

Erzeugung von VRML-Modellen aus medizinischen Volumendaten

Heidi Brockmann

Diplomarbeit

**Erstbetreuer: Prof. Dr. K.H. Höhne
Zweitbetreuer: Dipl.-Ing. Dr. W. Hansmann**

**Fachbereich Informatik
Universität Hamburg**

Hamburg, 2000

Danksagung

Diese Arbeit entstand als Diplomarbeit am Fachbereich Informatik der Universität Hamburg im Institut für Mathematik und Datenverarbeitung in der Medizin (IMDM) des Universitäts-Krankenhauses Hamburg-Eppendorf. Ich danke Herrn Prof. Dr. K.H. Höhne für die Betreuung der Arbeit und Herrn Dipl.-Ing. Dr. W. Hansmann für seine Unterstützung als Zweitbetreuer. Außerdem danke ich den Mitarbeitern des Instituts für Mathematik und Datenverarbeitung in der Medizin, insbesondere Herrn Dipl.-Inform. A. Pommert für die gute Zusammenarbeit und die Hinweise zu meiner Arbeit.

Zusammenfassung

Die Darstellung dreidimensionaler Objekte gewinnt im Bereich der Medizin zunehmend an Bedeutung. Sie ermöglicht neben der Veranschaulichung von intelligenten Volumendaten auch eine realistische Simulation von anatomischen und physiologischen Vorgängen (z.B. der Herzaktivität), die dem Betrachter einen unmittelbaren Zugang zu medizinischen Zusammenhängen ermöglicht.

Zur Darstellung von dreidimensionalen Objekten existiert eine Vielzahl von Visualisierungstechniken. Die Beschreibungssprache VRML ist jedoch das einzige plattformunabhängige Datenformat, das auch die Möglichkeit bietet, Objekte im Internet anzuzeigen und darin zu navigieren.

In dieser Arbeit wird auf der Basis des "Intelligenten Volumens" der VOXEL-MAN-Atlanten mit Hilfe des Marching Cubes-Algorithmus ein polygonales Oberflächenmodell für verschiedene medizinische Objekte erstellt. Zur Erzeugung eines VRML-Oberflächenmodells wird ein Konvertierungsprogramm entwickelt. Die Anwendung wird anhand dreier komplexer VRML-Modelle und einiger weiteren Beispielmodelle exemplarisch gezeigt, die anhand von Abbildungen in ihren einzelnen Schritten dargestellt werden. Abschließend erfolgt eine Bewertung des Entwicklungsprozesses sowie der Visualisierungsmöglichkeiten des zur Zeit gültigen VRML-Standards. Die im Rahmen dieser Arbeit erstellten interaktiven bzw. animierten VRML-Modelle werden auf ihre Zielsetzung hin untersucht und auf ihren praktischen Nutzen hin bewertet.

1.	Einleitung	1
1.1.	Problemstellung.....	1
1.2.	Zielsetzung	2
2.	Einführung in die Virtuell Reality Modelling Language (VRML) ...	4
2.1.	Entstehung von VRML	4
2.2.	Verfügbare Browser	5
2.3.	Das Konzept einer Szene	7
2.4.	Allgemeiner Aufbau einer VRML-Datei	9
2.4.1	Die Syntax in VRML	9
2.4.2	Definition von Datentypen und Feldinhalten.....	9
2.5.	Anforderungen an die Hard- und Software.....	12
3.	Stand der Forschung	13
3.1.	VRML-Anwendungen für spezielle Branchen und Berufsgruppen.....	13
3.2.	VRML-Modelle aus dem Bereich der Naturwissenschaften	14
3.3.	VRML-Projekte im Bereich der Medizin	17

Erzeugung von VRML-Modellen aus medizinischen Volumendaten	IV
3.3.1 Darstellungen mit selbstmodellierten Objekten	17
3.3.2 Darstellungen basierend auf medizinischen Datensätzen	21
4. Methode	25
4.1. VOXEL-MAN-Atlanten	27
4.2. Erzeugung von Polygonmodellen	28
4.2.1 Der Einsatz von Marching Cubes	28
4.2.2 Polygonreduzierung	29
4.3. Konvertierung intelligenter Volumenmodelle in eine VRML-Datei	31
4.3.1 Darstellung der Polygone durch Sets	32
4.3.2 Positionierung der Objekte.....	34
4.3.3 Beleuchtung, Farbgebung und Oberflächeneigenschaften.....	34
4.4. Bedienoberfläche.....	37
4.5. Generierung von Text und Hyperlinks.....	39
4.6. Gestaltung von Hintergrund in VRML	41
4.7. Erzeugung dynamischer VRML-Modelle.....	44
4.7.1 Interaktion	44
4.7.2 Animation.....	48
4.7.3 Viewpoints	49

5. Ergebnisse und Anwendungen	52
5.1. Einführende Experimente.....	52
5.2. Modell 1: Interaktive Darstellung von Gefäßen des Kopfes.....	57
5.3. Modell 2: Interaktive Darstellung von Schnittflächen des Kopfes	60
5.4. Modell 3: Interaktion und Animation komplexer Volumendaten.....	65
6. Schlußfolgerung und Ausblick	69
Literaturverzeichnis	72
Abbildungsverzeichnis	78
Glossar	80

1. Einleitung

Vor der Entwicklung der Virtual Reality Modeling Language (VRML) boten zahlreiche Virtual Reality (VR) Systeme die Möglichkeit, dreidimensionale Objekte darzustellen sowie durch geeignete Hardware in dieser definierten Welt zu navigieren. Jedes VR-System hatte dabei sein eigenes Dateiformat. Die Unterschiede waren nicht nur syntaktischer Natur, sondern sie waren schon durch die Grundkonzeption bedingt, was es schwer machte, die Daten unterschiedlicher Systeme auszutauschen. Die Schwierigkeiten, die die unterschiedlichen VR-Formate mit sich brachten, verdeutlichten die Notwendigkeit eines Standards.

Das Internet und somit das World Wide Web (WWW) erfuhren in den letzten Jahren einen sprunghaften Anstieg der Nutzung, der einen weiteren wichtigen Faktor für die Entstehung von VRML darstellte. Die Verschmelzung zwischen virtueller Realität und dem Internet mit einer konzeptionellen Äquivalenz zur Hypertext Markup Language (HTML) beinhaltete ein gewaltiges Wachstums- und Anwendungspotential.

Die *Virtual Reality Modeling Language* (VRML), ist eine Beschreibungssprache für interaktive dreidimensionale Welten im World Wide Web (WWW), die sich zum Internet-Standard für dreidimensionale Darstellungen entwickelt hat. Dreidimensionale Darstellungen können als VRML-Objekte ins Internet gestellt und somit medial nutzbar gemacht werden. Die Plattformunabhängigkeit von VRML sowie die kostengünstige Nutzung der benötigten Software bereiten diesem Web-basierten Standard eine erfolgreiche Zukunft und ein weit verbreitetes Anwendungsfeld.

1.1. Problemstellung

Die Grundlage zur Erzeugung von Modellen aus medizinischen Volumendaten mit Hilfe der Virtual Reality Modeling Language (VRML) bildet das Programm VOXEL-MAN, das mit seinen umfangreichen Atlanten den Ausgangspunkt für diese Arbeit darstellt. Zur Darstellung von VRML-Modellen im medizinischen Bereich muß jedoch noch ein Algorithmus für eine Methode entwickelt werden, die den detaillierten Werdegang von den Daten der VOXEL-MAN-Atlanten zur Gewinnung eines Polygonmodells bis hin zum VRML-Oberflächenmodell beschreibt.

Während der Marching Cubes-Algorithmus verwendet wird, um Polygondaten aus den Volumenmodellen der VOXEL-MAN-Atlanten zu erzeugen, ist ein Konvertierungsprogramm erforderlich, das aus diesen Polygondaten ein VRML-Oberflächenmodell erstellt.

1.2. Zielsetzung

Neben der Entwicklung eines Konvertierungsprogramms sollen im Laufe der Untersuchungen folgende Fragen beantwortet werden:

- Welche Interaktions- und Animationsmöglichkeiten bietet VRML ?
- Welche Probleme erzeugt die Komplexität der erstellten Polygondaten für VRML ?
- Wie weit lassen sich die Polygondaten ohne Verlust der realistischen Darstellungskraft reduzieren ?
- In wieweit sind komplexe Oberflächenmodelle in VRML navigierbar ?
- Ist es sinnvoll, Texturen einzusetzen und kann durch Texturen ein realistischer Eindruck vermittelt werden ?

Die Arbeit ist wie folgt gegliedert:

Kapitel 2 beinhaltet eine Einführung in die Virtual Reality Modeling Language. Die Entstehung von VRML wird in Unterkapitel 2.1 erläutert, während das Unterkapitel 2.2 auf die Verfügbarkeit und Anwendung der VRML-Browser eingeht. In diesem Zusammenhang geht das Unterkapitel 2.5 auf die Anforderungen an die Hardware und Software ein. Die hierarchische Anordnung von Objekten, die eine VRML-Szene bestimmen sowie die Vererbung von Objekteigenschaften ist Gegenstand von Unterkapitel 2.3. Die Syntax und Semantik der Beschreibungssprache VRML ist Inhalt des Unterkapitels 2.4.

Kapitel 3 stellt den aktuellen Stand der Forschung dar. VRML hat in allen Bereichen Einzug gehalten, die mit Informationsverarbeitung und Visualisierung arbeiten. Die ständig wachsende Popularität von VRML und deren Einbindung ins World Wide Web führte zu einer sehr umfangreichen Forschungstätigkeit und zu Forschungsprojekten, die in dieser Arbeit nur ansatzweise erläutert werden können. Lediglich auf den Bereich der medizinischen Anwendungen wird in dieser Arbeit detaillierter eingegangen.

Das Kapitel 4 mit dem Titel "Methode" erläutert die Entwicklungsschritte, die ausgehend vom VOXEL-MAN bis hin zum VRML-Oberflächenmodell erforderlich sind. Unterkapitel 4.1 geht allgemein auf den Zusammenhang dieser Arbeit mit dem "Intelligenten Volumenmodell" des VOXEL-MAN-Programms ein.

Die Generierung von Dateien aus den zugrunde liegenden Datensätzen erfolgt mit Hilfe des Marching Cubes-Algorithmus, der im Unterkapitel 4.2 erläutert wird. Die Konvertierung intelligenter Volumenmodelle und deren Darstellung in VRML ist Gegenstand von Unterkapitel 4.3. Zur einfacheren Handhabung ist in dieser Arbeit eine Bedienoberfläche entwickelt worden, die in Unterkapitel 4.4 näher erläutert wird. Die Kombination eines VRML-Oberflächenmodells im Zusammenhang mit Text und Hyperlinks wird im Unterkapitel 4.5 dargelegt. Die Erzielung von räumlichen Effekten durch die Gestaltung von geeigneten Hintergrundinformationen liegt dem Unterkapitel 4.6 zugrunde. Die durch Interaktion und Animation erzeugte Dynamik in VRML wird in Unterkapitel 4.7 erläutert.

Die Erläuterungen der im Rahmen dieser Diplomarbeit entstandenen VRML-Anwendungen sind Gegenstand von Kapitel 5. Der Werdegang hin zu komplexeren VRML-Modellen wird im Unterkapitel 5.1 dargelegt. Die Darstellung sämtlicher Gefäße des Kopfes kombiniert mit Interaktionsmöglichkeiten wird im Unterkapitel 5.2 erläutert und ausgewertet. Die Möglichkeit Schnitte des Kopfes interaktiv als VRML-Anwendung darzustellen ist die Zielsetzung des Unterkapitels 5.3. Die Problematik komplexe Volumenmodelle in VRML interaktiv sowie animiert darzustellen wird in Unterkapitel 5.4 erläutert und ausgewertet. Die beiliegende CD-ROM beinhaltet die in dieser Arbeit entstandenen und beschriebenen VRML-Modelle und VRML-Beispieldateien sowie zur Darstellung den Cosmo Player als Plug-In unter Netscape.

Die Schlußbetrachtung und der Ausblick sind Inhalt des 6. Kapitels. Im Anschluß an das Literaturverzeichnis findet sich ein Glossar, in dem die wichtigsten Begriffe im Zusammenhang mit VRML erläutert werden.

2. Einführung in die Virtuell Reality Modelling Language (VRML)

Die Virtual Reality Modeling Language (VRML) ist eine Beschreibungssprache für die Modellierung von interaktiven 3D-Welten und -Objekten. VRML ist in der Lage, statische und animierte Objekte zu repräsentieren und kann Hyperlinks zu anderen Medien wie Sound und Video sowie zu HTML-Text herstellen. Sowohl Browser [Seidman] für VRML als auch Autorenwerkzeuge [Couch] [Roehl] für die Erstellung von VRML-Dateien sind für viele Plattformen verfügbar.

2.1. Entstehung von VRML

Mark Pesce und Toni Parisi entwickelten die Idee eines Datenformates, das es ermöglichen sollte, 3D-Objekte oder -Welten im World Wide Web (WWW) herunterzuladen und anschließend darin zu navigieren. Der Name dieser neuen Sprache wurde durch HTML (*Hyper-Text Markup Language*) inspiriert und lautet VRML, was zunächst *Virtual Reality Markup Language* bedeutete, später aber zumeist mit *Virtual Reality Modeling Language* übersetzt wurde. Das entsprechende Datenformat bekam den Dateisuffix *.wrl* [Matsuba 98][Kloss 98].

Der von Pesce und Parisi entwickelte erste 3D-Browser namens "Labirinth" wurde 1994 auf der WWW-Konferenz in Genf vorgestellt und so den Interessenten zugänglich gemacht.

Nachdem eine VRML-Spezifikation vorlag, die im Oktober 1994 auf der zweiten WWW-Konferenz in Chicago als Entwurfsversion vorgestellt und anschließend von der VRML-Gemeinschaft diskutiert und verbessert wurde, kam im April 1995 die erste offizielle Version der VRML1.0 Spezifikation heraus. Zu diesem Zeitpunkt standen bereits einige Browser als Prototypen zur Verfügung [Kloss 98].

Ende 1995 wurde von den Firmen Sony und SGI sowie Mitgliedern der *VRML Architecture Group* (VAG) [Web3D Consortium] ein Vorschlag für VRML2.0 entwickelt. Dieser Vorschlag beinhaltete Erweiterungen zur Spezifikation von VRML1.0 und legte den Grundstein für den zukünftigen VRML-Standard. Mit Erscheinen der Spezifikation VRML2.0 wurde auch die Gründung des *VRML-Konsortiums* [Web3D Consortium] in den Weg geleitet. Hieran beteiligten sich 35 Firmen und Organisationen, darunter alle großen Software Firmen wie SGI, Netscape, SUN und Microsoft.

Im April 1997 wurde die Spezifikation unter dem Namen VRML97 zum *Draft International Standard 14772* der *International Standards Organization* erklärt. Im September 1997 wurde dieser vorläufige Entwurf endgültig als *International Standard (ISO 14772)* akzeptiert.

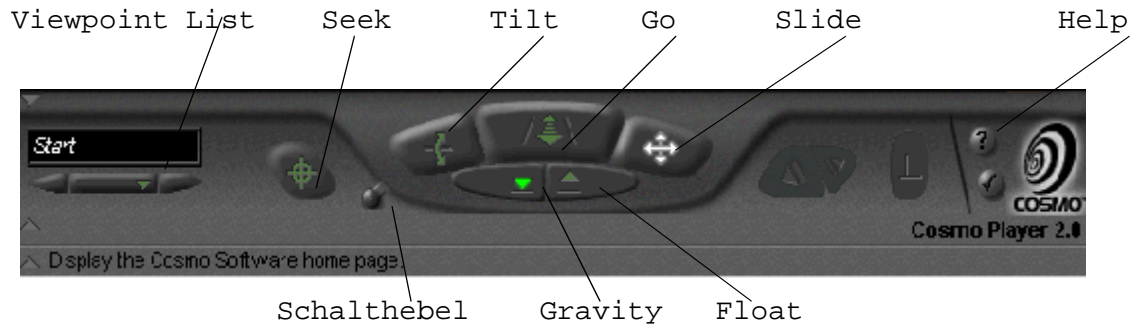
Damit ist VRML97 nun ein weltweit anerkannter und akzeptierter ISO-Standard und somit eine verlässliche Basis für Entwicklungen aller Art [VRML Org.].

2.2. Verfügbare Browser

Browser zur Darstellung von VRML-Dateien können unentgeltlich aus dem Internet geladen werden [Seidman]. Die meisten Browser sind als Plug-Ins unter Netscape oder dem Internet Explorer verfügbar. Nur wenige VRML-Browser, wie z.B. der VRWave Browser, sind als Standalone-Lösung implementiert. Es gibt viele Vergleichs- und Testseiten zum Thema VRML-Browser aus denen detaillierte Informationen zur Arbeitsweise des Browsers sowie zur Umsetzung des VRML-Standards entnommen werden können.

VRML selbst ist unabhängig von der Wahl des Browsers. Mit Ausnahme des *Cosmo Players* von der Firma Silicon Graphics, dem *WorldView Browser* der Firma Intervista sowie dem *Community Place Browser* der Firma Sony [VRML-Browser] unterstützen aber noch nicht alle Browser die vollständige Spezifikation von VRML.

In dieser Arbeit werden alle Modelle mit dem PC-basierten Cosmo Player Browser unter Netscape angezeigt. Die Navigationsmöglichkeiten des Cosmo Player Browsers werden in der Abbildung 1 erläutert.



GO: bewegen in horizontaler Ebene.

SLIDE: bewegen in vertikaler Linie.

TILT: Änderung des Blickwinkels, entspricht einer Drehung in allen Ebenen, die senkrecht zur Vertikalen stehen.

GRAVITY: hält den Benutzer am Boden, d.h. Höhen und Tiefen werden automatisch angepaßt.

FLOAT: fliegen und abheben.

SEEK: durch Anklicken des Objektes bewegt man sich automatisch näher an das Objekt heran.

VIEWPOINT: Text Display, enthält die Namen unterschiedlicher Blickpunkte.

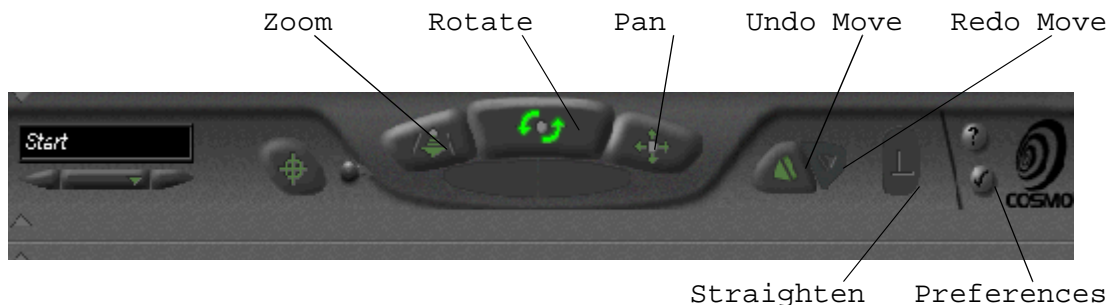
SCHALTHEBEL links neben Hauptboard (change control): ändert die Interaktionsschalter.

UNDO/REDO MOVE: letzte Bewegung rückgängig bzw. wiederholt.

STRAIGHTEN: bringt Betrachter in aufrechte Position, relativ zur horizontalen Ebene.

PREFERENCES: Aktivierung des Voreinstellungsmenüs.

Nach Betätigung des Schalthebels erfolgt ein Wechsel vom Modus "Walk" in den Modus "Examine"



ZOOM:	bewegt Objekt heran oder weg.
ROTATE:	läßt Objekte rotieren.
PAN:	Objekt im Zentrum, Viewer umrundet Objekt.

Abb. 1: Navigationsfunktionalität des Cosmo Player Browsers

2.3. Das Konzept einer Szene

Eine in VRML bestehende Polygonszene besteht aus einer Aufzählung ihrer Objekte, die hierarchisch angeordnet sind. Diese Hierarchie wird als *Szenegraph* bezeichnet und legt die Beziehung der Objekte untereinander fest. Bei einer Polygondarstellung beschreiben ein oder mehrere *Knoten* (engl. Nodes) die Geometrie des Objektes, weitere Knoten wiederum beschreiben die Materialeigenschaften wie z.B. die Farbe und das Licht.

Ein Knoten besteht aus einem oder mehreren *Feldern* (engl. Fields), die die Werte enthalten, welche den Zustand des jeweiligen Knotens parametrisch beschreiben. In einer Polygondarstellung enthalten die Felder eines Geometrienknotens die Punkte der Polygone sowie deren Indizes, die wiederum die Flächen der Polygone beschreiben. Die Zusammenfassung der Knoten einer Polygonszene, innerhalb der Hierarchie, wird durch *Gruppenknoten* (engl. Grouping Nodes) ermöglicht.

Um Interaktion und Animation in einer VRML-Datei einzubauen, werden Ereignisse (engl. Events) verwendet, die auf Knoten einwirken und Feldinhalte verändern können. Die Bearbeitung von Ereignissen in einem Knoten werden durch die Datentypen *eventIn*, *eventOut* und *exposedField* ermöglicht. Das Weiterreichen von Ereignissen zwischen Knoten übernehmen entsprechende *ROUTE* Befehle [Kloss 98].

Mögliche Knotentypen sind:

- Gruppenknoten
- Umgebungsknoten
- Geometrienknoten
- Knoten für Geometrieigenschaften
- Sensorknoten
- Interpolatorknoten
- Knoten für spezielle Verwendungen

Für die Erzeugung von Interaktionen, z.B. durch Bewegungen des Mauszeigers, gibt es in VRML eine Reihe von *Sensor-Knoten*, während vordefinierte Animationssequenzen mit Hilfe von *Interpolatoren* erzeugt werden können. Mit *Umgebungsknoten* lassen sich z.B. Kameransichten oder der Hintergrund beschreiben. Ebenso lassen sich hiermit Teile der Browserfunktionalität abschalten. *Geometrieknoten* umfassen die Standardgeometrien wie Würfel, Kugel, Zylinder und Kegel sowie selbstdefinierte Polygone, während unter *Geometrie Eigenschaftsknoten* untergeordnete Geometrieknoten zu verstehen sind. Unter *Knoten für spezielle Verwendungen* sind Knoten zu finden, die eine Schnittstelle zu anderen Programmiersprachen darstellen sowie Knoten mit denen sich z.B. Schalter realisieren lassen.

Die Abbildung 2 zeigt eine mögliche Knotenstruktur einer VRML-Polygonszene:

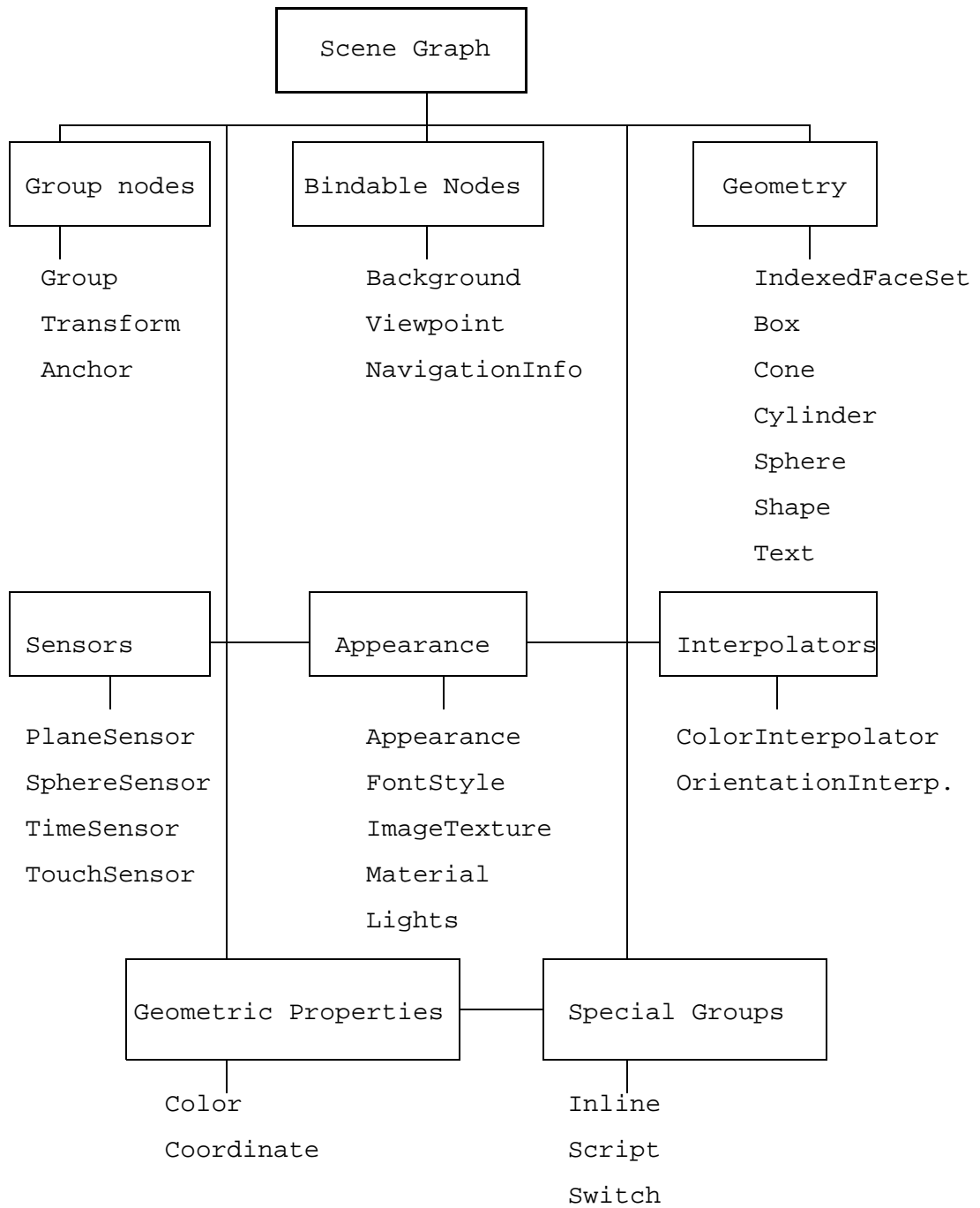


Abb. 2: Elemente eines möglichen Szenegraphen

2.4. Allgemeiner Aufbau einer VRML-Datei

Eine VRML-Szene entspricht einem gerichteten azyklischen Graphen. Dabei können Knoten andere Knoten enthalten sowie mehrfach in unterschiedlichen Knotentypen integriert sein. Eine Rekursion von gleichen Knotentypen ist möglich, aber nicht notwendig. Im folgenden wird auf die Details einer VRML-Datei näher eingegangen.

2.4.1. Die Syntax in VRML

Es existieren einige formale und syntaktische Anforderungen durch die sich eine Datei im VRML2.0 Format auszeichnet. Jede VRML-Datei beginnt mit einem Header, der den Browser über die Version der verwendeten Syntax informiert. Dieser lautet für die derzeit aktuelle Version:

#VRML V2.0 utf8.

Das Kürzel **utf8** steht für das dem Standard ISO 10646 entsprechende Kodierungsverfahren UTF-8, das die Darstellung des internationalen Zeichensatzes in VRML sicherstellt und eine Obermenge zum Unicode darstellt. Im Header darf nach dem „#“ Zeichen kein Leerzeichen stehen, da der VRML-Viewer ansonsten den Header als Kommentar interpretiert [Kloss 98].

Nach dem Header folgen Knoten, die Felder und Feldinhalte beinhalten, durch die die gewünschte Szene beschrieben wird.

In der VRML-Syntax wird die Groß- und Kleinschreibung unterschieden. Knotennamen müssen immer mit einem Großbuchstaben beginnen, während die Namen von Feldern und Ereignissen immer klein geschrieben werden müssen. So können auf den ersten Blick Knoten von Feldern unterschieden werden.

Feld-, Ereignis-, Prototyp- und Knotennamen dürfen nicht mit einer Zahl beginnen. Sie dürfen aber jedes druckbare ASCII-Zeichen außer (,“ ‘ # + , - . [] { und }) enthalten [Carey 97].

Der Definitionsteil eines Knotens wird in geschweiften Klammern eingeschlossen, während der Deklarationsteil von Feldern in eckigen Klammern eingeschlossen wird.

2.4.2. Definition von Datentypen und Feldinhalten

Die Feldefinition ist Bestandteil eines Knotens und beinhaltet den Feldtyp, den Datentyp und den Feldbezeichner, der jeweils mit einem Defaultwert belegt ist.

Es existieren insgesamt vier verschiedene Feldtypen: *eventIn* und *eventOut* dienen ausschließlich der Dynamisierung von Szenen. Der Feldtyp *exposedField* kann ebenfalls Bestandteil dynamischer Verfahren sein, da dessen Werte zur Laufzeit verändert werden können.

In statischen Szenen übernimmt *exposedField* jedoch die gleiche Funktion wie der zur Laufzeit nicht manipulierbare Feldtyp *field* [Carey 97].

Zu Erläuterung werden in Abbildung 3 einige Syntaxbeispiele dargestellt.

Feldtyp	Datentyp	Feldbezeichner	Default
Transform {			
eventIn	MFNode	addChildren	
eventIn	MFNode	removeChildren	
exposedField	SFVec3f	center	0 0 0
exposedField	MFNode	children	[]
exposedField	SFRotation	rotation	0 0 1 0
exposedField	SFVec3f	scale	1 1 1
exposedField	SFRotation	scaleOrientation	0 0 1 0
exposedField	SFVec3f	translation	0 0 0
field	SFVec3f	bboxCenter	0 0 0
field	SFVec3f	bboxSize	-1 -1 -1 }
IndexedFaceSet {			
eventIn	MFInt32	set_colorIndex	
eventIn	MFInt32	set_coordIndex	
eventIn	MFInt32	set_normalIndex	
eventIn	MFInt32	set_texCoordIndex	
exposedField	SFNode	color	NULL
exposedField	SFNode	coord	NULL
exposedField	SFNode	normal	NULL
exposedField	SFNode	texcoord	NULL
field	SFBool	ccw	TRUE
field	MFInt32	colorIndex	[]
field	SFBool	colorPerVertex	TRUE
field	SFBool	convex	TRUE
field	MFInt32	coordIndex	[]
field	SFFloat	creaseAngle	0
field	MFInt32	normalIndex	[]
field	SFBool	normalPerVertex	TRUE
field	SFBool	solid	TRUE
field	MFInt32	texCoordIndex	[] }

Abb. 3: Beispiele zur Syntax der Knoten

Wie bei anderen Programmiersprachen existieren in VRML viele verschiedene Datentypen, die jeweils das optimale Format für die mit ihnen durchgeführten Operationen besitzen.

Der Datentypbezeichnung vorangestellte SF (single-valued) bzw. MF (multiple-valued) gibt Auskunft darüber, ob im spezifizierten Feld ein einziger oder mehrere Werte desselben Datentyps aufgeführt sein können.

In dem folgenden Beispiel werden einige grundlegende Datentypen dargestellt:

SFNode / MFNode	:	VRML-Knoten
SFBool	:	Werte TRUE oder FALSE
SFColor / MFColor	:	Drei Fließkommawerte zwischen 0.0 und 1.0 nachdem RGB-Modell
SFint32 / MFInt32	:	32-Bit Integerzahl
SFRotation / MFRotation	:	Vier Werte, von denen die ersten drei den Vektor der Rotationsachse und der letzte Wert den Rotationswinkel (im Radiant) um diese Achse bilden.
SFString / MFString	:	String (utf8)
SFVec2f / MFVec2f	:	Zwei Fließkommazahlen (2D Vektor)
SFVec3f / MFVec3f	:	Drei Fließkommazahlen (3D Vektor)
SFFloat / MFFloat	:	Fließkomma Zahl(en) mit einfacher Genauigkeit
SFImage	:	Pixel Beschreibung einer Image-Map
SFTime / MFTime	:	Anzahl Sekunden seit dem 1. Januar 1970, 00:00:00

Abb. 4: Darstellung grundlegender Datentypen

Die VRML Architecture Group (VAG) hat durch den geschaffenen Standard VRML97 die vollständige Spezifikation aller Knoten dargelegt [Web3D Consortium].

Mit der Angabe der jeweiligen Feldbezeichner und den entsprechenden Defaultwerten wird die Spezifikation eines Knotens vervollständigt.

Während des Entwurfs einer Szene wird überwiegend mit Feldbezeichnern gearbeitet, denen situationsbedingt neue Werte zugeordnet werden. Um einen Defaultwert zu überschreiben, muß dem Feld ein Ersatzwert vom gleichen Datentyp zugewiesen werden [Nadeau 99].

2.5. Anforderungen an die Hard- und Software

VRML ist eine plattformunabhängige Sprache. Bei der Anwendung von VRML sollten jedoch einige Hardware-Mindestanforderungen berücksichtigt werden, die in [Kloss98] beschrieben werden.

- Zu Darstellung von VRML-Standardszenen für Personal-Computer (PCs) liegt die Mindestanforderung bei einem 100 MHz getakteten 486 und mindestens 16 MB Hauptspeicher, um gleichmäßige „Bild zu Bild“ Bewegungen zu erhalten. Ein Pentium mit 32 MB Hauptspeicher würde den Anforderungen jedoch besser genügen.
- Viele VRML-Browser benötigen 256 Farben als Minimum, so daß überprüft werden muß, ob der Bildschirm die Anzahl der Farben umsetzen kann [Matsuba 96] [Kloss98].

Für die Darstellung von komplexen VRML-Anwendungen reichen die o.g. Anforderungen bei weitem nicht aus wie weiterhin noch gezeigt wird. Je größer und leistungsstärker der Rechner desto schneller wird eine komplexe VRML-Darstellung angezeigt.

Die Grenzen einer komplexen Polygondarstellung in VRML sind bei maximal 10 Punkten pro Fläche und 15000 Indizes in einem Index Feld erreicht. Ein übergeordneter Gruppenknoten kann maximal 500 Kindknoten, z.B. Polygonddefinitionen, beinhalten. Das Limit bei der Farbzweisung innerhalb einer VRML-Datei ist bei 15000 Farben erreicht [Carey 97].

Schon dieser kurze Auszug aus den maximalen Darstellungsmöglichkeiten von VRML macht deutlich in welchem Umfang komplexe VRML-Dateien erstellt werden könnten.

An Software wird im einfachsten Fall neben dem schon erwähnten VRML-Browser sowie einem WWW-Browser lediglich ein einfacher Texteditor zur Erzeugung einer VRML-Szene benötigt. Ein VRML-Autorensystem bietet jedoch komfortablere Möglichkeiten, VRML-Szenen via Mausclick zu gestalten [Couch] [Roehl].

3. Stand der Forschung

Es gibt bereits eine beachtliche Anzahl von VRML-Anwendungen oder -Welten im Internet. Viele unterschiedliche Branchen wie z.B. die Industrie (Architektur, Werbung und Design, Autoindustrie, Luft und Raumfahrt etc.) oder Dienstleistungsbereiche (Immobilienhandel, Banken und Finanzdienstleistungen, Marketing, Internet-Service-Provider etc.), haben die Möglichkeit für sich entdeckt dreidimensionale interaktive Graphiken im World Wide Web einzubinden. Die Simulationseigenschaften von VRML sind aber auch für wissenschaftliche Anwendungen sehr interessant. Durch die Vielfalt des Angebotes im Internet kann im folgenden nur auf eine begrenzte Auswahl interessanter VRML-Modelle eingegangen werden.

3.1. VRML-Anwendungen für spezielle Branchen und Berufsgruppen

Die Autoindustrie hat schnell erkannt, daß sich VRML hervorragend für die Produktpräsentation im Internet anbietet. Für potentielle Kunden oder Fans ergibt sich dadurch die Möglichkeit, neue Modelle von außen sowie deren Innenausstattung visuell zu begutachten. Als Beispiel sei auf das VRML-Gittermodell der C-Klasse von Mercedes hingewiesen [Mercedes].

Aus dem Bereich der Architektur und Innenarchitektur werden zahlreiche Darstellungen im Internet gezeigt. VRML bietet neben den Ansichten von Häusern, die meist im zweidimensionalen Bildformat dargestellt werden, die Möglichkeit, einen Rundgang durch das Innere des Hauses vorzunehmen. Als Beispiel sei die Knowsley-Bibliothek genannt [Knowsley]. Banken und Finanzdienstleistungen haben einen erhöhten Bedarf an Visualisierungen von Wirtschaftsdaten. Die Darstellung von Wirtschaftsdaten wird zum größten Teil mit dreidimensionalen Graphiken realisiert. VRML bietet nicht nur die Möglichkeit dreidimensionale Objekte darzustellen, sondern ebenso die Möglichkeit, diese Daten medial zu nutzen, indem sie in eine entsprechende Internetseite eingebunden werden. Die Firma Visual Decision In3D (VDI) hat sich unter anderem auf die Visualisierung von wirtschaftlich orientierten Diagrammen spezialisiert und speichert diese auch im VRML-Format ab [Visual Decision].

Auch im Bereich der darstellenden Kunst werden zahlreiche Beispiele in VRML dargestellt. Eine Erweiterung der darstellenden Kunst ist nicht nur die Darstellung von dreidimensionalen Objekten, sondern das Hinzufügen von Bewegungen und Animationssequenzen. Es sei in diesem Zusammenhang der tanzende Roboter aus Japan (siehe Abbildung 5) als Beispiel genannt [Kikuro].

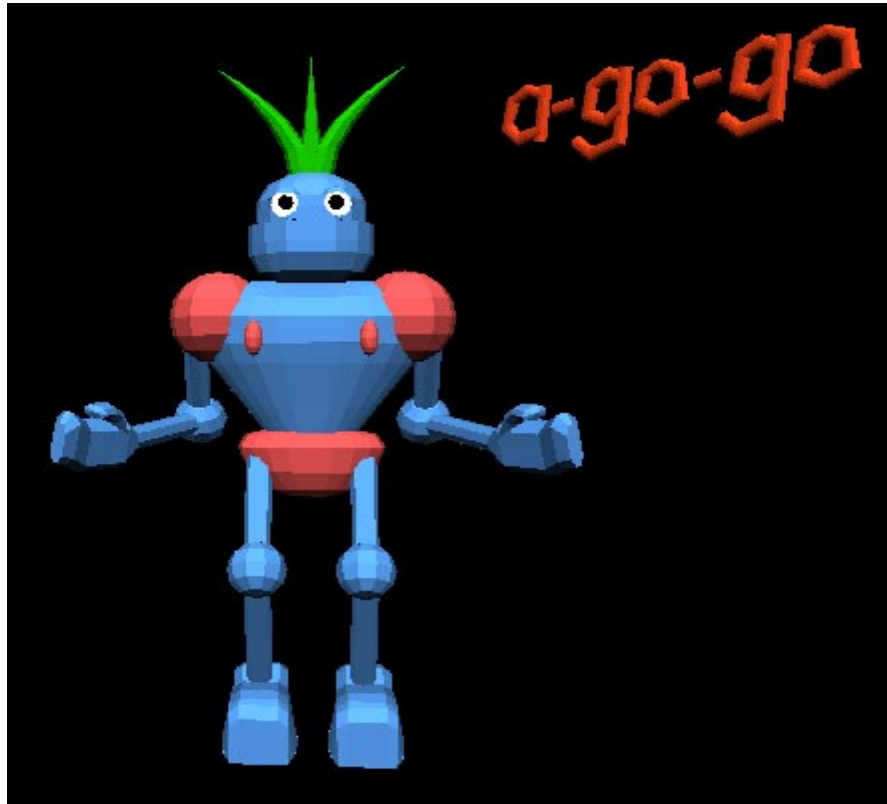


Abb. 5: Der tanzende Roboter: ein Beispiel für Animationen [Kikuro]

Der Bereich Unterhaltung und Spiele bietet viele Möglichkeiten zur Visualisierung. Als Beispiel sei hier Kubricks Zauberwürfel als beliebtes Geduldspiel genannt [Geometrek].

Diese Beispielmodelle und auch viele weitere hier nicht erwähnten VRML-Modelle stellen animierte sowie interaktiv beeinflussbare dreidimensionale Objekte mit unterschiedlicher Intention dar. Die Modelle dienen dem Zweck der Objektdarstellung, der Werbung sowie als Produktpräsentationen mit der Absicht, in Internet-Technologien wie dem World Wide Web auch dreidimensionale Darstellungen medial zu nutzen. Die potentielle Fähigkeit von VRML, zwischenmenschliche Interaktion in virtuellen Umgebungen zu realisieren, liegt bisher noch weitgehend in der Zukunft. Die Möglichkeit mittels VRML eine verteilte virtuelle Umgebung zu schaffen, welche von Netzgemeinschaften gemeinsam genutzt werden, wird hier bisher ebenfalls noch nicht umgesetzt.

3.2. VRML-Modelle aus dem Bereich der Naturwissenschaften

Im Bereich der Naturwissenschaften wird allein bei der Suche im Internet durch bekannte Suchmaschinen ein schier unerschöpfliches Spektrum an VRML-Anwendungen angeboten. Im folgenden werden deshalb nur drei Beispiele aus der Biologie, der Chemie und der Physik betrachtet, die nur einen kurzen repräsentativen Überblick über naturwissenschaftliche VRML-Modelle bieten können.

Ein sehr schönes VRML-Beispiel aus der Biologie ist das Gehirn der Fruchtfliege *Drosophila* (siehe Abbildung 6), dessen Modell an der Universität Freiburg erstellt wurde [Hansen].

Das Gehirn der Fruchtfliege ist als Polygonszene realisiert. Die einzelnen, farblich unterschiedlichen Bereiche des Gehirns sind dabei als separate Polygonmodelle definiert und werden in eine Polygonszene zusammen gefaßt. Auf welche Weise die Polygonmodelle entstanden sind, wird leider nicht bekannt gegeben. Die Interaktionsmöglichkeiten, die dieses VRML-Modell bietet, ist auf die Browserfunktionalitäten begrenzt.

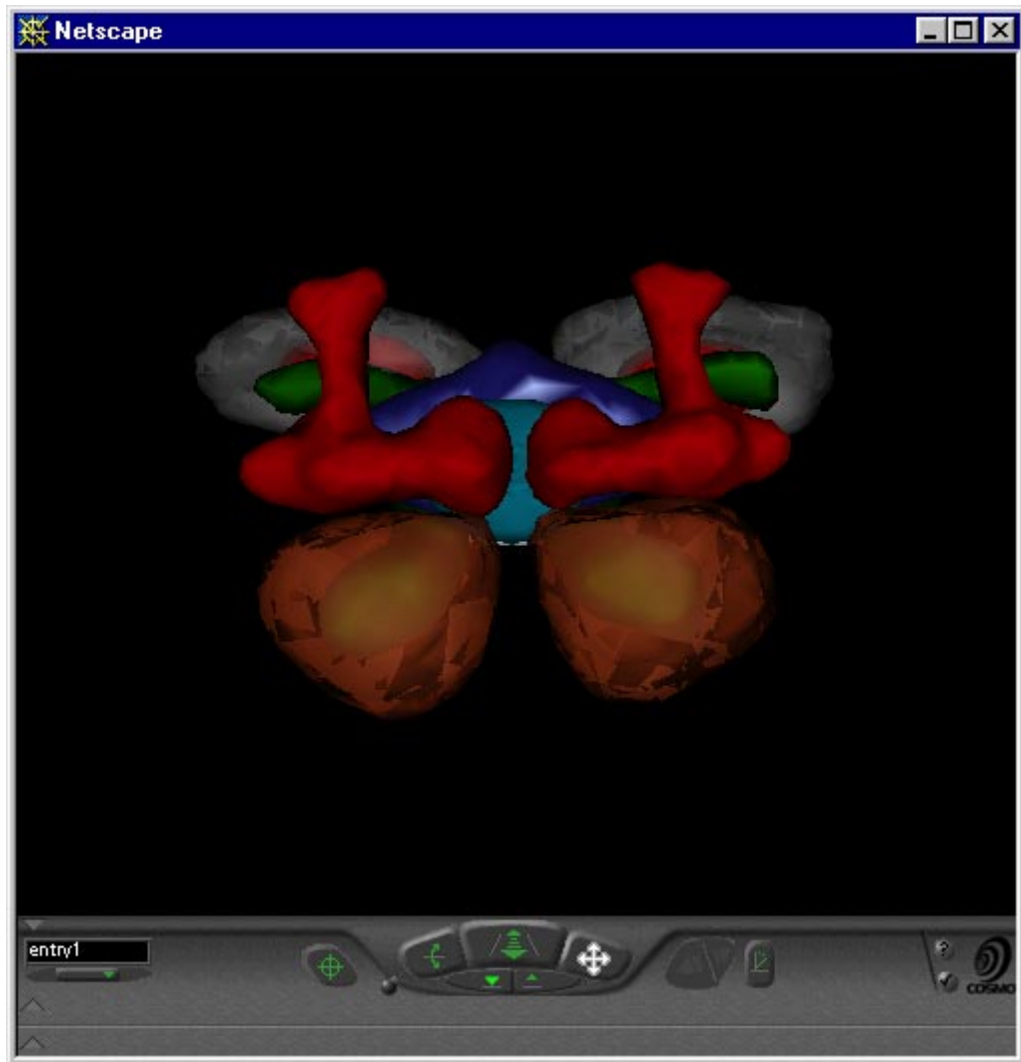


Abb. 6: Das Gehirn einer Fruchtfliege [Hansen]

In dem Bereich der Chemie, d.h. besonders im Bereich der organischen Chemie sowie der Biochemie, spielt die Darstellung von Molekülen und Molekülketten eine große Rolle. Viele Moleküle üben ihre Funktion durch eine Konformationsänderung ihrer räumlichen Struktur aus, z.B. die Ionenkanäle in den Neuronen des Gehirns oder das Rhodopsinmolekül in den Sehzellen des Auges. Ohne diese wäre kein Denken oder Sehen möglich. Die Visualisierung, z.B. mit VRML, bietet die Möglichkeit, durch die Interaktion zu prüfen, ob eine bestimmte Anordnung oder Konformitätsänderung überhaupt sinnvoll ist, und ob ein anderes Molekül dann überhaupt in die Bindungsstelle paßt [Keil].

Die Physik befaßt sich oft mit komplexen Modellen und Zusammenhängen, so daß auch hier ein Bedarf an Simulationsmöglichkeiten besteht. An der Universität von Syracuse gibt es einige

Lehrbeispiele in VRML, z.B. solche, die sich mit Wechselwirkungen von Teilchen, Elektromagnetismus, elektrischen Dipolen usw. befassen. Dabei werden Kräfte, die in bestimmte Richtungen wirken, als Felder oder Wolken von Vektorpfeilen dargestellt [Salgado].

Bei diesen drei Beispielen für naturwissenschaftliche VRML-Modelle steht die Visualisierung von dreidimensionalen Objekten sowie die Simulation von Problemsituationen im Vordergrund. Zum einen bietet die Visualisierung die Möglichkeit, räumliche Strukturen zum besseren Verständnis darzustellen. Zum anderen dient die Visualisierung der Verdeutlichung von Problembereichen innerhalb räumlicher Strukturen.

Der Vorteil einer dreidimensionalen VRML-Darstellung ist, daß durch die Browserfunktionalitäten verschiedene Perspektiven betrachtet werden können sowie die Möglichkeit besteht, sich in das Modell hinein zu bewegen. Dererlei Interaktionen bzw. Animationen können natürlich auch direkt in den VRML-Sourcecode eingebunden werden.

Zusammengesetzte Objekte wie beispielsweise die Teile des Gehirns der Fruchtfliege können unterschiedlich farblich gekennzeichnet sowie separat dargestellt und von allen Seiten betrachtet werden.

3.3. VRML-Projekte im Bereich der Medizin

In dem Bereich der Medizin kann VRML ebenfalls zweckbezogen eingesetzt werden. Ein großes Anwendungsgebiet ergibt sich in der Visualisierung von medizinischen Volumendaten nach VRML, welches Ziel dieser Arbeit ist. Ebenso kann VRML dazu genutzt werden, Untersuchungsmethoden an medizinischen Objekten zu simulieren, um so einen Beitrag zur medizinischen Lehre und Ausbildung zu leisten.

Im folgenden sollen zwei Arten von Modellen unterschieden werden: Zum einen die Darstellungen mit selbstmodellierten Objekten und zum anderen Darstellungen basierend auf medizinischen Datensätzen.

3.3.1. Darstellungen mit selbstmodellierten Objekten

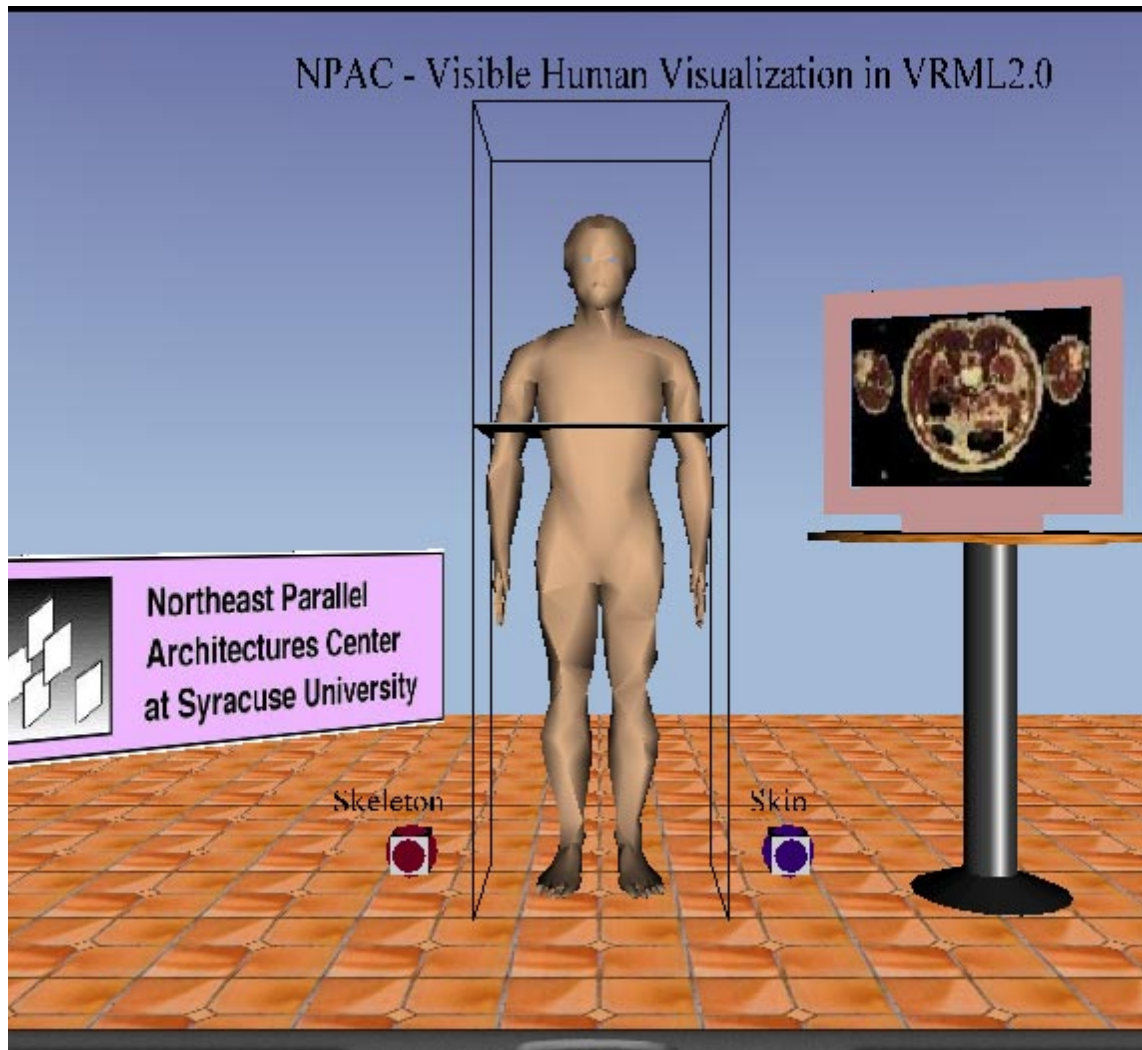


Abb. 7: Interaktive Darstellung Visible Human Schnittbilder [Zeynep]

Die Universität von Syracuse hat auch im Bereich der Medizin VRML-Modelle entwickelt. An einem hier in VRML erzeugten Oberflächenmodell eines Menschen kann via Maus eine horizontale Ebene vertikal verschoben werden (siehe Abbildung 7). Analog zu dieser Koordinatenveränderung werden auf dem daneben stehenden Monitor die jeweiligen Schnitte aus dem Datensatz des Visible Human angezeigt. Diese Schnitte werden als zweidimensionale Graphiken angezeigt. Ein generierter Button überläßt dem Benutzer die Auswahl zwischen der Anzeige des Skelettes oder der Hautoberfläche des Menschen, welche als selbstmodellierte Objekte generiert wurden [Zeynep].

Die Hautoberfläche sowie das Skelett sind selbsterstellte Objekte, die als *IndexedFaceSet* Darstellungen realisiert worden sind. Bei diesem Modell sind keine der Visible Human Daten nach VRML konvertiert worden.

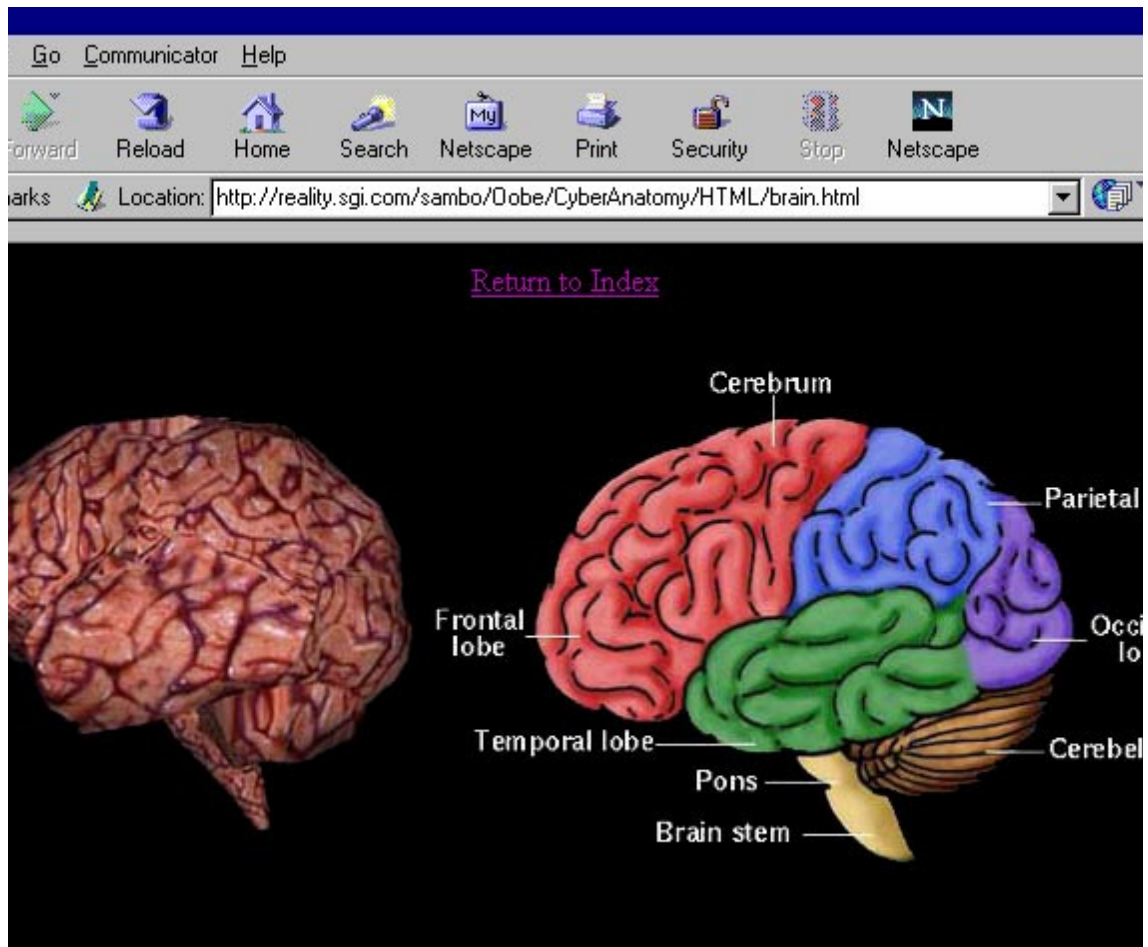


Abb. 8: Gehirndarstellung aus der Cyberanatomie [SGI]

Silicon Graphics Inc. hat eine Cyberanatomie in VRML erstellt, wobei Teile des Skelettes sowie einzelne Organe dargestellt werden können [SGI].

Die einzelnen Objekte sind leider nur als Bilder und nicht als VRML-Modelle ins Internet gestellt worden (siehe Abbildung 8), so daß Sichten aus anderen Blickwinkeln nicht möglich sind. Des weiteren können über die Definition der Objekte nur Mutmaßungen getroffen werden.

In der Abbildung 8 wird neben der beschrifteten Darstellung ein vereinfachtes künstliches Objekt in der abstrakten Form eines Gehirns gezeigt. Auf die Oberfläche des selbstmodellierten Objektes ist ein zweidimensionales Bild des Gehirns "gemapped" wurden, so daß in vereinfachter Form das

Aussehen des Gehirns nachempfunden werden kann. Es ist sehr fraglich, wie diese Darstellung z.B. aus einem anderen Blickwinkel erscheint.

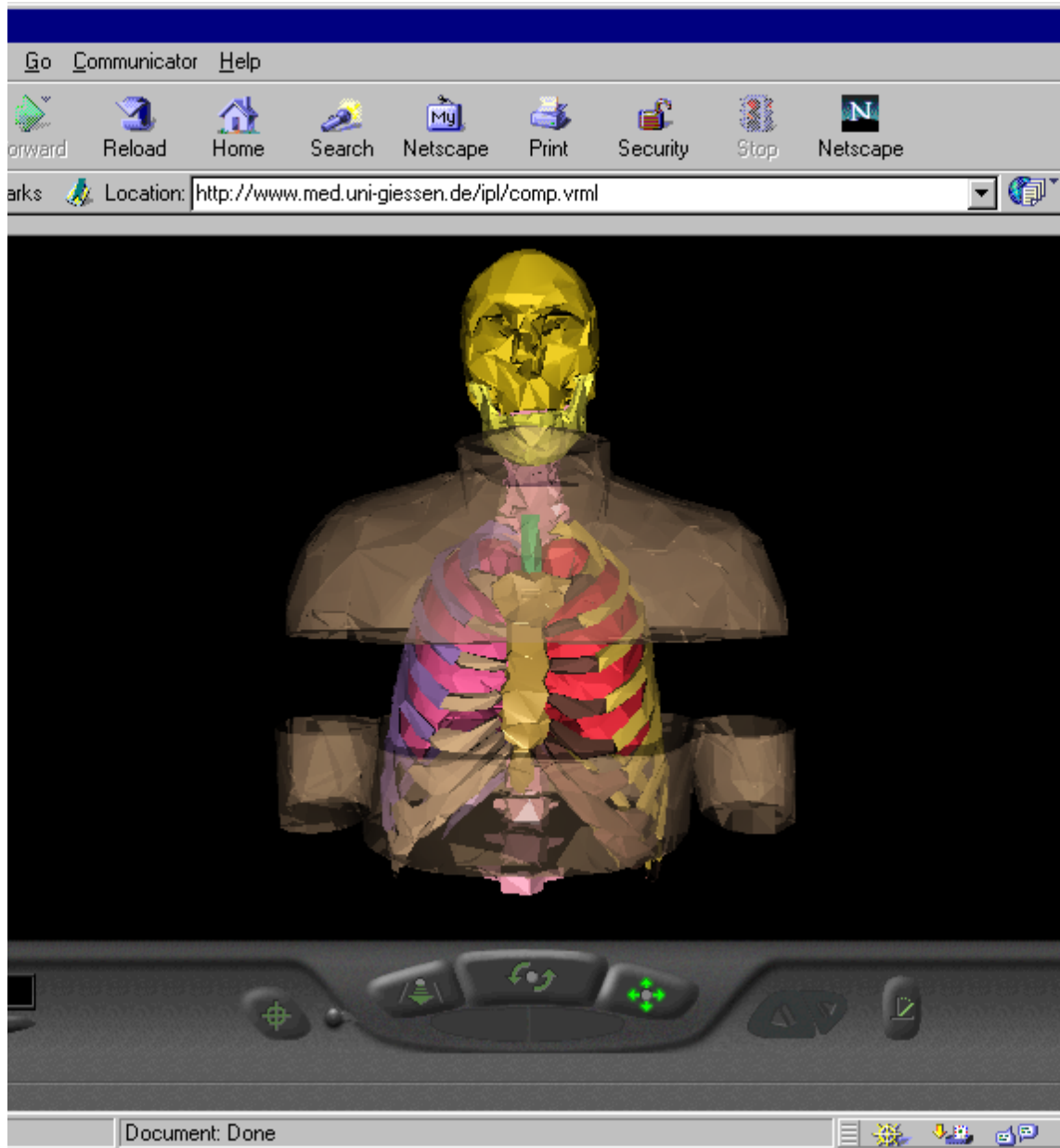


Abb. 9: Grobe VRML- Darstellung des Thorax und des Kopfes [Kriete]

Die Universität in Giessen hat derzeit zwei VRML-Modelle ins Web gestellt: Zum einen eine grobe polygonelle Darstellung eines Thorax in VRML1.0 (siehe Abbildung 9), zum anderen eine

grobe polygonelle Darstellung eines animierten Skellettes in VRML2.0. Bei beiden Modellen ist die Herkunft der Daten zur Polygondarstellung unbekannt [Kriete].

Die groben Kantenübergänge dieser Polygondarstellung ließen sich durch Materialeigenschaften, die VRML bietet, entsprechend abrunden, um somit glattere Flächenübergänge zu schaffen.

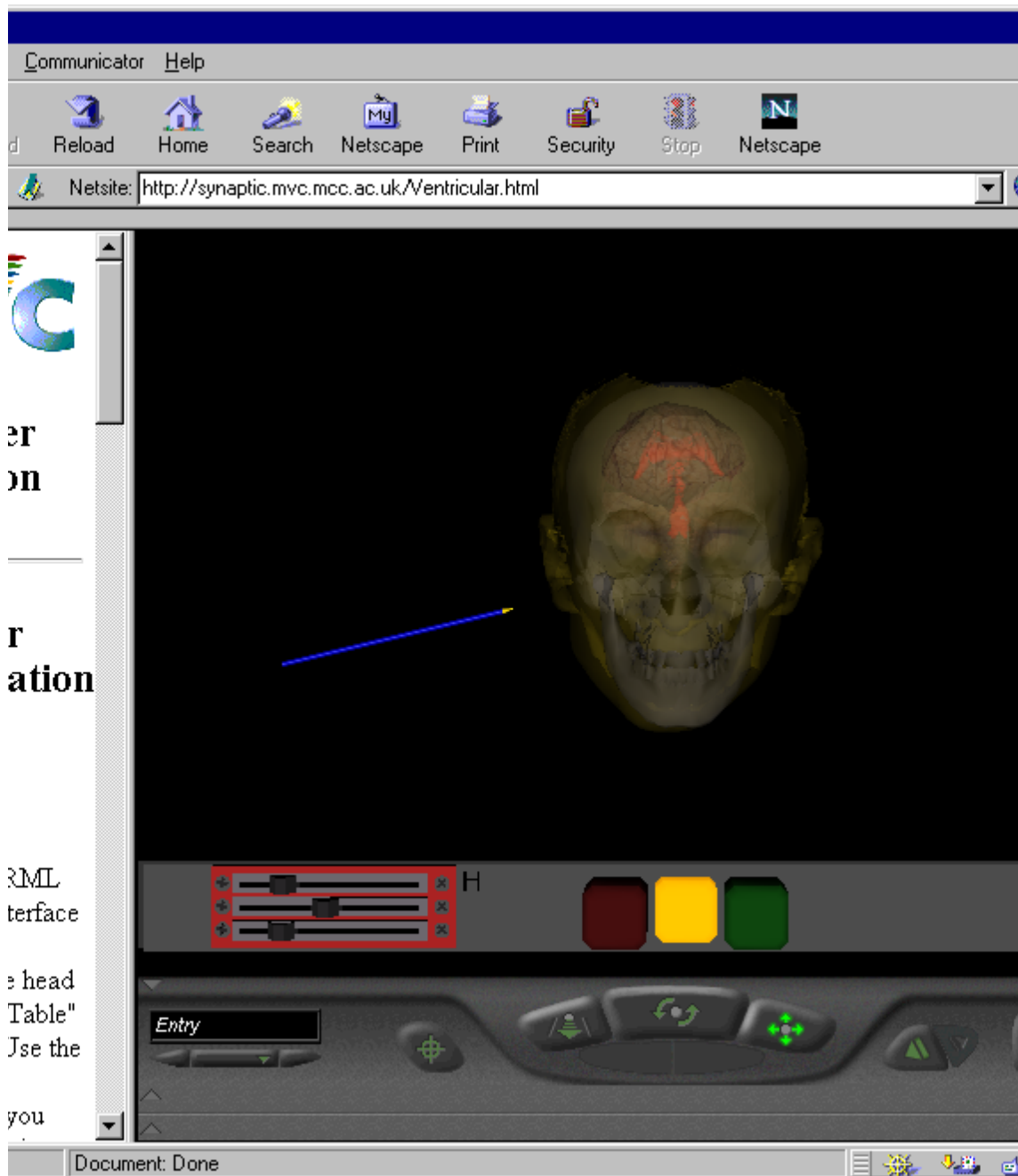


Abb. 10: Interaktive Darstellung einer Kathederisierung eines Ventrikels [John]

Das Visualisation Centre der Universität von Manchester hat drei selbstdefinierte VRML-Simulationen in VRML2.0 einschließlich zusätzlicher Java Programmierung erstellt. Neben einer Lumbalpunktion wurde hier eine Bestrahlungstherapie des Auges sowie eine Ventrikel-Kathederisierung (siehe Abbildung 10) sehr eindrucksvoll dargestellt [John].

Dieses Beispiel ist eins der wenigen VRML-Modelle, die über die Darstellung von menschlichen Körperteilen hinausgeht und die Simulation von Untersuchungsmethoden in interaktiver Form darstellt.

3.3.2. Darstellungen basierend auf medizinischen Datensätzen

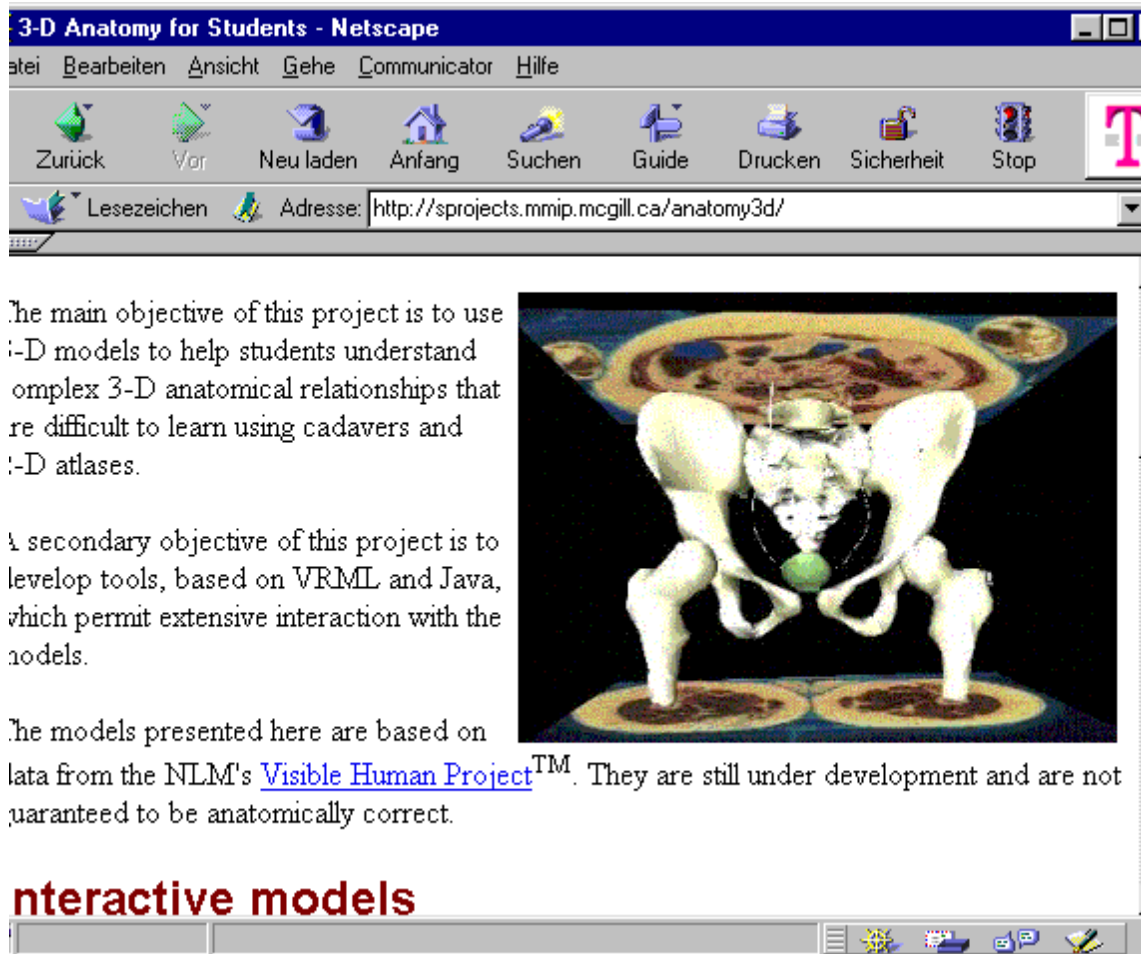


Abb. 11: Integration von VRML-Modellen zwischen Visible Human Schnitten [McGill]

Auf der Basis des Visible Human Datensatzes hat die McGill University ein VRML-Modell für den anatomischen Unterricht für Medizinstudenten erarbeitet.

Es ist dort beispielsweise die Anatomie des Beckens als VRML-Darstellung zu sehen, während darüber und darunter der Schnitt zum Visible Human sichtbar ist (siehe Abbildung 11). Eine strukturierte Liste stellt beliebige anatomische Teile zur Auswahl [McGill].

In diesem Beispiel wird der Visible Human Datensatz mit der Interaktionsmöglichkeit von VRML kombiniert.

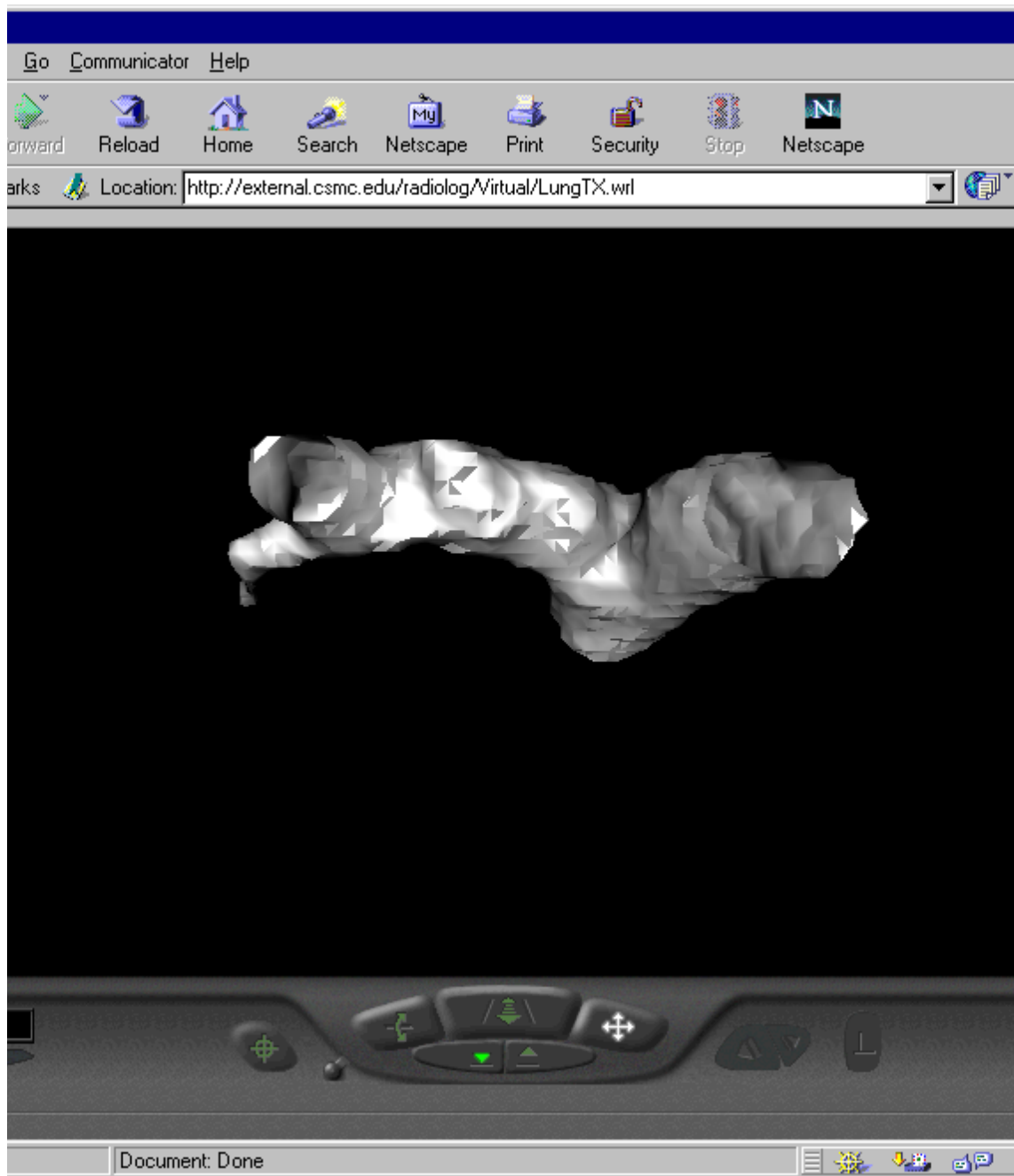


Abb. 12: Darstellung eines Bronchus in VRML [Feinberg]

Das Cedar-Sinai Medical Center hat ebenfalls eine kleine Auswahl von VRML-Modellen ins Internet gestellt. Es handelt sich hierbei ebenfalls um Polygondarstellungen, die aus CT- bzw. MR-Bilddatensätzen erzeugt wurden (siehe Abbildung 12). Diese Modelle haben keine Farb-
informationen und sind ohne Animation und Interaktion dargestellt [Feinberg].

Dieses Modell wirkt vor allem, wenn es interaktiv bewegt wird, als wäre es zum Teil durchsichtig und stellt kein richtiges Volumen dar. Damit in VRML ein Volumenmodell auch wie ein Volumen erscheint, muß die Innenseite sowie die Außenseite des Objektes explizit definiert werden. In diesem Beispiel ist dieses nicht geschehen, so daß dieses Objekt so eine durchsichtige Wirkung erzielt.

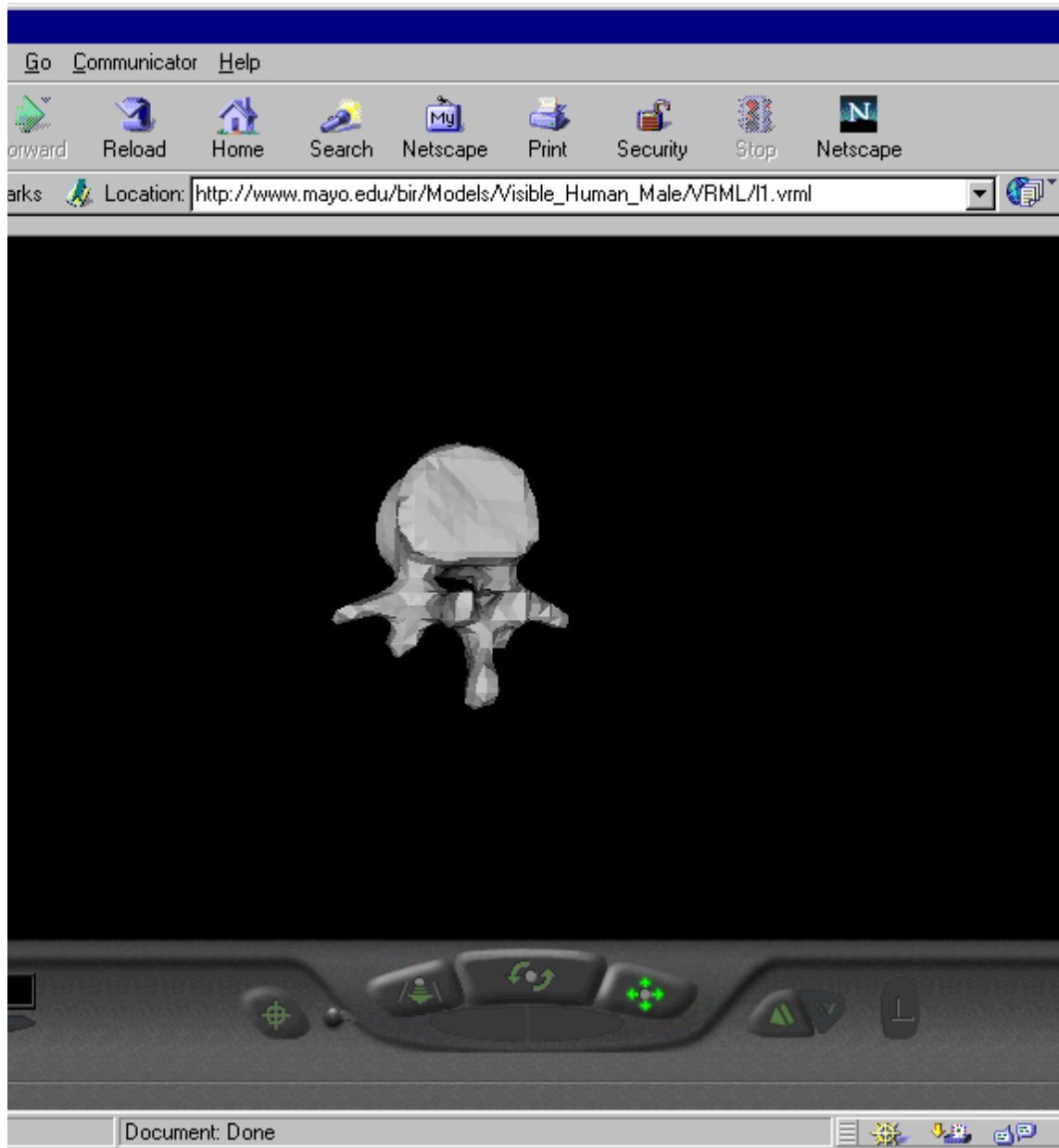


Abb. 13: Darstellung eines Lendenwirbels in VRML [National Library of Medicine]

Die National Library of Medicine hat einzelne Objekte des Visible Human wie Knochen, Aterien, Venen etc., als einzelne VRML-Modelle ins Web gestellt. Die Abbildung 13 zeigt als Beispiel die Darstellung eines Lendenwirbels. Diese VRML-Modelle sind ohne Oberflächeneigenschaften wie Farbgebung, Beleuchtung usw. sowie ohne Animation dargestellt. Zum Teil sind die einzelnen Objekte außerhalb des Browserfensters positioniert d.h. die Objekte müssen über die Browserfunktionalität in das Browserfenster gezogen werden. Die Datenmengen der Polygone sind nicht reduziert, so daß einige VRML-Modelle mehrere MegaBytes an Daten umfassen [National Library of Medicine].

Zusammenfassend lassen sich folgende Feststellungen treffen: der Vorteil der selbstdefinierten Objekte liegt vor allem in einer deutlichen Reduktion der Polygone und der entsprechenden Flächen. Bei diesen Modellen wird lediglich eine simplere Oberfläche beschrieben und dargestellt. Der Nachteil dieser Oberflächendarstellung ist, daß es sich immer um eine idealisierte Darstellung handelt, da auftretende Höhen, Tiefen, Einschnitte etc. der Objekte nicht berücksichtigt werden.

Darstellungen basierend auf medizinischen Datensätzen berücksichtigen dieses Manko und spiegeln realistische Darstellungen wieder. Der Nachteil medizinischer Datensätze ist die deutlich höhere Anzahl an Polygonen und Flächen. Wird die Anzahl der Polygone und Flächen zu groß, wird eine VRML-Datei schnell unhandhabbar für einen VRML-Browser. Dies wird durch die längere Ladezeit der VRML-Datei deutlich. Auch das Bewegen von Objekten mit hoher Polygonanzahl wird nicht mehr unmittelbar vom Browser ausgeführt.

Es stellt sich die Frage inwieweit sich die Vorteile beider Ansätze verbinden lassen. Diese Frage soll im weiteren Verlauf dieser Arbeit behandelt werden.

4. Methode

In diesem Abschnitt werden die in dieser Arbeit erstellten und verwendeten Methoden zur Erzeugung von VRML-Modellen aus medizinischen Volumendaten beschrieben. Ausgangspunkt sind jeweils Modelle, die mit Hilfe des Systems VOXEL-MAN aus räumlichen Schnittbildsequenzen erzeugt und als "intelligentes Volumen" gespeichert wurden.

Der VOXEL-MAN ist ein umfassendes Programmsystem für die Visualisierung medizinischer Volumendaten, bestehend aus drei wesentlichen Komponenten:

- das Visualisationsmodul für die hoch qualifizierte Wiedergabe [Tiede 98]

- das Segmentationsmodul [Höhne 92]
- das semantische Netzwerk der Wissensbasis (knowledge base) [Pommert 94]

Mit Hilfe dieses Systems ist eine große Anzahl von anatomischen und radiologischen Atlanten erzeugt worden, die eine dreidimensionale Exploration der menschlichen Anatomie ermöglichen [Höhne: CD-ROM 95]. Dieses sehr umfangreiche Paket ist allerdings nur auf leistungsfähigen und gut ausgebauten UNIX- Workstations lauffähig.

Daneben besteht die Möglichkeit, mit dem sogenannten VOXEL-MAN Junior bereits mit VOXEL-MAN vorberechnete Filme mit zwei Freiheitsgraden auch am PC anzusehen [Höhne: CD-ROM 98] [Höhne: CD-ROM 2000] [Schubert 99] [Höhne: RSNA 96] [Dittmer 94]. Diese Filme bleiben dabei natürlich auf die vorher vom Autor ausgewählten Ansichten beschränkt. In dieser Arbeit soll statt dessen die freie Interaktion des Anwenders mit den zu erstellenden VRML-Modellen unterstützt werden.

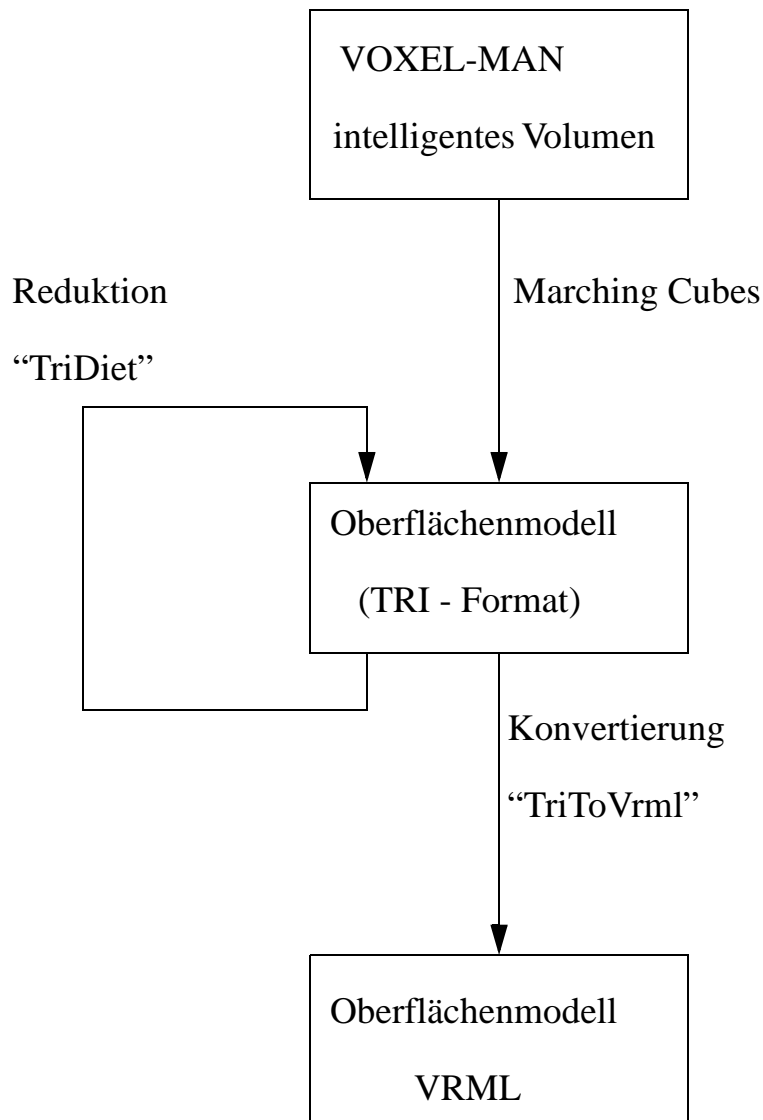


Abb. 14: Erzeugung von VRML-Modellen aus intelligenten Volumen im Überblick

Die Erzeugung eines Polygonmodells aus den VOXEL-MAN-Atlanten erfolgt mit Hilfe des Marching Cubes-Algorithmus (siehe Abbildung 14). Auf dieses entstandene Polygonmodell kann mit Hilfe des von Frank Wilmer [Wilmer93] erstellten Programms eine Reduzierung der Polygone vorgenommen werden. Das reduzierte oder nicht reduzierte Polygonmodell kann nun durch das in dieser Arbeit erstellte Konvertierungsprogramm "TriToVrml" in eine VRML-Anwendung konvertiert werden.

4.1. VOXEL-MAN-Atlanten

Die VOXEL-MAN-Atlanten basieren auf Datensätzen, die mit radiologischen bildgebenden Verfahren (z.B. Computertomographie (CT) und Kernspintomographie (MRI)) gewonnen wurden [Dössel 2000]. Die Beschreibung der menschlichen Anatomie in einem dreidimensionalen Atlas umfaßt neben dem räumlichen Wissen ebenfalls ein abstraktes symbolisches Wissen. Beide Aspekte sind im sogenannten “intelligenten Volumenmodell” (siehe Abbildung 15) zusammengefaßt [Höhne 95] [Pommert 94].

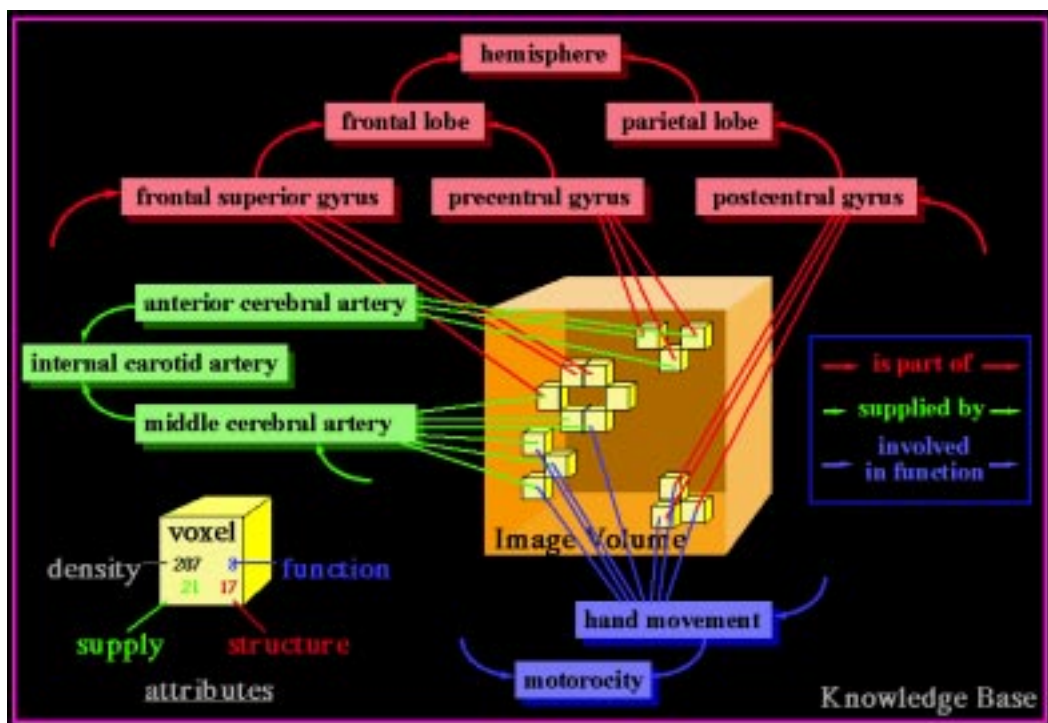


Abb. 15: Darstellung des im VOXEL-MAN verwendeten intelligenten Volumenmodells [Höhne 95]

Die obere Ebene eines “intelligenten Volumens” stellt eine symbolische Beschreibung anatomischer Objekte und ihrer Beziehungen dar, während die untere Ebene die räumliche Beschreibung dieser Strukturen beinhaltet. In Übereinstimmung mit den verschiedenen Bereichen der Anatomie bietet diese Struktur auch die Möglichkeit, zwischen verschiedenen Aspekten der Anatomie, z.B. der Morphologie, der funktionalen Anatomie sowie der Blutversorgung in graphischer oder textueller Darstellung, zu unterscheiden [Höhne 95].

VOXEL-MAN-Atlanten wurden bereits für verschiedene Ausschnitte des menschlichen Körpers erstellt, z.B. Gehirn, Schädel, Innere Organe, Hand, Fuß, Fötus, sowie diverse Erkrankungen [IMDM].

4.2. Erzeugung von Polygonmodellen

4.2.1. Der Einsatz von Marching Cubes

Marching Cubes ist ein Algorithmus, der es ermöglicht, aus dreidimensionalen Volumendaten ein Polygonmodell zu erstellen. Dieser Algorithmus, der inzwischen als Standardalgorithmus zum Erstellen von Isooberflächen (Konturflächen mit gleicher Intensität im Datenvolumen) gilt, wurde 1987 von William E. Lorensen und Harvey E. Cline veröffentlicht [Lorensen 87].

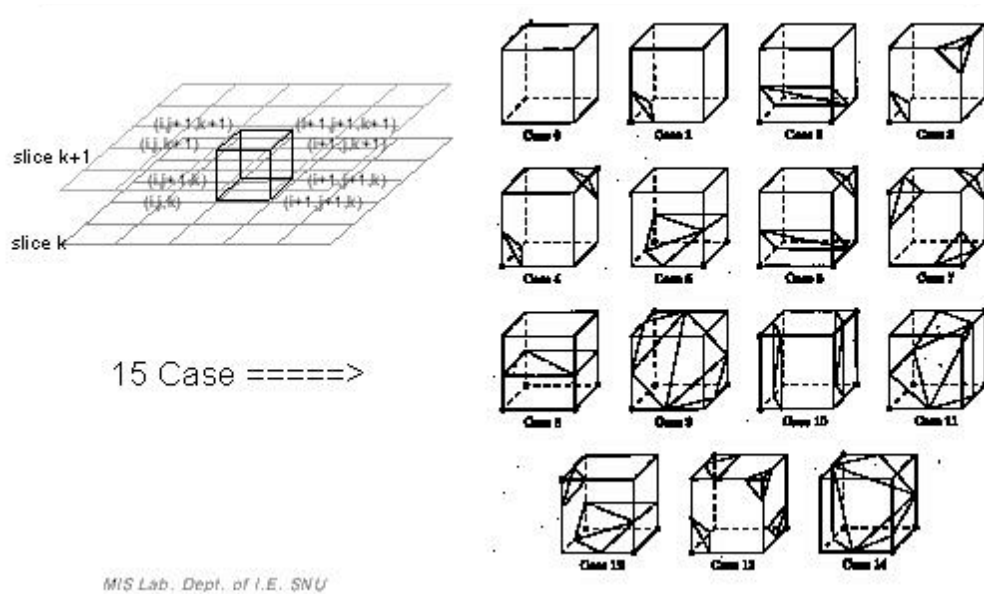


Abb. 16: Die 15 Fälle des Marching Cubes Algorithmus [Lorensen 87]

Das Haupteinsatzgebiet ist dort, wo spezielle Hardware und Software zur Graphikverarbeitung zur Verfügung steht und die Darstellung von dreidimensionalen Daten auf sehr großen Datenmengen basiert. So lassen sich mit Marching Cubes aus einem großen Volumen die (möglicherweise viel kleineren) interessierenden Oberflächen als Polygonmodelle extrahieren. Dazu wird im Vorwege ein Schwellwert festgelegt und anschließend eine Triangulierung erzeugt. Diese teilt den Raum in Bereiche mit Werten größer bzw. kleiner als der Schwellwert auf (siehe

Abbildung 16). Dazu wird der dreidimensionale Raum in kleine Würfel aufgeteilt, die unabhängig voneinander verarbeitet werden können. Da keine Abhängigkeiten zwischen den Würfeln berücksichtigt werden müssen, ist das Verfahren recht schnell, d.h. die Laufzeit steigt linear mit der Anzahl der Würfel an.

Voraussetzung ist, daß alle Daten in einem kartesischen Koordinatensystem angeordnet sind, und daß zu jedem Voxel des Raumes ein Wert gespeichert ist.

Für die Erzeugung der Polygone wird nun jedem Voxel eine 0 oder eine 1 zugeordnet, je nachdem, ob sich sein Wert ober- oder unterhalb der Isofläche befindet. Für jeden Würfel stehen 8 Bits zur Verfügung, die beschreiben, wie die Isofläche diesen Würfel schneidet. Werden die 8 Bits zu einem Byte zusammengefaßt, ergibt sich daraus ein Index von 0-255 möglichen Fällen. Durch Symmetrien und Rotationen lassen sich die Fälle auf 15 reduzieren. Gespeichert wird dabei jeder Eckpunkt eines Polygons, welche sich auf der Kante des Würfels befindet. Die genaue Position des Punktes auf der Kante eines Würfels wird durch lineare Interpolation gewonnen. Dabei wird errechnet an welcher Stelle der Schwellwert erreicht sein müßte.

Bei der Reduzierung auf 15 Fälle wurden im ursprünglichen Algorithmus einige Möglichkeiten übersehen, so daß Löcher in der Isofläche entstehen können. J. Wilhelms und A. van Gelder beschreiben diesen Fehler und erläutern die Korrektur des Marching Cubes-Algorithmus [Wilhelms 90]. Dieser korrigierte Algorithmus wird in dieser Arbeit verwendet.

Es gibt inzwischen viele verschiedene Abwandlungen von Marching Cubes, in die verschiedene Erweiterungen eingegangen sind, z.B. um andere Typen von Eingabedaten verarbeiten zu können oder Darstellungsfehler zu minimieren. Zu erwähnen sind hier etwa Marching Tetraeder oder Dividing Cubes [Schroeder 98].

4.2.2. Polygonreduzierung

Der Marching Cubes Algorithmus generiert aus dreidimensionalen Volumendaten eine geschlossene Polygonoberfläche. Die Größe der Polygone ist dabei abhängig von der Voxelgröße, d.h. bei kleinen Voxeln steigt die Anzahl der Polygone pro Fläche sehr hoch an.

Frank Wilmer [Wilmer 93] beschreibt in seiner Diplomarbeit ein Verfahren, das unter Berücksichtigung der Krümmung der Objektoberfläche die Auflösung dem Objekt anpaßt und so zu einer Datenreduktion bei minimalem Qualitätsverlust gelangt. Das Ziel seiner Arbeit ist, die Polygonanzahl bei möglichst großer Detailgenauigkeit der Oberfläche zu minimieren. Die

Eigenschaft einer durchdringungsfreien und orientierbaren Eingangsoberfläche bleibt durch die Reduktion unbeeinflusst.

Das Verfahren beginnt mit der Bildung freier Polygone durch Zusammenfassen von Polygonen ähnlicher Neigung. Voraussetzung für die Zuordnung eines Polygons zu einer Region sind zwei Bedingungen. Zum einen muß ein Polygon mit mindestens einem Polygon der Region eine Kante teilen. Zum anderen ist der von den Oberflächennormalen gebildete Winkel für jedes Polygonpaar kleiner als ein vorgegebener Toleranzwinkel α . Kann ein Polygon keiner Region zugeordnet werden, wird eine neue Region mit diesem Polygon initialisiert. Sind alle Polygone zugeordnet, wird die äußere Kontur jeder Region ermittelt, so daß eine Beschreibung eines "übergeordneten" Polygons im dreidimensionalen Raum entsteht.

Der nächste Schritt ist die Retriangulation der entstandenen Polygone. Dazu ist für jedes Polygon eine Projektion auf einer Ebene nötig. Als Projektionsebene wird die Ebene gewählt, die senkrecht zum Normalenvektor des Polygons steht. Wird festgestellt, daß das projizierte Polygon nicht überschneidungsfrei ist, wird es nicht weiter bearbeitet, sondern es werden die Originalpolygone verwendet.

In der weiteren Verarbeitung werden redundante Knoten der Polygonkontur entfernt. Dies sind vor allem Knoten, die mit ihrem Vorgänger- und Nachfolgerknoten eine Gerade bilden. Die abschließende Triangulation der Polygone übergibt wieder eine regelmäßige Polygonstruktur. Bei einer überschneidungsfreien Projektion läßt sich die Triangulation in der Projektionsebene durchführen. Das Ergebnis kann unmittelbar auf das Polygon im dreidimensionalen Raum übertragen werden.

Die durch den Marching Cubes Algorithmus generierten Oberflächen lassen sich durch das TriDiet Verfahren von Frank Wilmer je nach Wahl des Toleranzwinkels α zwischen 30% und 60% reduzieren. Der Toleranzwinkel kann von dem Anwender in dem von ihm implementierten Programm "TriDiet" durch die Wahl des Reduktionslevels (zwischen 1 und 7) gewählt werden. Wird ein kleiner Toleranzwinkel α gewählt (Reduktionslevel 1), bleibt die Detailtreue des Originals erhalten die Reduktion bleibt dabei entsprechend gering. Ein zu groß gewählter Toleranzwinkel α (entspricht Reduktionslevel 7) erreicht ein hohes Maß an Reduktion sowie eine Vergrößerung des Objektes, welche nicht unbedingt erwünscht ist. Ein mittlerer gewählter Toleranzwinkel α (entspricht Reduktionslevel 4) erreicht jedoch eine gute Datenreduktion ohne nennenswert sichtbaren Qualitätsverlust.

4.3. Konvertierung intelligenter Volumenmodelle in eine VRML-Datei

Das mit Hilfe des Marching Cubes-Algorithmus erzeugte Oberflächenmodell ist im VOXEL-MAN eigenen "TRI-Format" abgelegt. Das im Rahmen dieser Arbeit in der Programmiersprache C geschriebene Konvertierungsprogramm liest zunächst die generierte TRI-Datei ein. Aus der eingelesenen TRI-Datei werden Polygonkoordinaten sowie die im VOXEL-MAN zugewiesene Farbe des Objektes in die auszugebene VRML-Datei übertragen. Zur Veranschaulichung dient in Abbildung 17 der Ausschnitt einer konvertierten VRML-Datei.

```
#VRML V2.0 utf8
Background { skyColor 0.6 0.6 0.6 }
Transform {
  translation -24.75 -20.42 -104.00
  scale 0.5 0.5 0.5
  rotation 1 0 0 -1.57
  children [ Shape {
    appearance Appearance { material Material {
      diffuseColor 0.90 0.80 0.80 }}
    geometry IndexedFaceSet {
      creaseAngle 3.14 solid FALSE
      coord Coordinate {
        point [
          48.00 42.00 99.00, # 0
          49.00 42.00 98.00, # 1
          (...)
          48.00 40.00 110.00, # 330
          52.00 40.00 110.00 # 331
        ] }
      coordIndex [
        0 1 2 -1, # 0
        2 1 3 -1, # 1
        (...)
        326 331 329 -1, # 632
        327 329 331 -1, # 633
      ] } }
  ] }
}
```

Abb. 17: Codebeispiel einer VRML-Datei

In dem im Rahmen dieser Arbeit implementierten Konvertierungsprogramm "TriToVRML" werden zum einen die Polygonkoordinaten sowie ihre Indizes zur Flächendarstellung automatisiert dargestellt. Die ebenso automatisiert dargestellte Farbzuzuweisung wird wie die Polygone aus der eingelesenen Datei im "TRI-Format" übernommen. Zum anderen werden folgende zusätzliche Informationen automatisiert dargestellt. Beginnend mit dem Header, der bereits in Unterkapitel 2.4.1 erläutert wurde, erfolgt die Definition des Hintergrundes. Der Defaultwert des Hintergrundes ist die Farbe Schwarz, welche hier durch ein helles Grau ersetzt wird.

Der Gruppenknoten *Transform* läßt die Positionierung, Skalierung sowie eine Rotation um 90 Grad um die x-Achse zu, so daß das Objekt aufrecht aus dem Bildschirm herausblickend dargestellt wird. Die generierten Polygonkoordinaten der medizinischen Objekte können in VRML durch Verwendung von *Sets* umgesetzt werden. Die allgemeine Erläuterung von *Sets* erfolgt in dem folgenden Unterkapitel 4.3.1. Die Darstellung der medizinischen Objekte als Oberflächenmodell wird in VRML durch den Knoten *IndexedFaceSet* möglich. Während die Polygonkoordinaten der medizinischen Objekte als Punktkoordinaten übernommen werden, werden die Indizes der Punktkoordinaten als Flächenpunkte verwendet. Hinter dem Doppelgattersymbol "#", das als Kommentarzeichen dient, wird die Anzahl der Polygonpunkte bzw. -flächen angegeben. Materialeigenschaften wie "*creaseAngle 3.14*" und "*solid FALSE*" glätten die kantigen Flächenübergänge und lassen das Objekt als Volumen erscheinen.

Nicht automatisiert dargestellt werden Animationen bzw. Interaktionen. Diese können nach Konvertierung eines VRML-Oberflächenmodells manuell nachbearbeitet werden.

4.3.1. Darstellung der Polygone durch Sets

Geometrien dreidimensionaler Graphiken bestehen häufig aus vielen Eckpunkten, die mit Linien verbunden sind. Aus einem geschlossenen Verbund von Linien bzw. Punkten ergibt sich jeweils eine polygonale Fläche. Die Gestalt einer Geometrie teilt sich schließlich in eine Vielzahl von Polygonen auf. Jede Darstellungsebene setzt sich aus einer Gruppe von Elementen zusammen, die als Set bezeichnet wird. Einige VRML-Browser, z.B. WorldView oder VRWave, erlauben in diesem Zusammenhang das Umschalten zwischen verschiedenen Darstellungsmodi. So lassen sich die Geometrien einer VRML-Szene wahlweise im Modus Körper, Drahtgitter oder Punktwolke darstellen.

Der Standard VRML97 bietet für jeden Darstellungsmodus einen eigenen Knoten zur Konstruktion von Geometrien.

Die Generierung einer Geometrie als Punktwolke erfolgt mittels des Knotens *PointSet*. Zum einen besteht dieser Knoten aus einer Liste von Punkten und zum anderen aus einer Liste von Farben, die den Punkten zugeordnet sind [Carey 97].

Ein Drahtgittermodell wird mit dem Knoten *IndexedLineSet* erzeugt. Hier werden die Punkte mit Linien verbunden. In der Abbildung 18 wird ein VRML-Codebeispiel dargestellt.

```
#VRML V2.0 utf8
Background { skyColor 0.6 0.6 0.6 }
Transform {
  translation -39.11 -51.08 -100.72
  scale 0.5 0.5 0.5
  rotation 1 0 0 -1.57
  children [
    Shape { appearance Appearance {
      material Material { diffuseColor 0.00 0.00 0.00}}
      geometry IndexedLineSet {
        coord Coordinate {
          point [
            81.00 55.90 8.00, # 0
            (...)
            80.00 125.00 194.28 # 8138
          ] }
        coordIndex [
          96 95 85 -1, # 0
          (...)
          5917 5735 5737 -1, # 16268
        ]}}
  ]}
}}
```

Abb. 18: Codebeispiel einer Drahtgitterdarstellung in VRML

Die Darstellung von Körpern ist prinzipiell identisch zu der Drahtgitter-Darstellung, wobei der Knoten *IndexedLineSet* mit dem Knoten *IndexedFaceSet* auszutauschen ist. Die Umrisse werden automatisch als geschlossene Polygone dargestellt. Zu beachten ist jedoch die Reihenfolge der Punkte, die bei der Gitterdarstellung noch unerheblich ist. Wird die Reihenfolge nicht eingehalten, entstehen "Löcher" in den Geometrien.

In VRML sind Polygone lediglich einseitig definiert, d.h. sie werden beim Rendering nur von einer Seite dargestellt. Dies ist darin begründet, daß Polygone id. B. zu geschlossenen Körpern

zusammengesetzt werden, die lediglich eine äußere Gestalt haben sollen. Der Rendering-Aufwand läßt sich durch die Darstellung der Außenseite von Objekten erheblich reduzieren. Innenwände von Objekten müssen explizit definiert werden [Ames 96].

Für die Definition der Innenwände muß lediglich das Feld *solid* mit dem Wert "FALSE" belegt werden damit das dargestellte Objekt als Volumen angezeigt werden kann. Diese Definition ist durch das Konvertierungsprogramm "TriToVrml" bereits automatisiert worden.

4.3.2. Positionierung der Objekte

Die Position der nach VRML konvertierten Volumenmodelle ist durch die Werte der definierten Punkte vorgegeben. In VRML entspricht der Mittelpunkt des Bildschirms dem Koordinatenursprung, so daß die mit positiven Werten belegten Punkte des Volumenmodells rechts oben außerhalb des Sichtbereichs des Bildschirms liegen.

Für die Positionierung der nach VRML konvertierten intelligenten Volumenmodelle bieten sich zwei Möglichkeiten an. Zum einen die Möglichkeit, die Polygonkoordinaten bereits während des Konvertierungsprozesses um den Koordinatenursprung zu normieren. Zum anderen bleibt die Möglichkeit, im lokalen Koordinatensystem des zu konvertierenden Objektes eine Transformation der Koordinaten in den Koordinatenursprung vorzunehmen.

Die erste der aufgezählten Möglichkeiten hat den entscheidenden Nachteil, daß beim Zusammenfügen von mehreren medizinischen Objekten in eine VRML-Datei alle Objekte übereinander um den Koordinatenursprung liegen und somit nicht mehr anatomisch korrekt zueinander gruppiert werden. Der Grund hierfür ist der errechnete Mittelwert, der für jedes Objekt separat ermittelt wird. Es wäre zwar denkbar, einen für alle Objekte des Datensatzes allgemeingültigen Mittelwert zu verwenden, aber auch dies hätte entscheidene Nachteile. So würden beispielsweise kleine Objekte, z.B. die Linse eines Auges, sehr klein und weit weg dargestellt werden, während große Objekte, z.B. die Kopfhaut, sehr nah und groß erscheinen. Dies bedingt eine manuelle Nachbearbeitung der konvertierten VRML-Dateien zur optimalen Darstellung der Objekte, welche nicht wünschenswert ist.

Die zuletzt erwähnte Möglichkeit, die generierten Koordinaten zu übernehmen und die erzeugten Objekte durch eine Translation in ein lokales Koordinatensystem zu überführen, behebt diesen Nachteil. Beim Zusammenfügen von mehreren medizinischen Objekten braucht lediglich nur für ein Objekt eine Translation vorgenommen werden, alle weiteren Objekte gruppieren sich

anatomisch korrekt durch ihre exakten Koordinaten zu diesem Objekt. Dies ist ein klarer Vorteil gegenüber einer Normierung der Punktkoordinaten und wurde somit in dem in dieser Arbeit entwickelten Konvertierungsprogramm "TriToVml" umgesetzt.

4.3.3. Beleuchtung, Farbgebung und Oberflächeneigenschaften

Oberflächeneigenschaften wie Farbe und Beleuchtung [Foley 94] etc. verleihen den Objekten einen individuellen Charakter. Um auf diese Möglichkeit zuzugreifen, wird der Objektknoten *Appearance* verwendet. Dieser wiederum enthält u.a. einen weiteren Objektknoten *Material*, dessen Felder die Zuweisung verschiedener Oberflächeneigenschaften erlauben. Die Felder dieses Knotens beschreiben in welcher Weise Licht von einem Objekt reflektiert wird, so daß eine Farbwahrnehmung entsteht.

Bei der Darstellung von Farben verwendet VRML das RGB Farbmodell.

Innerhalb der VOXEL-MAN-Atlanten sind den medizinischen Objekten bereits Farbwerte zugewiesen worden. Im Konvertierungsprozeß werden die im Voxel-Man vergebenen Farbwerte eingelesen und als *diffuseColor* Werte in der VRML-Datei wieder ausgegeben. Dieser Prozeß ist durch das Konvertierungsprogramm automatisiert. Die im folgenden erläuterten weiteren Lichtquellen können durch manuelle Nachbearbeitung der VRML-Datei hinzugefügt werden.

Das Feld *ambientIntensity* spezifiziert wieviel ambientes Licht (Umgebungslicht) eine Oberfläche reflektieren soll. Ambientes Licht ist omnidirektional und abhängig von der Anzahl der Lichtquellen, aber nicht von ihren Positionen relativ zur Oberfläche. Je größer dieser Wert (zwischen 0 und 1) ist, desto heller erscheint die Farbe des Objektes. In der Abbildung 19 wird der VRML-Code dargestellt.

```
Shape {
  appearance Appearance {
    material Material {
      diffuseColor      1.00  0.50  0.00
      ambientIntensity  0.2
      emissiveColor     0      0      0
      shininess         0.2
      specularColor     0      0      0
      transparency      0
    } }
}
```

Abb. 19: VRML-Codebeispiel für die Farbgebung und Beleuchtung

Das Feld *diffuseColor* wird von allen VRML-Lichtquellen reflektiert, abhängig vom Winkel zwischen einer Oberfläche und der jeweiligen Lichtquelle selbst. Je direkter die Oberfläche dem Licht gegenüberliegt desto mehr diffuses Licht wird reflektiert. Im Gegensatz dazu modelliert das Feld *emissiveColor* leuchtende Objekte. Von dem Objekt wird kein Licht emittiert und es kann keine anderen Objekte beleuchten.

Das Feld *shininess* beschreibt die relative Größe von Glanzlichtern und damit die Glattheit einer Oberfläche. Niedrige Werte produzieren einen eher matten Schimmer, während höhere Werte schärfere und schmalere Lichtreflexe bewirken. Das Feld *specularColor* enthält, ergänzend zu *shininess*, einen Spektralfarbwert. Dieser wird in Abhängigkeit vom Winkel zwischen der Oberfläche und Lichtquelle sowie vom Winkel zwischen Oberfläche und Betrachter in die Berechnung der Farbübergänge vom Glanzlicht zur Objektfarbe einbezogen.

Das Feld *transparency* beschreibt die Durchsichtigkeit eines Objektes und kann in Werten zwischen 0 und 1 angegeben werden [Kloss 98].

In eine VRML-Szene können eine oder auch mehrere Beleuchtungsquellen eingebunden werden. Der Knoten *DirectionalLight* entspricht einer gleichmäßigen Beleuchtung wie durch Tageslicht. Eine punktförmige Lichtquelle kann durch den Knoten *PointLight* erzeugt werden, während der Knoten *SpotLight* Scheinwerferlicht erzeugt. Über entsprechende Felder kann die Farbe der Beleuchtung, die Richtung sowie die Helligkeit gewählt werden [Kloss 98].

Texturen zählen auch zu den Oberflächeneigenschaften, die, wenn ein geometrischer Körper mit einer Textur versehen wird, einen sehr realistischen Eindruck vermitteln. Gleichzeitig läßt sich mit der Verwendung von Texturen viel Rechenkapazität einsparen bzw. die Übertragungsdauer der Dateien verringern. Soll z.B. ein Bild im GIF oder JPEG Format als Textur eingesetzt werden, wird der *ImageTexture* Knoten verwendet. Das im Rahmen dieser Arbeit entwickelte Modell 2, verwendet Texturen um die Schnitte des Kopfes auf einem Bildschirm darzustellen. Die Abbildung 20 zeigt daraus ein VRML-Codebeispiel.

```
DEF Bild Transform {
    children [
    DEF SW Switch {
        whichChoice 0
        choice [
        Shape {
appearance Appearance {texture ImageTexture {url"jpg\v010.jpg"}}
geometry DEF SCREEN IndexedFaceSet {
        ccw FALSE
```

```

creaseAngle 3.14
coord Coordinate {
  point [ 0.14 -0.05 0.16, 0.14 0.22 0.11,
         -0.14 0.22 0.11, -0.14 -0.05 0.16 ]}
coordIndex [ 2 1 0 -1, 0 3 2 ]}}
Shape {
appearance Appearance {texture ImageTexture {url"jpg\v020.jpg"}}
geometry USE SCREEN }
}}

```

Abb. 20: VRML-Codebeispiel für die Verwendung von Texturen

Wird ein abstraktes Muster pixelweise definiert, so können mit dem *PixelTexture* Knoten Muster auf eine Objektoberfläche gebracht werden. Mit Hilfe des *MovieTexture* Knotens kann ein MPEG Video auf alle Seiten eines animierten Objektes projiziert werden [Kloss 98].

4.4. Bedienoberfläche

Die in dieser Arbeit entwickelte Methode spiegelt sich in der erstellten Bedienoberfläche wieder. Das in Tcl/Tk geschriebene Oberflächenmenü bietet mehrere Auswahlbuttons an, die in der Abbildung 21 dargestellt sind.

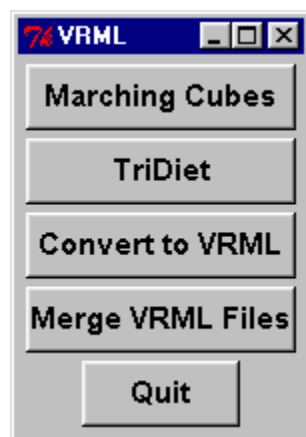


Abb. 21: Darstellung der erstellten Bedienoberfläche

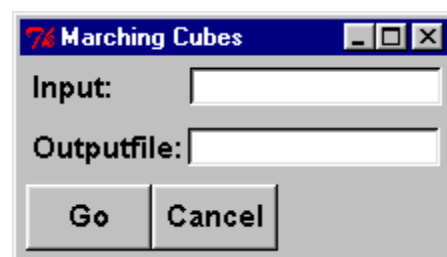


Abb. 22: Darstellung des Untermenüs "Marching Cubes" der Bedienoberfläche

Durch Anwählen des Marching Cubes Buttons öffnet sich ein weiteres Menü, das zum einen den Namen eines im intelligenten Volumen beschriebenen Objektes und zum anderen einen selbstgewählten Namen für die zu erstellende Datei erwartet. Durch Klicken auf den Go-Button wird der Marching Cubes-Algorithmus gestartet. Obige Abbildung 22 veranschaulicht die erläuterte Funktionsweise.

Die Funktionsweise des TriDiet Buttons ist ähnlich. Abbildung 23 zeigt das Menü des TriDiet Verfahrens. Durch einen Browse-Button läßt sich eine Eingabedatei auswählen. Des weiteren kann der oben beschriebene Reduktionsgrad zwischen eins und sieben gewählt werden.

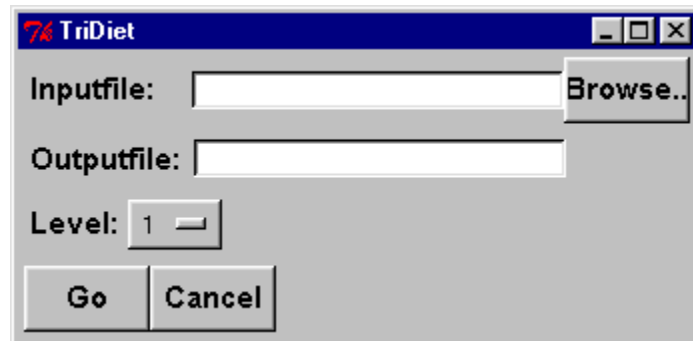


Abb. 23: Darstellung des Untermenüs "TriDiet" der Bedienoberfläche

Durch die Anwahl des "Convert to VRML" Buttons öffnet sich ein weiteres Menü (siehe Abbildung 24). Es gibt zusätzlich die Möglichkeit, Kommentare, die als Indizies im VRML-Code angezeigt werden, anzeigen zu lassen. Werden Objekte generiert, die sich aus einer hohen Anzahl von Polygonen zusammensetzen, sollte auf eine Indizierung der Polygonpunkte und -flächen verzichtet werden, um die Datenmenge der VRML-Datei möglichst klein zu halten.

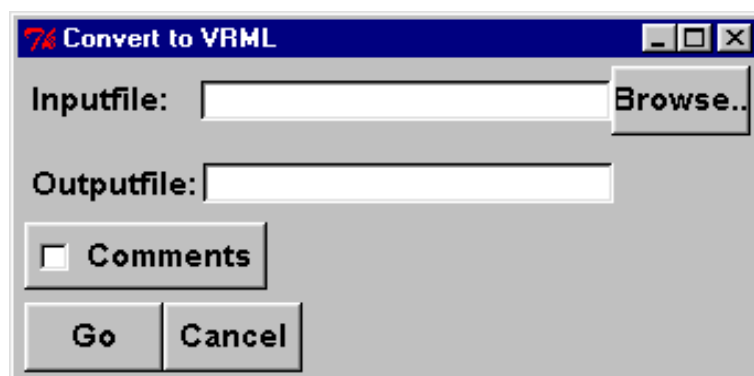


Abb. 24: Darstellung des Untermenüs "Convert to VRML" der Bedienoberfläche

Hinter dem Button “Merge VRML Files” steht ein weiteres im Rahmen dieser Arbeit entstandenes Programm, das die Möglichkeit bietet, VRML-Dateien in eine Datei zusammenzufügen (siehe Abbildung 25).

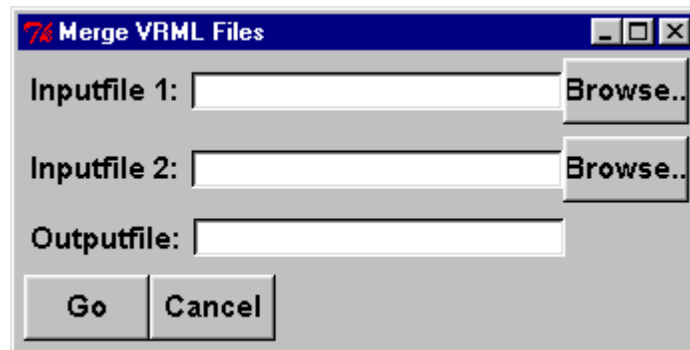


Abb. 25: Darstellung des Untermenüs “Merge VRML Files” der Bedienoberfläche

4.5. Generierung von Text und Hyperlinks

Für die Definition von Hyperlinks innerhalb einer VRML-Datei ist es gleichgültig, ob eine HTML-Seite adressiert wird oder eine andere VRML-Datei. Ebenso ist es gleichgültig, ob sich das Zielobjekt auf dem lokalen Rechner befindet oder irgendwo im Internet.

In VRML können Hyperlinks durch den *Anchor* Knoten sowie dessen Felder realisiert werden. Als Gruppenknoten kann der *Anchor* Knoten diverse andere Knoten als Kinder beinhalten.

Textinformationen werden im Knoten *Text* und dessen Feld *string* als in Anführungszeichen stehende Zeichenketten angegeben. Der Knoten *FontStyle* übernimmt dabei die Formatierungen des Textes. Neben der Festlegung des Schrifttyps, der Schriftgröße und des Schriftstils kann ebenso die Ausrichtung (linksbündig, zentriert, rechtsbündig) des Textes festgelegt werden. Texte können auch horizontal sowie vertikal ausgerichtet werden [Carey 97].

Abbildung 26 zeigt als Codebeispiel die Definition von Text sowie eines Hyperlinks:

```

Transform {
  children Anchor {
    url "inline\rotate\rotrvartery.wrl"
    description "single view and rotate right vertebral artery"
    children [

```

```

Transform { children [
  Shape {
    geometry DEF TEXT1 Text {
      string ["Right Vertebral Artery" ]
      fontStyle DEF RFS FontStyle {
        size 1.5
        justify "LEFT"
        family "SANS"
      } }
    appearance Appearance {
      material Material { diffuseColor 0 0 0 } } }
  Transform { translation 9 0 0 children [
    Shape { geometry Box {size 20 2.5 .5 }
    appearance Appearance {
      material Material {
        transparency 0.8 diffuseColor 0 0 1}} } ] } ] }

```

Abb 26: VRML-Codebeispiel für Text und Hyperlinks

Es kann aber auch eine VRML-Datei in den aktuellen Quellcode eingebunden werden, ohne dabei die aktuell geladene Datei zu ersetzen. Hierzu wird der *Inline* Knoten benötigt, dessen hauptsächliches Ziel es ist den Quelltext in mehrere Module aufzuspalten und diesen somit übersichtlicher zu gestalten. Gleichzeitig wird durch die Aufspaltung in kleinere Module eine schnellere Ladezeit erreicht [Kloss 98].

Abbildung 27 zeigt dazu einen VRML-Codeausschnitt indem zwei mittels GZIP komprimierte Dateien in den Quellcode eingebunden werden.

```

Group { children [
  Background { skyColor 0.8 0.8 0.8 }
  Transform {
    rotation 1 0 0 -1.57
    translation -20.64 -26.27 -54.72
    scale 0.5 0.5 0.5
    children [
      Transform { children [
        Inline { url "inline\arterien.wrl.gz" }
        Inline { url "inline\venen.wrl.gz" } ] }
    ] } ] }

```

Abb. 27: VRML-Codebeispiel für die Einbindung eines Inline Knotens

4.6. Gestaltung von Hintergrund in VRML

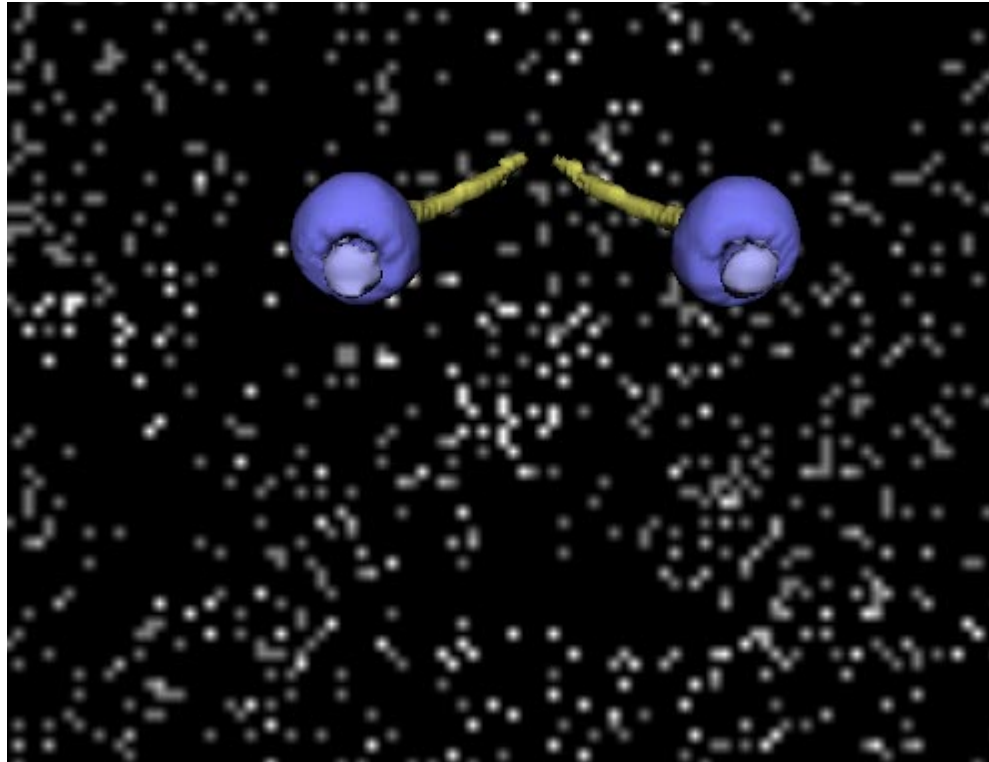


Abb. 28: Zweidimensionale Graphiken als Hintergrunddarstellung

In der obigen Abbildung 28 ist mit den Gestaltungsmöglichkeiten des Hintergrundes experimentiert worden. Der Background Knoten bietet die Möglichkeit, Himmel und Boden farblich bzw. durch den Einsatz zweidimensionaler Graphiken zu simulieren. Während der Boden als Halbkugel mit unendlichem Radius berechnet wird, umschließt der Himmel als vollständige Kugel die gesamte Szene in unendlicher Distanz. Beide Kugeln lassen sich über das Feld *groundColor* und *skyColor* mit Hilfe eines RGB-Farbwertes oder einer zweidimensionalen Graphik beliebig einfärben. Zusätzlich können Farbverläufe zwischen den konzentrischen Kreisen der Kugeln über die Felder *groundAngle* und *skyAngle* definiert werden [Carey 97].

Zur Veranschaulichung der Himmel und Boden Definition in VRML dient die Abbildung 29.

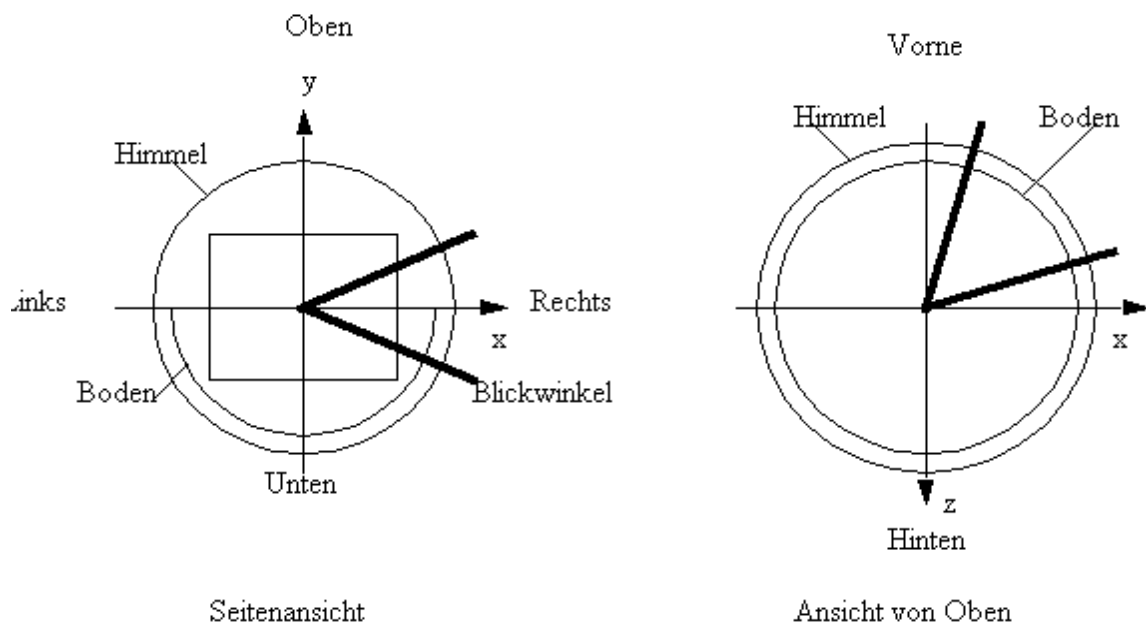


Abb. 29: Skizzierung der Definition von Himmel und Boden

Der folgende VRML-Codeausschnitt erläutert die Definition der Abbildung 30.

```
Background {  
  skyColor [ 0 0 0, 1.0 1.0 1.0 ]  
  skyAngle 1.6  
  groundColor [ 1 1 1, 0.8 0.8 0.8, 0.2 0.2 0.2 ]  
  groundAngle [ 1.2, 1.57 ] }  
}
```

Für die Farbgebung des Himmels und Bodens werden unterschiedliche Grauwerte verwendet, während für die Farbverläufe die jeweils angegebenen Winkelpositionen angegeben sind. Zur Erzeugung eines interpolierten Farbverlaufs muß die Anzahl der Farbwerte immer um eins höher sein als die Winkelwerte.

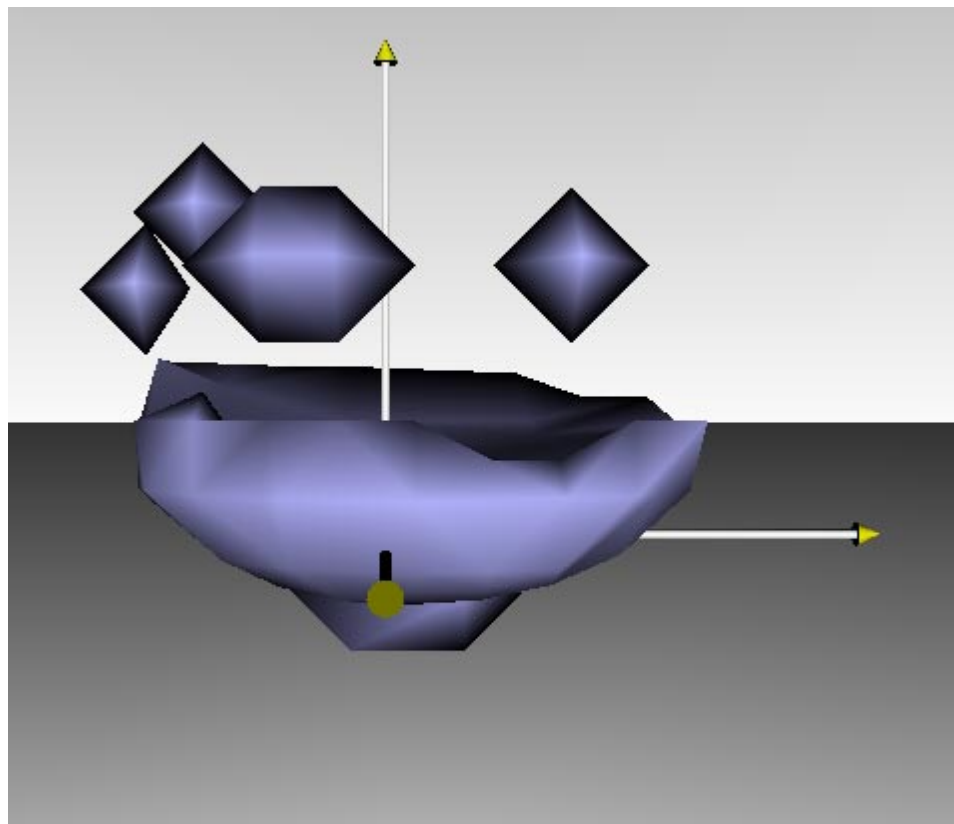


Abb. 30: Farbliche Darstellung von Himmel und Boden. Im Vordergrund ist eine Cornea von oben zu sehen. Die losgelösten Segmente sind Segmentierungsartefakte. Das Koordinatenkreuz veranschaulicht den Richtungswechsel, wenn die Cornea durch den Anwender interaktiv gedreht wird.

4.7. Erzeugung dynamischer VRML-Modelle

Die Dynamik, die VRML durch den Einsatz von Interpolatoren und Sensorknoten bietet, ist von statischer Natur. Das bedeutet, daß Animationssequenzen bzw. Interaktionsabläufe statisch im Programmcode definiert werden müssen. Innerhalb des VRML97 Standards ist das dynamische Nachladen von Programmcode nicht enthalten.

Die Animation wird durch Interpolator-Knoten realisiert, die Werte für die Zwischenphasen von einfachen Animationen liefern. "Einfach" bedeutet in diesem Fall, daß es sich um einen genau definierten Ablauf handelt, der vom Benutzer zwar ausgelöst, aber nicht gesteuert werden kann. Es ist ebenfalls möglich, im VRML-Quellcode eine beliebige Anzahl vordefinierter Kameras bzw. Blickpunkte durch den Knoten *Viewpoint* zu plazieren. Durch die Wahl eines Viewpoints aus dem Auswahlfeld des Steuermenüs des Browsers kann der Benutzer seinen Blickpunkt auf das Objekt verändern. Um Interaktion ohne das Steuermenü des Browsers bzw. des Plug Ins zu erreichen, sind Sensoren nötig. Sie sind die Quelle für Ereignisse. Mit ihnen kann der Benutzer interagieren, indem er z.B. einen Schalter betätigt und damit vordefinierte Ereigniskaskaden auslöst. Die verschiedenen Interpolatoren bzw. Sensorknoten sind in der Abbildung 2 aufgelistet.

4.7.1. Interaktion

Die Interaktion in VRML wird durch Sensor Knoten realisiert und stellt die Quelle für Ereignisse dar. Ein besonders wichtiger Sensor ist der *TimeSensor*. Er registriert nicht nur laufende Zeit, sondern generiert auch von sich aus fortlaufende Zeitsignale. In der Beispieldatei "ColorSkin.wrl" wurde der *TimeSensor* zur Steuerung der Farbanimation eingesetzt. Ebenso sind in der Abbildung 44 des in dieser Arbeit entstandenen zweiten Modells die Knöpfe mit Hilfe des *TimeSensors* farbanimiert worden.

Der *PlaneSensor*, der *TouchSensor* und der *SphereSensor* zählen zu den sogenannten Drag-Sensoren, die auf Bewegung des Mauszeigers reagieren.

Die Verbindung eines Objektes mit einem *SphereSensor* erlaubt eine beliebige Drehung des Objektes um einen Drehpunkt. Dies wird in der Beispieldatei "InteractiveCornea.wrl" (siehe Abbildung 30) eingesetzt. Wird auf die Linse geklickt und wird der Mausknopf gehalten, so kann die Linse beliebige Rotationen ausführen. Hinzu kommt, daß in diesem Beispiel der Einsatz von mehreren Sensoren, die mit einem Objekt verbunden werden, experimentiert worden ist. Da pro

Objekt immer nur ein Sensor zur Zeit aktiv sein kann, ist dieses Modell um einen interaktiven Button ergänzt worden, der das Hin- und Herschalten zwischen dem *SphereSensor* und dem *TouchSensor* ermöglicht. Der *TouchSensor* ermöglicht in diesem Zusammenhang durch direktes Anklicken der Linse, das diese unsichtbar wird. Erneutes Klicken läßt die Linse wieder erscheinen. Das Klicken auf den Button (siehe Abbildung 31) läßt den jeweils anderen Sensor aktiv werden.

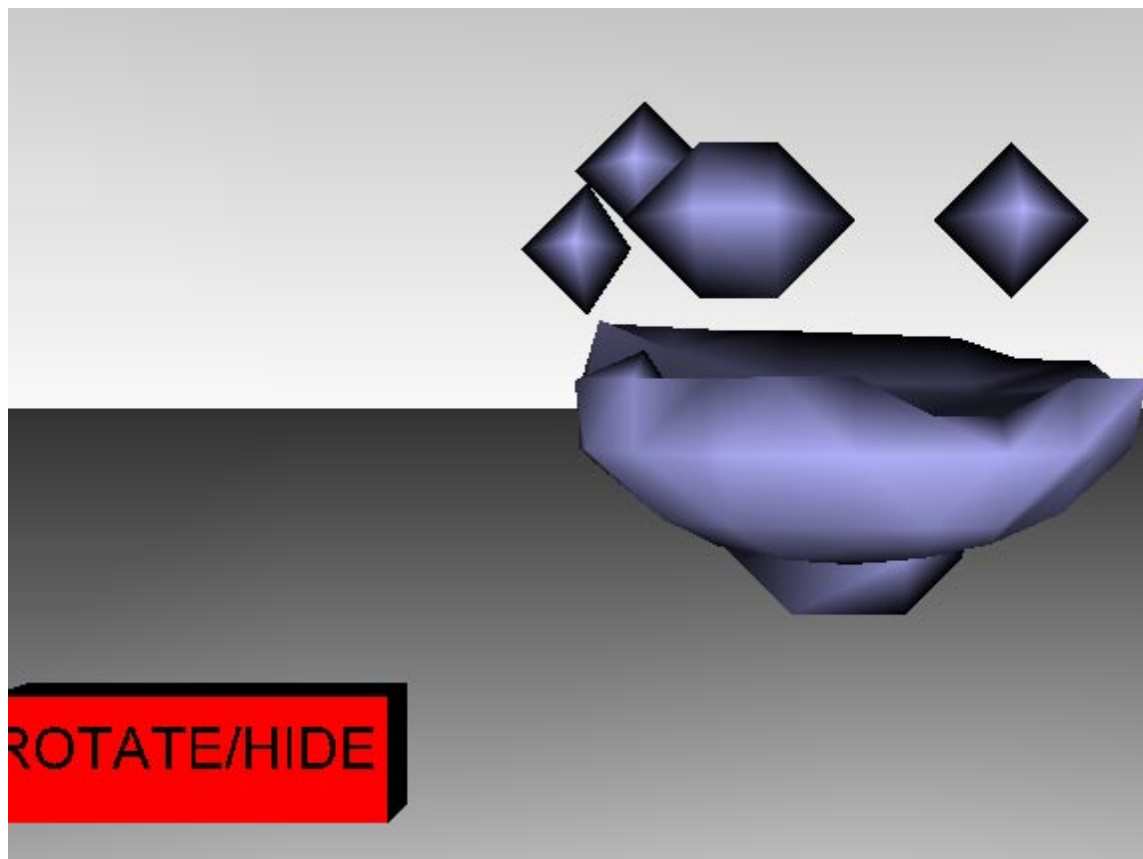


Abb. 31: Kombination von Sensoren durch einen Umschaltbutton am Beispiel Cornea

Das ein Objekt mit einem Drag-Sensor verbunden ist, ist lediglich durch den sich verändernden Mauszeiger zu erkennen, sofern dieser über das Objekt geführt wird.

Der *TouchSensor* ist u.a. in der Beispieldatei "InteractiveCornea.wrl" zum Einsatz gekommen. Durch Anklicken der Linse wird diese unsichtbar. Durch erneutes Anklicken wird die Linse wieder dargestellt (siehe Abbildung 31). Ebenso sind die Gefäße, die im Modell 1 dargestellt werden, mit dem *TouchSensor* verbunden.

Durch den *PlaneSensor* können Objekte innerhalb einer Ebene verschoben werden. In der Abbildung 32 wird ein VRML-Codebeispiel eines *PlaneSensors* gezeigt. Dem *PlaneSensor* wird dabei ein Bewegungsbereich in XY-Ebene zugeordnet. Durch Vertauschung der Achsen kann ein Bewegungsbereich in einer anderen Ebene definiert werden. Der in diesem Beispiel als Vertikal-Regler definierte Knoten gibt das Aussehen der verschiebbaren Ebene an. In diesem Beispiel ist dies eine rechteckige, blaue, leicht transparente Ebene, wie sie im VRML-Modell 2 verwendet wird.

```

Transform { translation 15 100 100
  rotation 0 0 1 -1.5
  children [
    DEF Vertikal_Bewegung PlaneSensor {
      minPosition 0 -135
      maxPosition 0 33
    }
    DEF Vertikal_Regler Transform {
      children Shape {
        appearance Appearance { material Material {
          diffuseColor 0 0 1 transparency .3 } }
        geometry Box { size 195 .25 195 }
      }
    }
  ]
}

```

Abb. 32: VRML-Codebeispiel für einen *PlaneSensor*

Damit eine Interaktion überhaupt stattfinden kann, ist es notwendig, daß Informationen zwischen definierten Knoten ausgetauscht werden können. Hierfür gibt es Ein- bzw. Ausgabefelder, die als Schnittstellen definiert sind.

Zur Weiterleitung von ausgelösten Ereignissen dienen die Ein- bzw. Ausgabeschnittstellen der Knoten. Ankommende Ereignisse sind Meldungen, die über ein *eventIn* Feld empfangen werden und meist eine Änderung eines Feldwertes beim empfangenden Knoten auslösen. Über das Feld *eventOut* kann der Knoten wiederum Meldungen an andere Knoten weitersenden, um beispielsweise die Änderung eines Feldwertes zu signalisieren. Die eigentliche Weiterleitung der Ereignisse erfolgt durch den ROUTE Befehl. Dies ist ein Kontrollmechanismus zur Steuerung der Kommunikation von Meldungen bzw. Ereignissen zwischen Knoten über ihre definierten Schnittstellen. Die Datentypen der beiden Schnittstellen (*eventIn* und *eventOut*) müssen dabei

übereinstimmen. Ist dies nicht der Fall, kann ein *Script* Knoten dazwischen geschaltet werden, so daß die Weiterleitung der Ereignisse möglich wird. Zudem kann ein *Script* Knoten Zustände und Parameter speichern und verwalten. Die darin enthaltenen Funktionen werden dann ausgeführt, wenn der Knoten ein entsprechendes Ereignis empfängt. Das Besondere am *Script* Knoten ist, daß das darin enthaltene Feld *url* auf Java Klassen verweisen bzw. direkt Javascript-Code enthalten kann [Carey 97].

Abbildung 33 zeigt ein Beispiel eines *Script* Knotens sowie die dazugehörigen ROUTE Befehle, die den in Abbildung 32 erläuterten PlaneSensor beeinflussen.

```

DEF S Script {
    eventIn      SFTime      touchTime
    field        SFInt32     choice      0
    eventOut     SFInt32     whichChoice
    eventOut     SFVec3f     PlanePos
    url "javascript:
        function initialize () {
            whichChoice = 0;    }
        function touchTime (value,time) {
            if (whichChoice == 12) whichChoice = 0;
            else ++whichChoice;
            choice = whichChoice ;
            if (choice == 0) PlanePos.y = 0;
            if (choice == 1) PlanePos.y = 10;
            if (choice == 2) PlanePos.y = 20;
            }"    }

ROUTE TS.touchTime TO S.touchTime
ROUTE S.whichChoice TO SW.whichChoice
ROUTE S.PlanePos TO Vertikal_Regler.set_translation

```

Abb. 33: VRML-Codebeispiel für einen Script Knoten und den dazugehörigen ROUTE-Befehlen

Das Ausführungsmodell (engl. Execution Model) beschreibt die Art und Weise wie *Script* Knoten oder ROUTE-Befehle innerhalb des VRML-Kontextes abgearbeitet werden. Während eines jeden

Signals im Browser werden alle potentiellen Ereignisse ausgewertet. Sobald ein Sensor oder ein Script ein anfängliches Ereignis generiert hat, prüft der Browser, ob ROUTE-Befehle zu anderen Knoten bestehen. Falls ja, wird das Ereignis entlang der ROUTE-Befehle zu anderen Knoten weitergeleitet. Diese anderen Knoten können antworten, indem sie immer weitere Ereignisse generieren. Dieser Prozeß entspricht einer Ereigniskaskade. Alle Ereignisse, die während einer Ereigniskaskade generiert werden, erhalten die gleiche Zeitmarke wie das anfängliche Ereignis. Sie werden so betrachtet als würden sie gleichzeitig ablaufen. Einige Sensoren generieren simultan verschiedene Ereignisse. In diesen Fällen kann jedes generierte Ereignis eine andere Ereigniskaskade initiieren [Carey 97].

4.7.2. Animation

Interpolatoren werden unabhängig von ihrer Position im Szenegraph gehandhabt. Sie sind immer aktiviert und reagieren entsprechend auf Eingabeergebnisse.

Für die Erzeugung von Animationssequenzen stehen Interpolationsfunktionen zur Verfügung mit denen lineare Interpolationen ausgeführt werden können. Die Interpolatorfunktion wird über dafür definierte Parameter spezifiziert [Carey 97].

In der Beispieldatei "ColorSkin.wrl" ist der Kopf mit einem *ColorInterpolator* verbunden worden. Dieser erzeugt ausgehend von einer Zahl von Farbwerten einen neuen interpolierten RGB-Farbwert. Zu beachten ist hierbei, daß die Eingabefelder als RGB-Farbwerte verarbeitet werden. Die Interpolation hingegen im HSV-Farbraum erfolgt [Carey 97].

Abbildung 34 zeigt die Definition des *ColorInterpolator* als VRML-Codeausschnitt aus der Datei "ColorSkin.wrl".

```
DEF Obj1 Transform { #skin, 8138 Punkte, 16268 Flaechen
  translation -37.92 -50.97 -95.47
  scale 0.5 0.5 0.5
  rotation 1 0 0 -1.57
  children [ Shape {
    appearance Appearance { material DEF M Material {
      shininess 1 diffuseColor 1.00 0.50 0.00 } }
    geometry IndexedFaceSet { creaseAngle 3.14 solid FALSE
      coord Coordinate {
        point [
          81.00 55.90 8.00,
```

```

        (...)]}
    coordIndex [
        96 95 85 -1,
        (...) ]}}
DEF Timer TimeSensor {
    cycleInterval 5.0
    loop TRUE      }
DEF Farbe ColorInterpolator {
    key [0.0,0.25, 0.5,0.75, 1.0]
    keyValue [ 1 0.5 0, 1 1 0, 1 0 1, 0 0 1,1 0.5 0  ]
} ]}
ROUTE Timer.fraction_changed TO Farbe.set_fraction
ROUTE Farbe.value_changed TO M.set_diffuseColor

```

Abb. 34: VRML-Code eines ColorInterpolator

Mit Hilfe des *OrientationInterpolator* kann die Lage bzw. die Orientierung eines Objektes verändert werden. Die Veränderung der Orientierung erfolgt durch fest definierte Schritte, deren Animation der Benutzer lediglich auslösen kann. Abbildung 35 zeigt die Definition eines *OrientationInterpolator* im Zusammenhang mit einem *TimeSensor*.

```

DEF TObj3 TimeSensor { cycleInterval 30.0 }
    DEF Rotierer3 OrientationInterpolator {
        key [ 0.125, 0.175, 0.25, 0.375, 0.5, 0.625, 0.75, 0.875,
1.0 ]
        keyValue [ 0 0 1 0.7, 0 0 1 1.1, 0 0 1 1.5,0 0 1 2.2,
0 0 1 3.14, 0 0 1 3.84, 0 0 1 4.5, 0 0 1 5.2, 0 0 1 6.28 ]}
ROUTE TObj3.fraction_changed TO Rotierer3.set_fraction
ROUTE Rotierer3.value_changed TO eyes.set_rotation

```

Abb. 35: VRML-Codeausschnitt eines OrientationInterpolator

Komplexe Darstellungen von Oberflächenmodellen lassen sich nur mit Verzögerungszeiten durch die Browserfunktionalitäten interaktiv beeinflussen. Ein Interaktion oder Animation, die direkt im Quellcode des VRML-Programms definiert ist, wird verzögerungsfrei vom Browser ausgeführt. Das Ausschalten der Browserfunktionalität kann durch den Knoten *NavigationInfo* möglich gemacht werden, der die Modi "None, Walk und Examine" zulässt.

4.7.3. Viewpoints

Ein *Viewpoint* Knoten beschreibt eine Kamera, die an einem bestimmten Ort innerhalb der Szene aufgehängt ist und in eine bestimmte Richtung schaut. Das Feld *position* beschreibt den Ort, an dem die Kamera hängt. Die Winkelöffnung der Kamera kann durch das Feld *fieldofView* eingestellt werden. Die Blickrichtung der Kamera wird durch das Feld *orientation* angegeben. Die ersten drei Werte entsprechen den Koordinaten eines zur Rotationsachse kollinearen Vektors, während der vierte Wert den Rotationswinkel um diese Achse darstellt. Im Booleschen Feld *jump* wird festgelegt, ob sich der Kamerawechsel sprunghaft oder interpoliert vollzieht. Die Default Einstellung TRUE bewirkt, daß der Kamerawechsel unmittelbar d.h. ohne Zwischenschritte erfolgt. Das *description* Feld vom Typ SFString ermöglicht es, den jeweiligen Blickpunkt mit einem Namen zu kennzeichnen.

Die Abbildung 36 veranschaulicht eine Kamerasicht, die die Szene des zweiten Modells von der linken Seite zeigt.

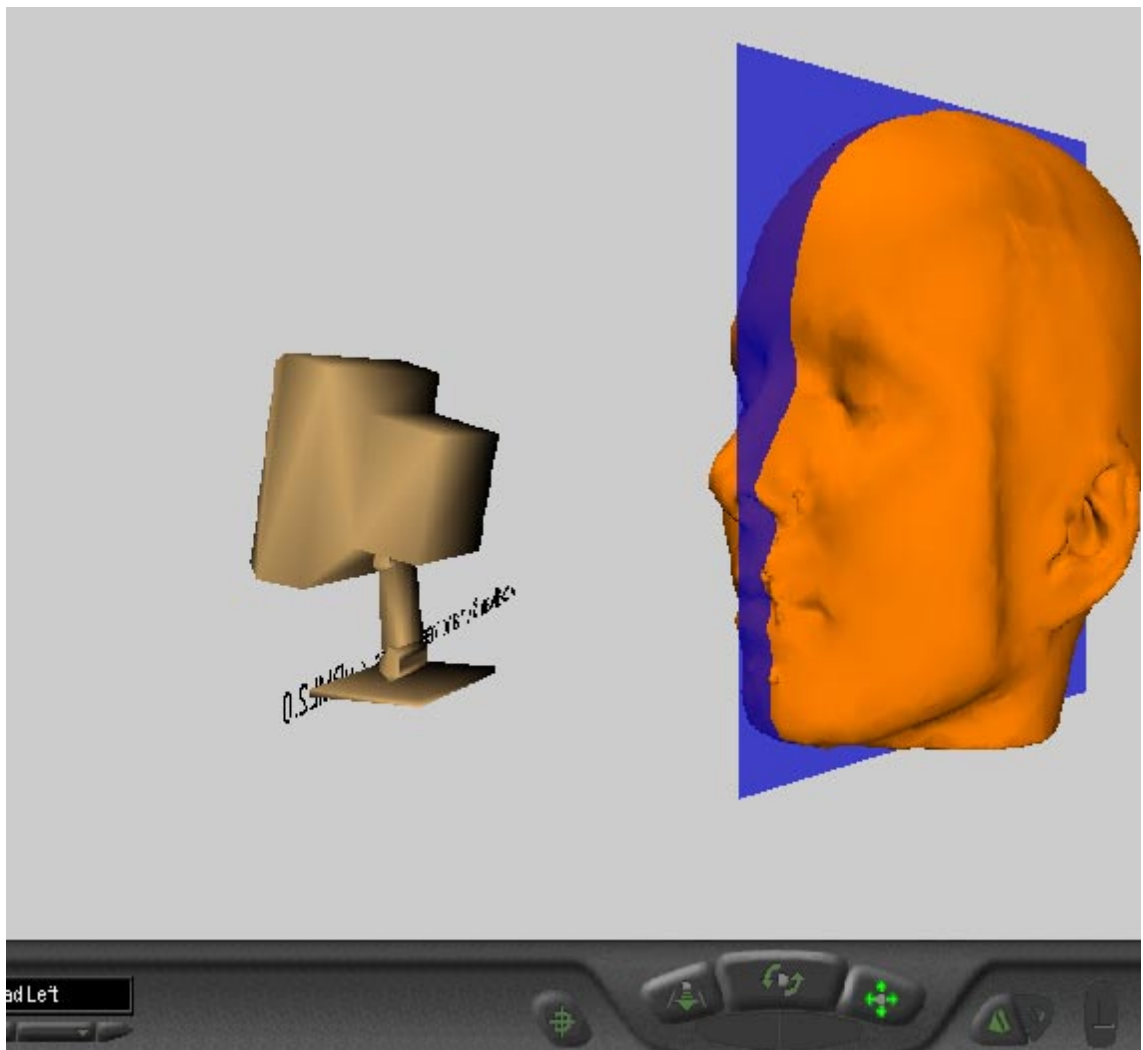


Abb. 36: Viewpoint: "Head Left" des zweiten Modells

Die meisten Browser ermöglichen es per Menü, zwischen den einzelnen vordefinierten Kameras zu wechseln. Beim Laden einer VRML-Datei ist der im Quelltext zu oberst definierte Viewpoint der, der geladen wird.

Der *NavigationInfo* Knoten wird konzeptuell als Kindknoten eines Viewpoints betrachtet und wird, falls dieser wechselt, vom Browser in den neuen Viewpoint als Kindknoten integriert.

Durch den *NavigationInfo* Knoten können Voreinstellungen des Browsers beeinflusst werden.

Das Feld *type* erlaubt es beispielsweise, Teile oder das ganze Navigationsmenü des Browsers auszuschalten, so daß dem Benutzer zur Navigation nur die vordefinierten Viewpoints erlaubt werden.

Der Viewpoint Knoten sowie der NavigationInfo Knoten haben gemeinsam, daß stets nur einer von jedem Typ gerade aktiv sein kann. Es können aber dennoch mehrere Knoten definiert sein. Der Browser verwaltet einen unabhängigen Stack für jeden Typ von einbindbaren Knoten [Carey 97].

Die Abbildung 37 zeigt als Beispiel VRML-Code für eine Liste von drei definierten Viewpoints:

```
#VRML V2.0 utf8
Viewpoint {
  position 0 1 100
  jump FALSE
  orientation 0 0 1 0
  description "Head Right" }
Viewpoint {
  position 0 1 100
  jump FALSE
  orientation 0 0 1 0
  description "Head Back" }
Viewpoint {
  position 0 1 100
  jump FALSE
  orientation 0 0 1 0
  description "Head Left" }
```

Abb. 37: Viewpoint Definitionen im VRML-Code

Der dritte Viewpoint dieses Codebeispiels mit dem Titel "Head Left" ist in der Abbildung 36 zu sehen. Mit der Wahl des Viewpoints "Start" wird die Ausgangsstellung wieder hergestellt.

5. Ergebnisse und Anwendungen

Im Rahmen dieser Diplomarbeit wurden drei verschiedene VRML-Modelle erstellt, an denen verschiedene Möglichkeiten zur Interaktion und Animation exemplarisch gezeigt werden sollen. Die VOXEL-MAN/brain [Höhne CD-ROM 95] erzeugt und anschließend manuell mit verschiedenen im vierten Kapitel erläuterten Methoden zur Interaktion und Animation erweitert. Die in den entstandenen Modellen eingesetzten Animationen und Interaktionen basieren auf dem Standard VRML97. Die Möglichkeiten, die dieser Standard bietet, sowie die unterschiedlichen Zielsetzungen der einzelnen Modelle werden im folgenden untersucht, beurteilt und bewertet.

Des Weiteren liegt dieser Arbeit eine CD-ROM bei. Die CD-ROM enthält die im Rahmen dieser Arbeit entstandenen VRML-Modelle 1 bis 3 sowie die erläuterten Beispieldateien und zusätzlich den CosmoPlayer-Browser als Plug-In unter Netscape.

5.1. Einführende Experimente

Ziel dieser Diplomarbeit ist es, VRML-Modelle zu erstellen, die auf Volumendaten basieren. Zur Darstellung dieser Volumenmodelle können unterschiedliche Darstellungsgeometrien verwendet werden. Die im wesentlichen verwendete Geometrie ist die Darstellung von Körpergeometrien, die das Volumen der einzelnen Objekte als Körperoberflächenmodelle wiedergeben. Eine andere Darstellungsgeometrie ist die Drahtgitterdarstellung. Die "Soft Tissues" des Kopfes beispielsweise beinhalten die Polygondaten und -flächen aller Weichteile des Kopfes und können sehr eindrucksvoll als Drahtgittermodell (siehe Abbildung 38) dargestellt werden. Durch die Komplexität der Polygondaten erzielt diese Drahtgitterdarstellung eine beeindruckende Wirkung, obwohl die Polygondaten vor der Konvertierung nach VRML mit einem Reduktionslevel 4 reduziert wurden. Vergleichsweise dazu ist die Drahtgitterdarstellung der Kopfhaut (siehe Abbildung 39), die nicht vor der Konvertierung nach VRML reduziert wurde, weitaus weniger beeindruckend.

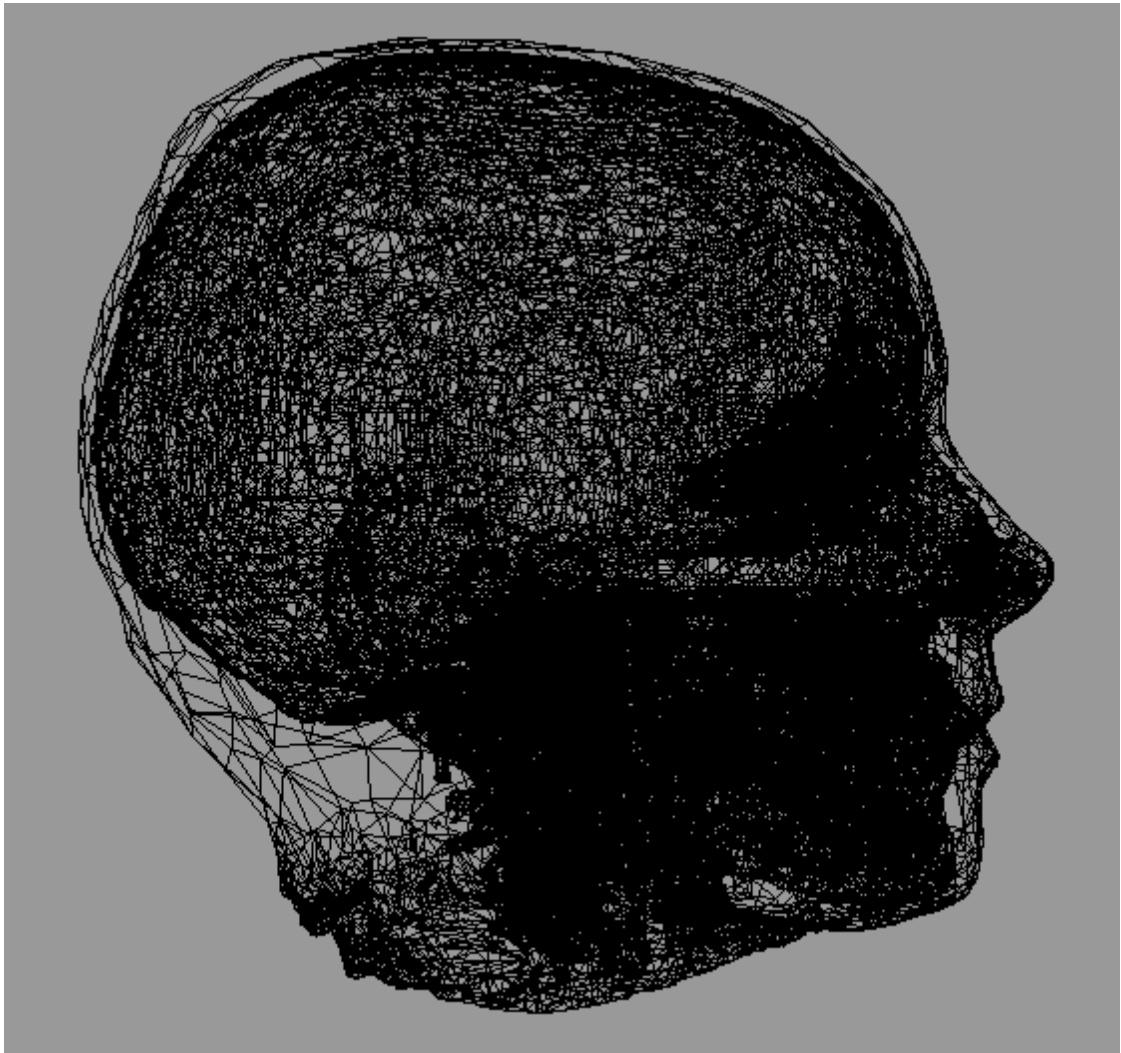


Abb. 38: Drahtgittermodell der Weichteile des Kopfes

Die folgende Abbildung 39 zeigt im Vergleich dazu die Kopfhaut als Drahtgittermodell. Die Polygondaten der Kopfhaut sind ohne Reduzierung nach VRML konvertiert worden. Die Polygondaten bestehen bei der Darstellung der Kopfhaut aus 8139 Polygonpunkten und 16269 Polygonflächen. Die VRML-Datei umfaßt dabei immer noch 1,1 MB. Im Vergleich hierzu besteht die bereits auf reduzierten Polygondaten konvertierte VRML-Datei der "Soft Tissues" aus 65230 Polygonpunkten und 130459 Polygonflächen. Insgesamt umfaßt diese Datei 8,9 MB.

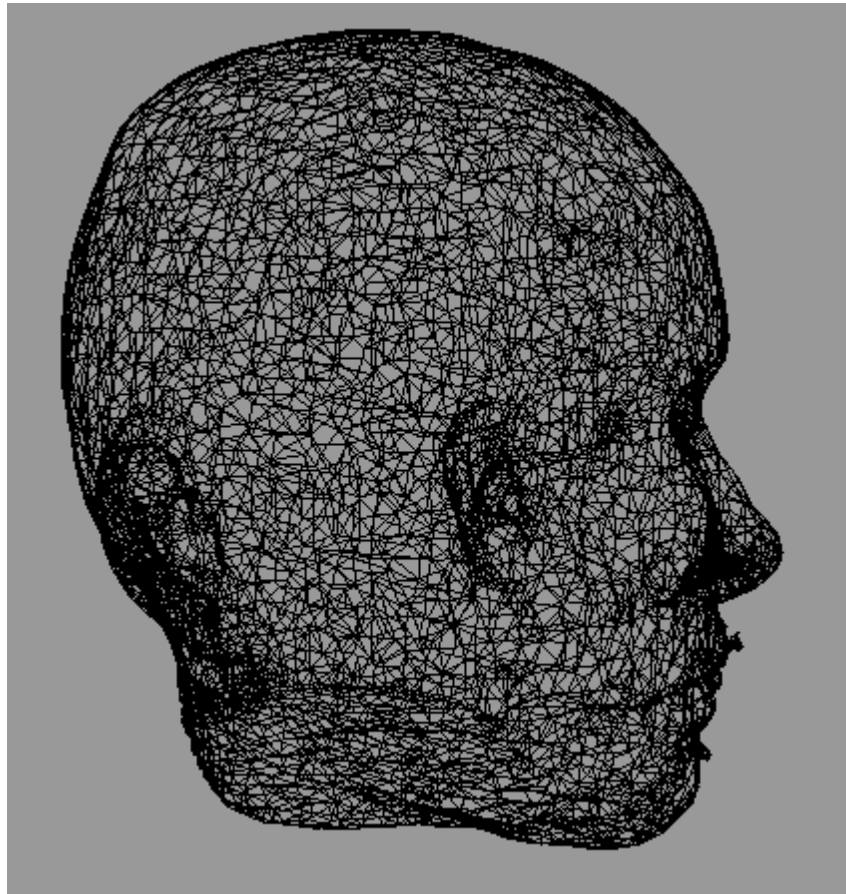


Abb. 39: Drahtgittermodell der äußeren Oberfläche der Kopfhaut

In der Untersuchung der Reduktionsergebnisse im Rahmen dieser Arbeit ist die bestmögliche Reduzierung mit dem Reduktionslevel 4 erreicht worden. Ab Reduktionslevel 5 sind Glättungen und Vereinfachungen der Objekte deutlich sichtbar. Mit dem Reduktionslevel 4 kann eine bis zu 50 prozentige Polygonreduktion ohne Verlust der realistischen Darstellungskraft erreicht werden.

Die Darstellung des Gehirns, das sich als Objekt mit vielen Windungen und Unebenheiten auszeichnet, eignet sich hervorragend für Versuche, die Komplexität der Polygondaten durch das Programm von Frank Wilmer [Wilmer 93] zu untersuchen. Während die Originaldaten nach VRML konvertiert wurden und in Abbildung 40 zu sehen sind, wurde dieses Objekt mit jedem möglichen Reduktionslevel reduziert. Ein optimales Reduktionsergebnis von ca. 45 % konnte ohne Einbußen der realistischen Darstellungskraft des Gehirns erreicht werden und ist im Vergleich zu den Originaldaten in Abbildung 41 dargestellt. Die Verwendung eines höheren

Reduktionslevels als der hier verwendete führt zu sichtbaren Glättungen der Gehirnstrukturen mit deutlichen Einbußen des realistischen Aussehens.

Der Dateiumfang der aus den Originaldaten erstellten VRML-Datei umfaßt 15,2 MB. Die reduzierte VRML-Datei des Gehirns umfaßt nur noch 8,3 MB. Während die reduzierte Version des VRML-Oberflächenmodells nur noch 65624 Polygonpunkte und 132376 Polygonflächen umfaßt, beinhaltet das Polygonmodell der vollständigen Darstellung des Gehirns in VRML 121518 Polygonpunkte und 245224 Polygonflächen.

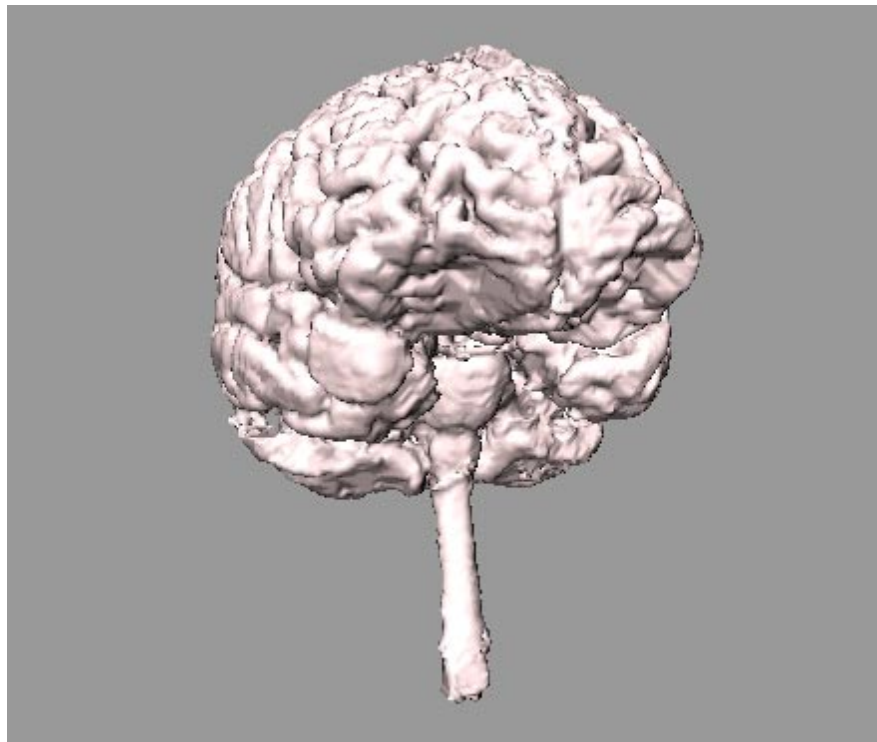


Abb. 40: Darstellung des aus den Originaldaten konvertierten VRML-Modells des Gehirns (15,2 MB)

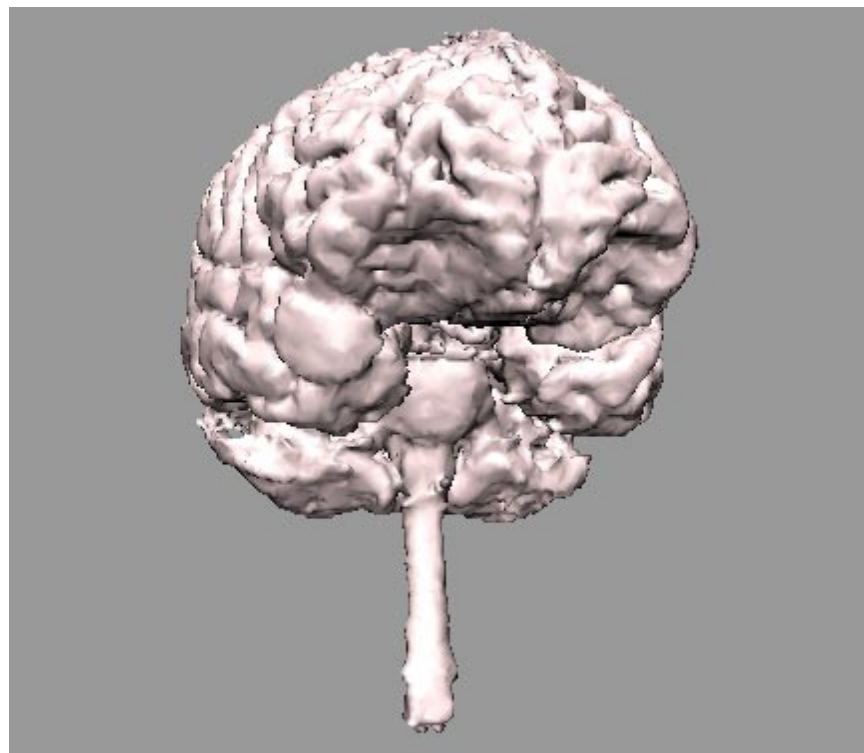


Abb. 41: Darstellung des aus den reduzierten Daten konvertierten VRML-Modells des Gehirns (8,3 MB)

Die räumliche Darstellungskraft von VRML-Objekten kann durch die Gestaltung des Hintergrundes unterstützt werden und ist in einzelnen Experimenten untersucht worden. Abbildung 28 zeigt z.B. eine zweidimensionale Graphik eines Sternenhimmels, der als Hintergrund für die Darstellung der Augen eingesetzt wurde. Ein dreidimensionaler Effekt kann durch die Darstellung von Himmel und Boden, wie sie in Abbildung 30 eingesetzt wurden, erreicht werden. In diesem Bereich sind noch eine Vielzahl von Varianten möglich. Im Rahmen dieser Arbeit wurde nur experimentell darauf eingegangen.

Bevor es zur Entwicklung der umfangreicheren Modelle kam, wurde mit einzelnen Objekten die Interaktions- sowie Animationsmöglichkeiten, die VRML bietet, getestet. Die Verwendung von Farbanimationssequenzen wurde z.B. durch die Kopfdarstellung der Datei "ColorSkin.wrl" getestet. Interaktionen anhand der Linse des Auges können durch das Öffnen der Datei "InteraktiveCornea.wrl" ausgeführt werden und sind in den Abbildungen 30 und 31 dargestellt. Das Ergebnis dieser Experimente bildeten den Anfang für die Entwicklung der komplexeren zum Teil animierten und interaktiv beeinflussbaren folgenden Modelle.

5.2. Modell 1: Interaktive Darstellung von Gefäßen des Kopfes

Dieses VRML-Beispielmodell, das sich aus beschrifteten Hyperlinks und der Darstellung der Gefäße des Kopfes zusammensetzt (siehe Abbildung 42) zeichnet sich durch seinen modularen Aufbau aus. Die geschaffene modulare Struktur des Modells dient dem schnelleren Laden der umfangreichen Dateien.

Das Modell 1 wird durch die Datei "vessels.wrl" aufgerufen, welches die Arterien sowie die Venen des Kopfes im Mittelpunkt des Browserfensters dargestellt. Dabei sind alle Arterien des Kopfes zu einer Datei namens "arterien.wrl" zusammengefaßt worden. Ebenfalls wurden alle Venen des Kopfes in eine Datei "venen.wrl" zusammengefaßt. Beide Dateien werden nachträglich in den Browser geladen. Das Nachladen der medizinischen Objekte gestaltet zum einen die Datei "vessels.wrl" übersichtlicher und zum anderen wird die nur 21 KB umfassende Datei "vessels.wrl" schneller geladen. In der Statuszeile, unten im Browserfenster, wird das Nachladen der Datei "arterien.wrl", die 1,9 MB umfaßt, und der Datei "venen.wrl", die 3,8 MB umfaßt, prozentual zur Ladezeit angegeben. Wären alle Dateien in eine Datei zusammengefaßt, würde diese Datei von ca. 6 MB einige Minuten an Ladezeit in Anspruch nehmen, bevor sie im Browserfenster angezeigt würde.

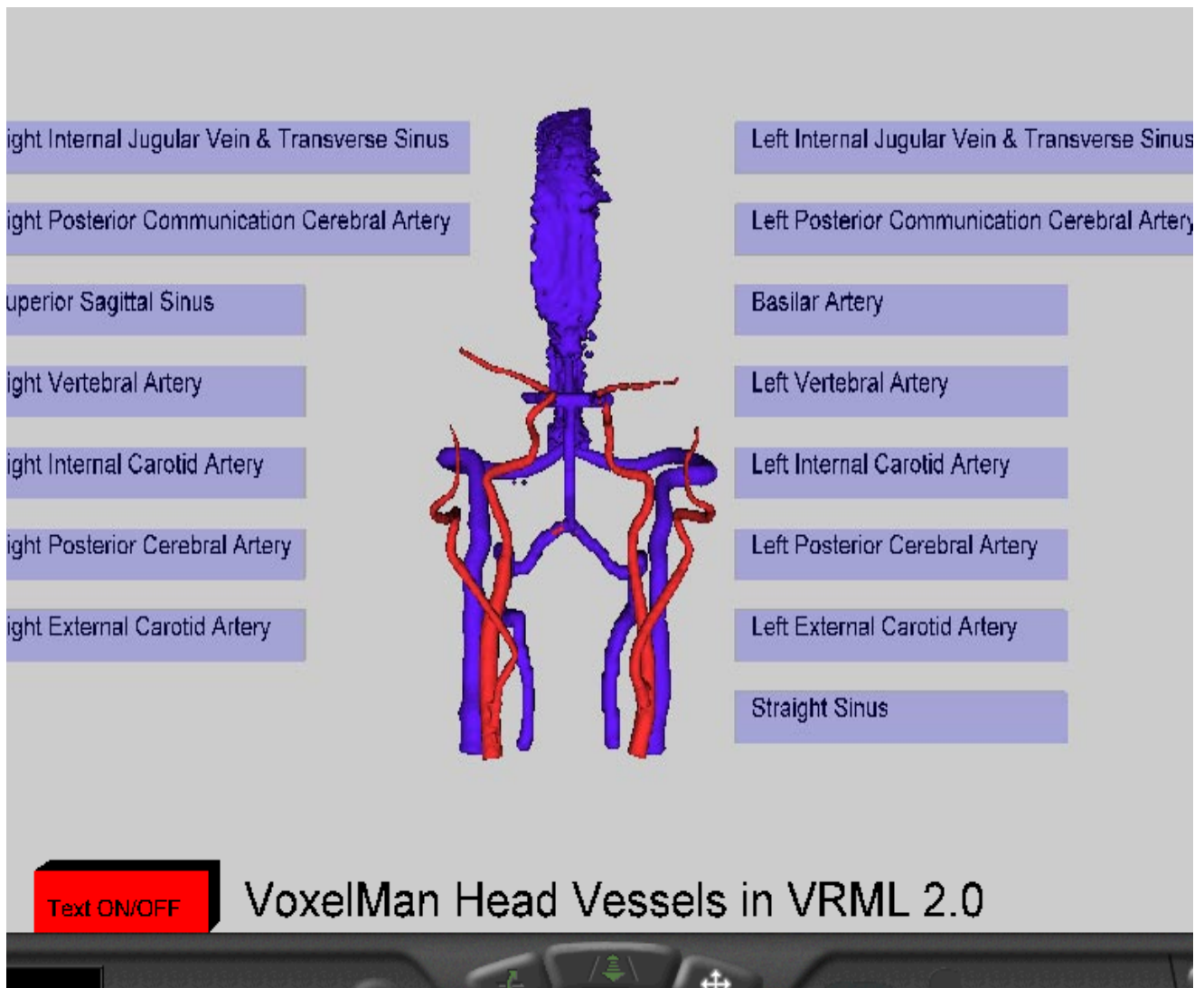


Abb. 42: Modell 1: Text und Hyperlinks

In diesem Modell sind die violett hinterlegten Kästen mit den Schriftzügen rechts und links neben den dargestellten Gefäßen als Hyperlinks im VRML-Programm eingebunden.

Es wird die eingebundene Sprungzieldatei in das Browserfenster geladen, wenn auf einen der violett hinterlegten Hyperlinks geklickt wird. In diesem Modell können so die einzelnen Venen bzw. Arterien angezeigt werden. Die Abbildung 43 zeigt eine Sprungzieldatei eines Hyperlinks.

Unabhängig davon können die Hyperlinks durch den dargestellten Button unsichtbar gemacht werden, so daß lediglich die Venen und Arterien sichtbar sind. Diese sind zudem ebenfalls interaktiv beeinflussbar, indem durch Anklicken der Gefäße eine statisch festgelegte Auswahl

angezeigt wird. Bei wiederholtem Anklicken der Gefäße werden jeweils nur die Arterien, nur die Venen und wieder alle Gefäße des Kopfes gemeinsam in einer Endlosschleife angezeigt.

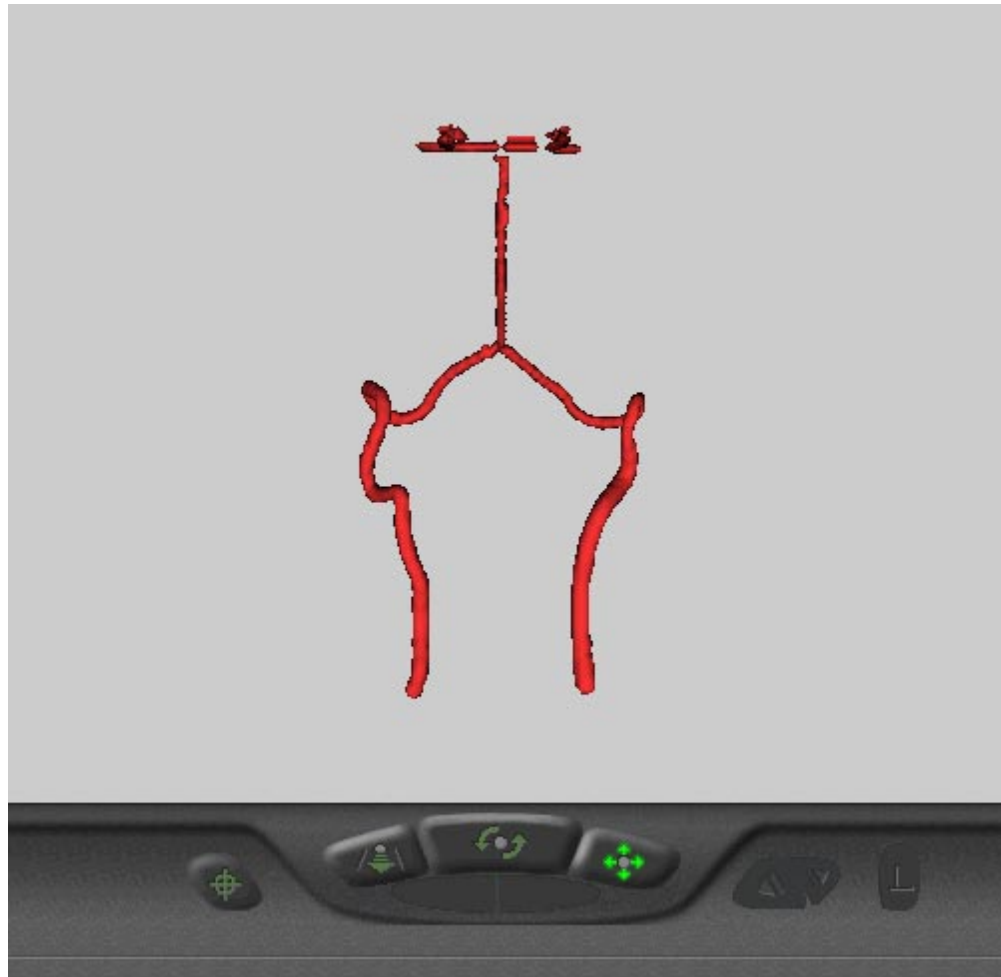


Abb. 43: Darstellung ausgewählter Halsarterien als Sprungzieldatei eines Hyperlinks

Der Einsatz von Hyperlinks innerhalb dieser VRML-Datei ermöglicht es, in diesem Modell eine detaillierte Darstellung der einzelnen Objekte, die Venen bzw. Arterien, darzustellen.

Ziel bei der Entwicklung dieses Modells war es, bestmögliche Interaktionsergebnisse bei Erstellung von modularen Strukturen zu erreichen, so daß lange Ladezeiten der komplexen Objekte auszu-schließen sind.

Fazit:

Dieses Ziel wurde erreicht, denn die Interaktionsmöglichkeiten werden unmittelbar vom Browser ausgeführt, sofern die Dateien geladen sind. Das Laden der relativ kleinen Datei "vessels.wrl" geschieht sehr schnell. Das Nachladen der komplexeren Dateien "arterien.wrl" und "venen.wrl"

dauert je nach Rechnergeschwindigkeit entsprechend länger, wird aber prozentual zur Ladezeit in der Statuszeile des Browsers mitgeteilt.

5.3. Modell 2: Interaktive Darstellung von Schnittflächen des Kopfes

Das Ziel des in dieser Arbeit entstandenen zweiten Modells ist es, interaktive Schnitte des Kopfes anzuzeigen. Aus medizinischer Sicht sind dabei horizontale, vertikale sowie transversale Schnitte von Interesse. Die Erfahrungen der Erstellung von modularen Strukturen aus dem ersten Modell wurden in diesem Modell weiterverarbeitet. Zur übersichtlichen Gestaltung der modularen Struktur zeigt die hierarchisch übergeordnete Datei "Button.wrl" (siehe Abbildung 44) durch die beschrifteten Buttons die zur Auswahl stehenden Schnittebenen an.

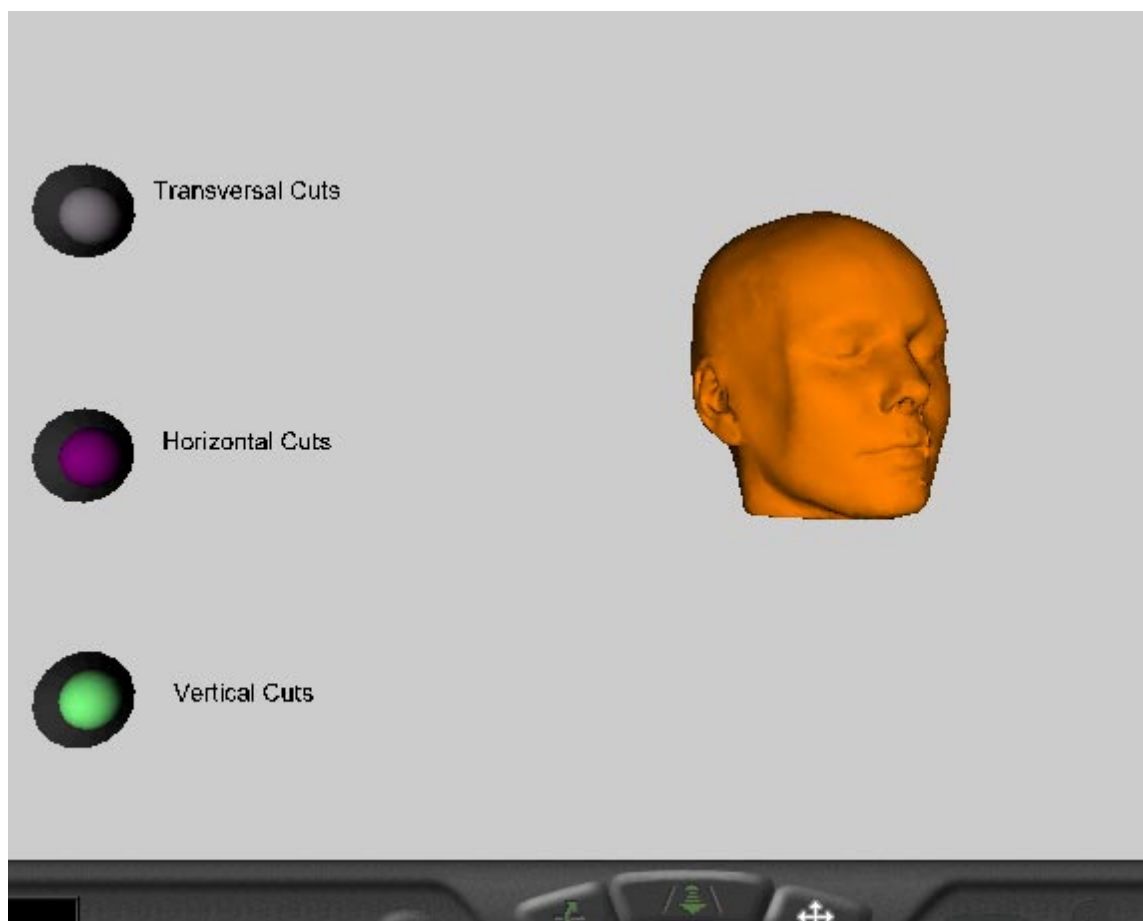


Abb. 44: Modell 2: Auswahl interaktive Darstellungen von Schnittflächen des Kopfes

Die Auswahl einer möglichen Schnittebene kann durch Anklicken eines farbanimierten Buttons vorgenommen werden, der als Hyperlink realisiert wurde, so daß die aufgerufene Sprungzieldatei die Datei "Button.wrl" vollständig ersetzt. Diese gewählte Möglichkeit, jeweils nur eine Schnittebene zur Zeit anzuzeigen, reduziert die Komplexität des VRML-Codes erheblich. Über die "Zurück" bzw. "Back" Taste des Web-Browsers wird die zuvor geladene Datei "Button.wrl" wiederhergestellt, so daß jederzeit eine andere Schnittebene ausgewählt werden kann.

Eine weitere Animation läßt das Oberflächenmodell des Kopfes in einer Endlosschleife um die y-Achse rotieren.

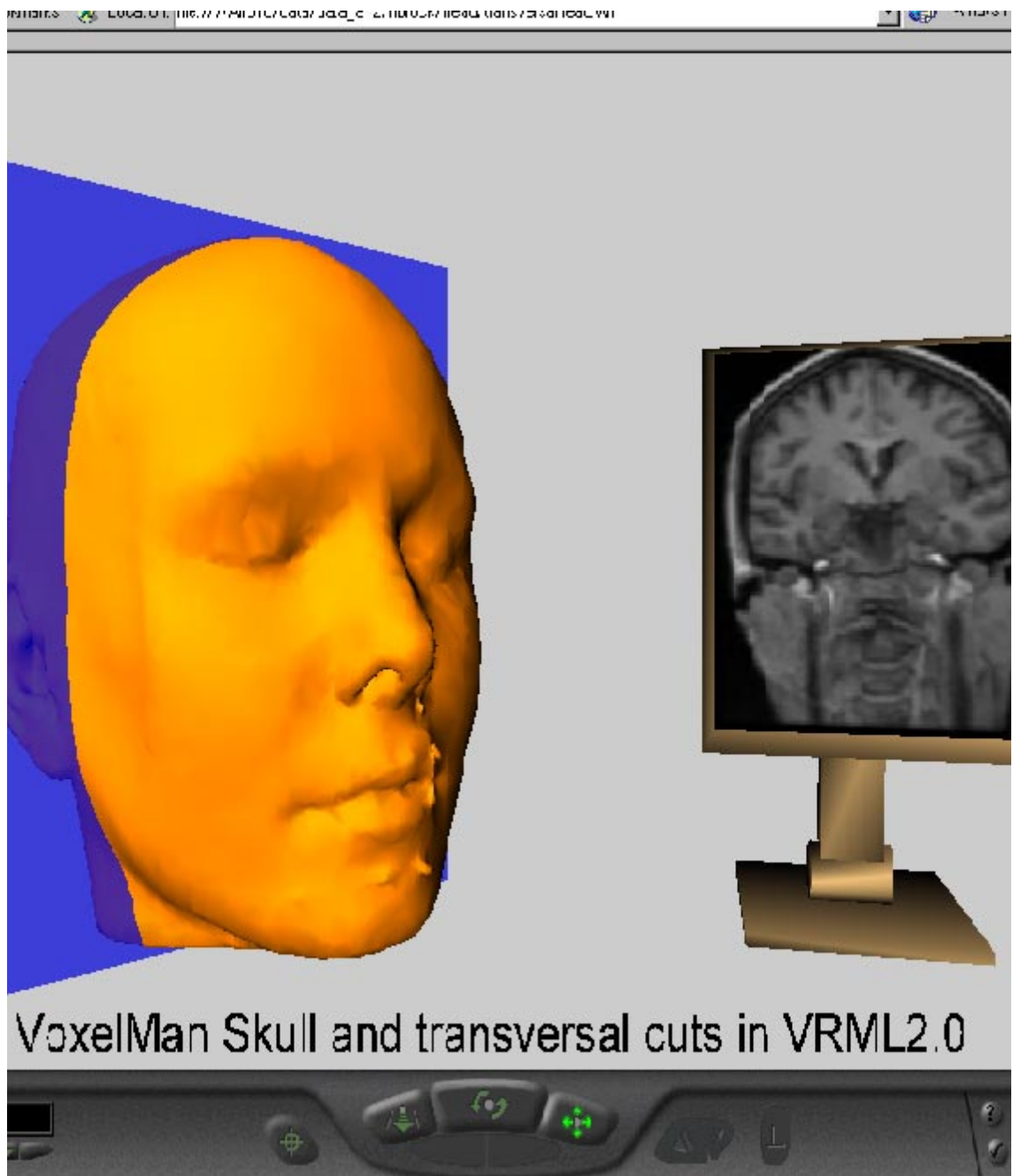


Abb. 45: Transversale Schnitte des Kopfes

Die Abbildung 45 zeigt den Kopf als VRML-Oberflächenmodell, der durch eine blaue transversale Ebene geschnitten wird. Auf dem Bildschirm rechts, neben dem Kopf, wird die entsprechende Schnittebene als Schwarz/Weiß-Bild im JPG-Format angezeigt. Wird der Mauszeiger auf den Kopf geführt verändert er sein Symbol und deutet damit die

Interaktionsmöglichkeit an. Durch einen Mausklick verschiebt sich die Ebene in vordefinierten Schritten und zeigt dabei auf dem Bildschirm die jeweiligen dazugehörigen Schnittbilder an. Die Darstellung und Interaktion des Kopfes sowie die Anzeige der Schnittbilder erfolgt analog auch für die horizontalen (siehe Abbildung 46) bzw. vertikalen (siehe Abbildung 47) Schnitte.

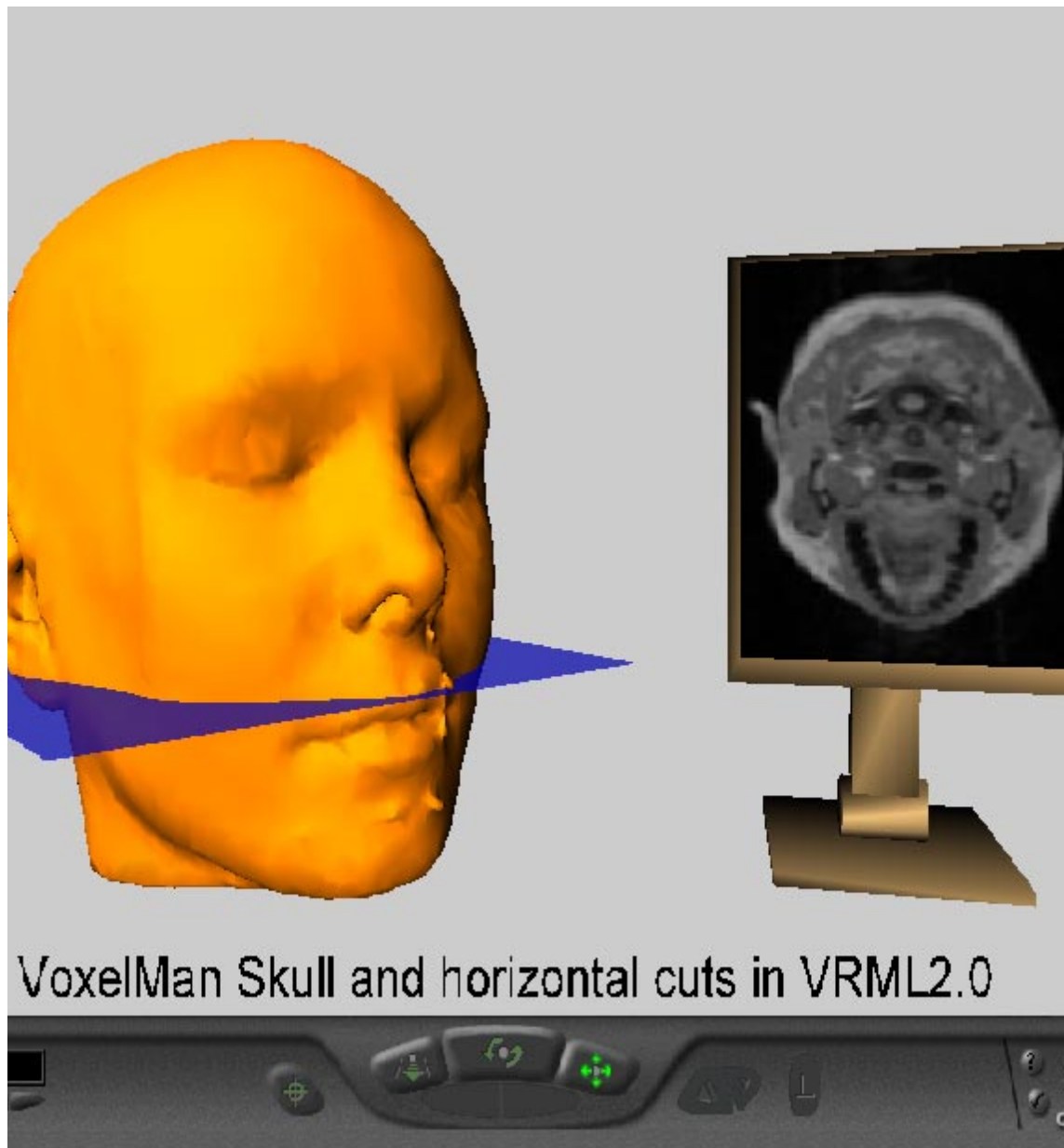


Abb. 46: Horizontale Schnitte des Kopfes

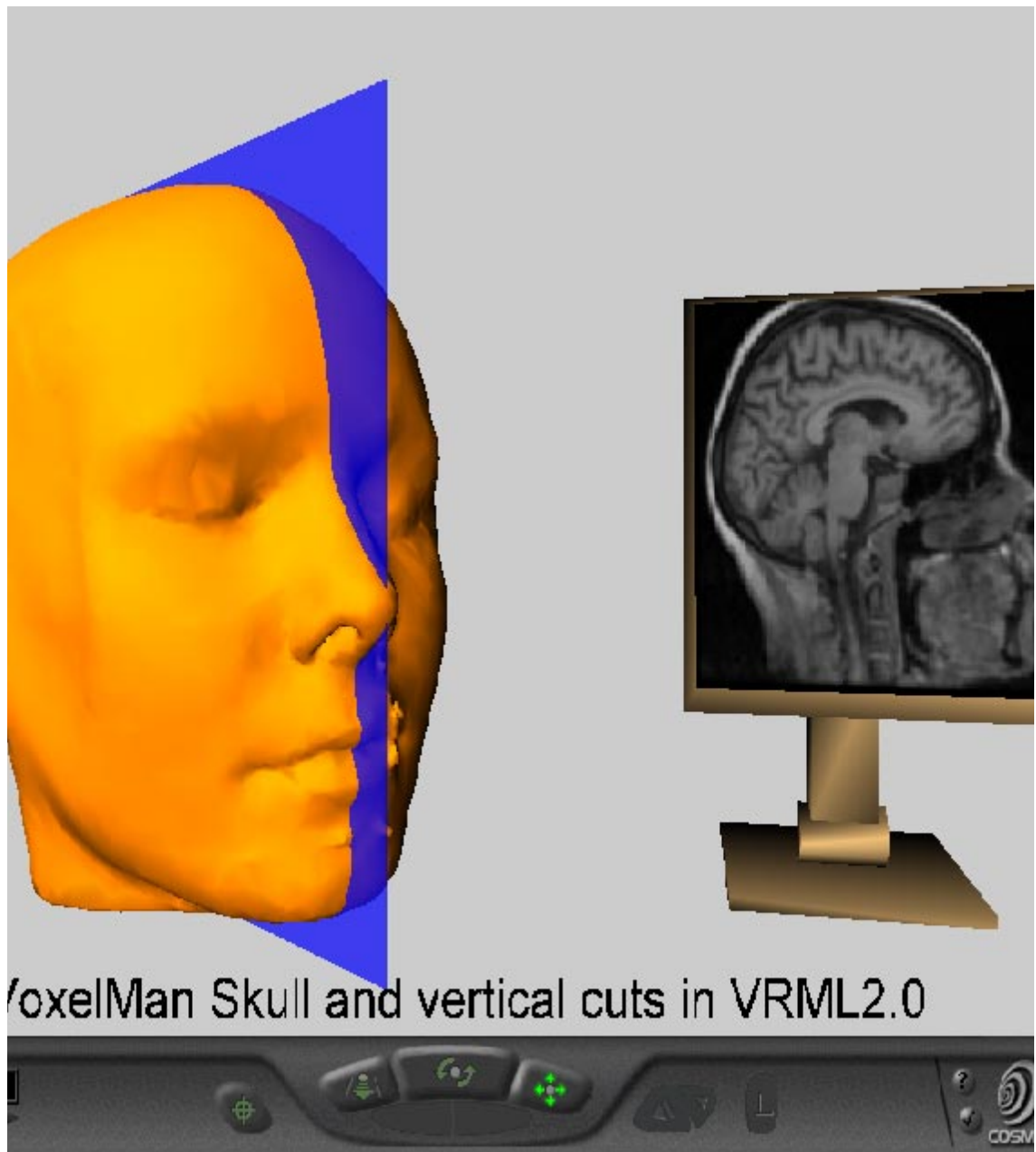


Abb. 47: Vertikale Schnitte des Kopfes

Durch den Einsatz von Texturen in diesem Modell lassen sich die Schnittflächen sehr realistisch darstellen. Gleichzeitig wird durch die Verwendung von Texturen viel Rechenkapazität sowie Übertragungsbandbreite eingespart.

Die Wahl der Schrittweiten, die mit jedem Mausklick vorgenommen wird, ist ebenfalls im VRML-Code fest vorgegeben. Dabei ist es durchaus vorstellbar, die Schritte so fein zu wählen

sowie die Interaktionsmöglichkeit so zu automatisieren, daß auf dem Bildschirm ein Film abläuft und die Bewegung der Ebene fließend wirkt. In diesem Modell war dies jedoch nicht beabsichtigt, sondern es sollte eine mögliche Benutzerinteraktion veranschaulicht werden.

Diese Interaktionsdynamik ist dabei rein statisch, d.h. die Schrittfolge sowie die Zuordnung der zweidimensionalen Graphiken ist fest vorgegeben im Programmcode. Die Möglichkeit, zur Laufzeit und somit dynamisch Bilder oder Programmteile nachzuladen, ist in der VRML-Spezifikation nicht enthalten.

Die statische Struktur des VRML97-Standards ist als großer Nachteil zu bewerten, denn gerade bei komplexen Darstellungen, die interaktiv beeinflussbar sein sollen, ist ein dynamisches Nachladen zur Laufzeit absolut notwendig, damit das Anzeigen der Darstellungen sowie Interaktionen innerhalb einer Szene handhabbar werden.

Alle Dateien haben zu der Browserfunktionalität auch die Möglichkeit, die Kamerasichten durch vordefinierte Viewpoints zu verändern. Ganz links in der Browsersteuerleiste kann im Viewpointauswahlfenster jeweils eine Kamerasicht ausgewählt werden. Die Kamerasichten der Kopfdarstellungen sind in allen Dateien des zweiten Modells identisch definiert.

Eine Kamera kann im dreidimensionalen Raum beliebig positioniert und ihre Linse in eine bestimmte Richtung ausgerichtet werden. In diesem Modell kann der Kopf durch die Auswahl der Viewpoints von allen Seiten betrachtet werden. Mit der Viewpointauswahl "Start" wird die Ausgangsstellung wiederhergestellt, ohne daß die Datei erneut geladen werden muß.

Fazit:

Medizinische Objekte, die aus Untersuchungsdaten gewonnen werden um als Polygonmodell dargestellt zu werden, sind in dem Umfang ihrer Definition äußerst komplex. Die in diesem Modell dargestellte Hautoberfläche ist bereits in ihrer Komplexität durch das Programm von Frank Wilmer [Wilmer 93] verringert worden. Trotzdem enthält die Hautoberfläche des Kopfes als ASCII-basierendes VRML-Oberflächenmodell immer noch 8138 Punkte und 16268 Flächen. Eine entsprechende VRML-Datei, die die Hautoberfläche darstellt, umfaßt 972 KB.

Die Komplexität der Daten ist das Hauptproblem für die Darstellung in VRML. Der Ladevorgang einer VRML-Datei wird dadurch, je nach Rechnergeschwindigkeit, verlangsamt, ebenso die Ausführung der Navigationsmöglichkeiten. Damit die Komplexität der Dateien möglichst gering gehalten wird, sind die Schnitte in dem erläuterten Modell in mehrere Dateien aufgespalten

worden. Die Idee, zweidimensionale Graphiken als Textur auf einem separaten Monitor darzustellen, ist auch aus der Komplexitätsproblematik entstanden. Zuerst sollten die Graphiken auf das Objekt selbst gemappt werden, das als jeweils geschnittenes Objekt dargestellt werden sollte. Der statische VRML97 Standard läßt allerdings ein dynamisches Nachladen der geschnittenen Hautoberflächen Objekte nicht ohne externe Programmierung zu. Das statische Verhalten von VRML97 erfordert jedoch die fest definierte Angabe der jeweiligen Schnittobjekte. Der Umfang dieser statischen Datei setzt sich dann aus der Anzahl der Schnitte, multipliziert mit dem Umfang der Hautoberfläche (972 KB) zusammen. Bei z.B. zehn Schnitten würde diese statische Datei ca. 10 MB betragen. Mit viel Geduld wäre der VRML-Browser in der Lage, dieses Objekt darzustellen, die Navigierbarkeit wird aber unzumutbar langsam. Der Umfang für die Darstellung des Monitors mit den dazugehörigen Graphiken als Texturen macht im Vergleich dazu nur ca. 10 KB aus, so daß eine Diskussion über die Wahl der Möglichkeiten alleine durch die Komplexität der Daten schon entschieden ist.

5.4. Modell 3: Interaktion und Animation komplexer Volumendaten

Inwieweit komplexe Modelle in VRML handhabbar sind und ob diese Modelle noch navigiert bzw. animiert werden können, wird in diesem Modell untersucht.

Das folgende Modell 3 hat drei vordefinierte Animationen eingebaut, die ausgelöst werden können, wenn einer der drei Buttons mit der Aufschrift "animate eyes, animate skin und animate brain" angeklickt wird. Das animierte Objekt dreht sich einmal komplett um die Y-Achse und bleibt dann an seiner korrekten Position wieder stehen.

Die Interaktionen können durch die drei Buttons "Eyes, Skin und Brain" ausgelöst werden. Dabei wird das jeweilige Objekt unsichtbar. Durch erneutes Klicken auf einen der drei Buttons "Eyes, Skin und Brain" wird das Objekt wieder dargestellt.

Beide Arten von Buttons sind unabhängig voneinander definiert, d.h. Animationen sowie Interaktionen können unabhängig voneinander ausgeführt werden. Wird z.B. die Hautoberfläche weggeklickt wie in der Abbildung 49 dargestellt, und anschließend der Button "animate eyes" angeklickt, so rotieren die Augen um die Y-Achse und bleiben nach exakt einer Umdrehung an ihrer anatomisch korrekten Position stehen, d.h. die Sehnerven verschwinden teilweise wieder ins Innere des Gehirns.

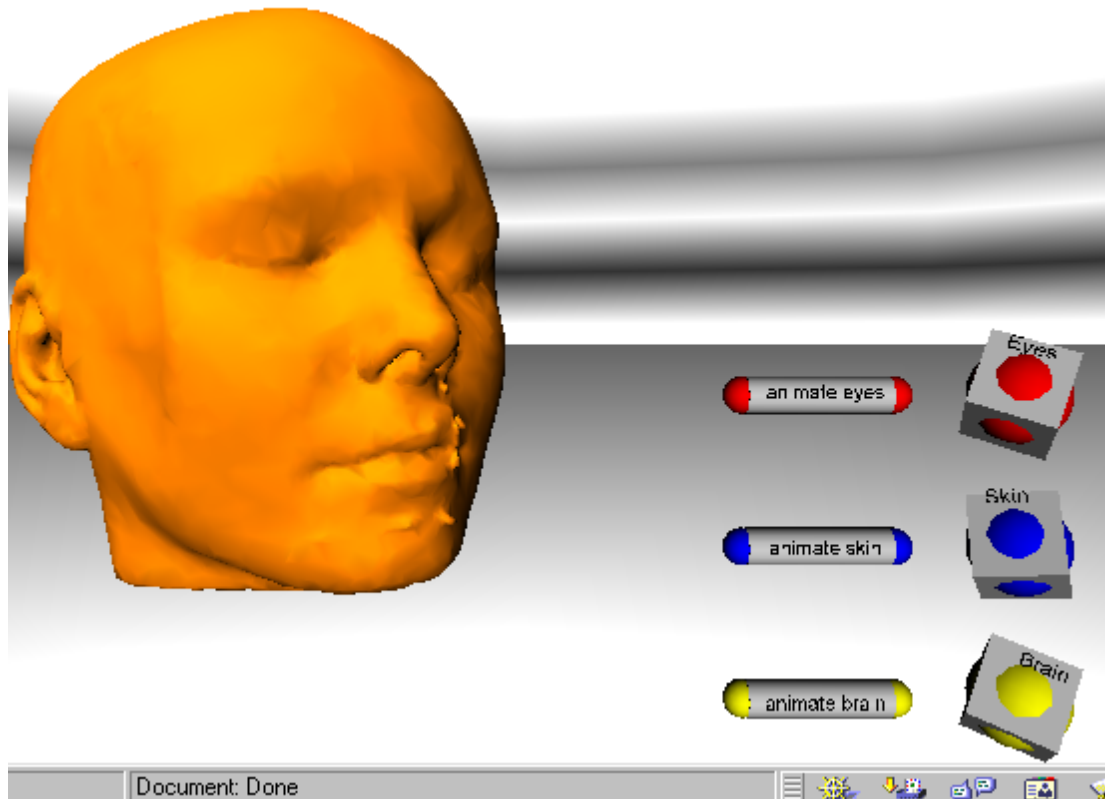


Abb. 48: Interaktion und Animation komplexer Oberflächenmodelle

Die Abbildung 48 zeigt die Datei "animator.wrl". Zu sehen ist auf dem ersten Blick der Kopf als Hautoberflächenmodell sowie sechs unterschiedliche Buttons. Durch modulares Aufteilen der Gesamtstruktur des Modells wird die Hautoberfläche separat in das Browserfenster geladen, ebenso das Gehirn sowie beide Augen einschließlich der Sehnerven. Da diese Objekte relativ komplex sind, dauert das Laden je nach Rechnergeschwindigkeit einige Zeit. Damit der VRML-Code noch mehr optimiert werden kann, werden die einzelnen Dateien mit Hilfe von GZIP komprimiert an den Browser übergeben. Das Plug-In erkennt selbstständig, daß es sich um komprimierte Dateien handelt und entpackt diese vor dem Anzeigen automatisch. In der Statuszeile des Browsers wird der Nachladevorgang prozentual zur Ladezeit angegeben. Für das Entpacken der Dateien gibt es leider keinen Hinweis in der Statuszeile; während des Entpackens ist dort "Document done" zu lesen. Die Dateien sind aber erst vollständig geladen, wenn sich der Mauszeiger, der auf einen der sechs Buttons zeigt, verändert und somit die Interaktion bzw. Animation andeutet.

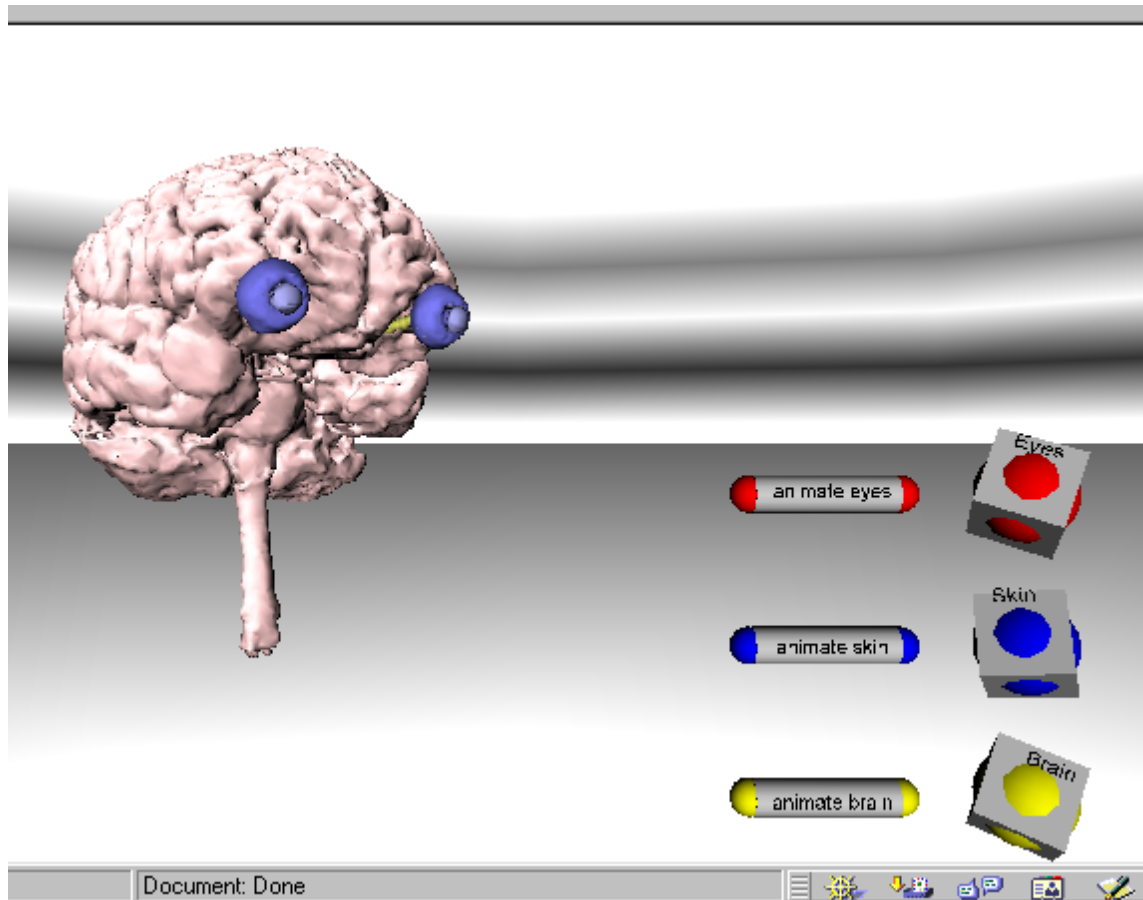


Abb. 49: Interaktion und Animation des Gehirns sowie der Augen

Die komplexe Darstellung des Gehirnes umfaßt in ihrer Definition 65624 Punkte und 132376 Flächen. Die Datei an sich ist 8,3 MB (gezippt ca. 1,16 MB) groß. Die Darstellung der Augen setzt sich aus der Definition des linken sowie rechten Augapfels, der linken und rechten Linse und des linken und rechten Sehnerves zusammen. Diese Oberflächenmodelle sind in eine Datei zusammengefaßt, die insgesamt 757 KB (gezippt ca. 100 KB) umfaßt. Zusammen mit der bereits beschriebenen Hautoberfläche, die als Datei 952 KB (gezippt ca. 140 KB) umfaßt, werden ca. 10 MB Daten in das Browserfenster geladen. Eine derartige Komplexität macht die Navigationsmöglichkeiten, die der Browser zur Verfügung stellt, extrem träge und nicht mehr handhabbar, so daß das Steuer Menü des Browsers im Programmcode ausgeschaltet wurde. Animations- bzw. Interaktionssequenzen sind über die sechs Buttons direkt im VRML-Code definiert und werden auch unmittelbar ausgeführt, sofern die Dateien vollständig geladen wurden.

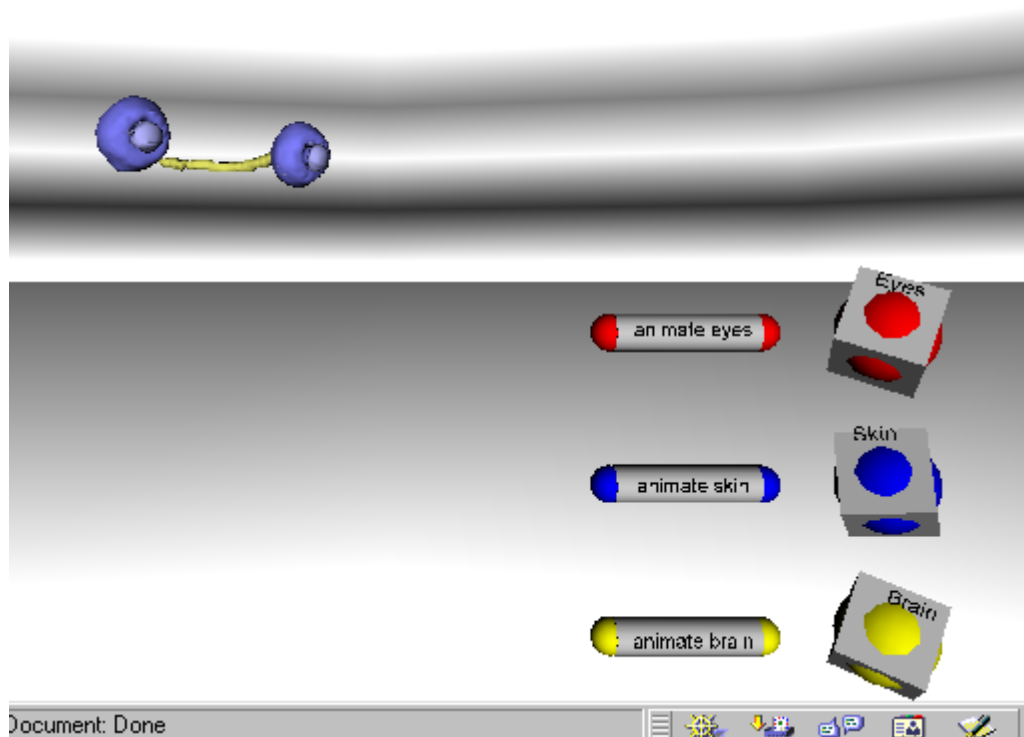


Abb. 50: Interaktion und Animation der Augen

Gleichzeitige Animationen und Interaktion eines Objektes sind ebenfalls möglich. Wird wie in diesem Beispiel (siehe Abbildung 50) die Hautoberfläche weggeklickt und anschließend der Button “animate skin” angeklickt, so wird eine Umdrehung der zur Zeit nicht sichtbaren Hautoberfläche ausgeführt. Dies ist zu sehen, wenn unmittelbar danach wieder der Button “Skin” angeklickt wird und so der Rest der Umdrehung sichtbar gemacht wird.

Fazit:

Interaktionen bzw. Animationen komplexer Darstellungen in VRML sind möglich und werden unmittelbar vom Rechner ausgeführt, sofern sie direkt im VRML-Code definiert werden. Das Navigieren einer gesamten Szene durch das Browsersteuermenü wird je nach Komplexität der Szene mit erheblichen Verzögerungen ausgeführt. Auf diese Möglichkeit sollte bei komplexen Modellen verzichtet werden. Durch Definitionen von Kamerasichten (Viewpoints) im VRML-Code können jedoch auch komplexe Szenen interaktiv gesteuert werden.

6. Schlußfolgerung und Ausblick

In dieser Arbeit wurde eine Methode entwickelt, implementiert und angewendet, mit der aus medizinischen Volumendaten VRML-Oberflächenmodelle erstellt werden können. Des weiteren wurden folgende Fragestellungen untersucht:

- Welche Interaktions- und Animationsmöglichkeiten bietet VRML ?
- Welche Probleme erzeugt die Komplexität der erstellten Polygondaten für VRML ?
- Wie weit lassen sich die Polygondaten ohne Verlust der realistischen Darstellungskraft reduzieren ?
- Inwieweit sind komplexe Oberflächenmodelle in VRML navigierbar ?
- Ist es sinnvoll, Texturen einzusetzen und kann durch Texturen ein realistischer Eindruck vermittelt werden ?

Wie das Programm VOXEL-MAN, auf das diese Arbeit aufbaut, wurde das Konvertierungsprogramm in der Programmiersprache C implementiert. Während die Polygonkoordinaten sowie die in den VOXEL-MAN-Atlanten zugewiesenen Farben im Konvertierungsprozeß unverändert übernommen wurden, stellte sich bei der Entwicklung heraus, daß die Positionierung der Objekte über zwei unterschiedliche Wege zu erreichen war: zum einen die Normierung der Polygonkoordinaten um den Koordinatenursprung, der sich im Mittelpunkt des Bildschirms befindet; zum anderen die Möglichkeit, eine Translation der Polygonkoordinaten in den Koordinatenmittelpunkt vorzunehmen. Die zuerst genannte Möglichkeit hat den entscheidenden Nachteil, daß die Normierung der Polygondaten mit Hilfe eines für jedes Objekt errechneten Mittelwertes bestimmt wird, so daß mehrere zusammengefügte Objekte nicht mehr anatomisch korrekt zueinander liegen, sondern übereinander im Koordinatenursprung. Wird ein allgemein gültiger Mittelwert verwendet, hat das den Nachteil, daß kleine Objekte, wie z.B. die Linse des Auges, sehr klein und weit im Hintergrund dargestellt werden, während große Objekte, wie z.B. die Kopfhaut, sehr groß und nah erscheinen.

In dieser Arbeit wurde deshalb die zweite Möglichkeit realisiert, wobei die generierten Polygonkoordinaten übernommen wurden und die Position der Objekte durch eine Translation in ein lokales Koordinatensystem festgelegt wurde. Dies hat den entscheidenden Vorteil, daß beim Zusammenfügen von mehreren Objekten nur für ein Objekt eine Translation vorgenommen werden muß. Die weiteren Objekte gruppieren sich anatomisch korrekt durch ihre exakten Koordinaten zu diesem Objekt.

Im Umgang mit VRML zeigte sich, daß dieses Beschreibungsformat ausgezeichnete Interaktions- und Animationsmöglichkeiten bietet. Besonders hervorzuheben ist, daß gezielt Interaktionen und Animationssequenzen im VRML-Code definiert werden können. Dies bietet die Möglichkeit, einzelne medizinische Objekte (z.B. ein einzelnes Auge) statt einer komplexen Szene über die Browserfunktionalität zu beeinflussen.

Zu komplexe Oberflächenmodelle sind in VRML nicht ohne Verzögerungszeiten navigierbar. Es stellte sich heraus, daß das Öffnen komplexer VRML-Dateien im Browser erhebliche Ladezeiten in Anspruch nahm.

Die Komplexität von Polygondaten innerhalb einer in VRML beschriebenen Szene stellte eines der grundlegenden Probleme dieser Arbeit dar, deren Lösung auf zwei konzeptionell unterschiedlichen Wegen erprobt wurde. Zum einen eine erfolgreiche Polygonreduzierung vor der Konvertierung nach VRML durch das Programm von Frank Wilmer [Wilmer 93]. Zum anderen erfolgte eine Reduzierung der Daten innerhalb des VRML-Codes. Dies ist z.B. durch eine Aufteilung der VRML-Modelle in modulare Strukturen möglich, die einzeln nachgeladen werden. Ebenso kann eine Reduzierung der Datei durch Zerlegung in mehrere Dateien und deren Verbindung durch Hyperlinks erreicht werden. Der Einsatz von Texturen als Ersatz komplexer Polygondaten trägt ebenfalls zur erheblichen Reduzierung der Datenmenge bei. Des weiteren können dem VRML-Browser komprimierte Dateien mit Hilfe von GZIP übergeben werden. Durch Instanziierung wird nicht nur der VRML-Code übersichtlicher, sondern auch erheblich die Datenmenge reduziert.

Durch die Reduzierung der Polygondaten durch das "TriDiet" Programm, d.h. die Polygonreduzierung vor dem Konvertierungsprozeß, bestand die Gefahr des Verlustes der realistischen Darstellungskraft der Modelle. Es stellte sich heraus, daß bis zu 45% Reduzierung der Polygone mit nur geringfügiger Verschlechterung des visuellen Eindrucks reduziert werden können. Alles was über 45% Reduzierung der Polygone hinausging führte zu einer erheblichen Reduzierung der realistischen Darstellung.

Bei der Verwendung von Polygondaten ergeben sich sehr schnell komplexe Datenmengen, die die Handhabung dieser Modelle erschweren. Im Laufe dieser Arbeit ergab sich, daß durch die Verwendung von Texturen die Datenmenge noch mehr reduziert werden konnte, ohne daß der realistische Eindruck der Darstellung beeinträchtigt wurde.

Zusammenfassend ist festzustellen, daß die statischen Eigenschaften des aktuellen VRML-Standards VRML97 die Komplexität der VRML-Modelle zur erheblichen Größen führen. Es wäre deshalb sinnvoll, diesen VRML-Standard durch dynamische Eigenschaften bzgl. des Laufzeitverhaltens zu verbessern. Dies ist im jetzigen Standard jedoch noch nicht vorgesehen. An der zukünftigen Entwicklung von VRML arbeiten u.a. das Web-3D-Consortium sowie viele Hersteller, die sich auch bisher an der Entwicklung von VRML und zur Zeit an einer Erweiterung des bestehenden Standards unter dem Namen VRML99 beteiligen. Im Februar 1999 hat unter diesem Namen die vierte Virtual Reality Modeling Language Konferenz in Paderborn stattgefunden, in denen verschiedene Erweiterungen wie die Einbindung von Java in VRML diskutiert wurden. Außerdem gibt es mehrere internationale Workshops, die auch unter dem Namen VRML-NG zu finden sind, wobei "NG" für Next Generation steht, die ähnliche Ziele wie VRML99 [Gerd] [Reddy] verfolgen. Die Entwicklung zu einem neuen Standard ist noch im vollem Gange, so daß zu diesem Zeitpunkt eine Vielzahl von Prognosen im Internet zu finden sind. Eins ist allerdings deutlich zu erkennen: VRML hat mit dem bisherigen Standard ein festes Standbein als dreidimensionales Beschreibungsformat mit Interneteinbindung, das mit einem evtl. neuen Standard nur noch mehr erweitert werden kann. Nach Erweiterung des jetzigen Standards könnte in Zukunft die Darstellung komplexer VRML-Modelle erheblich erleichtert werden. Es könnten darüber hinaus weitaus komplexere Modelle entwickelt werden, deren Details dynamisch nachgeladen werden können. Dies würde weiterhin die Möglichkeit für eine Vielzahl von medizinischen Darstellungen und Simulationen medizinischer Anwendungen bieten.

Literaturverzeichnis

- [Ames 96] A. *Ames*, D. R. *Nadeau*, J. L. *Moreland*, The VRML2.0 Sourcebook, Wiley Computer Publishing USA 1996.
- [Amman 97] E. *Amman*, Programmierung animierter Welten, International Thomsor Publishing GmbH Bonn 1997.
- [Carey 97] R. *Carey*, G. *Bell*, The Annotated VRML2.0 Reference Manual, Addison Wesley Longman Verlag GmbH Bonn, Paris, New York, Sydney 1997.
- [Chaco] VRML-Test Suite
<http://www.chaco.com/community/vrml/test>
- [Couch] VermelGen: A Free VRML2.0 Editor
<http://www.vlc.com.au/VermelGen>
- [Crispen] VRML-Tutorial Links
<http://home.hiwaay.net/~crispen/vrmlworks/tutorials/links.html>
- [Cromwell] VRML 2.0 Moving World Demonstration
<http://demo.westlake.com/vrml2/tutorial.html>
- [Dittmer 94] U. *Dittmer*, Einbettung 'intelligenter Bilder' in die Quick-Time-Architektur Studienarbeit, Fachbereich Informatik, Universität Hamburg, 1994.
- [Dössel 2000] O. *Dössel*, Bildgebende Verfahren in der Medizin: von der Technik zu medizinischen Anwendung, Springer-Verlag Berlin, Heidelberg, 2000.
- [Feinberg] VRML-Models
<http://www.csmc.edu/radiolog%5Fold/virtual/LungTX.wrl>
- [Finke 98] N. *Finke*, O. *Kober*, O. J. *Bott*, A. *Terstappen*, Virtual Connections: Dynamische Generierung virtueller Welten, in iX Press, Heinz Heise Verlag Hannover, Dezember 1998, S. 124-129.
- [Flanagan 98] D. *Flanagan*, JAVA in a Nutshell, Deutsche Ausgabe für Java 1.1, 2. Auflage, O'Reilly Verlag Köln 1998.

- [Foley 94] J. D. *Foley*, A. *van Dam*, S. K. *Feiner*, J. F. *Hughes*, R. L. *Phillips*,
Grundlagen der Computergraphik: Einführung, Konzepte, Methoden,
Addison-Wesley Verlag Bonn, Paris, New York, Sydney 1994.
- [Geometrek] Kubricks Zauberwürfel
<http://play.at/geometrek>
- [Gerd] First Time in Europe: VRML 99
<http://www.c-lab.de/vrml99/>
- [Hansen] Flybrain 3D Models
<http://www.flybrain.org/Flybrain/html/vrml/models/dros4.wrl>
- [Hartman 96] J. *Hartman*, J. *Wernecke*, The VRML Handbook, Addison Wesley Longman
Verlag GmbH Bonn, Paris, New York, Sydney 1996.
- [Höhne 92] K.H. *Höhne*, W.A. *Hanson*, Interactive 3D-segmentation of MRI and CT
volumes using morphological operations, J. Comput. Assis.Tomogr. 16, 2, S
285-294, 1992.
- [Höhne 95] K.H. *Höhne*, B. *Pflessner*, A. *Pommert*, M. *Riemer*, T. *Schiemann*,
R. *Schubert*, U.*Tiede*, A new Representaion of Knowledge Concerning Humar
Anatomy and Function, in Nature Med., Vol. 1, No. 6, S.506-511, 1995.
- [Höhne: CD-
Rom 95] K.H. *Höhne*, (ed.): VOXEL-MAN, Part 1: Brain and Skull, Version 1.0
Springer-Verlag Electronic Media, Heidelberg, (CD-ROM, ISBN 3-540-14517-
6), 1995.
- [Höhne 96] K.H. *Höhne*, B. *Pflessner*, A. *Pommert*, M. *Riemer*, T. *Schiemann*,
R. *Schubert*, U.*Tiede*, A Virtual Body Model for Surgical Education anc
Rehearsal, IEEE Computer 29, 1, S. 25-31, 1996.
- [Höhne:
RSNA 96] K.H. *Höhne*, K. *Priesmeyer*, M. *Riemer*, T. *Schiemann*, R. *Schubert*, U.*Tiede*
A. *Pommert*, H.-C. *Wulf*, Exploring the Visible Human via "intelligent movies"
Radiology 201, P (1996), 564. (abstract). Exhibit at RSNA '96, award 'cur
laude'.

- [Höhne: Vol. K.H. *Höhne*, A. *Pommert*, Volume Visualization. In Toga, A. W., Mazziotta, J
Vis. 96] C. (ed.): Brain Mapping, Academic Press, San Diego, CA, ch. 17, S. 423-443
1996.
- [Höhne: CD- K.H. *Höhne*, (ed.): VOXEL-MAN Junior: Interactive 3D Anatomy and
ROM 98] Radiology in Virtual Reality Scenes, Part 1: Brain and Skull. Springer-Verlag
Electronic Media, Heidelberg, 1998.
- [Höhne: CD- K.H. *Höhne*, (ed.): VOXEL-MAN Junior, Part 2: Inner Organs. 3D Navigator
ROM 2000] through Topographie and Radiological Anatomy. Springer-Verlag Electronic
Media, Heidelberg, 2000.
- [IMDM] Systems: VOXEL-MAN
[http://www.uke.uni-hamburg.de/institute/imdm/idv/forschung/vm/
index.en.html](http://www.uke.uni-hamburg.de/institute/imdm/idv/forschung/vm/index.en.html)
- [John] Web-Based Surgical Simulators and Medical Education Tools
<http://synaptic.mvc.mcc.ac.uk/simulators.html>
- [Kass] VRML 2.0 Testing Resources
<http://www.itl.nist.gov/div897/ctg/vrml/vrml2test.html>
- [Keil] Virtual Reality Modeling Language in Chemistry
<http://www.pc.chemie.tu-darmstadt.de/vrml/p53dna/full/p53dna.wrl>
- [Kikuro] Shockwave and VRML A-GO-GO
<http://plaza.across.or.jp/~kikuro/robo2.wrl>
- [Kloss 98] J. H. *Kloss*, R. *Rockwell*, K. *Szabó*, M. *Duchrow*, VRML97: Der neue Standard
für interaktive 3D-Welten im World Wide Web, Addison Wesley Longman
Verlag GmbH Bonn, Paris, New York, Sydney 1998.
- [Knowsley] Virtual Library
<http://www.knowsley.gov.uk/vrml/huyton/library.wrl>
- [Kriete] VRML 2.0 Kopf- und Thorax Präsentation
<http://www.uni-giessen.de/ipl/comp.vrml>
<http://www.uni-giessen.de/ipl/tryit.wrl>

- [Lea 97] R. *Lea*, K. *Miyashita*, JAVA for 3D and VRML Worlds, New Riders Publishing Indianapolis, USA 1997.
- [Lorensen 87] W. E. *Lorensen*, H. E. *Cline*, Marching Cubes: A High Resolution 3D Surface Construction Algorithm, in IEEE Computer Graphics and Applications, IEEE Computer Society, Los Alamitos July 1987.
- [Matsuba 96] S. N. *Matsuba*, B. *Roehl*, VRML das Kompendium: Einführung, Arbeitsbuch Nachschlagewerk, Markt&Technik Buch- und Software Verlag Haar bei München 1996.
- [McGill] 3D Anatomy for Students
<http://external.csmc.edu/radiolog/Virtual/default.html>
- [Mercedes] VRML-Gittermodell
http://www.mbi.mercedes-benz.com/e/Products/pkw/c_klasse/auto.wrl
- [Mitra] VRML 2.0 Tests
<http://earth.path.net/vrml2tests>
- [Nadeau 99] D. R. *Nadeau*, Building Virtual Worlds with VRML, in: IEEE Computer Graphics and Applications, IEEE Computer Society, Los Alamitos March/April 1999.
- [National Library of Medicine] Graphic Models
http://www.mayo.edu/bir/Models/Visible_Human_Male/VRML/11.vrml
- [Ozdemir] VRML-Tutorials
<http://www.npac.syr.edu/projects/tutorials/VRML/tutorials.html>
- [Pommert 94] A. *Pommert*, R. *Schubert*, M. *Riemer*, T. *Schiemann*, U. *Tiede*, K.H. *Höhne* Symbolic modeling of human anatomy for visualization and simulation, In Robb, R. A. (ed.): Visualization in Biomedical Computing 1994, Proc. SPIE 2359. Rochester, MN, S. 412-423, 1994.

- [Pommert 95] A. *Pommert*, M. *Riemer*, T. *Schiemann*, R. *Schubert*, U. *Tiede*, K.H. *Höhne*, Three-Dimensional Imaging in Medicine: Methods and Applikations in Computer Interated Surgery: Technology and Clinical Applikations, MIT Press Cambridge, MA 1995, S. 155-174.
- [Reddy] VRML-NG Proposal Requirement: Double-Precision
<http://www.ai.sri.com/~reddy/geovrml/double/>
- [Roehl] Bernie's VRML-Links
<http://ece.uwaterloo.ca:80/~broehl/vrml/>
- [Roehl 97] B. *Roehl*, J. *Couch*, C. *Reed-Ballreich*, T. *Rohaly*, G. *Brown*, Light Night VRML2.0 with Java, Ziff-Davis Press Emeryville California 1997.
- [Salgado] VRML-Gallery of Electromagnetism
<http://suhep.phy.syr.edu/courses/vrml/electromagnetism/dipolerl.wrl.gz>
- [Schubert 99] R. *Schubert*, B. *Pflessner*, A. *Pommert*, K. *Priesmeyer*, M. *Riemer*, T. *Schiemann*, U. *Tiede*, P. *Steiner*, K.H. *Höhne*, Interactive volume visualization using "intelligent movies". In Westwoos, J.D. et al. (eds.): Medicine meets Virtua Reality, Proc. MMVR '99, Health Technology and Informatics 62, IOS Press Amsterdam S. 321-327, 1999.
- [Seidman] VRML-Browsers Overview
<http://www.construct.net/tools/vrml/browsers.html>
- [Schroeder 98] W. *Schroeder*, K. *Martin*, B. *Lorensen*, The Visualization Toolkit 2nd Edition, Prentice Hall PTR, Upper Saddle River, New Jersey 1998.
- [SGI] Cyber Anatomy 101
<http://reality.sgi.com/sambo/Oobe/Cyberanatomy>
- [Stampe 93] D. *Stampe*, B. *Roehl*, J. *Eagon*, Virtual Reality Creations, Waite Group Press, Corte Madera, California 1993.
- [Tiede 98] U. *Tiede*, T. *Schiemann*, K.H. *Höhne*, High quality rendering of attributec volume data. In Ebert, D. et al. (ed.): Proc. IEEE Visualisation '98. IEEE Computer Society Press, Los Alamitos, CA, S. 255-262, 1998.

- [Turau 99] V. **Turau**, Techniken zur Realisierung Web-basierter Anwendungen, in: Informatik-Spektrum No.22, Springer-Verlag Berlin, Heidelberg, Februar 1999.
- [Visual Decision] Portfolio
<http://www.vdi.com/products/gallery/portofoli.htm>
- [VRML Browser] VRML2.0 Browser:
<http://www.sgi.com/>
<http://www.intervista.com/>
<http://sonypic.com/>
<http://www.iicm.edu/vrwave>
- [VRML Org.] VRML-Specifications
<http://www.vrml.org/Specifications/VRML97/>
- [Web Consortium] 3D VRML Architecture Group (VAG): <http://www.vrml.org/vag>
 VRML Consortium: <http://www.vrml.org>
 The VRML Repository: <http://www.web3d.org/vrml/vrml.htm>
- [Wilhelms 90] J. **Wilhelms**, A. **van Gelder**, Topological considerations in isosurface generation in: IEEE Computer Graphics and Applications, IEEE Computer Society, Los Alamitos Vol 24 No 5, S. 79-86, 1990.
- [Wilmer 93] F. **Wilmer**, Reduktion der Oberflächenbeschreibung triangulierter Oberflächen durch Anpassung an die Objektform, Diplomarbeit im Institut für Mathematik und Datenverarbeitung in der Medizin in Hamburg, 1993.
- [Zeynep] NPAC - Visible Human Visualisation in VRML 2.0
<http://www.npac.syr.edu/projects/3Dvisiblehuman/VRML/VRML2.0/MEDVIS/>

Abbildungsverzeichnis

- Abb. 1 Navigationsfunktionalität des Cosmo Player Browsers
- Abb. 2 Darstellung eines möglichen Szenegraphen
- Abb. 3 Beispiele zur Syntax der Knoten
- Abb. 4 Darstellung grundlegender Datentypen
- Abb. 5 Der tanzende Roboter: ein Beispiel für Animationen [Kikuro]
- Abb. 6 Das Gehirn einer Fruchtfliege [Hansen]
- Abb. 7 Interaktive Darstellung Visible Human Schnittbilder [Zeynep]
- Abb. 8 Gehirndarstellung aus der Cyberanatomie [SGI]
- Abb. 9 Grobe VRML-Darstellung des Thorax und des Kopfes [Kriete]
- Abb. 10 Interaktive Darstellung einer Katheterisierung eines Ventrikels [John]
- Abb. 11 Integration von VRML-Modellen zwischen Visible Human Schnitten [McGill]
- Abb. 12 Darstellung eines Bronchus in VRML [Feinberg]
- Abb. 13 Darstellung eines Lendenwirbels in VRML [National Library of Medicine]
- Abb. 14 Erzeugung von VRML-Modellen aus intelligenten Volumen im Überblick
- Abb. 15 Darstellung des intelligenten Volumenmodells des Programms VOXEL-MAN [Höhne 95]
- Abb. 16 Die 15 Fälle des Marching Cubes Algorithmus [Lorenzen 87]
- Abb. 17 Codebeispiel einer VRML-Datei
- Abb. 18 Codebeispiel einer Drahtgitterdarstellung in VRML
- Abb. 19 VRML-Codebeispiel für die Farbgebung und Beleuchtung
- Abb. 20 VRML-Codebeispiel für die Verwendung von Texturen
- Abb. 21 Darstellung der erstellten Bedienoberfläche
- Abb. 22 Darstellung des Untermenüs "Marching Cubes" der Bedienoberfläche
- Abb. 23 Darstellung des Untermenüs "TriDiet" der Bedienoberfläche
- Abb. 24 Darstellung des Untermenüs "Convert to VRML" der Bedienoberfläche
- Abb. 25 Darstellung des Untermenüs "Merge VRML Files" der Bedienoberfläche
- Abb. 26 VRML-Codebeispiel für Text und Hyperlinks
- Abb. 27 VRML-Codebeispiel für die Einbindung eines Inline Knotens
- Abb. 28 Zweidimensionale Graphiken als Hintergrunddarstellung
- Abb. 29 Skizzierung der Definition von Himmel und Boden
- Abb. 30 Farbliche Darstellung von Himmel und Boden.

- Abb. 31 Kombination von Sensoren durch einen Umschaltbutton am Beispiel Cornea
- Abb. 32 VRML-Codebeispiel für einen PlaneSensor
- Abb. 33 VRML-Codebeispiel für einen Script Knoten und den dazugehörigen Route-Befehlen
- Abb. 34 VRML-Codebeispiel eines ColorInterpolator
- Abb. 35 VRML-Codebeispiel eines OrientationInterpolator
- Abb. 36 Viewpoint: "Head Left" des zweiten Modells
- Abb. 37 Viewpoint Definitionen im VRML-Code
- Abb. 38 Drahtgittermodell der Weichteile des Kopfes
- Abb. 39 Drahtgittermodell der äußeren Oberfläche der Kopfhaut
- Abb. 40 Darstellung des aus den Originaldaten konvertierten VRML-Modells des Gehirns (15,2 MB)
- Abb. 41 Darstellung des aus den reduzierten Daten konvertierten VRML-Modells des Gehirns (8,3 MB)
- Abb. 42 Modell 1: Text und Hyperlinks
- Abb. 43 Darstellung ausgewählter Halsarterien als Sprungzieldatei eines Hyperlinks
- Abb. 44 Modell 2: Auswahl interaktive Darstellungen von Schnittflächen des Kopfes
- Abb. 45 Transversale Schnitte des Kopfes
- Abb. 46 Horizontale Schnitte des Kopfes
- Abb. 47 Vertikale Schnitte des Kopfes
- Abb. 48 Interaktion und Animation komplexer Oberflächenmodelle
- Abb. 49 Interaktion und Animation des Gehirns sowie der Augen
- Abb. 50 Interaktion und Animation der Augen

Glossar

ASCII

American Standard Code for Information Interchange (Standardzeichensatz)

Browser

(engl. browse = schmökern) Ein Programm, das das überblickartige Durchblättern von Dateien, Programmen und Systemkomponenten am Bildschirm ermöglicht. Im Unterschied zum Editor können die angesehenen Objekte meist nicht verändert werden.

Grundtypen

Zu Grundtypen zählen die grafischen Primitiven Würfel und Quader (Knoten Box), Kegel (Knoten Cone), Zylinder (Knoten Cylinder) und die Kugel (Knoten Sphere).

HSV

Das HSV (hue, saturation, value) Farbmodell benutzt ein zylindrisches Koordinatensystem. Die Teilmenge des Raumes, in der das Modell definiert ist, ist eine sechsseitige Pyramide. Die Spitze der Pyramide entspricht der Helligkeit $v=0$, während die entgegen gesetzte Seite $v=1$ die relativ kräftigen Farben enthält. Der Farbton h wird durch den Winkel um die vertikale Achse festgelegt. Die Komplementärfarben liegen in der HSV Pyramide um 180 Grad versetzt, einander gegenüber. Die Sättigung s ist eine Zahl zwischen Null auf der Mittellinie der v -Achse und Eins auf den rechteckigen Seiten der Pyramide. Die Sättigung wird relativ zur Farbpalette gemessen, die das Modell darstellt.

HTML

Hypertext Markup Language

ISO

International Organisation for Standardization

Java Applet

Java-Programm, das in Web-Dokumente eingebunden werden kann. Der Code von Applets wird über das Netz übertragen und von einem Interpreter ausgeführt.

KB (MB)

Abkürzung für KiloByte (MegaByte)

Plug-In

Erweiterung, die für bestimmte HTML-Browser realisiert sind. Während der HTML-Browser (z.B. Netscape, Internet-Explorer u.a.) die gesamte Web-Kommunikation übernimmt, wird das Plug-In lediglich als Hilfsanwendung, z.B. zur Darstellung einer dreidimensionalen Szene, gestartet.

Rendering

Das Umsetzen eines dreidimensionalen Objektes auf dem Computerbildschirm.

RGB

RGB steht hierbei für die additiven Leuchtfarben Rot, Grün, Blau, aus denen sich sämtliche natürlich vorkommende Farben mischen lassen.

SDSC

San Diego Supercomputer Center

Skalierung

verändern der Größe eines Objektes

Translation

Bewegung im Raum

utf8

ISO Standard 10646: Kodierungsverfahren UTF-8, das die Darstellung des internationalen Zeichensatzes in VRML sicherstellt.

VAG

VRML Architecture Group

Virtual Reality

Die vom Computer erzeugte Simulation einer dreidimensionalen Umgebung, bei der der Anwender Inhalte der Umgebung sowohl sehen als auch manipulieren kann.

VRML

Virtual Reality Modeling Language

Erklärung

Ich versichere an Eides Statt durch meine eigene Unterschrift, daß ich die vorstehende Arbeit selbständig und ohne fremde Hilfe angefertigt und alle Stellen, die wörtlich oder annähernd wörtlich aus Veröffentlichungen entnommen sind als solche kenntlich gemacht und mich auch keiner anderen als der angegebenen Literatur bedient habe. Diese Versicherung bezieht sich auch auf die in der Arbeit gelieferten Zeichnungen, Skizzen, bildlichen Darstellungen und desgleichen.

Mit der späteren Einsichtnahme in meine schriftliche Diplomarbeit erkläre ich mich einverstanden.