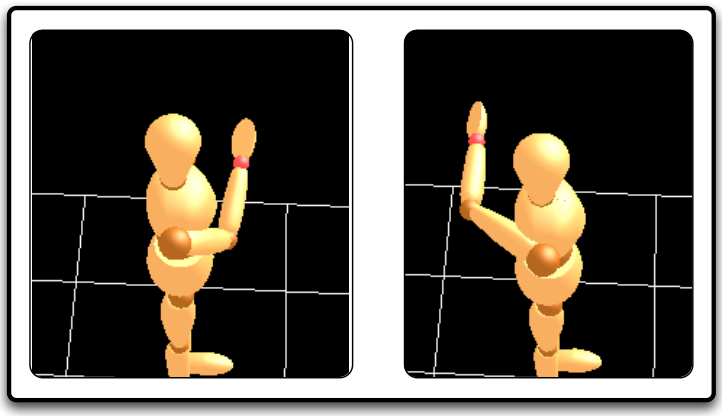


Abbildung 3.7.: Beispiel: Abhängigkeiten von Joint-Limitations anhand eines Armes. Im linken Bild ist die Gelenkstellung der Schulter möglich, wenn der Arm vor dem Körper sein soll. Die rechte Stellung ist im Allgemeinen nicht möglich, wenn der Arm hinter dem Körper ist, obwohl die Schulter in beiden Fällen nach links rotiert wird.

3.3. Interpolation von Posen

Die Interpolation zwischen zwei Key-Posen soll den Aufwand des Annotierers reduzieren, indem die Frames zwischen zwei Schlüsselposen automatisch berechnet werden. Dadurch beschränkt sich die Arbeit des Annotierers nur noch auf die wichtigsten Schlüsselposen, die für die zu annotierende Geste charakteristisch sind. Dieser Abschnitt erläutert die verwendete Methode SLERP (Spherical Linear Interpolation) und die benötigten mathematischen Grundlagen über Rotationen und Quaternionen.

3.3.1. Rotationen

Rotationen gehören zu den *geometrischen Transformationen*, d.h. Operationen, die auf Objekte in einem dreidimensionalen Raum ausgeführt werden können. Neben Rotationen gibt es noch Translationen. Eine Translation ist die Bewegung eines Punktes

innerhalb eines Raumes von einer Position auf eine andere. Da Skelettmanipulationen über die Rotation von Gelenken erfolgt, liegt der Schwerpunkt bei dieser Arbeit auf den Rotationen, so dass Translationen vernachlässigt werden können.

Die Definition von Rotationen kann auf vielfältige Art und Weise geschehen. Neben *Eulers Rotationstheorem* findet man in der Literatur häufig auch *Euler-Winkel* als Definition für Rotationen (siehe Abschnitt 3.3.1.1) [Dam u. a. 1998]. In der Computergrafik werden jedoch *Quaternionen* (siehe Abschnitt 3.3.1.2) zur Darstellung von Rotationen bevorzugt. Im Folgenden sollen die letzten beiden Rotationsmethoden vorgestellt werden, wobei die Rotation mit Hilfe von Quaternionen die wichtigste ist, da die hier realisierte Interpolation über SLERP mit Hilfe von Quaternionen definiert wird (siehe Abschnitt 3.3.2).

3.3.1.1. Euler-Winkel und Rotationsmatrizen

Eine häufige Methode, um eine Rotation zu beschreiben sind *Euler-Winkel*. Bei der Darstellung als *Euler-Winkel*, wird eine Rotation in drei sequentielle Rotationen um die Hauptachsen des kartesischen Koordinatensystems aufgeteilt. Die entsprechenden Winkel (θ_x, θ_y, θ_z) geben dabei den Rotationswinkel für die jeweilige Achse (x-, y- oder z-Achse) an. Eine Möglichkeit, Euler-Winkel umzusetzen, sind Rotationsmatrizen. Dabei existiert für jede Achse eine eigene Rotationsmatrix. Die Rotation erfolgt dabei durch die Multiplikation der Matrizen mit dem Ortsvektor des zu rotierenden Punktes. Das Ergebnis beschreibt dann den Ortsvektor des neuen rotierten Punktes. Zu beachten ist, dass die Rotation im Allgemeinen nicht kommutativ ist. Diese Rotationsmatrizen haben folgende Form:

$$X = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta_x & -\sin\theta_x \\ 0 & \sin\theta_x & \cos\theta_x \end{bmatrix} \tag{3.3}$$

$$Y = \begin{bmatrix} \cos\theta_y & 0 & \sin\theta_y \\ 0 & 1 & 0 \\ -\sin\theta_y & 0 & \cos\theta_y \end{bmatrix} \tag{3.4}$$

$$Z = \begin{bmatrix} \cos\theta_z & -\sin\theta_z & 0 \\ \sin\theta_z & \cos\theta_z & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.5}$$

Matrix (3.3) beschreibt dabei die Rotation um die x-Achse. Matrix (3.4) rotiert um die y- und Matrix (3.5) um die z-Achse. Zusätzlich unterscheidet man zwischen der Rotation bezüglich des Lokalen- oder des Weltkoordinatensystems, deren Ergebnisse im Allgemeinen nicht äquivalent sind. Beispielsweise existieren für die Rotationsmatrix $Z - Y - X$ zwei verschiedene Lösungen, je nachdem welches Bezugskoordinatensystem gewählt wurde[4]). Wählt man nun das Weltkoordinatensystem, so wird das

[4]Diese Rotationsmatrix gibt die Reihenfolge der Rotationen an und nicht die Reihenfolge der Matrixmultiplikationen.

Objekt nacheinander um die Achsen des globalen Koordinatensystems rotiert: zuerst um die z-Achse, dann um die y-Achse und zum Schluss um die x-Achse. Soll im Gegensatz dazu das lokale Koordinatensystem als Bezugssystem dienen, rotiert man zuerst um die lokale z-Achse, anschließend um die *neue y-Achse* und schlussendlich um die *neue x-Achse*.

3.3.1.2. Quaternionen-Rotationen

Quaternionen wurden 1843 vom irischen Mathematiker Sir William Rowan Hamilton entdeckt, als er nach einem Weg suchte, eine Divisionalgebra für Tripel von komplexen Zahlen zu finden. Heute werden Quaternionen auch verwendet, um Rotationen darzustellen[5].

Allgemein hat ein *Quaternion* folgende mathematisch Form[6]:

$$Q = w + \mathrm{i}x + \mathrm{j}y + \mathrm{k}z \qquad \text{mit } x, y, z \in \mathbb{R}$$

und für die Imaginärteile i, j, k gelten:

$$i^2 = j^2 = k^2 = ijk = -1.$$

Häufig wird auch nur ein 4-Tupel (w, x, y, z) mit den vier Koeffizienten als Darstellung für ein Quartenion verwendet. Eine neuere Schreibweise ist die Schreibweise als Summe eines reellen Skalars mit einem Vektor $\mathbf{v} \in \mathbb{R}^3$:

$$Q \mathrel{\widehat{=}} w + \mathbf{v} \mathrel{\widehat{=}} (w, \mathbf{v}).$$

Schreibt man diese Komponenten aus, so erhält man folgende Form:

$$w + \mathbf{v} = w + \hat{\mathbf{i}}x + \hat{\mathbf{j}}y + \hat{\mathbf{k}}z$$

mit

$$\begin{aligned}
\hat{\mathbf{i}} &\mathrel{\widehat{=}} 0 + 1i + 0j + 0k \\
\hat{\mathbf{j}} &\mathrel{\widehat{=}} 0 + 0i + 1j + 0k \\
\hat{\mathbf{i}} &\mathrel{\widehat{=}} 0 + 0i + 0j + 1k
\end{aligned}$$

Möchte man nun mit Hilfe von Quaternionen Rotationen durchführen, so wird der reelle Anteil w des Quaternions Q als der Drehwinkel interpretiert, während die drei imaginären Anteile des Vektors v die Ausrichtung der Rotationsachse bestimmen. Die Rotation R_q mit Hilfe von Quaternionen erfolgt dann mit:

$$R_q(\mathbf{P}) = \mathbf{q_e p q_e^{-1}} \ .$$

[5] Die Ausführungen über Quaternionen-Rotationen sind angelehnt an das zweite Kapitel des Buches *Methoden der Computeranimation* [Jackel u. a. 2006].

[6] Eine Zusammenfassung der wichtigsten Rechenregeln für Quaternionen findet sich in Anhang A.

Hierbei bezeichnet $\mathbf{q_e}$ ein Einheitsquaternion:

$$\begin{aligned}\mathbf{q_e} &= (\cos\alpha, \mathbf{n}\sin\alpha) \\ &= (\cos\alpha, n_x\sin\alpha, n_y\sin\alpha, n_z\sin\alpha) \qquad \text{mit } |\mathbf{n}| = 1.\end{aligned} \tag{3.6}$$

$\mathbf{q_e}$ repräsentiert dabei den Drehwinkel, sowie die Ausrichtung des Drehachsenvektors. Die Inverse $\mathbf{q_e^{-1}}$ eines Einheitsquaternions $\mathbf{q_e}$ ist äquivalent zu seinem komplementären Quaternion:

$$\mathbf{q_e^{-1}} = (s, -\mathbf{v}) = \overline{\mathbf{q_e}}.$$

$$\mathbf{p} = (0, \mathbf{r}) \tag{3.7}$$

ist die Darstellung des Punktes $\mathbf{P}$ als Quaternion, dessen Skalaranteil Null ist und dessen Imaginärteil die Koordinaten von $\mathbf{P}$ sind. Führt man nun die Multiplikation dieser drei Quaternionen nach den im Anhang vorgestellten Regeln durch, so ergibt sich:

$$R_q(\mathbf{P}) = (0, (s^2 - \mathbf{v^2})\mathbf{r} + 2\mathbf{v}(\mathbf{v}\cdot\mathbf{r}) + 2s\mathbf{v}\times\mathbf{r}).$$

Mit (3.6) und (3.7) erhält man dann folgende allgemeine Formel für die Rotation eines Punktes mit Hilfe von Quaternionen:

$$\begin{aligned}R_q(\mathbf{P}) &= (0, \mathbf{r}(\cos^2\alpha - \sin^2\alpha) + 2\mathbf{n}(\mathbf{n}\cdot\mathbf{r})\sin^2\alpha + 2\cos\alpha\sin\alpha(\mathbf{n}\times\mathbf{r})) \\ &= (0, \mathbf{r}\cos 2\alpha + (1-\cos 2\alpha)\mathbf{n}(\mathbf{n}\cdot\mathbf{r}) + \sin 2\alpha(\mathbf{n}\times\mathbf{r})).\end{aligned}$$

Dieses Ergebnis lässt sich wie folgt interpretieren: Wird ein Quaternion $\mathbf{p} = (0, \mathbf{r})$ von links mit dem Einheitsquaternion $\mathbf{q_e}$ und von rechts mit dem Inversen des Einheitsquaternionen $\mathbf{q_e^{-1}}$ multipliziert, so erhält man die Drehung eines durch den Vektor $\mathbf{r}$ bestimmten Punktes $\mathbf{P}$ um die Achse $\mathbf{n}$ mit dem Winkel 2α (Abbildung 3.8 skizziert diese Rotation).

Daraus ergibt sich folgende allgemeine Gleichung für die Rotation eines beliebigen Punktes $\mathbf{P}$ um den Winkel α:

$$\overbrace{(0, \begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{z}\end{bmatrix})}^{R_q(\mathbf{p})} = \overbrace{(\cos\frac{\alpha}{2}, \begin{bmatrix}n_x\sin\frac{\alpha}{2}\\ n_y\sin\frac{\alpha}{2}\\ n_z\sin\frac{\alpha}{2}\end{bmatrix})}^{q_e=(s,\mathbf{v})} \overbrace{(0, \begin{bmatrix}x\\ y\\ z\end{bmatrix})}^{p=(0,\mathbf{r})} \overbrace{(\cos\frac{\alpha}{2}, \begin{bmatrix}-n_x\sin\frac{\alpha}{2}\\ -n_y\sin\frac{\alpha}{2}\\ -n_z\sin\frac{\alpha}{2}\end{bmatrix})}^{q_e^{-1}=(s,-\mathbf{v})}.$$

3.3.2. Interpolation mit SLERP

In dieser Arbeit wurde SLERP (Spherical Linear Interpolation) zur automatischen Berechnung neuer Zwischenposen verwendet. Ken Shoemake stellte SLERP erstmals 1985 in seinem Artikel *Animating Rotation with Quaternion Curves* vor [Shoemake 1985]. Sie ist eine auf Quaternionen basierende Interpolationstechnik, die die Interpolation auf einem Bogen auf der Einheitssphäre beschreibt und dadurch eine

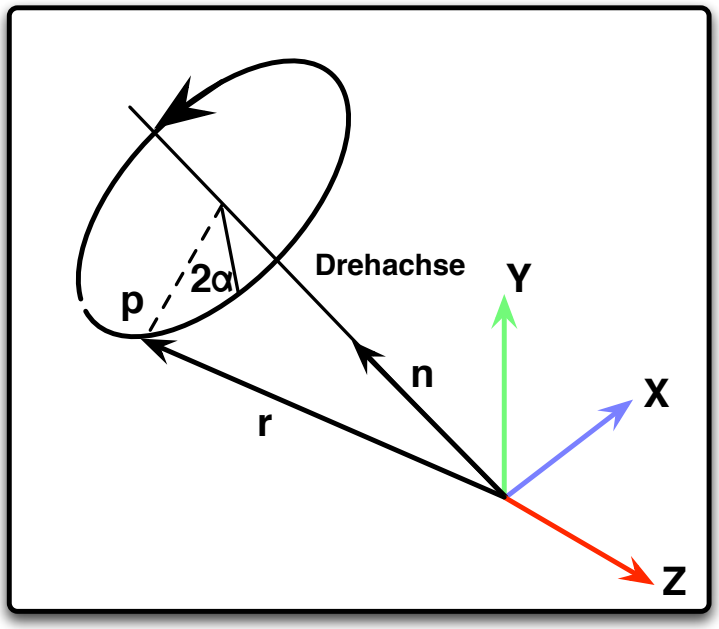


Abbildung 3.8.: Bild skizziert die Rotation mit Quaterionen. Punkt P wird um 2α rotiert.

gleichmäßige Beschleunigung erreicht [Dam u. a. 1998]. Im Gegensatz dazu interpoliert die lineare Interpolation LERP (Linear Interpolation) nicht auf dem Bogen der Einheitssphäre, sondern auf einer direkten Linie zwischen den beiden Schlüsselposen. Dadurch ist bei einer Animation der Schlüsselposen eine Beschleunigung zu erkennen (siehe Abbildung 3.9). Die Gleichung, um die sphährische Interpolation mit SLERP zu realisieren, hat folgende Form:

$$Slerp\ (q_1, q_2, t) = \frac{\sin(1-t)\ \alpha}{\sin\alpha}\ q_1 + \frac{\sin t\ \alpha}{\sin\alpha}\ q_2 \qquad \text{mit } t \in [0,1], \tag{3.8}$$

Dabei wird von Quaternion q_1 nach Quaternion q_2 interpoliert, während die Variable t die Schrittweite der Interpolation angibt (Abbildung 3.10 zeigt den Verlauf und die Beschleunigung der SLERP-Interpolation).

Wie in Abbildung B.1 im Anhang zu sehen ist, weist SLERP bei Interpolation mit mehreren Key-Frames keine weichen (*smoothen*) Kurven auf. An den Schlüsselpositionen entstehen ruckartige Richtungsänderungen, aufgrund der ruckartigen Wechsel zwischen den Rotationsachsen. Ebenso entsteht dann eine unterschiedliche Beschleunigung und somit ein Verlust der konstanten Bewegungsgeschwindigkeit[7].

Eine weitere Möglichkeit wäre die Interpolation mit Hilfe der Positionen des End-Effektors. Dabei werden die Punkte, die auf der Gerade zwischen zwei Positionen

[7] In ihrem Artikel *Quaternions, Interpolation and Animation* stellen Dam et al. Techniken vor, wie diese Nachteile behoben werden können [Dam u. a. 1998].

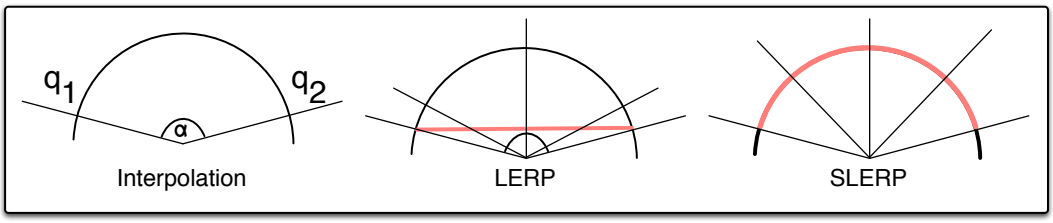


Abbildung 3.9.: Illustration der Interpolation von LERP und SLERP. Das linke Bild zeigt den Winkel α zwischen Anfangsquaternion q_1 und Endquaternion q_2. Das mittlere Bild zeigt die Interpolation mit LERP. Dabei wird auf der geraden zwischen Anfangs- und Endquaternion (rote Linie) linear interpoliert, d.h. die Gerade wird in gleich große Intervalle aufgeteilt. Der Winkel zwischen den einzelnen Segmenten ergibt dann den zu rotierenden Winkel. Zu erkennen sind die unterschiedlich großen Zwischenwinkel. Im Gegensatz dazu interpoliert SLERP auf dem Kreisbogen, so dass die Zwischenwinkel alle gleich groß sind (siehe rechtes Bild).

des End-Effektors liegen, interpoliert. Anschließend wird die Pose mit Hilfe des IK-Algorithmus generiert, indem der End-Effektor iterativ die interpolierten Positionen durchläuft. Da beim IK-Algorithmus jedoch unerwartete bzw. unnatürliche Posen entstehen können, ist das Interpolationsergebnis von den Ergebnissen des IK-Algorithmus abhängig. Sind keine Beschränkungen der Gelenke implementiert, so kann es eventuell zu vielen unnatürlichen Zwischenergebnissen kommen (siehe Abschnitt 3.2.2.2).

3.4. Dateiformat zum Austausch von 3D-Daten

Zum Austausch von 3D-Daten existieren zahlreiche Formate unterschiedlicher Firmen und Arbeitsgruppen. In diesen Formaten werden Informationen zur Darstellung einer Szene gespeichert. Dazu gehören neben den Geometriedaten auch Informationen zur Szenendarstellung wie Kamera- oder Lichteinstellungen sowie Daten zur Animation der Objekte.

Um dem Benutzer von ANVIL die Möglichkeit zu geben, sein eigenes spezifisches Skelett definieren und verwenden zu können, soll ANVIL in der Lage sein, eine externe Datei mit einer neuen Skelett-Topologie einzulesen. Um die Wahl des dazu benötigten 3D-Grafikprogramms offen zu lassen, sollte es ein Standard sein, der von vielen gängigen 3D-Grafikprogrammen unterstützt wird. Denn ohne Zuhilfenahme von 3D-Grafikprogrammen ist die Erstellung von komplexen Skelett-Topologien recht schwierig. In dieser Arbeit wurde COLLADA verwendet, weil es ein offener und gut dokumentierter Standard ist, der auf XML basiert. Zudem ist COLLADA flexibel,

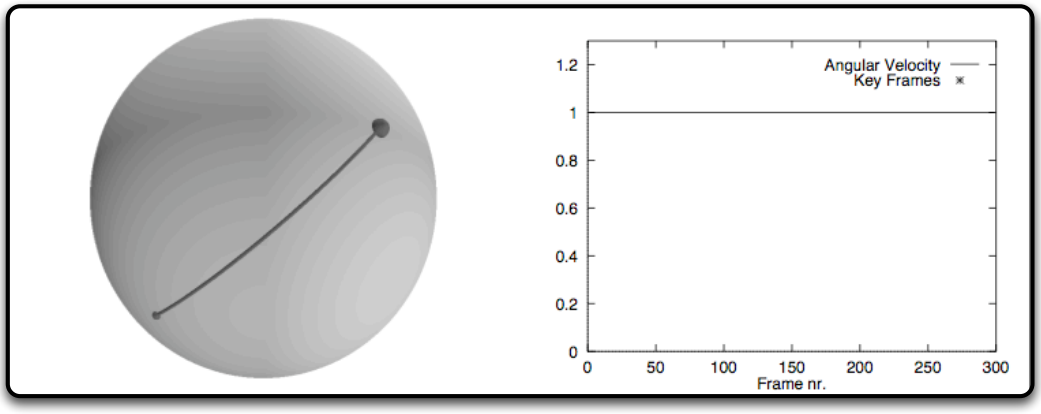


Abbildung 3.10.: Illustration der Interpolation und Beschleunigung von SLERP. Die Interpolation erfolgt auf dem Bogen der Einheitssphäre. Rechts ist die gleichmäßige Beschleunigung in der Tabelle zu erkennen (entnommen aus [Dam u. a. 1998]).

da es ein allgemeiner Standard für 3D-Daten ist, und sich nicht nur auf bestimmte Aspekte, wie Motion-Capture-Daten, konzentriert. Nachdem einige Alternativen vorgestellt werden, wird im anschließenden Abschnitt der Einsatz von COLLADA beschrieben.

3.4.1. Alternative Formate

Ein bekanntes Format zur Speicherung von 3D-Daten ist beispielsweise FBX der Firma Autodesk[8]. Der Austausch des Formats mit den unterschiedlichen Programmen erfolgt dabei über ein *FBX-Plugin*. Ein Vorteil von FBX liegt in der hohen Kompatibilität zwischen Maya und 3dsMax, den bekanntesten 3D-Programmen, da beide gleichzeitig auch von Autodesk entwickelt werden. Mittlerweile verwenden auch andere Firmen wie Microsoft oder Softimage dieses Format. Mit einem FBX-Konverter ist es auch möglich, andere Formate, wie auch COLLADA (.dae), in das FBX Format zu konvertieren. Jedoch ist FBX kein offener und nicht dokumentierter Standard.

Ein Alternative ist der offene Standard *VRML (Virtual Reality Markup Language)*[9], der mit Hilfe von speziellen Viewern oder Plugins für Internetbrowser, die Darstellung von 3D-Inhalten in Echtzeit erlaubt, so dass Anwender sogar interaktiv in der Szene agieren können. Mittlerweile wurde VRML vom ISO-Standard X3D[10] abgelöst.

Eine Möglichkeit, hierarchisch strukturierte Skelette zu speichern bietet *BVH (Biovision Hierarchical Data)*. BVH wurde von Biovision, eine auf Motion-Caption spezialisierte Firma, entwickelt, um Motion-Capture-Daten zu speichern. Im Gegensatz

[8] http://www.autodesk.de
[9] http://www.web3d.org/x3d/specifications/vrml/ISO-IEC-14772-VRML97/
[10] http://www.web3d.org/x3d

zu COLLADA speichert BVH die Daten nicht in einer XML-Struktur, sondern in einer Ascii-Textdatei ab. Dies erschwert das Einlesen und die maschinelle Handhabung im Vergleich zu XML, da es für die meisten Programmiersprachen (Java, C++, Perl usw.) bereits Parser und Bibliotheken für den Umgang mit XML gibt. Ähnlich wie bei BVH ist auch ASF (Acclaim Skeleton File)/AMC(Acclaim Motion Capture data) des Spieleherstellers Acclaim[11] ein Ascii-Dateiformat zur Speicherung von Motion-Capture-Daten. Dieser Standard wurde von Acclaim entwickelt, um die hauseigenen Motion-Caputure-Daten zu speichern. Im Gegensatz zu BVH wurde die Skelett-Topologie in eine separate Datei (ASF) ausgelagert, so dass die AMC-Datei nur die Motion-Caputure-Daten beinhaltet. Dies erlaubt die Wiederverwendung eines Skeletts für mehrere Motion-Capture-Daten.

In dieser Arbeit wurde jedoch COLLADA verwendet, da COLLADA ein offener und gut dokumentierter Standard ist, der auf XML basiert.

3.4.2. Einsatz von COLLADA

COLLADA (COLLAborative Design Activity)[12] ist ein offener und auf XML-basierter Standard zum Austausch von 3D-Daten zwischen 3D-Applikationen. Dieser Standard wurde von der Khronos-Gruppe[13] spezifiziert, um einen einheitlichen Standard zu schaffen, der von vielen Anwendungen und Plattformen unterstützt wird. Mittlerweile wird COLLADA auch von vielen Firmen wie Sony, NVIDIA, ATI oder Google unterstützt[14]. In COLLADA können komplette Szenen inklusive Texturen, Kamera- und Lichteinstellungen oder Animationen abgebildet werden. Seit Version 1.5 ist es auch möglich, kinematische Informationen wie kinematische Ketten und Skelett-Topologien zu speichern.

COLLADA wurde in dieser Arbeit eingesetzt, um eigene Skelett-Topologien und die berechneten Animationen zu speichern. Da die Szene auch in COLLADA hierarchisch strukturiert ist, wird das Skelett als hierarchischer Unterbaum in die Szene eingebettet (siehe Abschnitt C). Zusätzlich können kinematische Ketten und die Benennung der dazugehörigen Gelenke, die für den IK-Algorithmus benötig werden, ebenfalls in COLLADA abgespeichert werden. Diese finden sich unter der Bibliothek für kinematische Ketten. Diese Definition ist jedoch optional, da kinematische Ketten erst ab Version 1.5 unterstützt werden und für die meisten 3D-Programme noch keine Export-Plugins für diese Version existieren. Dafür stellt ANVIL die Möglichkeit bereit, nachträglich in einem Editor unter ANVIL, die kinematischen Ketten zu definieren und die Gelenke zu benennen. Um die erstellten Posen und Gesten in einem 3D-Grafikprogramm wiederzuverwenden, sollen die Animationen ebenfalls in COLLADA exportiert werden können. Diese Animationen werden unter `<animation>` gespeichert).

[11] http://www.acclaim.com/
[12] http://www.collada.org
[13] http://www.khronos.org/
[14] Eine Liste mit den unterstützenden Firmen und deren Produkte findet man unter https://collada.org/mediawiki/index.php/Portal:Products_directory.

Ausreichend für diese Arbeit sind folgende „Bibliotheken“:

- `<asset>`
- `<library_geometries>`
- `<library_visual_scenes>`
- `<library_kinematics_models>`
- `<library_animations>`

Eine kurze Zusammenfassung dieser Bibliotheken findet sich in Anhang C.

Fazit: Die hier dargestellten Inhalte dienen als Grundlage für die Umsetzung der in dieser Arbeit gesetzten Ziele. Dazu gehören Aufbau mit H-Anim und Steuerungsmöglichkeiten eines Skeletts mit IK und FK. Ebenfalls wurde der Interpolationsalgorithmus SLERP vorgestellt, der für die Interpolation zwischen den Key-Posen benötigt wird. Diese berechneten Zwischenposen sind notwendig, um eine Animation der Geste zu erhalten. Bevor allerdings auf die Realisierung skelettbasierter Gestenannotation eingegangen wird, soll im nun folgenden Kapitel das Benutzerinterface vorgestellt werden, das eine intuitive Bedienung ermöglichen soll.

4. Design der grafischen Benutzerschnittstelle

Folgendes Kapitel gibt einen Überblick über die im Rahmen dieser Arbeit erstellte Applikation. Ziel dieser Applikation ist die Schaffung eines inutitiven Benutzerinterfaces für die Annotation von Gesten. Dieses Interface soll mit Hilfe eines manuell manipulierbaren 3D-Skeletts realisiert werden. Die Manipulation des Skeletts erfolgt dabei über Inverse und Forward Kinematics. Das Ergebnis soll dann animiert und zur Weiterverwendung in anderen 3D-Applikationen in das COLLADA-Format exportiert werden können. Umgesetzt wurde diese Applikation in Java (Version 6) und Java3D (Version 1.5) als fester Bestandteil von ANVIL. Dabei gliedert sich die Implementierung in vier Hauptaufgaben:

1. die Modellierung des Skeletts mit Java und Java3D.
2. die Implementierung der Manipulations- und Interpolationsalgorithmen.
3. die grafische Benutzeroberfläche.
4. die Einbindung in ANVIL.

4.1. Überblick und Workflow

Dieser Abschnitt skizziert zunächst die allgemeinen Anforderungen an das System und beschreibt anschließend den Workflow zur Erstellung neuer Posen.

4.1.1. Allgemeine Eigenschaften des Systems

Die hier vorgestellte Erweiterung wurde in Java implementiert, um dieses Tool in den vorhandenen Code eingebetten zu können, da auch ANVIL bereits in Java programmiert wurde. Dabei wurde auf Java in der Version 6 gesetzt. Zur Darstellung der Szene und des Skeletts wurde die Grafikbibliothek *Java3D* verwendet. Java3D ist eine Grafikbibliothek von Sun, mit der dreidimensionale Szenen in Java programmiert werden können (siehe Abschnitt 3.1). Die Gründe der Entscheidung für Java3D liegen zum einen in der sehr guten API, zum anderen in der problemlosen Integration in eine Java Umgebung.

Die Hauptaufgabe des Tools besteht darin, eine visuelle und intuitive Annotation von Gesten und Posen unter ANVIL zu ermöglichen. Die Visualisierung einer Pose

erfolgt über ein 3D-Skelett, das über inverse Kinematik verändert werden kann. Zusätzliche Hilfsmittel sollen den Aufwand bei dieser skelettbasierten Annotation von Gesten in vertretbarem Rahmen halten.

Um ein gutes Ergebnis bei der Nachbildung einer Pose zu erhalten, bekommt der Benutzer eine Vorlage der Originalpose in Form eines Screenshots geliefert. Dieser Screenshot wird aus dem aktuellen Videoframe mit der Originalpose entnommen und kann direkt hinter das Skelett platziert werden. Die Position und Größe des Sprechers auf dem Video können von Frame zu Frame stark variieren. Gründe dafür sind wechselnde Kameraperspektiven und Zoomeinstellungen der Aufzeichnungen. Daher soll der Benutzer die Möglichkeit haben, das Hintergrundbild frei zu bewegen und zu skalieren. Um diesen Prozess zu beschleunigen, soll dies automatisch durch Markierung der Schultern auf dem Screenshot geschehen. Zusätzlich soll der Benutzer die volle Kontrolle über die Betrachtung auf die Szene erhalten. Dies wird über eine freie Kontrolle der Szenenkamera ermöglicht.

Die Erstellung neuer Posen am 3D-Skelett durch inverse Kinematik ermöglicht dem Kodierer, nur noch den End-Effektor (z.B. die Hand) an die gewünschte Position bewegen zu müssen. Auf diese Weise können einfach und durch visuelle Kontrolle präzise Schlüsselposen annotiert werden. Zwischen diesen Schlüsselposen wird durch geeignete Interpolation die Geste vervollständigt. Diese Key-Frame-Animation erspart dem Kodierer die aufwändige Arbeit, jeden Frame der Sequenz zu annotieren, um eine Geste zu erstellen.

Um eine hohe Flexibilität bezüglich der Skelett-Topologie zu gewährleisten, ist es möglich eine COLLADA-Datei mit der Definition eines Skeletts einzulesen. Ausserdem kann der Benutzer mit dieser Applikation eigene kinematische Ketten definieren und auch abspeichern.

4.1.2. Workflow

Folgender Abschnitt soll den Workflow zur Erstellung und Modifikation von Posen beschreiben. Posen werden über den *Pose Sequence Editor* erstellt oder angepasst. Dieser wird über den *Attribute Editor* aufgerufen, da Posen in ANVIL als Attribute eines Tracks repräsentiert werden. Zum *Attribute Editor* gelangt man durch Hinzufügen eines neuen Elements (Angabe einer neuen Start- und Endzeit) oder durch Editieren eines vorhandenen Elements im *Annotation Board* von ANVIL . Im *Pose Sequence Editor* wird dann der gewünschte Frame ausgewählt, während die Pose selbst im *Pose Editor* erstellt wird. Über den *Pose Viewer* kann der Benutzer die Posenerstellung zu jeder Zeit von allen Winkeln betrachten. In diesem Fenster kann auch die Animation der Geste verfolgt werden. Eine weitere Komponente der Benutzerschnittstelle ist der *Skeleton Markup Editor* (siehe Abbildung D.9). Dieser wird automatisch vor der Annotation der ersten Pose aufgerufen, wenn keine kinematischen Ketten in der Skelettbeschreibung definiert wurden. Dieser Workflow wird in Abbildung 4.1 illustriert.

1. Um Posen in ANVIL annotieren zu können, muss mindestens ein Track in der

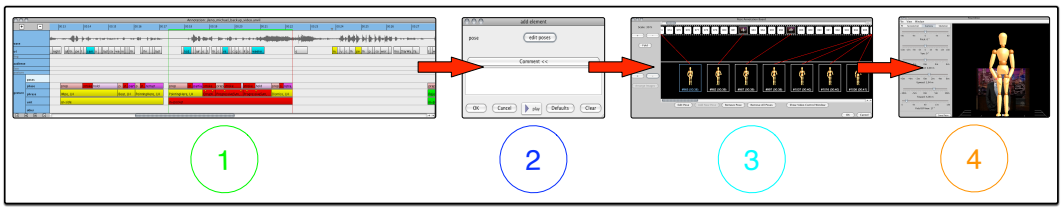


Abbildung 4.1.: Workflow zur Erstellung und Modifizierung von Posen. **Schritt 1:** ANVIL Annotation Board, **Schritt 2:** Attribute Editor, **Schritt 3:** Pose Sequence Editor, **Schritt 4:** Pose Editor)

Spezifikationsdatei das Attribut *TimestampedPose* (siehe Abschnitt 5.3) enthalten. Mit diesem Attributtyp wird die Annotation von Posen gekennzeichnet. Dazu muss im *Annotation Board* von ANVIL ein neues Element angelegt oder ein vorhandenes editiert werden.

2 Der `Attribute Editor` wird automatisch aufgerufen, sobald die Start- und Endzeit eines neuen Tracks angegeben oder die Editierfunktion eines vorhandenen Elements aufgerufen wurde. Über den Button *edit poses* gelangt man dann zum `Pose Sequence Editor`.

3 Der `Pose Sequence Editor` dient zur Übersicht der erstellten Posen und erlaubt die Steuerung des Videos. Von hier aus können vorhandene Posen gelöscht bzw. verändert oder neue Posen hinzugefügt werden. Zusätzlich können von hier aus die annotierten Gesten ständig mit der Originalvorlage abgeglichen werden. Dazu dienen die Screenshots von den Posen und des Videoframes sowie die Skelettanimation. Der `Pose Viewer` mit seinen drei verschiedenen Ansichten auf die Szene dient ebenfalls zur Kontrolle der Animation. Die Manipulation des Skeletts selbst, geschieht im `Pose Editor`.

4 Der `Pose Editor` dient zur Anpassung alter und zur Erstellung neuer Posen. Über IK kann der Benutzer das Skelett manipulieren und ausrichten. Hilfsfunktionen, wie die automatische Ausrichtung und Skalierung des Hintergrundbildes, dienen zur Unterstützung bei der Nachbildung der Posen. Zusätzlich kann der Benutzer die Szenenkamera frei verändern, um eine bessere Perspektive zu erhalten. Nach Fertigstellung einer Pose kehrt man zum `Pose Sequence Editor` zurück.

4.2. Grafische Benutzeroberfläche

Die komplette grafische Benutzeroberfläche wurde mit Swing umgesetzt und fest in ANVIL integriert. Die Funktionalität dieser Erweiterung wurde in vier Komponenten unterteilt:

1. *Pose Sequence Editor*: Kontrolle und Übersicht des Videos zur Posenannotation.
2. *Pose Editor*: Manipulation des Skelettes zur Erstellung neuer Posen.
3. *Pose Viewer*: Kontrolle der annotierten Geste und Animation.
4. *Skeleton Markup Editor*: Bearbeitung und Anpassung importierter Skelett-Topologien.

Im Folgenden werden diese Komponenten vorgestellt. Hier soll nur ein kleiner Überblick über die Komponenten gegeben werden. Eine genauere Übersicht der Benutzeroberfläche und deren Bedienung findet sich im Anhang D.

4.2.1. Pose Sequence Editor

Der *Pose Sequence Editor* (siehe Abbildung D.3) ist Ausgangspunkt jeder neuen Posen- bzw. Gestennannotation, da über ihn entweder neue Posen erstellt oder bestehende editiert werden können. Zusätzlich dient er zur Kontrolle des Videos und gibt eine Übersicht über die bereits annotierten Posen. Er erlaubt entweder über einen Schieberegler oder über die Frameleiste frei im Video zu springen. In der unteren Leiste werden Screenshots der bereits erstellten Posen zur besseren Übersicht abgelegt. Durch Wegblenden von nicht-annotierten Frames in der oberen Hälfte, kann direkt zwischen der Originalpose und der konstruierten Pose verglichen werden. Die Originalposen werden dazu als Screenshot vom Video direkt über dem Bild der Pose platziert.

4.2.2. Pose Editor

Zum Erstellen neuer Posen wird der *Pose Editor* verwendet (die verschiedenen Panels sind in den Abbildungen in Anhang D zu finden). Neben der Möglichkeit das Skelett über IK oder FK zu manipulieren, bietet der Editor auch die Möglichkeit das Hintergrundbild und die Kamera anzupassen. Dadurch kann der Benutzer Anpassungen an der Szene so vornehmen, dass er eine optimale Vorlage erhält, um Posen nachzustellen. Hierbei hat der Benutzer die Wahl zwischen einer Figur und einer Skelett-Ansicht. Die Figurenansicht ist jedoch nur möglich, wenn das Skelett auch Meshdaten enthält. Wird das Skelett nur über eine Definition einer Skelett-Topologie aufgebaut, so ist nur die Skelettansicht möglich, d.h. ein Skelett bestehend nur aus Kugeln und Quadern (Abbildung 4.2 veranschaulicht die verschiedenen Ansichten).

4.2.3. Pose Viewer

Der so genannte *Pose Viewer* (Abbildung D.8) dient unter anderem als zusätzliche Kontrolle bei der Erstellung von neuen Posen. Denn durch die zusätzlichen drei Ansichten auf das Skeletts, die frei verstellbar sind, erhält der Benutzer eine noch bessere Möglichkeit die Pose während der Erstellung zu kontrollieren. Zusätzlich wird

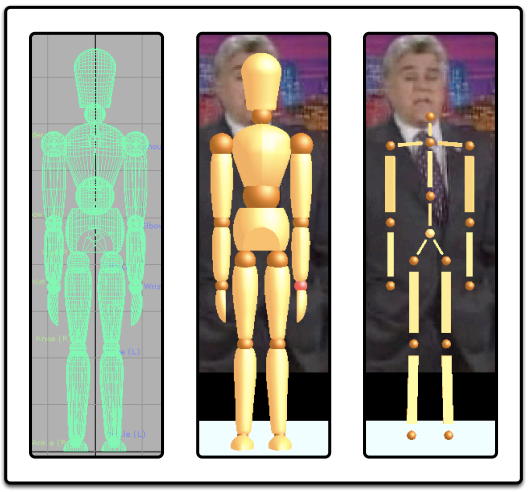


Abbildung 4.2.: Links ist die Figur-Ansicht des Skeletts in Maya zu sehen, während die mittlere Abbildung die Figue-Ansicht unter ANVIL zeigt. Auf dem rechten Bild ist das Skelett aus den Java3D Primitiven *Box* und *Sphere* aufgebaut.

auch die Animation des Skeletts im *Pose Viewer* angezeigt. Diese Animation dient zum Abgleich der kompletten Geste mit dem Originalvideo.

4.2.4. Skeleton Markup Editor

Mit dem *Skeleton Markup Editor* werden kinematische Ketten definiert, falls die geladene COLLADA-Datei keine Ketten spezifiziert (Abbildung D.9). Neben der Definition der beiden Arme, die diese Erweiterung als Minimum vorschreibt, können auch noch weitere kinematische Ketten über die Angabe der Wurzel und des End-Effektors festgelegt werden.

Fazit: Die hier dargestellten Elemente des Interfaces sollen dem Benutzer eine intuitive Bedienung und ein schnelles Annotieren ermöglichen. Der *Pose Sequence Editor* soll die Navigation im Video während dem Annotieren erleichtern und dem Benutzer einen Überblick über die annotierten Posen verschaffen. Manipulationen des Skeletts finden im *Pose Editor* statt. Dieser bietet Hilfsmittel wie die Platzierung eines Screenshots hinter dem Skelett oder die freie Kameraeinstellung, die den Arbeitsprozess optimieren sollen. Mit Hilfe des *Pose Viewers* kann die Pose und Animation aus allen Perspektiven betrachtet werden. Um kinematische Ketten zu definieren, steht der *Skeleton Markup Editor* zur Verfügung. Wie die eigentliche Implementierung des 3D-Skeletts und die anschließende Integration in ANVIL umgesetzt wurden, soll im nachfolgenden Kapitel erörtert werden.

5. Implementierung und Integration

5.1. Modellierung des Skeletts in Java3D

Die Hauptklasse zur Abbildung von allgemeinen Skeletten ist die abstrakte Klasse `Skeleton`, die alle möglichen Skelette repräsentieren soll. Mit dieser Generalisierung soll es dann prinzipiell möglich sein, auch nicht-menschliche Skelette zu implementieren. Von dieser abstrakten Klasse wird `Humanoid` abgeleitet, welches allgemein ein menschenähnliches Skelett darstellen soll (Abbildung 5.1 zeigt das Klassendiagramm eines Skeletts). Mit der Einschränkung auf ein menschliches Skelett, soll sicherge-

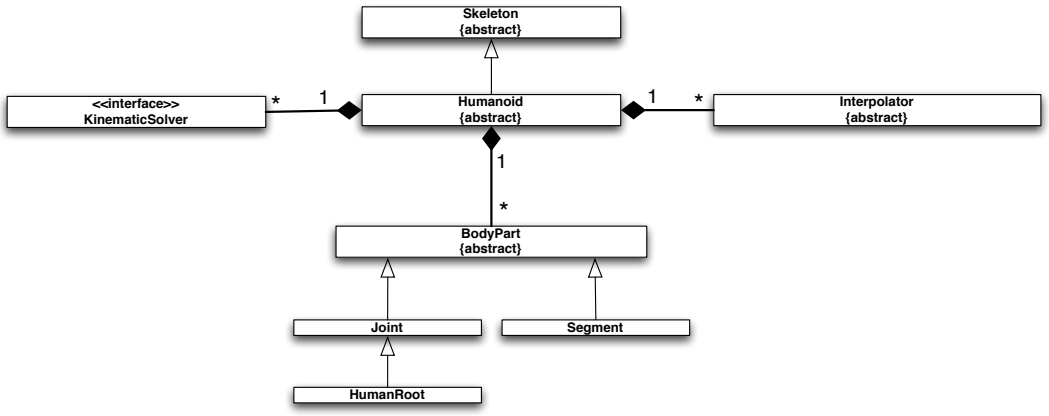


Abbildung 5.1.: Klassendiagramm und Abhängigkeiten eines Skeletts.

stellt werden, dass von einer Grundstruktur des Skeletts ausgegangen werden kann. Zum Beispiel besitzt ein menschliches Skelett mindestens zwei Arme, die wiederum aus mindestens dem Schulter-, Ellbogen- und Handgelenk aufgebaut sind. Zusätzlich stellt `Humanoid` alle Methoden zur Verfügung, um das Skelett zu steuern oder zu manipulieren. Dazu müssen spezielle Klassen geschrieben werden, die entweder die Schnittstelle `IKSolver` oder `FKSolver` implementieren (siehe Abschnitt 5.2). Beide Schnittstellen sind wiederum vom Typ `KinematicSolver`. Zur Berechnung der Interpolation beinhaltet `Skeleton` mindestens einen `Interpolator` (siehe Abschnitt 5.2.2).

`BodyPart` ist eine abstrakte Klasse zur Darstellung von Gelenken (`Joint`) und Knochen (`Segment`). Die hierarchische Struktur des Skeletts ergibt sich aus dem rekursiven Aufbau der Einzelelemente. Zur Veranschaulichung zeigt Abbildung 5.2 dies am Beispiel des linken Arms. Hier ist zu sehen, dass ein Arm aus den Klassen `Joint`

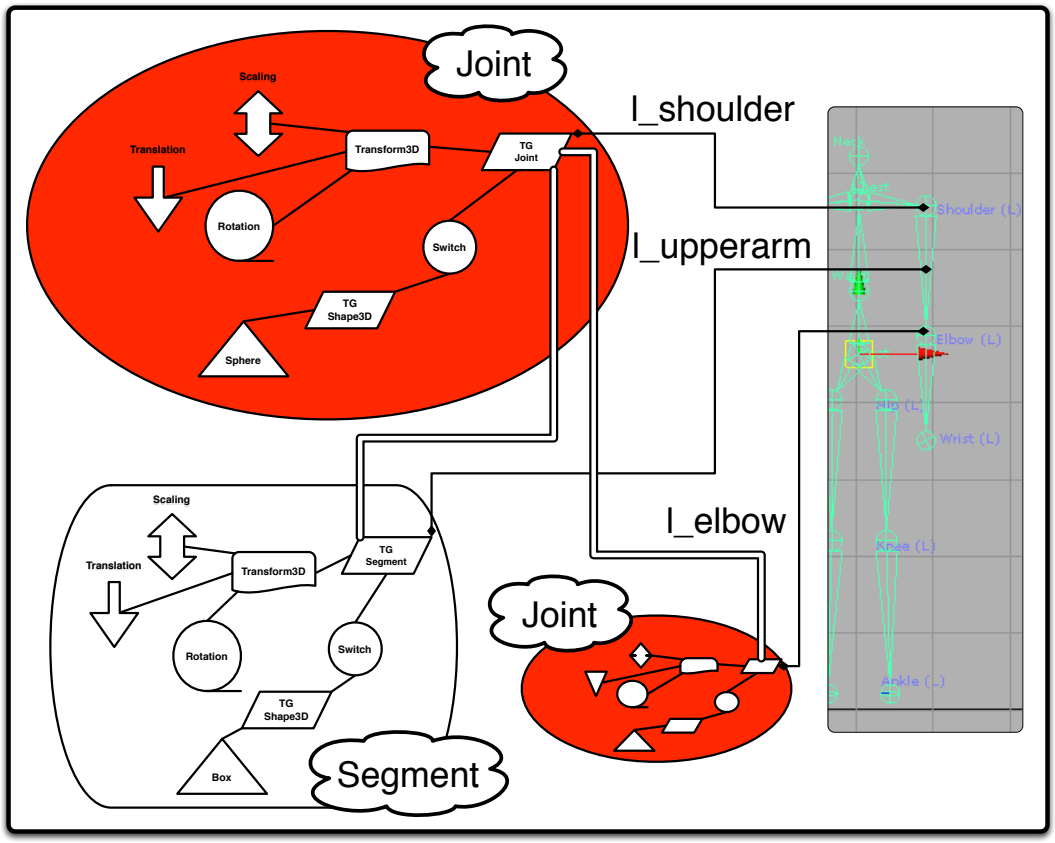


Abbildung 5.2.: Umsetzung eines Armes in eine hierarchische Struktur in Java3D.

und `Segment` besteht. Um nun die hierarchische Skelettstruktur zu erhalten, besitzt jeder `BodyPart` Verweise zu seinen direkten Kindern. Dazu erhält jeder `BodyPart` eine neue `TransformGroup` (siehe Abschnitt 3.1), um Transformationen wie Translation, Skalierung und Rotation ausführen zu können. Auf diese `TransformGroup` werden dann die initialen Transformationen angewendet und die Kinderknoten (Gelenke und Knochen) angehängt. Im abgebildeten Beispiel wären dies die Kinderknoten: „linker Oberarm“ (*l_ upperarm*) und „linker Ellbogen“ (*l_ ellbow*). Die hierarchische Struktur ergibt sich aus den hierarchischen Beziehungen zwischen den `TransformGroups` der Eltern- und der Kinderknoten (siehe Abbildung 5.3). `HumanRoot` ist die Wurzel des Skeletts. Sollte das Skelett später verschoben werden können, so werden die Translationen auf `HumanRoot` ausgeführt. Abbildung 5.3 zeigt, wie der hierarchische Aufbau des Standardskeletts dieser Arbeit, in Java3D aufgebaut wurde.

Durch diese hierarchische Struktur kann das in Abschnitt 3.2.1 vorgestellte Konzept, eines hierarchisch-strukturierten Skelettes, umgesetzt werden. Diese hierarchische Struktur erlaubt eine hierarchische Kontrolle, was das Hauptkonzept des Szenegraphen unter Java3D ausnutzt. Dieses Konzept bewirkt, dass die Transformationen (Translation, Skalierung und Rotation) auf einen Knoten des Szenegraphen

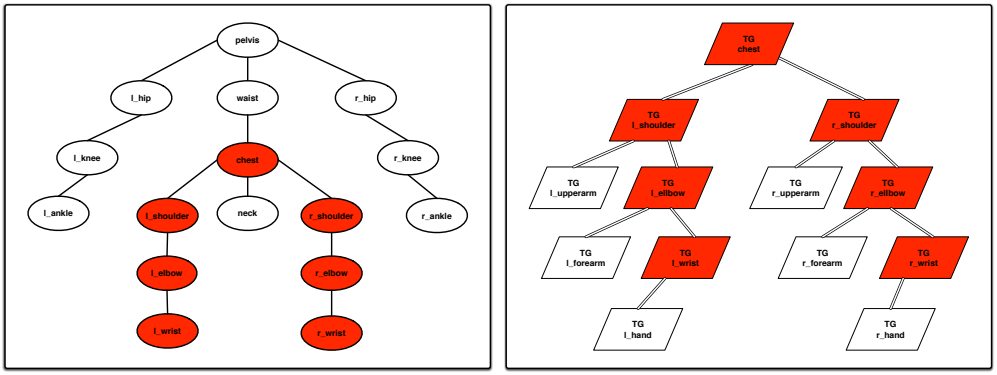


Abbildung 5.3.: Links: Hierarchische Struktur der Gelenke des verwendeten Skeletts nach H-Anim. Rechts: Äquivalente Abbildung des Skeletts als hierarchische Java3D Struktur.

sich immer relativ zur Ausrichtung der Elternknoten beziehen [Selman 2002]. Umgekehrt bedeutet dies, dass jede Veränderung eines Elternknotens sich rekursiv auf alle seine Kinderknoten auswirkt, so dass ihre relative Position bezüglich des Elternknotens nicht verändert wird. Lässt man zum Beispiel die Schulter eines Arms rotieren, bewegen sich zwar alle Kinderknoten (Oberarm, Ellbogen usw.) mit, so dass sich deren absolute Position bezüglich der Weltkoordinaten verändert, die relative Ausrichtung und Position zu ihrem direkten Elternknoten bleiben jedoch erhalten. Abbildung 5.4 veranschaulicht dies am Beispiel der linken Schulter. Obwohl nur auf die `TransformGroup` der linken Schulter eine Rotation durchgeführt wurde, bewegen sich alle anderen Elemente mit, der Arm bleibt aber in seiner „Form“ erhalten.

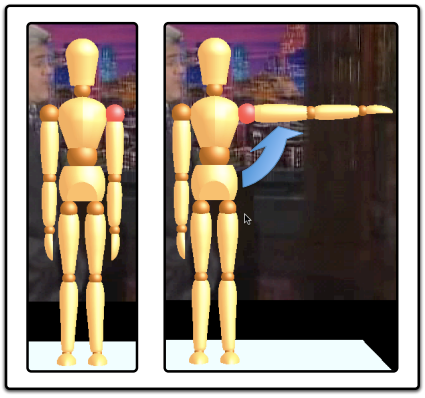


Abbildung 5.4.: Das linke Bild zeigt die Figur in der Ausgangsstellung. Auf dem rechten Bild wurde die Schulter um 90° in der Z-Achse rotiert. Dabei verbleiben alle Elemente unterhalb der Schulter in der relativen Position zu ihren Elternknoten. Dadurch verändert sich die „Form“ des Armes nicht.

5.2. Manipulation des Skeletts

Der folgende Abschnitt stellt die java-technische Umsetzung zur Manipulation des Skelettes dar. Zur Zeit existieren zwei Möglichkeiten das Skelett zu verändern:

1. über das „direkte“ Verändern des Skeletts mit Hilfe von Inverse und Forward Kinematics.

2. über die automatische Interpolation zwischen zwei Schlüsselposen.

Beide wurden in separate Klassen ausgegliedert, um das Austauschen der Algorithmen auch zur Laufzeit zu ermöglichen.

5.2.1. KinematicSolver

`KinematicSolver` ist ein Interface (Schnittstelle), das alle Algorithmen zur Veränderung eines Skeletts zusammenfasst. Sie unterteilt sich in die beiden Hauptkonzepte: Inverse- und Forward-Kinematik. Diese beiden Methoden werden über die Interfaces `IKSolver` und `FKSolver` repräsentiert (siehe Abbildung 5.5). Direkte Ableitungen dieser Klassen implementieren dann die expliziten Algorithmen. `CCDIKSolver` im-

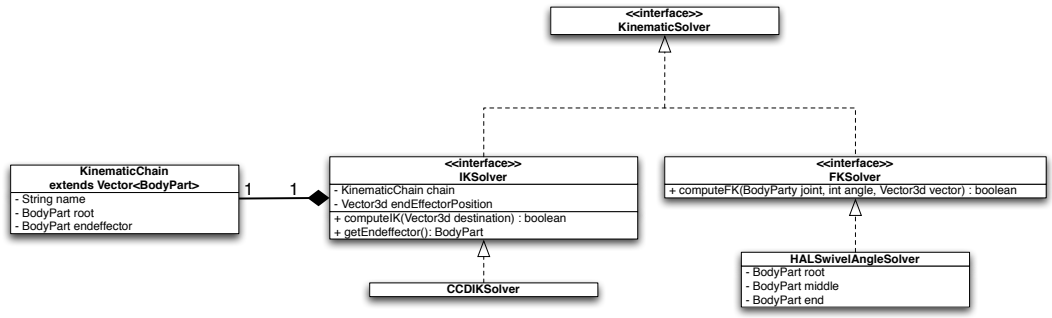


Abbildung 5.5.: Klassendiagramm des `KinematicSolver` zur Manipulation des Skeletts.

plementiert beispielsweise direkt das Interface `IKSolver` und ist für die Berechnung des CCD-Algorithmus (siehe Abschnitt 3.2.2.4) zuständig. Dafür wird für jede kinematische Kette ein eigener `CCDIKSolver` zur Verfügung gestellt. Über die Klasse `KinematicChain`, die Implementierung einer kinematischen Kette, werden die dafür benötigten Informationen bereitgestellt. Zu diesen Informationen gehören u.a. die Definition des End-Effektors, sowie die Wurzel der jeweiligen Kette. Zur Feinjustierung der Posenannotation dienen die Algorithmen zur Manipulation des Skelettes über FK. So erlaubt die Klasse `HALSwivelAngleSolver` (HAL=Human-Arm-Like) die Veränderung des „Swivel-Angle“, d.h. die Positionierung des Ellbogens über die Rotation des Schultergelenks.

5.2.2. Interpolator

Mit Hilfe der Interpolation werden die Posen zwischen den Schlüsselposen automatisch berechnet, so dass das Skelett zur Laufzeit animiert werden kann. Anhand dieser Animation kann der Animationskünstler kontrollieren, inwieweit seine Annotation dem Originalvideo entspricht. Wie in Abschnitt 3.3.2 bereits erwähnt, wurde SLERP als Interpolationsalgorithmus implementiert. Die Berechnung von SLERP übernimmt die Klasse `SLERPInterpolator`, die von der abstrakten Klasse `Interpolator` abgeleitet wird. Dadurch können später andere Interpolationsalgorithmen implementiert und eingebunden oder ausgetauscht werden (siehe Abbildung 5.6). Beim Speichern der Annotation werden nur die Schlüsselposen in die ANVIL-Datei geschrieben. Die interpolierten Posen werden beim Einlesen der ANVIL-Datei berechnet. Listing 5.1 beschreibt das Hinzufügen von neuen Posen als Pseudo-Code. Beim Hinzufügen einer neuen Pose wird die Interpolation zum vorherigen und zur nächsten Pose berechnet.

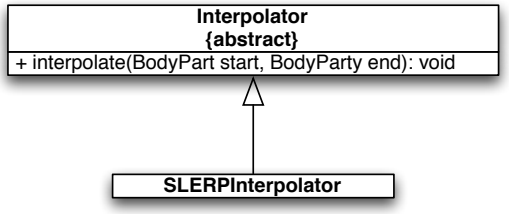


Abbildung 5.6.: `Interpolator` ist eine abstrakte Klasse, von der explizite Interpolationsalgorithmen, wie beispielsweise LERP, SLERP usw., abgeleitet werden. Dies ermöglicht das spätere Austauschen und Ergänzen von weiteren Interpolationstechniken.

5.3. Einbindung in ANVIL

Dieses Werkzeug wurde fest in ANVIL integriert, d.h. der vorhandene ANVIL-Code wurde erweitert. Zu diesem Zweck mussten neue Konzepte hinzugefügt werden, die das Lesen, die Speicherung und die Repräsentation von Posen ermöglichen.

5.3.1. Erweiterung der Konzepte

Um Posen in ANVIL hinzufügen zu können, wurde `AnnElement`, die Klasse zur Repräsentation von Annotationselementen, um eine weitere Liste erweitert. Diese Liste enthält Elemente vom neuen Typ `AnnTimestampedPoseAttribute`. Diese neue Klasse verwaltet die einzelnen annotierten Posen eines `AnnElements` und leitet sich direkt von der abstrakten Klasse `AnnAttribute` ab, die die Attribute eines Tracks darstellt. Eine Pose wird mit `TimestampedPose` abgebildet. `TimestampedPose` speichert dazu

```
1  void addNewPose(Pose pose){
2    Pose previousPose = getPreviousPose(pose);
3    Pose nextPose = getNextPose(pose);
4    if (previousPose != null && nextPose != null){
5      loeschePosen(previousPose.frame, nextPose.frame)
6      double t = 1 / (pose.frame-previousPose.frame);
7      berechneSLERP(previousPose, pose, t);
8      t = 1 / (nextPose.frame-pose.frame);
9      berechneSLERP(pose, nextPose, t);
10   }
11   else
12     if (nextPose == null){
13       double t = 1 / (pose.frame-previousPose.frame);
14       berechneSLERP(previousPose, pose, t);
15     }
16     else
17       if (previousPose == null){
18       double t = 1 / (nextPose.frame-pose.frame);
19       berechneSLERP(pose, nextPose, t);
20       }
21   fügePosehinzu(pose);
22 }
```

Listing 5.1: Pseudo-Code: Interpolation mit SLERP. Ab Zeile 4 wird überprüft, ob eine Pose zwischen zwei Schlüsselposen hinzugefügt wird. In diesem Fall werden alle interpolierten Posen gelöscht (Zeile 5) und dann die Interpolationen zwischen Vorgängerpose (Zeile 7) und Nachfolgepose (Zeile 9) berechnet. Ansonsten wird überprüft, welche Interpolationen durchgeführt werden müssen. Entweder zwischen neuer und Vorgängerpose (ab Zeile 12) oder zwischen neuer und Nachfolgerpose (ab Zeile 17).

die einzelnen Orientierungen von jedem Gelenk eines Skeletts, zu einem bestimmten Zeitpunkt ab. Die Speicherung der Orientierung erfolgt dabei als Quaternion in der `JointOrientation`-Klasse. Abbildung 5.7 illustriert die Klassenabhängigkeiten dieser Klassen.

5.3.2. Modifikation des Codes

Zum Aufruf der in Abschnitt 4.2 vorgestellten grafischen Benutzeroberfläche, mussten vorhandene Klassen um Methoden und Funktionen erweitert werden.

Aufruf der Benutzeroberfläche Um den *Pose Sequence Editor*, die Benutzeroberfläche zur Posenannotation, aufrufen zu können, wurde die Methode *showPoseEditor()* der `ElementEditWindow`-Klasse hinzugefügt. Diese Klasse wird aufgerufen, sobald ein neues Element in ANVIL hinzugefügt oder ein

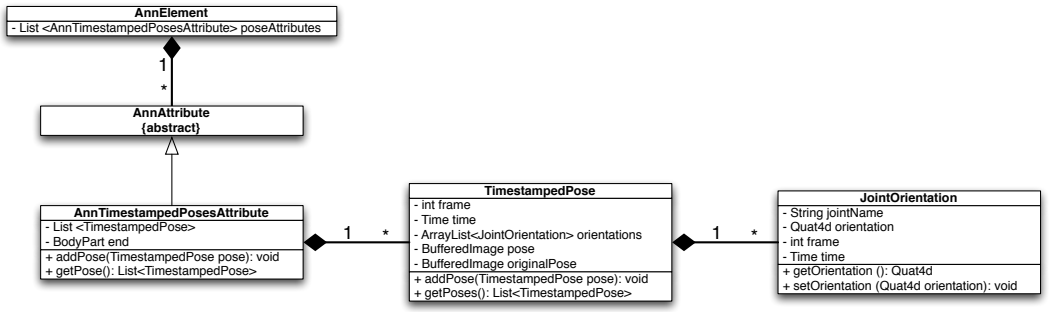


Abbildung 5.7.: Das Diagramm zeigt die Erweiterung der `AnnAttribute`-Klasse, damit Posen in ANVIL repräsentiert werden können. Es werden nur die wichtigsten Funktionen und Attribute der Klassen aufgezeigt.

vorhandenes editiert werden soll. *showPoseEditor()* ruft den *Pose Sequence Editor* auf und ist nur möglich, wenn für den entsprechenden Track das Attribut `TimestampedPose` in der Spezifikationsdatei angegeben wurde.

Einlesen Spezifikation Da ANVIL-Spezifikationen vom `AnnSpecReader` eingelesen werden, musste dieser so angepasst werden, dass das neue Attribut `TimestampedPose` erkannt und verarbeitet wird.

Speichern und Einlesen der Posen Um Posenannotationen zu speichern, wurde die Speicherroutine von ANVIL modifiziert. Wie in Abschnitt 5.3.1 geschildert, werden in der ANVIL-Datei nur die Schlüsselposen dauerhaft gespeichert. Wie diese Repräsentation in der ANVIL-Datei aussieht, zeigt Listing 5.2. Demnach wird eine Pose als Liste von Gelenkorientierungen dargestellt. Die Gelenkorientierung wird wiederum als 4-Tupel gespeichert. Diese Darstellung wurde gewählt, weil sie äquivalent zur 4-Tupel-Darstellung von Quaternionen (siehe Abschnitt 3.3.1.2) ist und somit die Rücküberführung in Quaternionen beim Einlesen problemlos ermöglicht. Dazu wurde der `AnnReader` angepasst, damit die neuen Strukturen aus der ANVIL-Datei wieder eingelesen werden können.

Import und Export von COLLADA Zum Einlesen von COLLADA-Dateien wurde ein COLLADA-Importer implementiert, der aus den COLLADA-Daten die Java-Repräsentationen und das Skelett erzeugt. Über den Exporter können die Animationen wieder exportiert und in 3D-Programmen weiterverwendet werden.

```
1
2  <attribute name="pose">
3    <pose time="18.72" frame="468">
4      <joint name="l_shoulder" x="-0.161089..." y="-0.085565..." z="
           0.144287..." w="0.972578..."/>
5      <joint name="r_ankle" x="0.0" y="0.0" z="0.0" w="1.0"/>
6      <joint name="neck" x="0.0" y="0.0" z="0.0" w="1.0"/>
7      <joint name="pelvis_root" x="0.0" y="-0.104528..." z="0.0" w="
           0.994521..."/>
8      <joint name="l_hip" x="0.0" y="0.0" z="0.0" w="1.0"/>
9      <joint name="l_wrist" x="-0.325567..." y="-6.103528..." z="1.598431..."
           w="0.945518..."/>
10     <joint name="chest" x="0.0" y="0.0" z="0.0" w="1.0"/>
11     <joint name="waist" x="0.0" y="0.0" z="0.0" w="1.0"/>
12     <joint name="l_knee" x="0.0" y="0.0" z="0.0" w="1.0"/>
13     <joint name="l_elbow" x="0.585044..." y="-0.140374..." z="-0.441138..."
           w="0.665893..."/>
14     <joint name="r_shoulder" x="0.101234..." y="-0.205466..." z="
           -0.065016..." w="0.971240..."/>
15     <joint name="r_knee" x="0.0" y="0.0" z="0.0" w="1.0"/>
16     <joint name="r_wrist" x="0.1908086..." y="3.575758..." z="-9.368022..."
           w="0.9816279..."/>
17     <joint name="r_elbow" x="0.4241346..." y="0.179389..." z="-0.261221..."
           w="0.8483469..."/>
18     <joint name="r_hip" x="0.0" y="0.0" z="0.0" w="1.0"/>
19     <joint name="l_ankle" x="0.0" y="0.0" z="0.0" w="1.0"/>
20   </pose>
21 </attribute>
```

Listing 5.2: In ANVIL wird eine Pose als Attribut eines Tracks gespeichert und enthält eine Liste von Gelenken (`joint`). Jede Pose wird eindeutig über die Zeit und Frame identifiziert. Jedes Gelenk enthält den Namen und die Orientierung des Gelenks. Die Orientierung wird als 4-Tupel mit den Werten für w, x, y und z gespeichert. Damit kann die Rückführung in Quaternionen problemlos erfolgen.

Fazit: Dieses Kapitel befasste sich mit der realen Umsetzung des vorgeschlagenen 3D-Skeletts, dessen Manipulation und der Interpolation der Posen in Java und Java3D. In diesem Zusammenhang wurde auf eine Klassenhierarchie geachtet, die den Austausch und die Erweiterung von Interpolations- und Manipulationsalgorithmen erlaubt. Für die Integration in ANVIL wurden neue Konzepte wie der neue Attribut-Typ `TimestampedPose` eingeführt. In Verbindung damit wurden weitere Klassen wie z.B. der `AnnSpecReader` angepasst, die den neuen Typ einlesen und die fertigen Annotationen mit den neuen Informationen abspeichern können. Die auf diese Weise realisierte Applikation wird im nun folgenden Kapitel evaluiert.

6. Evaluation

Ziel der im Rahmen dieser Arbeit erstellten Applikation war ein Interface, das eine intuitive und präzise Gestenannotation ermöglicht. Allerdings sollte der Arbeits- und Zeitaufwand dennoch in vertretbarem Rahmen bleiben. Eine Möglichkeit dies zu realisieren, war die Umsetzung der Gestenannotation mit Hilfe eines 3D-Skeletts. Die Integration verschiedener Hilfsmittel, wie beispielsweise die freie Einstellung der Szenenkamera, eine intuitive Manipulationsmöglichkeit des 3D-Skeletts mit Hilfe von IK, etc. (siehe Kapitel 4), sollten die Annotationsarbeit erleichtern. Um zu überprüfen, ob und inwiefern die gesetzten Ziele erreicht wurden, ist eine Evaluation dieser Applikation erforderlich. Dabei standen folgende Fragen im Mittelpunkt: Ist die Benutzung intuitiv und wie hoch ist der Mehraufwand? Eine Möglichkeit zur Beantwortung dieser Fragen ist ein Vergleich der Annotationszeiten der neuen und alten Methode. Dazu müssen ausgewählte Probanden die Applikation testen, die mit der Gestenannotation nicht vertraut sind. Dabei werden die Annotationszeiten gemessen und verglichen. Die Frage nach der Vertretbarkeit des Aufwandes kann nur durch einen direkten Vergleich zwischen den verschiedenen Annotationsmethoden geklärt werden. Wie diese Evaluation aufgebaut und durchgeführt wurde, soll im Folgenden dargestellt werden.

6.1. Aufbau

6.1.1. Probanden und Aufgabe

Für die Evaluation des Benutzerinterfaces wurden acht Probanden ausgewählt, die weder Vorkenntnisse im Bereich der Gestenannotation aufwiesen, noch mit der zu untersuchenden Applikation oder ANVIL vertraut waren. Bei den Probanden handelt es sich um Studierende oder Personen mit abgeschlossenem Hochschulstudium im Alter zwischen 21 und 30 Jahren. Unter den Probanden befanden sich sieben männliche und eine weibliche Person.

Die Aufgabe der Probanden bestand in der Annotation einer vorgegebenen Gestensequenz bestehend aus 123 Frames in ANVIL mit Hilfe der Funktionselemente dieser Applikation. Die betreffende Sequenz ist zuvor bereits mit einem Annotationsschema in ANVIL annotiert worden. Daher lagen die Gesten schon in ihre einzelnen Phasen untergliedert vor. Diese Phasen sollten sukzessive bearbeitet werden, indem sinnvolle Key-Posen anhand des 3D-Skeletts erstellt werden. Die Geste wurde dann über die Animation der Posen nachgebildet. Das Ziel war eine möglichst präzise Abbildung der Gestensequenz, d.h. dass die animierten Gesten den Originalgesten aus dem zugehörigen Video möglichst gut entsprechen. Nach Möglichkeit sollten hierbei

alle verfügbaren Hilfsmittel, wie die Veränderung der Szenenkamera, der Pose Viewer usw., genutzt werden.

6.1.2. Material und Ablauf

Das für die Evaluation verwendete Material beinhaltet ein Manual, das die notwendigen Arbeitsschritte zur Erstellung neuer Posen und zur Manipulation bestehender Posen erklärt (siehe Anhang E). Darüber hinaus soll ein Fragebogen die Einschätzung der Probanden feststellen. Er erfasst zunächst eine Gesamtbeurteilung der Applikation hinsichtlich Verständlichkeit, Einarbeitungszeit, Intuitivität, „Arbeitsspaß“, Zufriedenheit mit dem Ergebnis sowie einer Beurteilung, wie sinnvoll diese Art der Annotation erscheint. Zusätzlich erfasst er die Bewertung der einzelnen Funktionselemente der Benutzeroberfläche jeweils nach Übersichtlichkeit, Bedienungsfreundlichkeit, Zweckerfüllung und Intuitivität der Anwendung. Die betreffenden Funktionselemente umfassen den *Pose Sequence Editor*, den *Pose Editor* mit den verschiedenen Panels (Videoscreenshot-, End-Effektor- und Kamera-Panel) und den *Pose Viewer*. Als Bewertungsmaßstab sollten nach dem Schulnotensystem Noten zwischen 1 (sehr gut) und 6 (ungenügend) für jedes Element und jede Bewertungskategorie vergeben werden. Zusätzlich sollte für jede Bewertungskategorie der Grund für die jeweilige Benotung angegeben werden.

Die Probanden erhielten vor der Durchführung der Aufgabe das Manual zum Lesen. Danach wurde eine kurze Einweisung in die Benutzung von ANVIL und die relevanten Grundfunktionen der neuen Werkzeuge gegeben. Im Anschluss daran wurde die Gestensequenz von den Probanden eigenständig annotiert. Währenddessen wurde die Zeit für jedes einzelne annotierte Segment der Sequenz geloggt. Die Zeit lief, sobald im *Pose Sequence Editor* der Befehl *Add New Pose* oder *Edit Pose* betätigt wurde. Wenn der Befehl *Save Pose* gegeben wurde, wurde der Timer gestoppt. Beginn und Ende der gesamten Session wurden ebenfalls mitgeloggt. Abschließend füllten die Probanden die Fragebögen aus. Die Ergebnisse der Probanden in Bezug auf die Bearbeitung der Aufgabe und die Ergebnisse der Fragebögen werden im nächsten Abschnitt dargestellt.

6.2. Auswertung

Die Ergebnisse der von den Probanden bearbeiteten Aufgabe werden in der Datenanalyse und deren subjektive Bewertung in der Analyse der Fragebögen abgehandelt. Bevor diese Ergebnisse allerdings erläutert werden, soll an dieser Stelle zunächst auf die Komplexität einer Pose eingegangen werden. Denn diese ist unmittelbar relevant für eine angemessene Auswertung und Interpretation der Ergebnisse.

6.2.1. Komplexität einer Pose

Die Komplexität einer Pose ist eine entscheidende Variable im Zusammenhang mit der Annotationszeit. Denn der Zeitaufwand bei der Posenerstellung ist hauptsäch-

lich von der Größe des Unterschieds zwischen zwei Key-Posen abhängig[1]. *Komplexität* wird in dieser Arbeit daher als die Größe der Veränderung zwischen zwei Key-Posen definiert. Da die Komplexität verschiedener Posen variiert, ist die Darstellung der benötigten Annotationszeit pro Pose nur bedingt aufschlussreich. Ein Beispiel aus der hier zu annotierenden Sequenz zeigt, dass von Frame 489 zu Frame 506 eine größere Bewegungsveränderung vorliegt als zwischen Frame 506 und Frame 508. Daran kann man erkennen, dass die Veränderungen zwischen zwei Posen nicht immer konstant ist. Um dennoch eine Aussage über das Lernverhalten der Benutzer machen zu können, muss die Annotationszeit an der Komplexität relativiert werden. D.h. es musste ein Maß gefunden werden, das die „Leistung“ der Probanden während des Versuchs vergleichbar widerspiegelt. Der Verlauf dieser „Leistungskurve“ kann dann als Lernkurve der Probanden gesehen werden. Hierfür ist es nun notwendig die Komplexität in einem mathematischen Wert auszudrücken. Im Folgenden soll deshalb beschrieben werden, wie die Komplexität berechnet werden kann.

Die Veränderung der Pose eines Skeletts kann entweder über IK, also dem Verschieben des End-Effektors, oder über die Veränderung der Gelenkwinkel erfolgen[2]. Aus diesen Überlegungen heraus, habe ich folgende Formeln für die *Komplexität* K einer Pose abgeleitet:

$$K = T + R \tag{6.1}$$

K beschreibt die Komplexität einer Pose. T ist die Summe der Wegstrecken aller End-Effektoren und R die Summe aller Winkelveränderungen. T berechnet sich

[1] Natürlich vergeht auch Zeit bei der Benutzung der Hilfsmittel, wie der Ausrichtung und Positionierung der Screenshots. Jedoch steht die Verwendung dieser Hilfsmittel in engem Zusammenhang mit der Komplexität einer Pose. Ist diese gering, kann wahrscheinlich auf die Benutzung dieser Hilfsmittel verzichtet werden. Daher sollte diese Zeit bei der Abschätzung der Komplexität nicht ins Gewicht fallen.

[2] Um herauszufinden, um welchen Winkel sich die Gelenke verändert haben, werden zwei Orientierungen eines Gelenks zu verschiedenen Zeiten (Anfangspose und Endpose einer Sequenz) betrachtet. Um die Winkelveränderung zwischen beiden Orientierungen zu erhalten wird eine Interpolation mit SLERP und dem Faktor t=1 durchgeführt (siehe Abschnitt 3.3.1.2). Das Ergebnis ist ein Quaternion q, das die Rotation repräsentiert, die das Gelenk von der Anfangspose in die Endpose überführt. Für q gilt:

$$q = (w, x, y, z) = (\cos\frac{\alpha}{2}, n_x \sin\frac{\alpha}{2}, n_y \sin\frac{\alpha}{2}, n_z \sin\frac{\alpha}{2}).$$

Daraus ergibt sich folgender Zusammenhang:

$$w = \cos\frac{\alpha}{2}.$$

Daraus lässt sich der gesuchte Winkel α leicht berechnen:

$$\alpha = 2 \arccos w.$$

durch:

$$T = \left(\frac{T_1 + T_2}{2}\right) n, \quad \text{mit } n = \begin{cases} 1, & \text{wenn eine Hand bewegt wurde,} \\ 2, & \text{wenn beide Hände bewegt wurden.} \end{cases} \tag{6.2}$$

Da das verwendete Skelett nur 2 End-Effektoren hat, wird durch diese Konstante dividiert. Somit wird T maximal, wenn beide Hände in die entgegensetzte Richtung bewegt werden. Die Wegstrecke wäre damit die doppelte Armlänge für jeden Arm. Die Summe der Winkelveränderungen R berechnet sich wie folgt:

$$R = \left(\frac{\alpha_0 + \alpha_1 + \cdots + \alpha_m}{7}\right) m, \tag{6.3}$$

wobei m die Anzahl der tatsächlich veränderten Gelenke ist und α_i die einzelnen Winkelveränderungen sind. Die Konstante 7 steht für die maximale Anzahl der beweglichen Gelenke in dem verwendeten Skelett. Demnach beschreibt R die durchschnittliche Rotation der veränderten Gelenke, verteilt auf alle Gelenke, die verändert werden können. Den maximalen Wert erreicht R, wenn also alle sieben Gelenke um 2π rotiert wurden.

Um zu zeigen, dass diese Formel die Komplexität einer Pose richtig beschreibt, betrachten wir zuerst die Summanden von K zusammen. Je größer die Bewegung der Hände (T) und die durchschnittliche Winkelveränderung der Gelenke (R) ist, desto größer wird die Posenkomplexität K. Dies erscheint mir eine zutreffende Vorstellung von Komplexität zu sein. Betrachtet man nun die einzelnen Summanden getrennt, so lassen sich folgende Aussagen treffen. Die Beträge von T und R werden kleiner, wenn nicht alle Hände bewegt bzw. alle Gelenke rotiert wurden. Das bedeutet, dass eine Translation mit einer Hand die Komplexität der gesamten Veränderung senkt, genauso wie die Rotation von wenigen Gelenken. Daraus ergibt sich die maximale Komplexität von $T_1 + T_2 + 2\pi$, wobei T_1 und T_2 die doppelte Armlänge ist.

6.2.2. Datenanalyse

Die Gesamtdauer vom Öffnen bis zum Schließen von ANVIL variiert zwischen 28 und 50 Minuten bei einem Mittelwert von 23,5 Minuten. Die reine Annotationszeit beträgt zwischen 15 und 34 Minuten (Mittelwert = 19 Minuten). Die Auswertung der Ergebnisse der annotierten Sequenzen zeigt, dass von den 123 möglichen Posen maximal 18 Posen annotiert wurden. Jeder Proband hat jedoch mindestens 13 Posen annotiert. Es handelt sich dabei jeweils um die Anfangs- und Endpose der einzelnen Phasen einer Geste. Abbildung 6.1(a) stellt die benötigte Annotationszeit pro Pose dar. Aus Gründen der Übersichtlichkeit werden hier nur die Kurven der ersten fünf Probanden gezeigt. Es sind ausschließlich die Frames aufgeführt, die von jedem Probanden annotiert wurden.

Abbildung 6.1(b) zeigt hingegen die über alle Probanden gemittelte Zeit und die jeweilige Standardabweichung, die pro Frame benötigt wurde, um die dargestellte Pose zu annotieren (schwarze Kurve). Sie zeigt gleichzeitig auch die Annotationszeit,

die ein *Experte* für die gleiche Gestensequenz unter gleichen Bedingungen (türkisfarbene Kurve) bzw. für die Annotation mit Hilfe eines Annotationsschemas benötigt (blaue Kurve). *Experte* bedeutet hier, dass die Person Erfahrungen im Umgang mit ANVIL und der Annotation mit diesem Werkzeug hat. Die Probanden benötigen im Durchschnitt wesentlich mehr Zeit als der Experte bei skelettbasierter Annotation.

Abbildung 6.2 stellt einen Ausschnitt der verschiedenen Annotationsergebnisse dar. In der obersten ist die Annotationsvorlage zu sehen. Die zweite Reihe zeigt die generierten Posen, die aus der Annotation mit dem Annotationsschema und der Arbeit von Kipp u. a. [2007a] gewonnen wurden. Die dritte Reihe zeigt das Ergebnis des Probanden 5. Die Annotation des Experten befindet sich in der letzen Reihe. Die Ergebnisse auf Basis des Annotationsschemas beruhen auf der Auswahl von Labels. Abbildung 6.3 zeigt zum Vergleich das Eingabefenster zur Auswahl der verschiedenen Attribute und deren Auswahlmöglichkeiten.

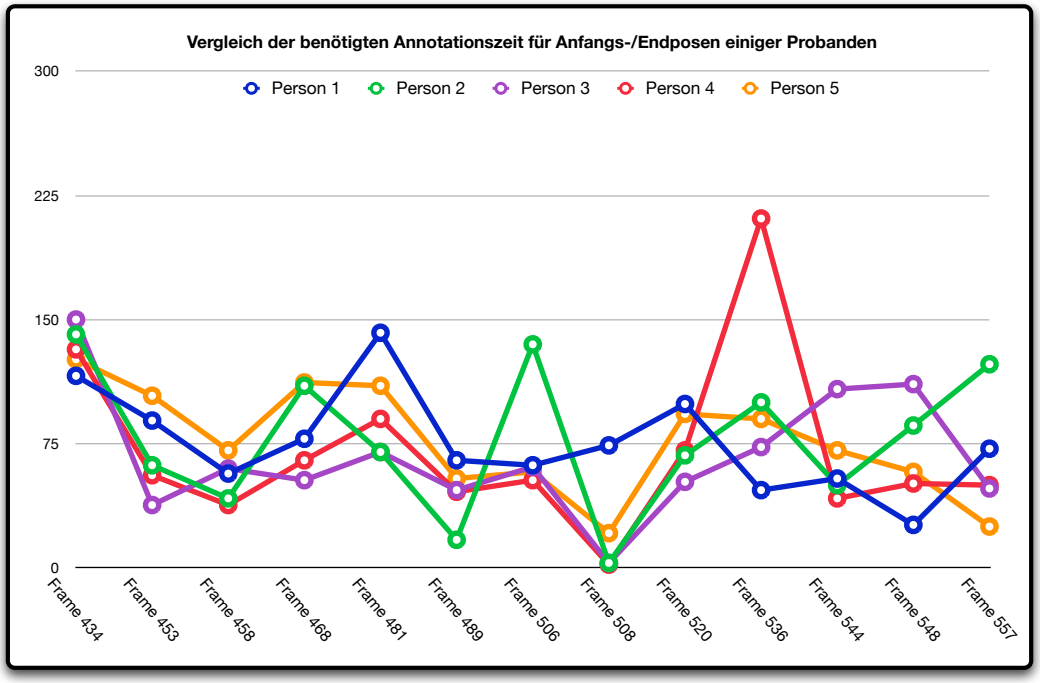
Die gemittelte Komplexität der einzelnen Posen, berechnet aus den Werten aller Probanden, ist in Abbildung 6.5 dargestellt. Abbildung 6.4 stellt hingegen, die „bereinigte“ Annotationszeit dar. D.h. hierbei wurde die Komplexität als Einflussfaktor entfernt, indem die reine Annotationszeit an ihr relativiert wurde. Die Lernkurve der Probanden und die des Experten nähern sich mit zunehmender Anzahl annotierter Posen an. Der größte Unterschied besteht in Bezug auf die erste Pose. Hier zeigt sich auch die größte Standardabweichung bei der Leistung der Probanden.

6.2.3. Analyse der Fragebögen

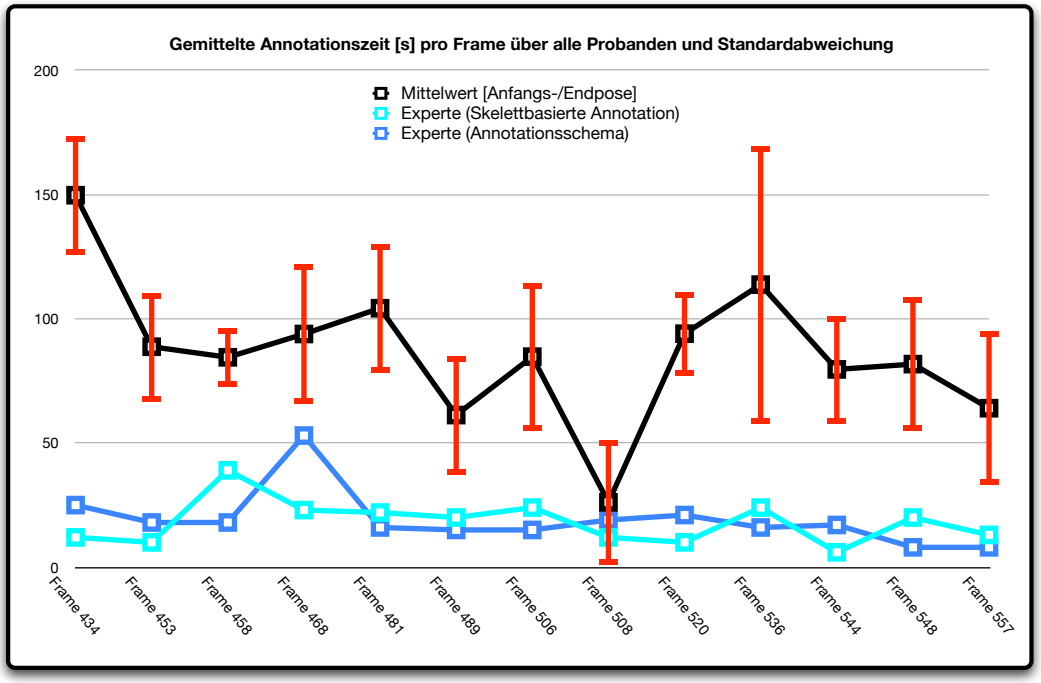
Die Auswertung der Fragebögen ergab bei der Beurteilung der einzelnen Funktionselemente, dass die vergebenen Noten zwischen *sehr gut* und *ausreichend* variieren. Der weitaus größte Teil der Noten liegt im Bereich zwischen *sehr gut* und *gut*; die Bewertung *ausreichend* wurde nur einmal vergeben. Abbildung F.1 im Anhang zeigt die Tabelle mit den vollständigen Ergebnissen und den Häufigkeiten der vergebenen Noten für alle Bewertungskategorien und Funktionselemente. Da einige Probanden nicht jedes Funktionselement benutzt haben, sind in der Tabelle bei einzelnen Kategorien weniger als die maximale Anzahl von acht Bewertungen eingegangen. In Tabelle 6.2.3 sind für einen Überblick lediglich die Durchschnittsnoten der übergeordneten Kategorien zusammengefasst.

Kategorie	Mittelwert
Gesamtbeurteilung	1,6
Pose Sequence Editor	1,3
Pose Editor	
Video-Screenshot-Panel	1,5
End-Effektor-Panel	1,4
Kamera-Panel	1,8
Pose Viewer	1,3

Tabelle 6.1.: Überblick über die Durchschnittsnoten der abgefragten Oberkategorien des Fragebogens.



(a) Annotationszeiten einzelner Probanden



(b) Gemittelte Annotationszeiten aller Probanden

Abbildung 6.1.: Grafik a) zeigt die einzelnen Annotationszeiten der ersten fünf Probanden. Grafik b) stellt die gemittelten Annotationszeiten aller Probanden mit der Standardabweichung pro Frame (schwarze Kurve) dar. Die türkisfarbene Kurve beschreibt die skelettbasierte Annotation des „Experten“, die blaue dessen Annotationszeit mit einem Annotationsschema.

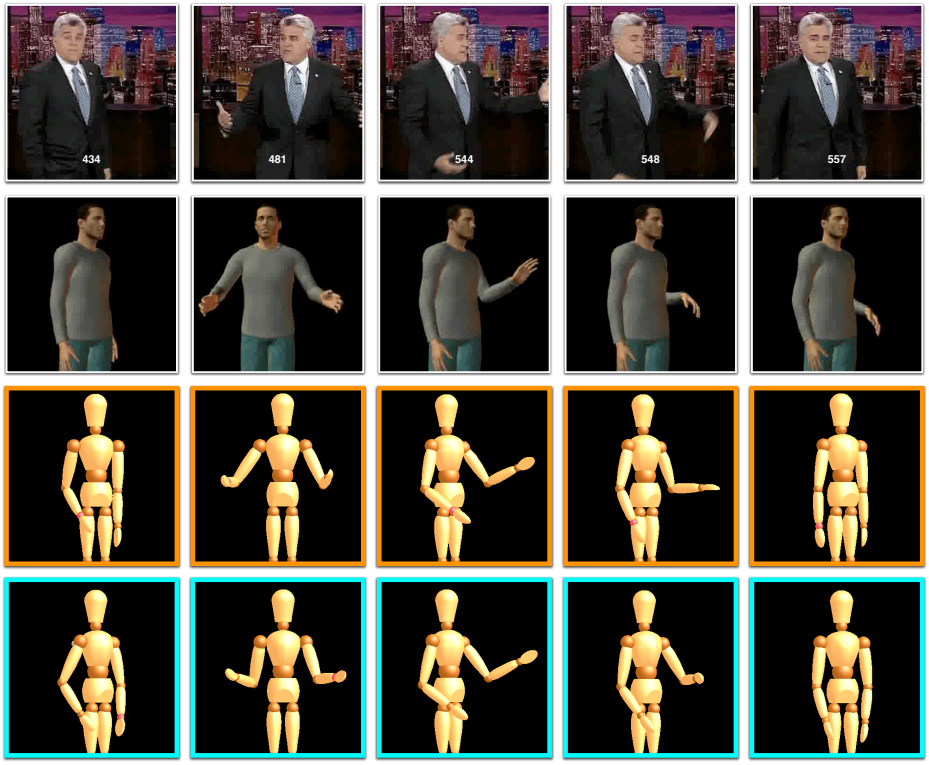


Abbildung 6.2.: Visueller Vergleich der Annotationsergebnisse. Die oberste Zeile zeigt den Originalsprecher, die zweite Zeile die generierten Posen aus dem Annotationsschema. In Zeile 3 ist das Ergebnis von Proband 5 zu sehen (orangeumrandet), während die letzte Zeile die annotierten Posen des Experten zeigen (türkisumrandet).

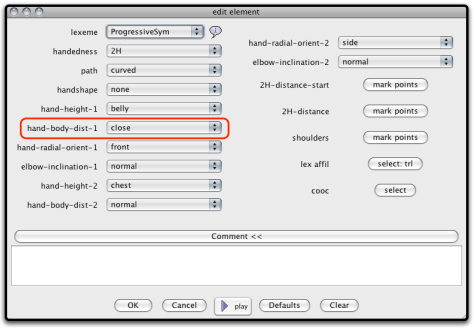


Abbildung 6.3.: Für die Beschreibung einer Pose müssen die verschiedenen Attribute gesetzt werden, wie der Abstand der Hand vom Körper (rot umrahmt). Je nach Anforderung und Genauigkeit variiert die Annotationszeit.

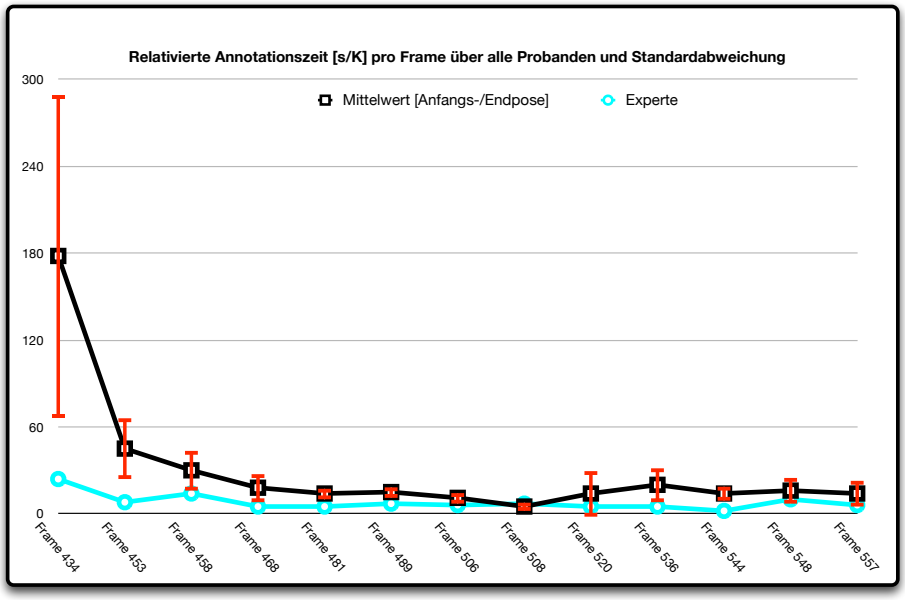


Abbildung 6.4.: Grafische Darstellung der Verbesserung im Verlauf der Annotation. Diese Werte wurden relativiert an der Komplexität einer Pose, d.h. $\frac{s}{K}$ gibt die Annotationszeit in Sekunden pro Komplexitätseinheit an.

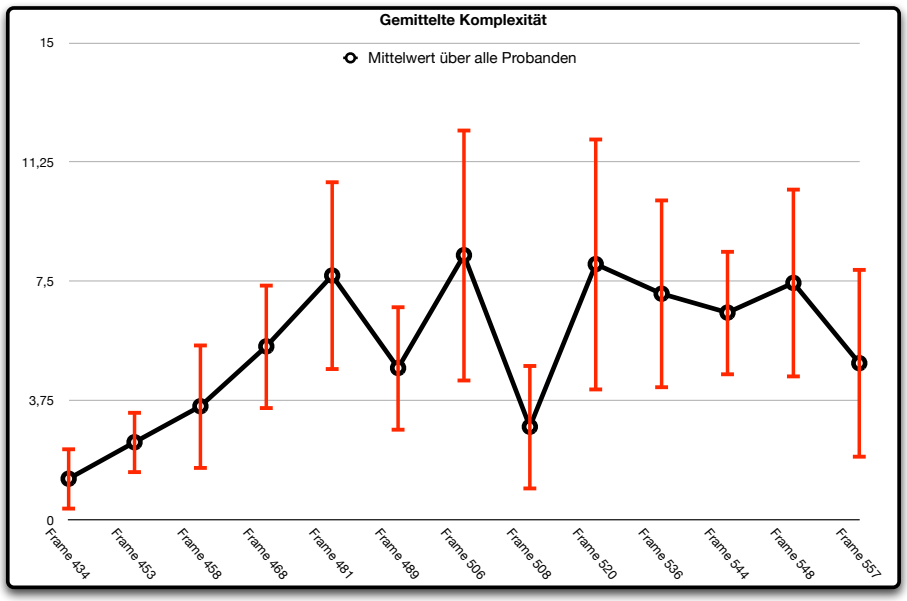


Abbildung 6.5.: Grafische Darstellung der gemittelten Komplexität aller Probanden.

6.3. Diskussion

Hintergrund dieser Evaluation war die objektive Überprüfung, ob die gesetzten Ziele erreicht wurden. Insbesondere sollte sie in erster Linie die Frage der Intuitivität dieser Applikation klären. Zusätzlich sollte sie Aufschluss geben, ob der vermutete Mehraufwand noch vertretbar ist. Bevor auf diese Aspekte eingegangen wird, sollen zunächst einige allgemeine Ergebnisse angesprochen werden.

Bei der Auswertung der Ergebnisse der annotierten Sequenzen fällt auf, dass die Annotation von 13 Posen ausreicht, um eine relativ präzise Abbildung der Gesten und einen natürlichen Bewegungseindruck in der Animation zu erreichen. Gründe für die Zeitdifferenz zwischen der Dauer einer Annotationssession und der reinen Annotationszeit sind u.a. die Verwendung der Animation zur Kontrolle, sowie die Selektion der Anfangs- und Endzeiten im *Annotation Board*. Über alle Probanden hinweg zeigt sich ein ähnlicher Verlauf der Annotationszeit. Alle auftretenden „Ausreisser“ beruhen darauf, dass hierbei die Reset-Funktion zum Zurückstellen des Skeletts betätigt wurde oder die manuelle Anpassung der Screenshots erfolgte. Individuelle Unterschiede in der Zufriedenheit mit dem eigenen Annotationsergebnis sind der Hauptgrund dafür, dass einige Probanden im Gesamtverlauf länger brauchten als andere. Die Unterschiede in der Annotationszeit der Probanden im Gegensatz zu der des Experten bei skelettbasierter Annotation (siehe Abbildung 6.1(b)) legen nahe, dass die Erfahrung im Umgang mit der Applikation durchaus eine Rolle zu spielen scheint.

Die Auswertung der Fragebögen, basierend auf den Noten, ergab eine insgesamt sehr positive Beurteilung der Applikation. Dies lässt zunächst darauf schließen, dass die Probanden die Applikation problemlos anwenden konnten und diese im Wesentlichen gut akzeptiert haben. Die durchgehend sehr guten Bewertungen in den Bereichen Zweckerfüllung und Intuitivität (ausgenommen in Bezug auf das Kamera-Panel) sprechen dafür, dass das Ziel eines intuitiven Benutzerinterfaces, dem subjektiven Eindruck der Probanden nach, erreicht werden konnte. Auch die im Fragebogen geäußerten Kommentare der Probanden scheinen dies zu bestätigen. Sie liefern jedoch auch zusätzlich genaue Hinweise auf die Stärken und Schwächen der einzelnen Applikationskomponenten. Eine ausgemachte Schwäche liegt beispielsweise in der Bedienung des Kamera-Panels im Pose Editor. Aufgrund der vielen Schieberegler scheint er den meisten Probanden zu unübersichtlich zu sein. Die Möglichkeit, die annotierte Sequenz sofort betrachten zu können, wurde hingegen als „motivationsfördernd“ angesehen. Ausserdem wurde die grafische 3D-Annotation als „leicht zugänglich„ bewertet, auch weil man das Ergebnis „sofort sieht“. Die Benutzung des *Pose Sequence Editors* wurde als „einleuchtend und intuitiv“ gelobt, u.a. weil die Ansicht der zu annotierenden Frames als „Filmstreifen“ aufgebaut wurde. Zusätzlich wurde der *Pose Viewer* begrüßt, da dieser „unterschiedliche Ansichten und eine visuelle Kontrolle ermöglicht“. Dies erlaubt die annotierte Pose bezüglich der Abbildungstreue mit dem Original und der „Realitätsangemessenheit“ bei der Animation zu vergleichen. Obwohl schon sieben Gelenke manipulierbar sind, wurde darüber hinaus der Wunsch geäußert, noch mehr Gelenke verändern zu können, da „manche Gelenke nicht ganz

fein bzw. gar nicht zu justieren“ seien. Natürlich müssen subjektive Aussagen in einem Fragebogen mit Vorsicht interpretiert werden, da sie unter Umständen verzerrt sein können. Der Fragebogen an sich könnte zu undifferenziert gewesen sein oder die Probanden neigen dazu, zu positiv zu bewerten. Daher muss die Überprüfung, ob das Ziel tatsächlich erreicht werden konnte, über ein objektiveres Maß erfolgen.

Die Frage nach der Intuitivität der Applikation kann nur anhand des objektiven Lernverhaltens der Probanden befriedigend geklärt werden: Bei einem intuitiven Tool sollten unerfahrene Probanden schnell in der Lage sein, die Applikation zu beherrschen und effizient damit zu arbeiten. Die Annotationszeitkurven allein können hier keinen Aufschluss geben. Der relativ ähnliche Verlauf der Annotationszeiten der einzelnen Probanden (Abbildung 6.1(a)) lässt allerdings vermuten, dass es für jede Pose einen konstanten Faktor geben muss, der die Schwankungen in der individuellen Annotationszeit verursacht. Ich halte die Komplexität der Pose für den entscheidenden Faktor. Der Vergleich der Schaubilder 6.1(a) und 6.5 bestätigt diese Vermutung. Denn die Annotationszeit der Probanden nimmt den analogen Verlauf wie die gemittelte Komplexität von allen Probanden: Steigt die Komplexität einer Pose, so steigt auch die Annotationszeit. Die einzige Ausnahme ist die erste Pose, bei der bei allen Probanden eine relativ hohe Annotationszeit vorliegt. Wird der Einfluss der Komplexität herausgerechnet (s/K), zeigt sich die reine „Lernleistung“ des Probanden (siehe Abbildung 6.4). Die über alle Probanden gemittelte Lernkurve spiegelt eine deutliche lern- bzw. erfahrungsbedingte Verbesserung im Verlauf der Aufgabenbearbeitung wider. Die erste Pose benötigt die meiste Zeit, obwohl ihre Komplexität nicht besonders hoch ist, wie Abbildung 6.5 zeigt. Die enorme Standardabweichung hierbei verdeutlicht die individuellen Unterschiede bei der ersten Konfrontation mit der Bedienung. Die starke Abnahme der benötigten Zeit zur zweiten Pose spricht dafür, dass nur eine Pose zur Orientierung ausreicht. Die Leistung der Probanden unterscheidet sich ab der fünften annotierten Pose kaum noch von der des Experten. Demnach ist die Annotation von fünf Posen ausreichend, um eine Leistung auf vergleichbarem Niveau, wie der Experte zu zeigen. Vergleicht man aber die reine Annotationszeit der Probanden, so stellt man fest, dass diese dennoch wesentlich höher ist als die des Experten. Zurückführen könnte man diesen Unterschied auf die verstärkte Nutzung der Hilfsmittel durch die Probanden.

Die Frage des Mehraufwandes kann dadurch beantwortet werden, dass die Leistung des Experten unter beiden Annotationsmöglichkeiten miteinander verglichen wird. Bei diesem Vergleich scheint kein deutlicher Unterschied in Bezug auf die Annotationszeiten zu bestehen (siehe Abbildung 6.1(a)). Die Annotation mit dem Annotationsschema beinhaltet lediglich die Auswahl von Labels. Daher hätte man eigentlich erwarten können, dass dies deutlich weniger Zeit erfordert als skelettbasierte Annotation. Tatsächlich ist der Aufwand zur Beschreibung einer Pose aber durch die Vielzahl der auszuwählenden Labels in vergleichbarer Größe wie der bei der manuellen Manipulation eines 3D-Skeletts. So muss beispielsweise bei der Annotation mit Labels, angegeben werden, in welchem Abstand die Hände vom Körper gehalten werden. Je nach Anforderung und Genauigkeit variiert auch hier die Annotationszeit und kann besonders bei kurzen Sequenzen höher liegen als bei der skelettbasierten

Annotation. Betrachtet man die Annotationsergebnisse in direktem Vergleich (siehe Abbildung 6.2), wird deutlich, dass die skelettbasierte Annotation gegenüber der Annotation mit Hilfe eines Annotationsschemas qualitativ gleichwertig ist. Dies trifft sogar für den Vergleich mit der Leistung eines Probanden zu, die sich nur unwesentlich von der des Experten unterscheidet. Der Vorteil der Manipulation am 3D-Skelett ist allerdings, dass die Posen variabel angepasst werden können, d.h. die Präzision kann gezielt beeinflusst werden. Beim Annotationsschema kann über die Auswahl hinaus kein Einfluss auf das Ergebnis genommen werden. Dadurch sehen Posen, die mit denselben Labels versehen wurden, auch immer gleich aus. Dies muss jedoch nicht genau der zu annotierenden Pose entsprechen. Alles in allem sprechen die Ergebnisse des Vergleichs der beiden Annotationsmöglichkeiten dafür, dass kein großer Mehraufwand durch skelettbasierte Annotation entsteht. Allerdings würde die rein skelettbasierte Annotation keine semantische Information über die Geste beinhalten. Sofern zusätzlich semantische Informationen benötigt werden, sollte sie immer in Kombination mit einem geeigneten Annotationsschema verwendet werden.

Fazit: Die Evaluationsergebnisse verdeutlichen, dass die zentralen Fragen sehr befriedigend beantwortet werden konnten. Erstens zeigte sich, dass die Applikation intuitiv ist, weil der Umgang damit in kürzester Zeit erlernt werden kann. Dies konnte sowohl durch die subjektiven Eindrücke der Probanden bestätigt werden als auch durch die objektive Leistung, die frei von bewussten Verzerrungen ist. Zweitens konnte kein erheblicher Mehraufwand festgestellt werden. Der Annotationsaufwand bei skelettbasierter Annotation entsprach in etwa genau dem eines Annotationsschemas. Ein Mehraufwand ergibt sich dann, wenn auch semantische Informationen benötigt werden, da in diesem Fall beide Annotationsmethoden verwendet werden müssten. Mit diesen Ergebnissen kann das Ziel eines Interfaces zur intuitiven und präzisen Posenannotation insgesamt als erreicht betrachtet werden.

7. Konklusion

Im letzten Kapitel dieser Arbeit soll zunächst noch einmal die Umsetzung der skelettbasierten Gestenannotation zusammengefasst werden. Abschließend folgt noch ein Ausblick auf zukünftige Arbeiten und mögliche Erweiterungen.

7.1. Zusammenfassung

Bisherige Annotationstools bieten bei der räumlichen Beschreibung einer Geste in der Regel nur eine grobe Approximation. Präzise Beschreibungen sind bei diesen Tools mit einem sehr hohen Aufwand in Form von langen Trainings- und Annotationszeiten verbunden. Das Ziel dieser Arbeit war es daher, eine Gestenbeschreibung zu ermöglichen, die sowohl sehr präzise, als auch intuitiv zu bedienen ist. Durch die intuitive Bedienung soll die Trainingszeit reduziert werden und die Durchführung der Annotationen relativ schnell möglich sein. Die zentrale Idee bei der Umsetzung ist die Verwendung eines 3D-Skeletts mit intuitiven Kontrollen. Diese intuitiven Benutzerschnittstellen sollen dem Benutzer helfen, sich in kurzer Zeit einzuarbeiten, um dann schnell und mit wenig Aufwand Posen zu erstellen. Zusätzlich wurden Funktionen zur Verfügung gestellt, um den Annotationsaufwand zu reduzieren und die Arbeit zu beschleunigen.

Zur Realisierung wurde ein 3D-Skelett entworfen, dessen Pose mit Inverse und Forward Kinematics verändert werden kann. Damit ist es dem Benutzer möglich, durch einfaches Verschieben der Handgelenke schnell Posen zu erstellen oder zu verändern. Um möglichst genaue Posenabbildungen zu erzielen, wird eine Vorlage der aktuell zu erstellenden Pose als Hintergrundbild in die Szene integriert. Zusätzlich hat der Benutzer die Möglichkeit die Szene aus mehreren Blickwinkeln zu betrachten, um bessere Ergebnisse erzielen zu können. Dies geschieht entweder über die Veränderung der Szenenkamera, des *Pose Editors* oder des *Pose Viewers*, der eine Mehrfachansicht aus verschiedenen Perspektiven auf das Skelett erlaubt. Der *Pose Sequence Editor* ermöglicht ein direktes Navigieren im Video und stellt annotierte Posen in einer Übersicht dar, so dass die Auswahl bestimmter Frames oder Posen schnell und mühelos erfolgen kann. Der *Skeleton Markup Editor* dient zur Definition von fehlenden kinematischen Ketten.

Über den Interpolationsalgorithmus SLERP ist die Animation der Posen möglich, ohne die aufwändige Annotation aller benötigten Frames. Für die Animation reichen die wichtigsten Schlüsselposen aus, da alle fehlenden Zwischenposen automatisch berechnet werden. Diese Animation kann synchron zum Originalvideo abgespielt werden und dient dem Benutzer zu Kontrolle und Vergleich mit der Originalgeste.

Durch die Verwendung von COLLADA, einem Dateiformat zum Austausch von 3D-Daten, ist die Wiederverwendbarkeit der Daten sichergestellt. Über COLLADA können Skelett-Topologien, die in 3D-Programmen wie Maya oder 3dsMax erstellt wurden durch diese Arbeit in ANVIL importiert und verwendet werden. Zur Weiterverwendung oder -bearbeitung in diesen 3D-Programmen können die erstellten Animationen und Schlüsselposen umgekehrt auch nach COLLADA exportiert werden.

Die abschließende Evaluation konnte zeigen, dass die Bedienung der entwickelten Werkzeuge intuitiv und daher schnell erlernbar ist. Zusätzlich konnte kein erheblicher Mehraufwand im Vergleich zur Annotation über ein Annotationsschema festgestellt werden, da der Annotationsaufwand bei skelettbasierter Annotation in etwa genau dem eines Annotationsschemas entspricht. Somit konnte gezeigt werden, dass das gesetzte Ziel eines Interfaces zur intuitiven und präzisen Gestenannotation erreicht wurde.

7.2. Zukünftige Arbeiten

Zum Abschluss dieser Arbeit werden mögliche Erweiterungen an diese Arbeit vorgestellt und diskutiert.

Zur Berechnung der Posen wurde der iterative Algorithmus CCD verwendet. Eine Erweiterung um analytische Methoden wäre sinnvoll, um je nach Gegebenheit zwischen den Algorithmen wechseln zu können. So können eventuell natürlichere Ergebnisse schneller erzielt werden. Beispielsweise könnte bei einer Erweiterung der Skelettfunktion um die Beweglichkeit des Oberkörpers, dieser mit CCD verändert werden, während die Manipulation der Arme mit einem analytischen Algorithmus erfolgt.

Da innerhalb dieser Arbeit keine Joint-Limitations implementiert wurden, sind auch unnatürliche Gelenkstellungen möglich. Möchte man jedoch diese reduzieren bzw. ganz ausschließen, so wären Beschränkungen der Gelenkfreiheiten eine mögliche Erweiterung dieser Arbeit. Zusätzlich könnten Gesten noch präziser annotiert werden, wenn weitere Gelenke, Körperteile und Rotationsachsen (z.B. Abwinkeln der Hände) zur Manipulation freigegeben werden. So wären Kopf- oder Oberkörperbewegungen sinnvolle Erweiterungen, um Gesten und Bewegungen noch natürlicher erscheinen zu lassen. Jedoch steigt mit steigender Manipulationsmöglichkeit auch die Annotationszeit und die Komplexität der Berechnungen, insbesondere wenn möglichst natürliche Posen entstehen sollen.

Derzeit können externe Skelett-Topologien importiert und verwendet werden. Jedoch bietet diese Arbeit noch keine Werkzeuge an, um das Skelett an andere Sprecher anzupassen. Sinnvoll wäre hier die Erweiterung des *Skeleton Markup Editors* um die Funktion, die Größe des Skelettes oder die Länge der einzelnen Gliedmaßen zu verändern. Somit könnten noch präzisere Posen und Gesten erstellt werden.

Eine weitere Möglichkeit diese Arbeit zu erweitern, wären zusätzliche Steuerungsmöglichkeiten, die noch intuitivere Skelettmanipulationen erlauben. Beispielsweise

bieten Multitouch-Screens mehr Möglichkeiten an, die Manipulationsfunktionen auszuführen, als eine Maus über ihre Tasten und das Mausrad. Auch so genannte *Tangible Interfaces*, also Schnittstellen, die dem Benutzer über reale Objekte die Interaktion mit dem System erlauben, könnten zur Skelettmanipulation genutzt werden. So könnte ein echtes mit Sensoren ausgestattetes Skelett dazu dienen, die Posen des 3D-Skeletts zu erstellen.

Literaturverzeichnis

Bro 2009
Brockhaus Multimedial. Mannheim : Brockhaus Multimedial, 2009

Altorfer u. a. 1997
Altorfer, Andreas ; Jossen, Stefan ; Wuermle, Othmar: Eine Methode zur zeitgenauen Aufnahme und Analyse des Bewegungsverhaltens. In: *Zeitschrift für Psychologie*, 1997

Auber und Prié 2005
Auber, Olivier ; Prié, Yannick: Advene: Active Reading through Hypervideo. (2005), S. 235–244. http://dx.doi.org/http://doi.acm.org/10.1145/1083356.1083405. – DOI http://doi.acm.org/10.1145/1083356.1083405. ISBN 1–59593–168–6

Barnes und Finch 2008
Barnes, Mark ; Finch, Ellen L.: COLLADA - Digital Asset Schema Release 1.5.0 / The Khronos Group Inc., Sony Computer Entertainment Inc. 2008. – Forschungsbericht

Bickmore und Cassell 2005
Bickmore, Timothy ; Cassell, Justine: Social Dialogue with Embodied Conversational Agents. In: Kuppevelt, J. van (Hrsg.) ; Dybkjaer, L. (Hrsg.) ; Bernsen, N. (Hrsg.): *Advances in Natural, Multimodal Dialogue Systems*, New York: Kluwer Academic, 2005

Bigbee u. a. 2001
Bigbee, Tony ; Loehr, Dan ; Harper, Lisa: Emerging Requirements for Multi-Modal Annotation and Analysis Tools. In: *Proceedings of the 7th European Conference on Speech Communication and Technology*, 2001

Boersma 2001
Boersma, Paul: Praat, a system for doing phonetics by computer. In: *Glot International* (2001), S. 341–345

Breiner 2005
Breiner, Tobias C.: *Dreidimensionale virtuelle Organismen*, Diss., 2005

Brugman und Russel 2004
Brugman, Hennie ; Russel, Albert: Annotating Multi-media / Multi-modal resources with ELAN. (2004)

Cary 1978
CARY, Mark S.: The Role of Gaze in the Initiation of Conversation. In: *Social Psychology* 41 (1978), Nr. 3, 269–271. http://www.jstor.org/stable/3033565. – ISSN 0147829X

Cassell 1989
CASSELL, Justine: Embodied Conversation: Integrating Face and Gesture into Automatic Spoken Dialogue Systems. In: *Spoken Dialogue Systems* (1989)

Cohn u. a. 2005
In: COHN, Jeffrey F. ; AMBADAR, Zara ; EKMAN, Paul: *Observer-Based Measurement of Facial Expression with the Facial Action Coding System.* Oxford Univesity Press, 2005

Dam u. a. 1998
DAM, Erik B. ; KOCJ, Martin ; LILLHOLM, Martin: Quaternions, Interpolation and Animation / Department of Computer Scines, University of Copenhagen. 1998. – Forschungsbericht

Dipper u. a. 2004
In: DIPPER, S. ; GÖTZE, M. ; STEDE, M.: *ANNIS A Linguistic Database for Exploring Information Structure.* Universitötsverlag Potsdam, 2004, S. 245–279

Ekman und Friesen 1978
EKMAN, Paul ; FRIESEN, W.V.: Facial Action Coding System. In: *Consulting Psychologists Press* (1978)

Frey 1999
FREY, Siegfried: *Die Macht des Bildes.* Hans Huber Verlag, 1999

Götze und Dipper 2006
GÖTZE, Michael ; DIPPER, Stefanie: ANNIS Complex Multilevel Annotations in a Linguistic Database. In: *Proceedings of the LREC 2006 Workshop on Merging and Layering Linguistic Information*, 2006

Gut und Milde 2003
GUT, Ulrike ; MILDE, Jan-Torsten: Annotation of Conversational Gestures using TASX and CoGesT. (2003)

Harrigan u. a. 2005
HARRIGAN, Jinni A. ; ROSENTHAL, Robert ; SCHERER, Klaus R. ; HARRIGAN, Jinni A. (Hrsg.) ; ROSENTHAL, Robert (Hrsg.) ; SCHERER, Klaus R. (Hrsg.): *The new Handbook of Methods in nonverbal Behavior Research.* Oxford Univesity Press, 2005

Hodgins u. a. 1999
HODGINS, J. K. ; O'BRIEN, J. F. ; BODENHEIMER, R. E.: Computer Animaiton.

In: Webste, J. G. (Hrsg.): *The Wiley Encyclopedia of Electrical and Electronics Engineering* Bd. 3. John Wiley and Sons, 1999, S. 686–690

Jackel u. a. 2006
Jackel, Dietmar ; Neunreither, Stephan ; Wagner, Friedrich: *Methoden der Computeranimation*. Springer Verlag, 2006

Kipp 2001
Kipp, Michael: Anvil – A Generic Annotation Tool for Multimodal Dialogue. In: *Proceedings of Eurospeech*, 2001, S. 1367–1370

Kipp 2003
Kipp, Michael: *ANVIL 4.0 - Annotation of Video and Spoken Language User Manual*, 2003

Kipp 2008
Kipp, Michael: Spatiotemporal Coding in ANVIL. In: *Proceedings of the 6th international conference on Language Resources and Evaluation*, 2008

Kipp u. a. 2007a
Kipp, Michael ; Neff, Michael ; Albrecht, Irene: An Annotation Scheme for Conversational Gestures: How to economically capture timing and form. In: *Journal on Language Resources and Evaluation - Special Issue on Multimodal Corpora* 41 (2007), December, Nr. 3–4, 325–339. http://www.springerlink.com/content/l7628q1145352652/

Kipp u. a. 2007b
Kipp, Michael ; Neff, Michael ; Kipp, Kerstin H. ; Albrecht, Irene: Towards Natural Gesture Synthesis: Evaluating gesture units in a data-driven approach to gesture synthesis. (2007)

Kitagawa und Windsor 2008
Kitagawa, Midori ; Windsor, Brian: *MoCap for Artists - Workflow and Techniques for Motion Capture*. Focal Press, 2008

Lander 1998
Lander, Jeff: Making Kine More Flexible. In: *Game Developer* (1998)

Lasseter 1987
Lasseter, John: Principles of Traditional Animation Applied to 3D Computer Animation. 21 (1987), Juli, Nr. 4, S. 35–44. http://dx.doi.org/http://doi.acm.org/10.1145/37401.37407. – DOI http://doi.acm.org/10.1145/37401.37407

Liu und Prakash 2003
Liu, Qiang ; Prakash, Edmond C.: The Parameterization of Joint Rotation with the Unit Quaternion. / Nanyang Technological University. 2003. – Forschungsbericht

Maestri 2006
MAESTRI, George: *Digital Character Animation 3*. Bd. 3. New Riders, 2006. – 325 S.

Magnenat-Thalmann u. a. 2008
MAGNENAT-THALMANN, Nadia (Hrsg.) ; JAIN, Lakhmi C. (Hrsg.) ; ICHALKARANJE, Nikhil (Hrsg.): *New Advances in Virtual Humans - ArtiPcial Intelligence Environment*. Springer-Verlag, 2008

Milde und Gut 2001
MILDE, Jan-Torsten ; GUT, Ulrike: The TASX-environment: an XML-based corpus database for time aligned language data. In: *IRCS Workshop of linguistic databases* (2001)

Neff u. a. 2008
NEFF, Michael ; KIPP, Michael ; ALBRECHT, Irene ; SEIDEL, Hans-Peter: Gesture Modeling and Animation Based on a Probabilistic Recreation of Speaker Style. In: *ACM Transactions on Graphics* 27 (2008), March, Nr. 1, 1-24. http://portal.acm.org/citation.cfm?doid=1330511.1330516

Parent 2001
PARENT, Rick: *Computer Animation: Algorithms and Techniques*. Morgan Kaufmann, 2001 http://www.siggraph.org/education/materials/HyperGraph/animation/rick_parent/Intr.html#Overview

Pitsch u. a. 2003
PITSCH, Karola ; GUT, Ulrike ; MILDE, Jan-Torsten: Multimodale bilinguale Korpora gesprochener Sprache: Korpuserstellung,-analyse und -dissemination in der TASX Umgebung. In: SEEWALD-HEEG, U. (Hrsg.): *Sprachtechnologie für die multilinguale Kommunikation - Textproduktion, Recherche, Übersetzung, Lokalisierung*, 2003

Rist u. a. 2003
RIST, Thomas ; ANDRÉ, Elisabeth ; BALDES, Stephan ; GEBHARD, Patrick ; KLESEN, Martin ; KIPP, Michael ; RIST, Peter ; SCHMITT, Markus: A Review of the Development of Embodied Presentation Agents and Their Application Fields. In: PRENDINGER, H. (Hrsg.) ; ISHIZUKA, M. (Hrsg.): *Life-Like Characters - Tools, Affective Functions, and Applications*, Springer, 2003, S. 377–404

Rohlfing u. a. 2006
ROHLFING, Katharina ; LOEHR, Daniel ; DUNCAN, Susan ; BROWN, Amanda ; FRANKLIN, Amy ; KIMBARA, Irene ; MILDE, Jan-Torsten ; PARRILL, Fey ; ROSE, Travis ; SCHMIDT, Thomas ; SLOETJES, Han ; THIES, Alexandra ; WELLINGHOFF, Sandra: Comparison of multimodal annotation tools. In: *Gesprächsforschung - Online-Zeitschrift zur verbalen Interaktion* 7 (2006), S. 99–123

Ruesch und Kees 1956
RUESCH, J. ; KEES, W.: *Nonverbal Communication.* Berkley: University od California Press, 1956

Savoye und Meyer 2008
SAVOYE, Yann ; MEYER, Alexandre: Multi-Layer Level of Detail For Character Animation. In: *Workshop on Virtual Reality Interaction and Physical Simulation VRIPHYS*, 2008

Schmidt 2005
SCHMIDT, Thomas: EXMARaLDA und Datenbank „Mehrsprachigkeit“: Konzepte und praktische Erfahrungen. (2005)

Schmidt und Wörner 2005
SCHMIDT, Thomas ; WÖRNER, Kai: *Erstellen und Analysieren von Gesprächskorpora mit EXMARaLDA*, 2005

Selman 2002
SELMAN, Daniel: *Java3D Programming.* Manning, 2002

Shoemake 1985
SHOEMAKE, Ken: Animating rotation with quaternion curves. New York, NY, USA : ACM, 1985. – ISSN 0097–8930, S. 245–254

Siegfried u. a. 1981
In: SIEGFRIED, Frey ; H.P., Hirschbrunner ; J., Pool ; W., Draw: *Das Berner System zur Untersuchung nonverbaler Inerakion: I. Die Erhebung des Rohdatenprotokolls.* P. Winkler, 1981, S. 203–236

Tolani u. a. 2000
TOLANI, Deepak ; GOSWAMI, Ambarish ; BADLER, Norman I.: Real-time inverse kinematics techniques for anthropomorphic limbs. In: TOLANI, Deepak (Hrsg.) ; GOSWAMI, Ambarish (Hrsg.) ; BADLER, Norman I. (Hrsg.): *Graphical Models and Image Processing* Bd. 62. Orlando, FL, USA : Academic Press, Inc., 2000. – ISSN 1077–3169, S. 353–388

Wahlster 2006
Kapitel Dialogue Systems Go Multimodal: The SmartKom Experience. In: WAHLSTER, Wolfgang: *SmartKom: Foundations of Multimodal Dialogue Systems.* Springer, 2006, S. 3–27

WEB3D Consortium
(Stand 10.2.2009)

Welman 1993
WELMAN, Chris: *Inverse Kinematics and Geometric Contraints for Articulated Figure Manipulation*, Simon Fraser University, Diplomarbeit, 1993

Wittenburg u. a. 2006
WITTENBURG, Peter ; BRUGMAN, Hennie ; RUSSEL, Albert ; KLASSMANN, Alex ; SLOETJES, Han: ELAN: a Professional Framework for Multimodality Research. (2006)

Zeppenfeld 2004
ZEPPENFELD, Klaus: *Lehrbuch der Grafikprogrammierung*. Spektrum Akademischer Verlag, 2004

A. Rechenregeln Quaternionen

Folgender Anhang soll nur die Hauptrechenregeln für Quaternionen vorstellen, um die hergeleiteten Gleichungen aus Abschnitt 3.3.1.2 nachrechnen zu können.

Hamilton-Regeln: Mit Hilfe der folgenden Regeln wird die Multiplikation zweier Imaginärteile beschrieben:

$$ij = -ji = k \tag{A.1}$$
$$jk = -kj = i \tag{A.2}$$
$$ki = -ik = j \tag{A.3}$$

Konjugation: Die Konjugation eines Quaternions erfolgt durch die Negation des Imaginärteils:

$$\overline{q} = w - \mathbf{v}$$

Betrag: Der Betrag eines Quaternions ist das Produkt eines Quaternions mit seiner Konjugation:

$$|q|^2 = q\overline{q} \tag{A.4}$$
$$= \overline{q}q \tag{A.5}$$
$$= w^2 + \mathbf{v} \cdot \mathbf{v} \tag{A.6}$$

Ein Quaternion dessen Betrag 1 ist, bezeichnet man auch als *Einheitsquaternion*. Solche werden zur Beschreibung von Rotationen benutzt. Eine wichtige Eigenschaft der Einheitsquaternionen ist, dass das Inverse eines Einheitsquaternions äquivalent zu seinem Konjugierten ist:

$$\mathbf{q_e^{-1}} = \overline{\mathbf{q_e}}.$$

Addition und Subtraktion: Die Addition und Subtraktion zweier Quaternionen erfolgt komponentenweise. Die Addition wird dabei folgendermaßen definiert:

Seien q und q' zwei Quaternionen mit $q = (w + v)$ und $q' = (w' + v')$, so gilt:

$$q + q' = (w + w') + (\mathbf{v} + \mathbf{v}').$$

Die Subtraktion erfolgt dabei analog zu der oben definierten Addition.

Multiplikation: Seien q und q' zwei Quaternionen mit $q = (w + v)$ und $q' = (w' + v')$, so ist die Multiplikation definiert als:

$$qq' = (ww') - \mathbf{v} \cdot \mathbf{v}' + \mathbf{v} \times \mathbf{v}' + w\mathbf{v}' + w'\mathbf{v},$$

wobei $\cdot$ das Skalarprodukt und $\times$ das Kreuzprodukt beschreibt.

Inverse: Das *Inverse* eines Quaternions wird definiert als

$$q^{-1} = \frac{1}{|q|}\overline{q}$$

B. Weitere Abbildungen der Interpolation mit SLERP

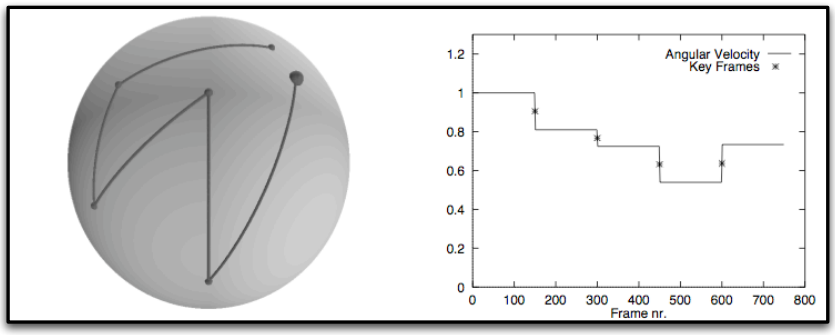


Abbildung B.1.: In dieser Abbildung ist zu sehen, dass SLERP bei mehr als zwei Key-Frames ruckartige Änderungen der Bewegungsrichtung erzeugt. Auch geht die gleichmäßige Beschleunigung verloren (entnommen aus Dam u. a. [1998]).

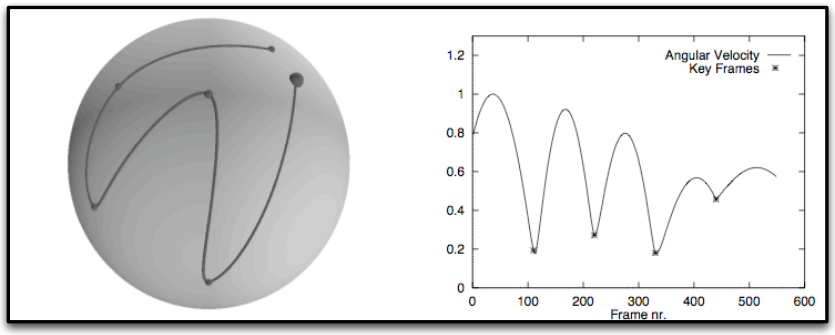


Abbildung B.2.: SQUAD erzeugt runde Kurven bei Richtungsänderungen an den einzelnen Schlüsselposen. Allerdings erhält man auch bei SQUAD im Allgemeinen keine gleichmäßige Beschleunigung (entnommen aus Dam u. a. [1998]).

C. COLLADA

Die folgende Übersicht soll keine umfassende Spezifikationsbeschreibung für COLLADA sein, sondern nur die für diese Arbeit wichtigen Punkte der COLLADA-Spezifikation auf konzeptueller Ebene erklären. Eine vollständige Beschreibung dieser Spezifikation mit kleinen Beispielen ist in Barnes und Finch [2008] zu finden. Ausreichend waren für ANVIL folgende „Bibliotheken“(library):

- `<asset>`
- `<library_geometries>`
- `<library_visual_scenes>`
- `<library_kinematics_models>`
- `<library_animations>`

library

Um die großen und komplexen Datenmengen verwalten zu können, unterteilt COLLADA die Datenmengen in kleinere Einheiten, die dann unter Bibliotheken wie *<library_geometries>*, *<library_visual_scenes>* und *<library_kinematics_models>* verwaltet werden. Denn bei einer einzigen großen Datenstruktur wird die Veränderung von vielen Daten erschwert. Listing C.1 beschreibt eine einfache `<library_geometries>` mit nur einem `<geometry>`-Objekt (siehe C). Diese „Bibliotheken“ dienen als Zusammenfassung von bestimmten Konzepten wie Geometriedaten oder Daten über die Kinematik.

```
<library_geometries>
  <geometry name="cube" id="cube123">
    <mesh>
      <source id="box-Pos"/>
      <vertices id="box-Vtx">
        <input semantic="POSITION" source="#box-Pos"/>
      </vertices>
    </mesh>
  </geometry>
</library_geometries>
```

Listing C.1: Collada <library_geometries>

<asset>

Unter <asset> werden Zusatzinformationen abgespeichert, die der Kommentierung des dazugehörigen Mutterknotens dienen. Darin können Informationen, wie Datum der letzten Änderung oder Schlüsselwörter abgelegt werden. Listing C.2 zeigt eine Beispielausprägung aus der Standard COLLADA-Datei für ANVIL.

```
<asset>
    <contributor>
        <author>quannguyen</author>
        <authoring_tool>Maya8.5 | ColladaMaya v3.05B</authoring_tool>
        <comments>Skeleton for Anvil Annotation Tool</comments>
      <source_data>file://anvil_skeleton.ma</source_data>
    </contributor>
    <created>2008-11-10T17:29:39Z</created>
    <modified>2008-11-10T17:29:39Z</modified>
    <unit meter="0.01" name="centimeter"/>
    <up_axis>Y_UP</up_axis>
</asset>
```

Listing C.2: Collada Asset

<geometry>

<geometry> sind die Hauptobjekte der <library_geometries>. Sie können geometrische Informationen, wie Form und Aussehen, einzelner Objekte der Szene enthalten. Dazu gehören Maßeinheiten, Beziehungen zwischen Punkten, Linien, Winkel und Oberflächen sowie Beschreibungen zur Erstellung des Drahtgittermodells. Diese Informationen werden in einem <mesh>-Objekt verwaltet. In Listing C.3 wird ein <geometry>-Objekt mit Daten zur Erstellung einer Mesh dargestellt.

```
<geometry id="l_ankle_joint_mesh" name="l_ankle_joint_mesh">
  <mesh>
    <source id="l_ankle_joint_mesh-positions" name="position"/>
    <source id="l_ankle_joint_mesh-normals" name="normal"/>
    <source id="l_ankle_joint_mesh-map1" name="map1"/>
    <vertices id="l_ankle_joint_mesh-vertices"/>
    <triangles material="blinn7SG" count="760">
      <input semantic="VERTEX" source="#l_ankle_joint_mesh-vertices" offset=
          "0"/>
      <input semantic="TEXCOORD" source="#l_ankle_joint_mesh-map1" offset="1
          " set="0"/>
      <p>0 0 1 ... 438</p>
    </triangles>
  </mesh>
</geometry>
```

Listing C.3: Collada <geometry>

Mesh

<mesh> beschreibt mit Hilfe von Informationen zu Kanten und Primitiven, Drahtgittermodelle der anzuzeigenden Objekte. Jede Kante beinhaltet dabei Informationen zur Position, Farbe, ihrer Texturkoordinate, sowie Informationen zu ihrem Normalenvektor. Die Art der Verbindung der Kanten wird anhand der Primitive festgelegt. Primitive können zum Beispiel Polygone, Dreiecke oder einfache Linien sein. Listing C.4 zeigt wie eine solche Mesh Definition aussehen kann.

```
<mesh>
  <source id="l_ankle_joint_mesh-positions" name="position">
    <float_array id="l_ankle_joint_mesh-positions-array" count="1146">
        0.018365 0.005967 -0.121921 ... 0.12344</float_array>
    <technique_common>
      <accessor source="#l_ankle_joint_mesh-positions-array" count="382"
          stride="3">
        <param name="X" type="float"/>
        <param name="Y" type="float"/>
        <param name="Z" type="float"/>
      </accessor>
    </technique_common>
  </source>
  <source id="l_ankle_joint_mesh-normals" name="normal">
    <float_array id="l_ankle_joint_mesh-normals-array" count="1146">0.148756
        0.048333 -0.987692 ... 1</float_array>
    <technique_common>
      <accessor source="#l_ankle_joint_mesh-normals-array" count="382"
          stride="3">
        <param name="X" type="float"/>
        <param name="Y" type="float"/>
        <param name="Z" type="float"/>
      </accessor>
    </technique_common>
  </source>
  <source id="l_ankle_joint_mesh-map1" name="map1">
    <float_array id="l_ankle_joint_mesh-map1-array" count="878">0.95 0 0.95
        ... 0.975</float_array>
    <technique_common>
      <accessor source="#l_ankle_joint_mesh-map1-array" count="439" stride="
          2">
        <param name="S" type="float"/>
        <param name="T" type="float"/>
      </accessor>
    </technique_common>
  </source>
  <vertices id="l_ankle_joint_mesh-vertices">
    <input semantic="POSITION" source="#l_ankle_joint_mesh-positions"/>
    <input semantic="NORMAL" source="#l_ankle_joint_mesh-normals"/>
  </vertices>
```

```
35   <triangles material="blinn7SG" count="760">
36     <input semantic="VERTEX" source="#l_ankle_joint_mesh-vertices" offset="0
         "/>
37     <input semantic="TEXCOORD" source="#l_ankle_joint_mesh-map1" offset="1"
         set="0"/>
38     <p>0 0 1 ... 438</p>
39   </triangles>
40 </mesh>
```

Listing C.4: Collada <mesh>

<visual_scene>

Die hierarchische Struktur der `<visual_scene>` wird in einem Szenegraphen organisiert. Durch den Szenegraph wird eine optimale Verarbeitung und Renderung der Daten ermöglicht. Listing C.5 beschreibt eine Beispielszene: ein Skelett mit den dazugehörigen Drahtgittermodellen für die Gelenke und Knochen. Die Skelett-Topologie wird über die Objekte `<node>` vom Typ `JOINT` definiert. Diese Struktur bildet das Skelett aus Abbildung C.1 in COLLADA ab. `<node>`

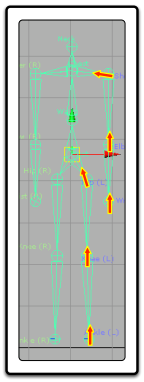


Abbildung C.1.: Abhängigkeiten innerhalb eines hierarchisch strukturierten Skeletts.

vom Typ `NODE` beinhalten dagegen Informationen zur grafischen Darstellung der Holzpuppe aus Abbildung 4.2. JOINT und NODE sind dabei Konstanten der COLLADA-Spezifikation. Jeder `<node>`-Knoten enthält Position und Orientierung. Über `<instance_geometry>` können dem Objekt bestimmte Geometriedaten aus der `<library_geometries>` Bibliothek zugewiesen werden. Dies geschieht über das *url*-Attribut des `<instance_geometry>`-Elementes.

```
<visual_scene id="VisualSceneNode" name="
    holzpuppe_skeleton_new_hierachy_scmoothed">
  <node id="pelvis_root" name="pelvis_root" type="JOINT">
    <translate sid="translate">-0.001043 4.63981 0</translate>
    <rotate sid="jointOrientZ">0 0 1 -61.5267</rotate>
    <rotate sid="jointOrientY">0 1 0 0</rotate>
    <rotate sid="jointOrientX">1 0 0 0</rotate>
    <node id="waist" name="waist" type="JOINT">
      ...
      <node id="l_wrist" name="l_wrist" type="JOINT">
        <translate sid="translate">1.39113 0.002608 -0.000683</translate>
        <rotate sid="jointOrientZ">0 0 1 -0.895174</rotate>
        <rotate sid="jointOrientY">0 1 0 -28.4733</rotate>
        <rotate sid="jointOrientX">1 0 0 90</rotate>
        <node id="l_wrist_joint" name="l_wrist_joint" type="NODE">
          <translate sid="translate">0.000622 -0.000307 -0.000158</translate
              >
          <rotate sid="rotateZ">0 0 1 61.5267</rotate>
          <rotate sid="rotateY">0 1 0 0</rotate>
          <rotate sid="rotateX">1 0 0 0</rotate>
          <scale sid="scale">0.814801 0.814801 0.814801</scale>
          <instance_geometry url="#l_wrist_joint_mesh">
            <bind_material>
              <technique_common>
                <instance_material symbol="blinn7SG" target="#dark_wood">
                  <bind_vertex_input semantic="TEX0" input_semantic="TEXCOORD
                      " input_set="0"/>
                </instance_material>
              </technique_common>
            </bind_material>
          </instance_geometry>
          <node id="l_hand_segment" name="l_hand_segment" type="NODE">
            <translate sid="translate">0.095233 -5.16902 1.01883</translate>
            <rotate sid="rotateZ">0 0 1 0</rotate>
            ...
            <scale sid="scale">1.22729 1.22729 1.22729</scale>
            <instance_geometry url="#l_hand_segment_mesh">
              ...
            </instance_geometry>
          </node>
        </node>
      </node>
      ...
    </node>
  </node>
</visual_scene>
```

Listing C.5: Collada <visual_scene>

<kinematics_model>

Seit Version 1.5 wurde COLLADA um das Konzept der *<kinematics_model>* erweitert. Dieses Element enthält kinematische Informationen wie Gelenke oder Verknüpfungen und wird unter *<library_kinematics_models>* zusammengefasst. `<library_kinematics_models>` kann dabei mehrere `<kinematics_model>`-Elemente besitzen.

`<kinematics_model>` beschränkt sich nur auf die kinematischen Beschreibungen ohne die physikalischen Parameter zu berücksichtigen. Mit `<kinematics_model>` ist es auch möglich, kinematische Ketten zu beschreiben. Über `<link>`-Elemente können definierte Gelenke verbunden werden. Dazu müssen diese Gelenke entweder über `<instance_joint>` unter `<kinematics_model>` oder über `<joint>` unter `<library_joints>` definiert werden. Listing C.6 zeigt die Verwendung von `<kinematics_model>` am Beispiel des rechten Arms.

```
<library_kinematics_models>
     <kinematics_model name="l_arm">
        <technique_common>
           <instance_joint name="Left Shoulder" sid="l_shoulder" url="#
               l_shoulder"/>
           <instance_joint name="Left Elbow" sid="l_elbow" url="#l_elbow"/>
           <instance_joint name="Left Wrist" sid="l_wrist" url="#l_wrist"/>
           <link name="chest_bone">
              <attachment_full joint="l_shoulder">
                 <link name="l_shoulder_bone">
                    <attachment_full joint="l_elbow">
                       <link name="l_elbow_bone">
                          <attachment_full joint="l_wrist">
                             <link name="l_wrist_bone"/>
                          </attachment_full>
                       </link>
                    </attachment_full>
                 </link>
              </attachment_full>
           </link>
        </technique_common>
     </kinematics_model>
</library_kinematics_models>
```

Listing C.6: Collada <library_kinematics_models>

<animation>

`<library_animations>` verwaltet hierarchische `animation`-Elemente, deren Ausführung hintereinander die eigentliche Animation darstellt. `animation`-Element beinhaltet die Key-Frame-Daten und den Interpolationsalgorithmus. Listing C.7 zeigt eine Beispielausprägung der Animations-Bibliothek.

```
<library_animations>
  <animation name="elliots_animations" id="all_elliot">
      <animation name="elliot's spells" id="spells_elliot">
          <animation id="elliot_fire_blast"/>
          <animation id="elliot_freeze_down"/>
          <animation id="elliot_ferocity"/>
      </animation>
     <animation name="jump" id="jump">
        <animation id="skeleton_root_translate">
            <source/><source/><sampler/><channel/>
        </animation>
        <animation id="left_hip_rotation">
            <source.../><source.../>
            <sampler>...</sampler><channel .../>
        </animation>
        <animation id="left_knee_rotation">
            <source .../><source .../>
            <sampler>...</sampler><channel .../>
        </animation>
        <animation id="right_hip_rotation">
            <source .../><source .../>
            <sampler>...</sampler><channel .../>
        </animation>
        <animation id="right_knee_rotation">
            <source .../><source .../>
            <sampler>...</    </animation>
      </animation>
  </animation>
</library_animations>
```

Listing C.7: Collada <library_kinematics_models>

D. Die grafische Benutzerschnittstelle

Die Illustration und Beschreibung der grafischen Benutzerschnittstelle kann als Anleitung zur Erstellung neuer Posen verwendet werden. Die Reihenfolge der Illustration ist identisch mit der Reihenfolge der Aufrufe bei der Annotation neuer Posen. Dabei werden einzeln alle Fenster und deren Funktionalität erklärt.

Anvil: Annotation Board

Um eine neue Pose erstellen zu können, muss ein neues Element über das *Annotation Board* hinzugefügt werden. Dies geschieht durch die Markierung der Start- und Endzeit (siehe Abbildung D.1).

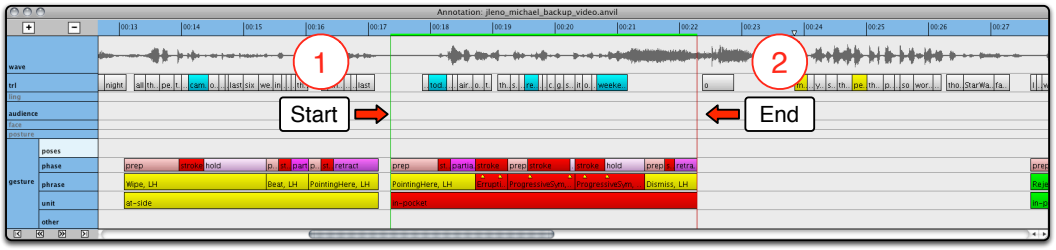


Abbildung D.1.: Das *Annotation Board* ist Ausgangspunkt jeder neuen Annotation. Mit Markierung des Start- und Endpunktes werden neue Elemente (Annotationen) hinzugefügt.

1 Startmarkierung

2 Endmarkierung

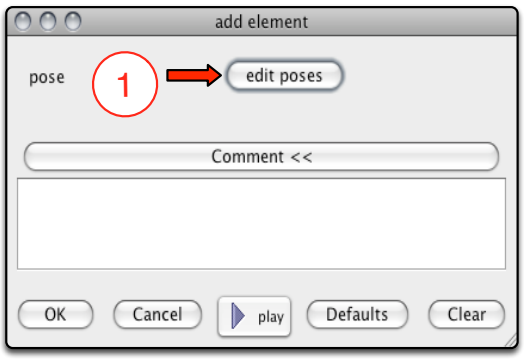


Abbildung D.2.: Über den Attribute Editor gelangt man zum *Pose Sequence Editor*.

Pose Sequence Editor

Der *Pose Sequence Editor* ist Ausgangspunkt neuer Posen sowie zur Veränderung und zum Löschen bereits existierender Posen. Neben den Funktionen zum Hinzufügen, Editieren und Löschen von Posen, stellt er auch Funktionen bereit, um im Video navigieren zu können. Zusätzlich bietet es die Möglichkeit, über Screenshots die annotierten Posen mit den Originalposen direkt zu vergleichen.

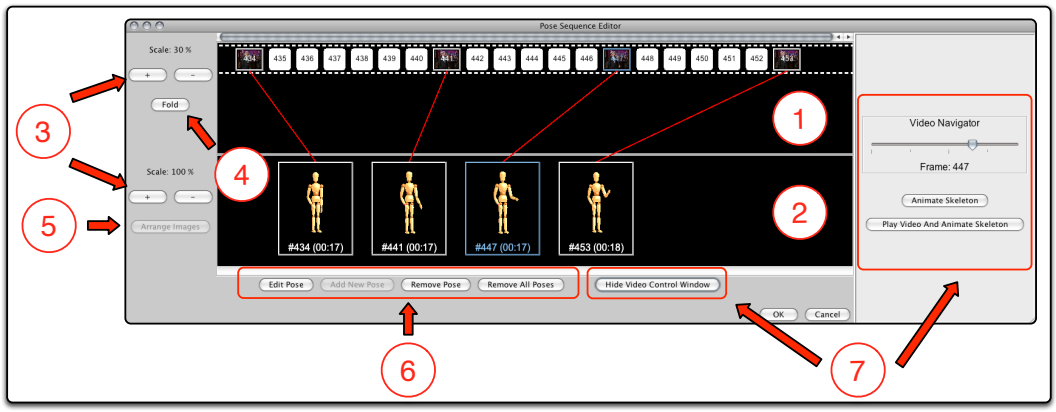


Abbildung D.3.: Pose Sequence Editor.

(1) Die obere Hälfte des *Pose Sequence Editors* stellt eine Übersicht der Videoframes dar. Noch nicht annotierte Frames sind mit weißem Hintergrund dargestellt. Frames, in denen eine Pose annotiert wurde, beinhalten einen Screenshot des dazugehörigen Videoframes. Diese Ansicht kann über die Zoom-Funktion verkleinert oder vergrößert werden, um eine bessere Übersicht zu erhalten. Zum besseren Vergleich der annotierten Posen mit den Originalposen, können alle nicht annotierten Frames ausgeblendet werden. Dabei werden die Screenshots

der Posen direkt untereinander angeordnet (siehe Abbildung D.4).

2 Die untere Hälfte des *Pose Sequence Editors* bildet die annotierten Posen ab. Auch diese Ansicht ist über die Zoom-Funktion separat veränderbar.

3 Über die markierten Buttons können die einzelnen Panels unabhängig voneinander vergrößert oder verkleinert werden.

4 Mit diesem Button werden nicht annotierte Frames aus- bzw. eingeblendet. Beim Ausblenden werden die unten dargestellten annotierten Posen direkt unter den zugehörigen Frames abgebildet. Dazu müssen die oberen Bilder mindestens genauso breit sein, wie die unteren Bilder.

5 Dieser Button dient zur Neuanordnung der unteren Screenshots. Er ist nur aktiv, wenn nicht annotierte Frames ausgeblendet wurden und die oberen Bilder mindestens die Breite der unteren Bilder besitzen (siehe Abbildung D.4).

6 Hier finden sich die Buttons zum Editieren, Löschen und Hinzufügen von Posen.

7 Über diesen Schalter wird die Videokontrolle eingeblendet, mit der man das Hauptvideo bedienen und die Animationen steuern kann (rot eingerahmt).

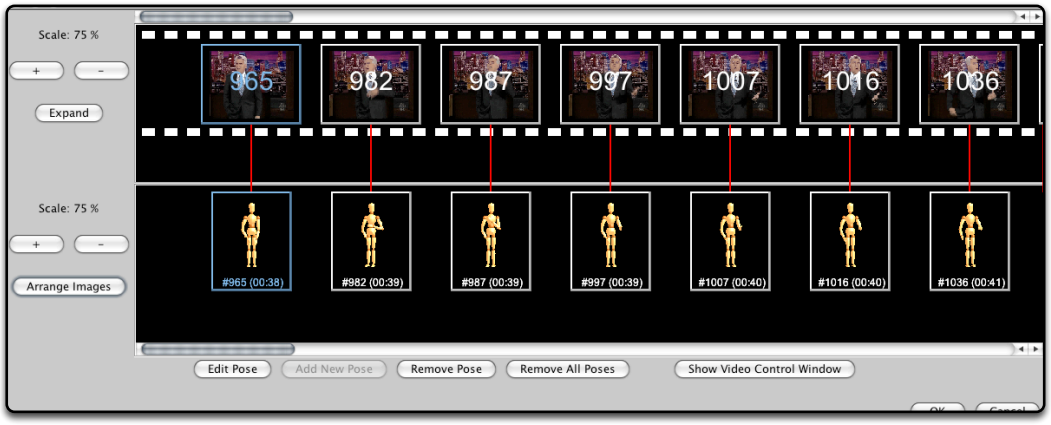


Abbildung D.4.: Der Pose Sequence Editor mit ausgeblendeten Frames und geordneter Ansicht der Screenshots.

Pose Editor

Der *Pose Editor* ist das Hauptfenster zur Erstellung neuer Posen und Bearbeitung vorhandener Posen. Er bietet Funktionen an, um die Arbeit bei der Erstellung von Posen zu erleichtern. Dazu gehört das Video-Screenshot-Panel, das Panel zur Veränderung der Kamera und ein Panel zur Auswahl der End-Effektoren.

Video-Screenshot-Panel

Das Video-Screenshot-Panel dient zur Justierung des Video-Screenshots im Pose Editor (Abbildung D.5).

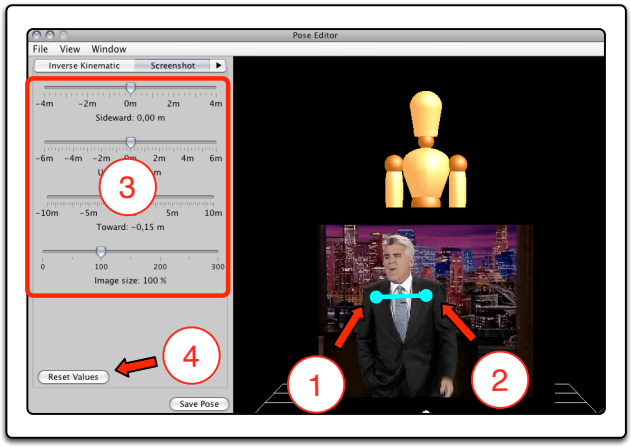


Abbildung D.5.: Einstellung des Video-Screenshot als Vorlage im Pose Editor.

1. Über die *Alt-Taste* wird das Bild in den Vordergrund gelegt, so dass man mit der linken Maustaste die rechte Schulter des Sprechers markieren kann.

2. Mit gedrückter *Alt-* und *Maus-Taste* wird die Schulter markiert. Zur Kontrolle wird eine Verbindungslinie zwischen den beiden Kontrollpunkten gezeichnet.

3. Über dieses Panel kann das Bild manuell nachjustiert werden. Es kann verschoben (x- und y-Richtung) und skaliert werden.

4. Über den *Reset-Button* können die Veränderungen wieder rückgängig gemacht werden.

End-Effektor-Panel

In diesem Panel können die verschiedenen End-Effektoren ausgewählt werden.

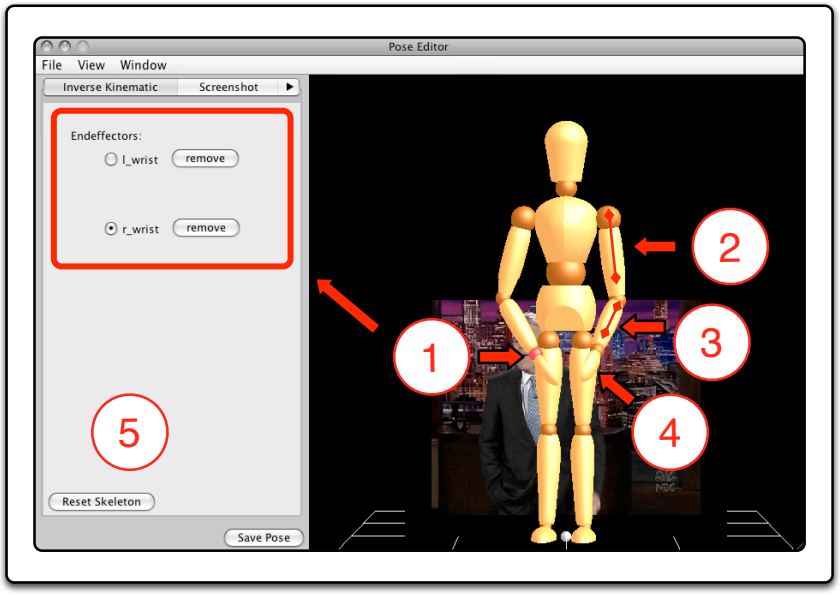


Abbildung D.6.: Erstellung der Pose im Pose Editor.

1. Die End-Effektoren können entweder über dieses Panel oder direkt über die Selektion des Handgelenks ausgewählt werden. Standardmäßig sind die Tasten „*l*“ für das linke Handgelenk und „*r*“ für das rechte Handgelenk als Shortcut vordefiniert.

2. Durch Auswahl der Schulter oder des Oberarms kann der *Swivel-Angle* (Positionierung des Ellbogens über die Winkelstellung der Schulter) verändert werden.

3. Durch Auswahl des Ellbogengelenks oder Unterarms kann der Ellbogen gebeugt werden.

4. Durch Auswahl der Hand kann die Orientierung der Handflächen über Rotation des Handgelenks verändert werden (ein Abwinkeln der Hände ist nicht möglich).

5. Mit dem *Reset-Button* kann das Skelett jeder Zeit in die neutrale Ausgangsposition versetzt werden.

Kamera-Panel

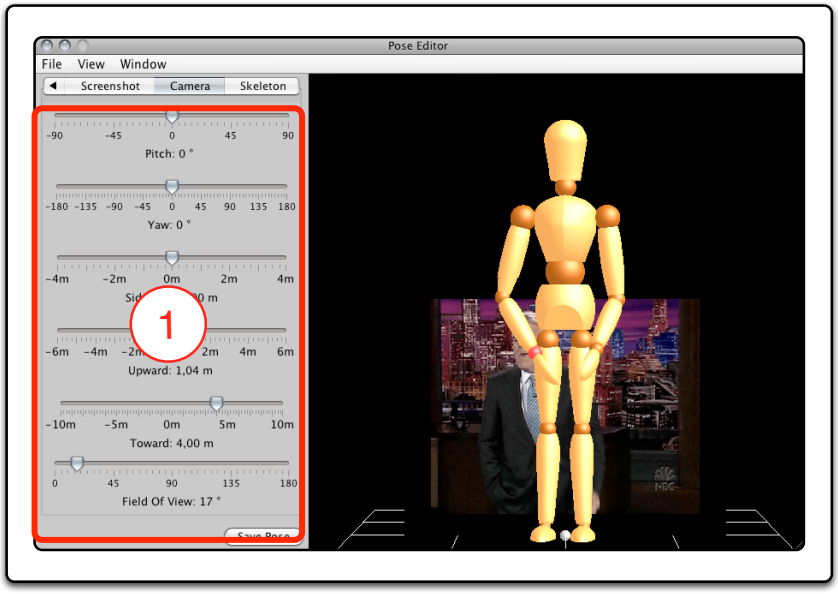


Abbildung D.7.: Panel zur Einstellung der Kamera im Pose Editor.

(1) Panel zur Veränderung der Kameraperspektive.

Pose Viewer

Der *Pose Viewer* dient zur Überwachung des Skeletts bei der Posenerstellung und Animation. Er hat drei separate Ansichten auf das Skelett und ermöglicht individuelle und unabhängige Veränderungen der einzelnen Ansichten (Abbildung D.8). Bei gedrückter *Alt*-Taste kann man die Kamera mit dem Scrollrad der Maus in die Szene rein- oder rausbewegen.

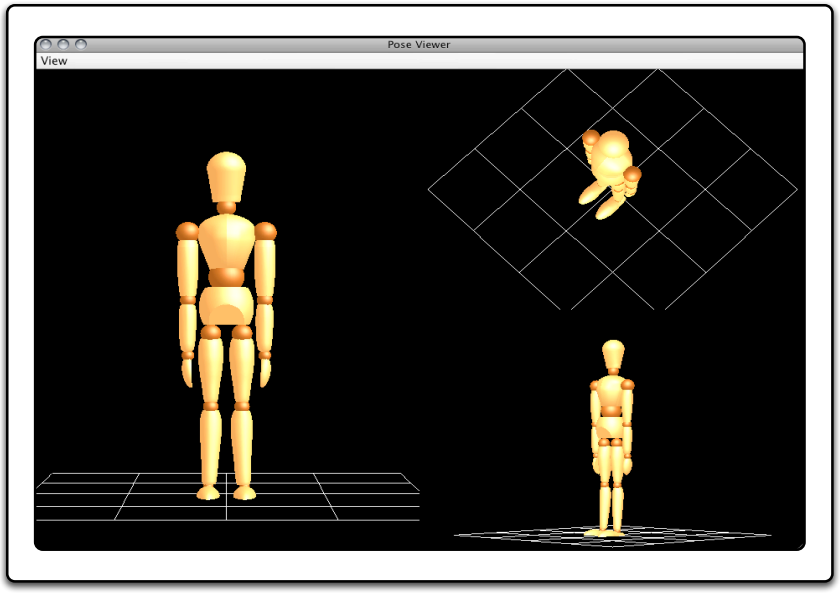


Abbildung D.8.: *Pose Viewer* mit drei verschiedene Ansichten auf die Szene.

Skeleton Markup Editor

Mit dem *Skeleton Markup Editor* können kinematische Ketten definiert werden (Abbildung D.9). Dieser Editor wird automatisch aufgerufen, wenn ein Skelett ohne kinematische Ketten geladen wird.

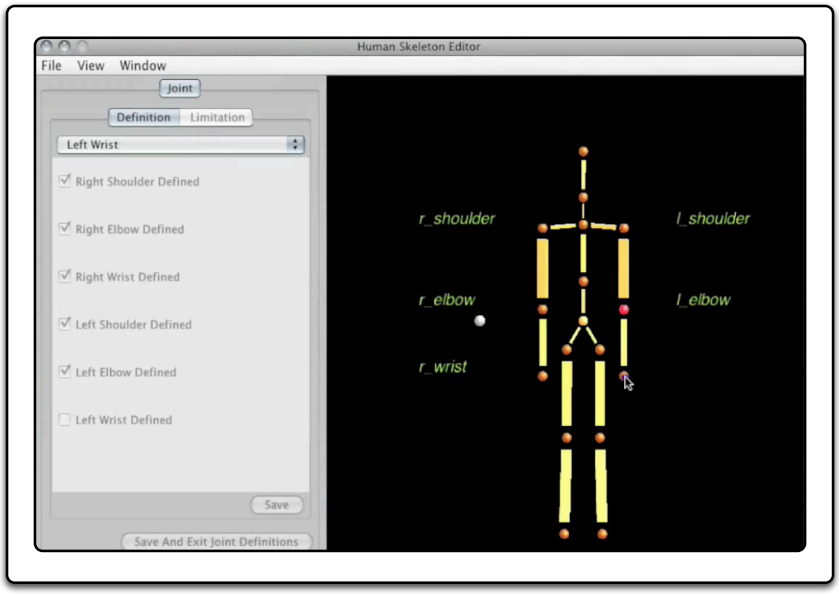


Abbildung D.9.: *Skeleton Markup Editor*

E. Detaillierter Workflow

Vorbereitung (Spezifikationsdatei anpassen): Um die Posenannotation in ANVIL zu aktiveren, muss in der Spezifikationsdatei neben dem Pfad zur Skelett-Topologie auch mindestens ein Track existieren, der das Attribut `TimestampedPose` beinhaltet.

Schritt 1 (Start- und Endzeit markieren):

- Markieren der Startzeit:
  - *Playline* (grüne Linie) mit linker Maustaste auf den gewünschten Zeitpunkt verschieben (darauf achten, dass man sich in der richtigen Spur befindet).
  - über rechte Maustaste Kontextmenü aufrufen und *Start* auswählen.
- Markieren der Endzeit:
  - *Recordline* (rote Linie) an den gewünschten Endzeitpunkt verschieben.
  - über rechte Maustaste Kontextmenü aufrufen und *End* auswählen. Daraufhin öffnet sich automatisch der *Attribute Editor* mit dem Attribut *pose*.

Schritt 2 (Pose Sequence Editor aufrufen):

- Wenn sich der *Attribute Editor* geöffnet hat, den Button *Edit Pose* klicken, um den *Pose Sequence Editor* aufzurufen.

Schritt 3 (Hinzufügen einer neuen Pose):

- Über den Button *Add New Pose* im *Pose Sequence Editor* den *Pose Editor* aufrufen.

Schritt 4 (Einstellung des Video-Screenshots):

- Mit der *Alt-Taste* das Hintergrundbild nach vorne holen. Solange die Taste gedrückt wird, bleibt das Bild im Vordergrund.
- Bei gedrückter *Alt-Taste* die rechte Schulter des Sprechers mit der linken Maustaste markieren.
- *Alt- und Maustaste* gedrückt halten und Mauszeiger zur linken Schulter bewegen.
- Wenn die linke Maustaste losgelassen wird, verschiebt und skaliert sich das Bild automatisch, so dass es sich direkt hinter dem Skelett befindet.

- Über den *Reset-Button* kann das Bild wieder in die Ausgangsposition und Ursprungsgröße zurückgesetzt werden.
- Bei Bedarf kann das Bild über die Schieberegler im linken Panel manuell angepasst werden.
- Mit der *Alt-Taste* kann das Bild jederzeit zur Kontrolle wieder in den Vordergrund verschoben werden.

Schritt 5 (Manipulation des Skeletts mit Hilfe von IK):

- Auswahl des End-Effektors über linkes Panel oder Shortcuts benutzen („*l*“ für linke Hand, „*r*“ für rechte Hand des Skeletts).
- Bei gedrückter linker Maustaste Arm bewegen.
- Über das Mausrad kann das Handgelenk in Richtung der z-Achse verschoben werden.
- Zur Feinjustierung können Schultergelenk, Ellbogen oder Hand rotiert werden.

Schritt 6 (Abspeichern der Pose):

- Pose über den *Save*-Button wird die neu erstellte Pose abgespeichert. Man gelangt wieder zum *Pose Sequence Editor*. Hier können dann erneut neue Posen erstellt oder die Animation der Geste kann von hier aus kontrolliert werden.

F. Weitere Abbildungen der Evaluation

	1	2	3	4	5	6	
	Sehr gut	Gut	Befriedigend	Ausreichend	Mangelhaft	Unbefriedigend	Mittelwert
ALLGEMEIN							
Verständlichkeit des Manuals	1	5	1				2
Subjektives Empfinden zur Einarbeitungszeit	6	2					1,25
Intuitivität der Bedienung des Tools	3	5					1,63
Arbeitsspaß	4	4					1,5
Zufriedenheit mit dem Endergebnis	4	3	1				1,63
Erweiterung sinnvoll	4	4					1,5
GESAMT Mittelwert							1,585
POSE SEQUENCE EDITOR							
Übersichtlichkeit	5	2	1				1,5
Bedienungsfreundlichkeit	5	3					1,38
Zweckerfüllung	8						1
Intuitivität	6	2					1,25
GESAMT Mittelwert							1,2825
POSE EDITOR (Video-Screenshot-Panel)							
Übersichtlichkeit	4	4					1,5
Bedienungsfreundlichkeit	3	5					1,63
Zweckerfüllung	6	2					1,25
Intuitivität	4	4					1,5
GESAMT Mittelwert							1,47
POSE EDITOR (End-Effektor-Panel)							
Übersichtlichkeit	4	2					1,33
Bedienungsfreundlichkeit	2	4	1				1,86
Zweckerfüllung	6	1					1,14
Intuitivität	4	3					1,43
GESAMT Mittelwert							1,44
POSE EDITOR (Kamera-Panel)							
Übersichtlichkeit	2	2	1				1,8
Bedienungsfreundlichkeit	1	3		1			2,2
Zweckerfüllung	3	2					1,4
Intuitivität	1	4					1,8
GESAMT Mittelwert	1						1,8
POSE Viewer							
Übersichtlichkeit	3	2					1,4
Bedienungsfreundlichkeit	5	1					1,17
Zweckerfüllung	5	1					1,17
Intuitivität	3	3					1,5
GESAMT Mittelwert							1,31

Abbildung F.1.: Diese Abbildung zeigt die Tabelle mit den Ergebnissen des Fragebogens.