Konzeption eines Datenbanksystemes zur Organisation audiovisueller Szenen und Realisierung eines Prototypen am Beispiel "MPEG-4 VRML-Profile"

Diplomarbeit

Technische Universität Ilmenau,

Fakultät für Elektrotechnik und Informationstechnik

Vorgelegt von:	Volker Neundorf
Matrikelnummer:	25312
Studiengang:	Medientechnologie
Studienrichtung:	Digitale Medien
Nummer der Arbeit:	2181 - 01D - 09
Verantwortlicher Professor:	Prof. Dr. Karlheinz Brandenburg
Betreuender wiss. Mitarbeiter:	Dipl.-Ing. Uwe Kühhirt

Bearbeitungszeitraum: 20.08.2001 – 19.02.2002

Danksagung

An erster Stelle gilt mein Dank den unerschöpflichen Informationsquellen des Internet. Aufgrund der hochaktuellen Problematik dieses Themas und teilweise noch nicht verabschiedeter Standards, hatte ich leider nur sehr wenige gute konventionelle Literaturquellen in Form von papiernen Büchern zur Verfügung. Ohne die Suchmöglichkeiten des Internet wäre es nicht möglich gewesen diese Arbeit zu schreiben.

Weiterhin möchte ich mich bei meinem Betreuer Dipl.-Ing. Uwe Kühhirt für seine Unterstützung und Anregungen bedanken. Dank auch an Dipl.-Ing. Marco Rittermann als meinen inoffiziellen (Vertretungs)Betreuer.

Viel zu wenig Platz und schriftliche Ausdrucksmöglichkeiten stehen mir zur Verfügung um mich bei meiner Familie, vor allem meinen Eltern und meinen Freunden zu bedanken, die mich ständig aufmunterten und vorantrieben.

„Der Weg ist das Ziel."
Homer (Dichter)

1. Einleitung

Die Abkürzung MPEG steht für „Moving Pictures Expert Group“[1]. Erst durch die Standardisierungen dieser Gruppe von Audio-, Video- und Programmierspezialisten stehen uns heute noch kaum erfassbare Möglichkeiten der digitalen Welt offen.
Standards wie MPEG-1, MPEG-2, heute MPEG-4 und künftig MPEG-7 stellen der Industrie als auch Forschungs-, Entwicklungs-, und Bildungseinrichtungen einheitliche Richtlinien und Grundlagen für kompatible Systeme und Anwendungen zur Verfügung.
MPEG-4 ist kein Produkt oder neues Dateiformat, vielmehr liefert dieser Standard Normen und Richtlinien zur Programmierung von Algorithmen und Anwendungen zur Kompression und Übertragung von audiovisuellen Daten.
Sind MPEG-1 und MPEG-2 vor allem Standards zur Datenkompression von A/V-Strömen, so basiert MPEG-4 auf einen objektorientierten Ansatz. Der wohl momentan allgemein bekannteste Standard aus dieser Gruppe ist MPEG-1 Audio Layer-3. Dieser unter dem Namen MP3 verbreitete Standard zur psychoakustischen Audiocodierung ist mittlerweile aufgrund seiner sehr guten Kompressions- und Klangeigenschaften zum Schrecken der kommerziellen Musikindustrie und gleichzeitig zum populärsten Tauschformat der Internetnutzer geworden.

Das MPEG-1 Projekt begann im Mai 1988 und wurde im November 1992 ein offizieller ISO Standard (ISO/IEC 11172). MPEG-1 beschreibt die Speicherung von A/V-Strömen bei einer Datenrate von 1,4 MBit/s zur Wiedergabe von interaktiven Filmen. MPEG-1 besteht aus drei Teilen: Systems, Audio und Video. Der „Systems“-Teil beschreibt hierbei den Decoder, die Synchronisation und die Zusammensetzung von Audio- und Videoströmen. Im „Video“-Teil werden die Algorithmen zur Speicherung von Videodaten anhand von z.B. Bewegungskompensation und interpolierter Prädiktion mittels DCT beschrieben. Der „Audio“-Teil implementiert Audiokompressionsalgorithmen über psychoakustische Modelle.

Das Hauptanliegen hingegen bei MPEG-2 war die Migration von analogen und digitalen TV. Die Qualität des Signals sollte hierbei 6MBit/s für das Composite

[1] Offizielle MPEG Homepage http://www.cselt.it/mpeg

und 9MBit/s für das Komponentensignal betragen. Des weiteren war die Benutzung von multiplen Audiokanälen vorgesehen.
MPEG-2 unterscheidet in den Programmstrom und den Transportstrom. Der Programmstrom ist vergleichbar mit dem MPEG-1 System Multiplex. Es werden ein oder mehrere elementare Ströme über eine gemeinsame Zeitbasis zusammengepackt. Der Transportstrom kombiniert ein oder mehrere elementare Ströme über eine oder mehrere unabhängige Zeitbasen. Transportströme finden vor allem Anwendung in Umgebungen in denen Übertragungsfehler und Datenverluste auftreten können.
Die Hauptnutzung von MPEG-2 liegt bei DVD-Produktionen, Satelliten und Kabelfernsehen sowie den Set Top Boxen. Für Broadcasting oder Streaming über Internet ist MPEG-2 aufgrund seiner hohen Datenrate, selbst für die nach heutigen Verhältnissen verfügbaren Internetanbindungen, nicht geeignet.

Das MPEG-4 Projekt begann im Juli 1993 und wurde in seiner ersten Version im März 1999 ein internationaler Standard (ISO/IEC 14496). Die überarbeitete zweite Version wurde im Januar 2000 veröffentlicht. Der MPEG-4 Standard realisiert die Kodierung von natürlichen und synthetischen audiovisuellen Objekten. MPEG-4 wurde entwickelt, um eine Unterstützung für zukünftige Anwendungen zu geben, wie solche Objekte in einer interaktiven Szene darstellbar sind und wie die A/V-Ströme übertragen und gespeichert werden können. Grundlegende Anforderungen waren u.a. die Skalierbarkeit von Encoder und Decoder, inhaltsbasierter Datenzugriff, differenzierte Möglichkeiten der Benutzerinteraktion sowie auch die Problematik der Rechteverwaltung und des Copyright.

Der hohe Grad an Parametrisierung solcher Szenen und ihrer enthaltenen Objekte prädestiniert MPEG-4 für Anwendungen wie das virtuelle Studio, computergenerierte Moderatoren oder auch interaktive 3D-Anwendungen für niedrige Bandbreiten, wie z.B. VRML, im Internet.

Die Ergebnisse dieser Arbeit sind ein Beitrag zur Verwirklichung des Gesamtprojektes IAVAS[2]. Das Projekt IAVAS entwickelt Verfahren zur MPEG-4 konformen Kodierung audiovisueller Inhalte, entwirft Beispielapplikationen und entsprechende Autorenwerkzeuge um die Möglichkeiten des MPEG-4 Standards zu nutzen. Einen großen Rahmen nimmt in diesem Projekt die Entwicklung von Anwendungen rund um das virtueller Studio der TU Ilmenau ein. Die benutzerabhängige Datenhaltung von Szenen des virtuellen Studios könnte somit eine der zukünftigen praxisnahen Beispielimplementierungen dieser Arbeit sein. Denkbar wäre eine Zugriffshierarchie von Szenenmodeler, Beleuchter und eines Operators für die Kamerafahrten. Abhängig von ihrem Status können diese Benutzer nur bestimmte Objekte einer Szene bearbeiten und verändern. Das Optimum an Arbeitseffizienz wäre erreichbar, da alle Bearbeiter einer Szene gleichzeitigen Zugriff auf „ihre“ Objekt haben. Möglich wird dies über die Baum- bzw. Knotenstruktur von MPEG-4 Szenen. Der Szenengraph koordiniert hier die räumliche und zeitliche Anordnung von natürlichen oder synthetischen audiovisuellen Objekten. Das grundlegende Problem ist, wie noch ausführlich zu sehen sein wird, die Bearbeitung von Teilbaumstrukturen innerhalb dieses Szenengraphen.

[2] Interaktive Audiovisuelle Anwendungssysteme
http://www.imt.tu-ilmenau.de/forschung/fo_00300.html

2. Stand der Technik

Beispiel: Virtuelles Studio

Die im virtuellen Studio auftretenden Daten lassen sich mittels MPEG-4 effizient beschreiben. Die im MPEG-4 Standard festgelegten *Profiles* für visuelle, auditive und graphische Objekte ermöglichen eine inhaltsbezogene Beschreibung der Szene, eine geringe Datenrate und die Erhaltung der Originaltreue. Eine gespeicherte Szene kann somit ohne Qualitätsverluste durch Datenreduktion nachbearbeitet werden. Ein wesentlicher Vorteil der Lösung für das virtuelle Studio der TU Ilmenau besteht darin, das die Datenquellen schon parametrisiert erzeugt werden (3D-Set, Avatare, Graphikmaterial, Texte etc). Es entfallen somit die Schritte der klassischen Bildverarbeitung[3] von Objekterkennung, Bildanalyse und Klassifikation der Objekte. Diese genaue Vorgabe von verwendbaren Parametern und Objekten macht es einfacher, entsprechende spezialisierte Softwarewerkzeuge für das virtuelle Studio zu entwickeln.

2.1 MPEG-4

Der wohl größte Vorteil von MPEG-4 ist seine Einbeziehung des Nutzers in eine weite interaktive Welt. Nicht nur die Auswahl und das Starten von Audio- oder Videoströmen ist verwirklicht worden, MPEG-4 geht den entscheidenden Schritt weiter und erlaubt dem Benutzer das komplette Ersetzen einzelner Objekte durch neue. So ist es möglich, sich für einen bestimmten Typ von Moderator zu entscheiden und auch dessen Sprache mittels verschiedenen *Text To Speech*[4] Parametern zu beeinflussen. Ebenso gelingt es Eigenschaften von natürlichen Objekten zu verändern oder Objekte ganz zu löschen. Wie wäre es, wenn man im TV während einer Verkaufssendung interaktiv z.B. die Farbe eines Kleidungsstückes ändern könnte? Oder, während man sich frei vor einem Spiegel bewegt, den heutigen Anzug in entsprechender Größe und Farbe auf das Spiegelbild projiziert bekommt?

[3] Carsten Köhn, Bildanalyse und Bilddatenkompression, S.11, Abb.1
[4] TTS Text To Speech, MPEG-4 Overview, 2.4.2 Synthesized Sounds

Der Unterschied zu vorhergehenden Standards, liegt in der strengen Objektorientiertheit von MPEG-4. Objekte können einzeln benutzt, zu Objektgruppen zusammengeschlossen werden oder mehrere Gruppen bilden *higher level audio-visual entities*. Dieses anordnen von Objekten und Objektgruppen resultiert schließlich in der *composition* der kompletten Szene.
In Anlehnung an das Internet und die drahtlose Kommunikation, wie Handheld, Mobilfunkgeräte oder zukünftige UMTS-Anwendungen, ist MPEG-4 in der Lage, Endgeräte mit niedrigen Bitraten mit den entsprechenden Inhalten zu versorgen. Ein weiteres Feature ist hierbei der *scalable content.* Diese Prinzip erlaubt die einmalige Schaffung eines bestimmten Inhaltes (*content*), der dann je nach Ausgabeplattform und Bandbreite den entsprechenden Systemumgebungen angepasst (*scalable*) wird. Dies geschieht unbemerkt vom Benutzer automatisch im Hintergrund der entsprechenden Anwendung. Ein ähnliches Prinzip wird im weiteren Verlauf bei XML-und der XSL Transformation näher betrachtet werden.
Visuelle Objekte werden in ihrer Lage durch Koordinaten im Raum und ihr zeitliches Auftauchen je nach Zeitbasis beschrieben und müssen so nur einmal übertragen werden. Änderungen der Objekte erfolgen dann durch Umwandlung der Änderungsparameter in die neuen Raum-Zeit-Koordinaten. Das ermöglicht eine sehr schnelle Reaktionszeit und das möglichst in Echtzeit durchgeführte Update einer Szene bei Interaktionen.

2.1.1 Layerstruktur eines MPEG-4 Client Terminal

Im Gegensatz zu MPEG-2 definiert MPEG-4 keine Transportschichteigenschaften, dafür aber eine sehr flexible Multiplex Struktur. Ein besonderes Merkmal von MPEG-4 ist die Entwicklung von Wiedergabemöglichkeiten von interaktiven Szenen über Broadcast (Client-Server) und Wiedergabe von Massenspeichermedien (DVD, magnetische Speichermedien etc.).
Die Quelle überträgt gemultiplexte Ströme von komprimierten audiovisuellen Objekten und die dazugehörigen Szenen– und Objektbeschreibungen. Der Empfänger demultiplext die A/V-Ströme und setzt die resultierenden dekomprimierten Objekte nach der Szenenbeschreibung zusammen. Mit der entstehenden Szene kann der Benutzer nun je nach vorgesehenen Möglichkeiten interagieren. Die Interaktionsinformationen können auf der Benutzerseite erfolgen oder zur Quelle

übertragen werden. Bevor eine Szene beim Benutzer dargestellt wird, durchlaufen die Objekte drei Schichten (siehe Abb. 1).

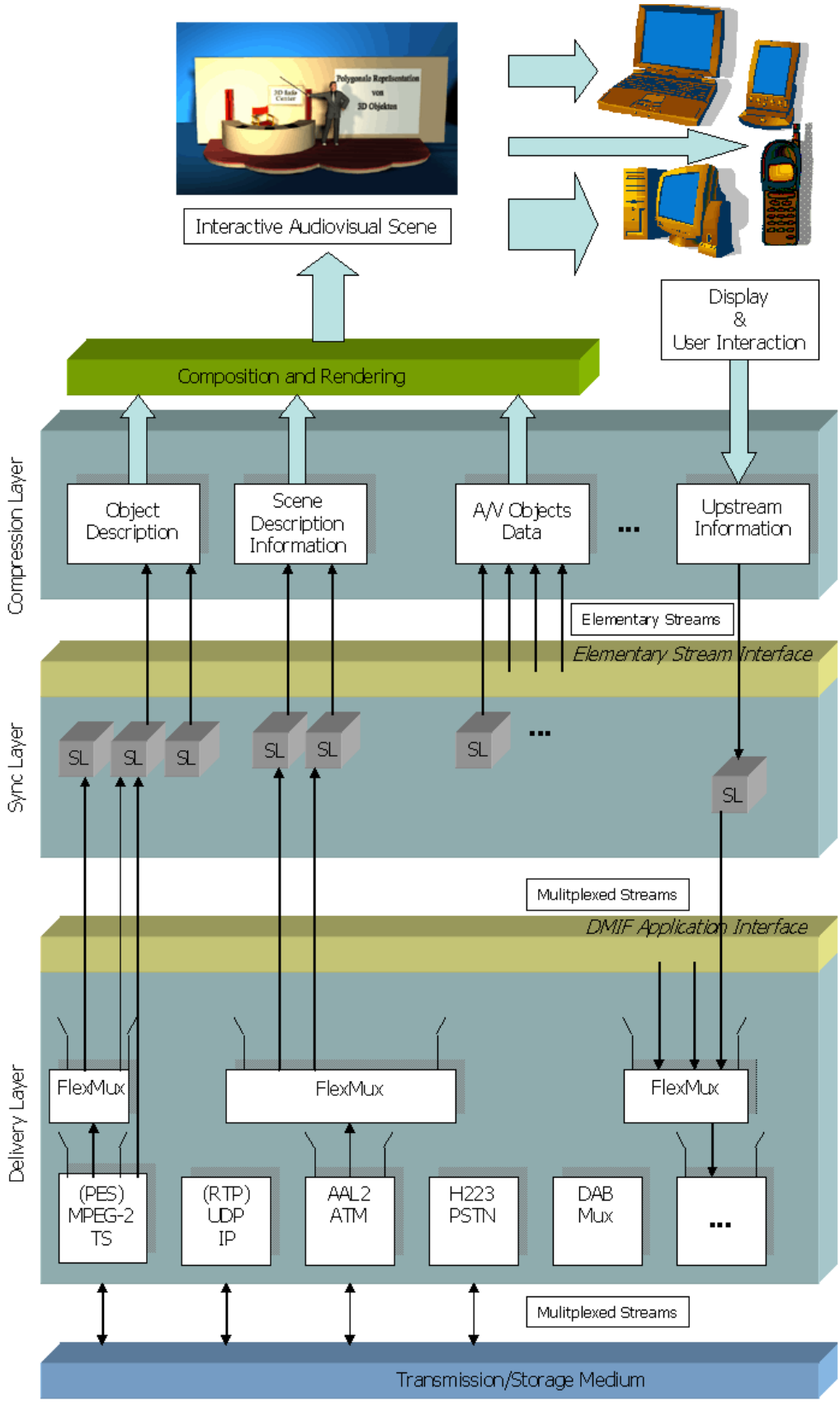

Abbildung 1 (2.1): Schichtenmodell MPEG-4 Client Terminal

Die Transportschicht (*delivery layer*) besteht aus dem Transport Multiplex (TransMux) und den MPEG-4 Multiplex (FlexMux). MPEG-4 definiert keine spezielle Transportschicht, sondern setzt auf vorhandenen Standards auf. Dies können Transportmöglichkeiten nach MPEG-2, Internetprotokollen oder auch Protokolle des digitalen Broadcastings sein. Es ist somit netzwerkunabhängig. Die zugehörige Schnittstelle heißt *DAI* für *Delivery Multimedia Interchange Format Application Interface.* Diese Schicht ist vom Medium unabhängig und transportbedingt.

Die Sychronisationsschicht (*sync layer*) passt die einzelnen Ströme (*elementary streams*) für die Kommunikation mittels der Schnittstelle für die elementary streams (*Elementary Stream Interface*) an. Es werden Zeit- und Synchronisationsdaten ebenso wie Informationen über gerichteten oder wahlfreien Zugriff übertragen. Diese Schicht ist von Medium und Transport unabhängig.

Die Kompressionsschicht (*compression layer*) decodiert Daten aus ihren ursprünglichen encodierten Format, den elementary streams, über das *Elementary Stream Interface* und stellt die notwendigen Operationen bereit, um die Objekte zu rekonstruieren und darzustellen. Die Kompressionsschicht enthält die benötigten Objekt Decoder. Diese Schicht ist medienabhängig und transportunabhängig.

Es ist möglich, zwischen den einzelnen Schichten Rechte auszuhandeln, um den Szeneninhalt oder einzelne Objekte zu schützen. *Intellectual Property Rights* (IPR) werden über IPMP Module verwaltet und organisiert.

Die letzte Schicht, die Präsentationsschicht, stellt die dekodierten Informationen auf dem Terminal des Benutzers dar.
Zusammenstellung, Präsentation und Rendering sind hierbei abhängig vom verwendeten System. Grafik beispielsweise kann aufwendig gerendert auf
High-End-Workstations und ebenso auf einem mobilen Gerät mit limitierten Grafikfähigkeiten wiedergegeben werden.
Der Benutzer kann jetzt mit der Szene interagieren und Informationen an den Server über den Rückkanal (*upstream channel*) senden.

2.1.2 Object Description Framework

Über das *Object Description Framework (OD)* werden die einzelnen Ströme identifiziert und innerhalb der Szene beschrieben. Ein OD beinhaltet ähnlich einer URL Zeiger auf die *elementary streams*. Es besteht aus mindestens einem *Elementary Stream Descriptor (ES Descriptor)* und optional aus Deskriptoren für *Object Content Information (OCI)* und Zugriffsdeskriptoren für IPMP. Durch dieses Einbringen von inhaltsbezogenen Kontrollpunkten an einer frühen Stelle der Verarbeitungskette ist eine umfangreiche Zugriffsbeschränkung bzw. Überwachung möglich.

Ein einfaches Beispiel ist die Möglichkeit einer Sprachauswahl (siehe Abb.2).

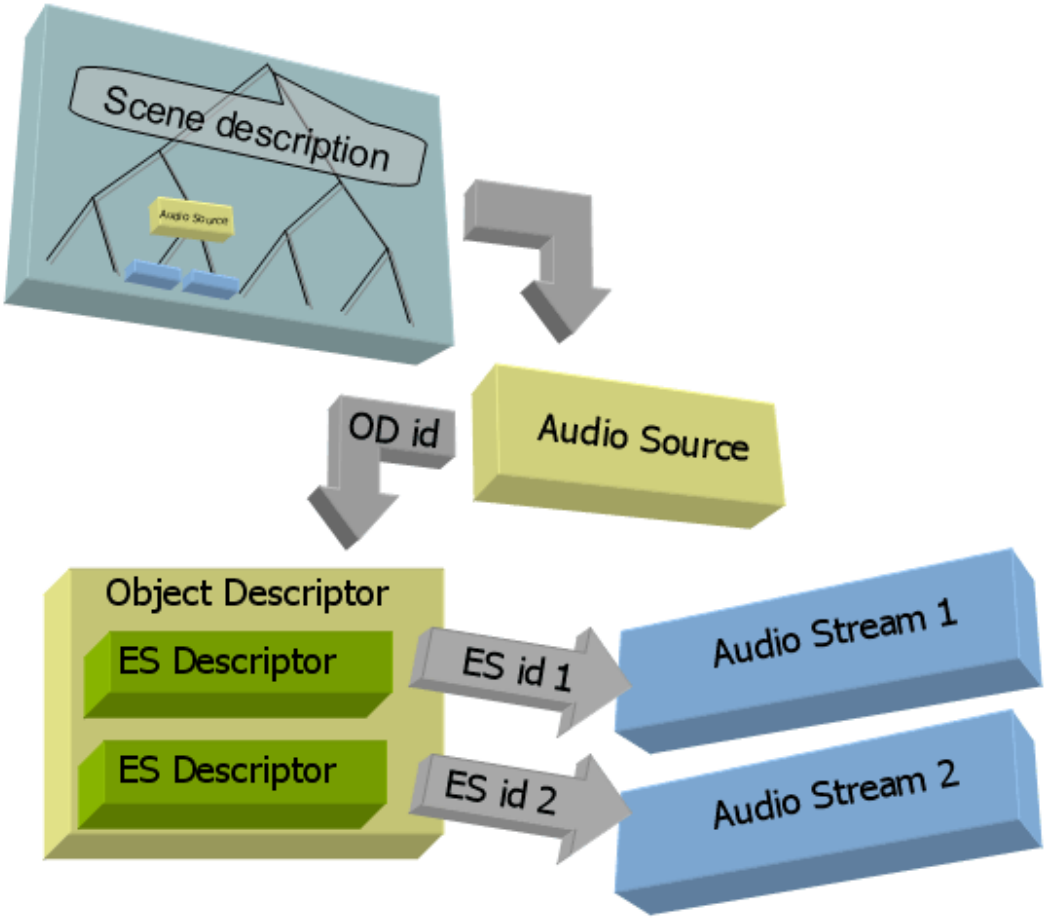

Abbildung 2 (2.2): Benutzung von OD bei einem Audioknoten für 2 Audio Streams

Die Komposition einer Szene erfolgt entsprechend der Szenenbeschreibung. Audiovisuelle Ströme werden in der Dekompressionsschicht dekodiert, mit anderen Objekten zusammengefügt und dem Benutzer präsentiert.

Mit Rendering wird in diesem Zusammenhang die semantische Interpretation der Knoten und der Interaktionen bezeichnet. Benutzerinteraktionen können von den verschiedensten Eingabegräten beeinflusst werden oder aber auch skriptgesteuert sein. Die Skriptsprachen ECMAScript und Java mittels der MPEG-J Schnittstelle stehen zur Verfügung. Über die MPEG-J Schnittstelle sind im Gegensatz zu den Möglichkeiten der Manipulation in VRML-Welten auch Zugriffe auf das Netzwerk und den Decoder möglich.

2.1.3 Anwendungsbereiche für MPEG-4

Fällt in den Medien das Schlagwort MPEG-4, so werden damit momentan die Verbesserungen von Codecs zur Video- und Audiokompression verbunden.
Aktuelle Codecs wie DivX[5] (Video) oder AAC-LD[6] (Audio) stellen letztendlich Weiterentwickelungen von Algorithmen des MPEG-2 Standards dar. Vor allem verbesserte Bildqualität bei niedrigen Bitraten und damit die Möglichkeit ca. 1h Videomaterial (PAL-Auflösung), in fast DVD Qualität auf einer herkömmlichen CD-ROM unterzubringen, verhalfen dem MPEG-4 Videocodec zum Durchbruch.
Ebenfalls verbesserte Qualität bei Audioübertragungen prädestiniert den AAC-LD Codec des Fraunhofer-Institutes für zukünftige Anwendungen, wie UMTS-Kommunikation und andere schmalbandige Kommunikationssysteme. Im Videosegment für UMTS-Anwendungen bietet sich ebenfalls MPEG-4 an, da es die Übertragung von Videoströmen über fehleranfällige Kanäle, wie Mobilfunknetze, unterstützt. Aus diesem Grund wurde MPEG-4 Video auch vom 3GPP[7]-Konsortium als ein Kompressionsstandard für UMTS ausgewählt.

Erste Prototypen von multimedialen MPEG-4 basierten Anwendungen zeigen deutlich wohin die Entwicklung gehen wird. Blaxxun[8] z.B. entwirft einen MPEG-4 kompatiblen Player, der sich für Live-Streaming 3D-Konferenzen eignet und auch virtuelle Einkäufe unterstützt. In einem virtuellen 3D-Modehaus kann man so beliebig Kleidung „anprobieren“ und diese aus allen Blickwinkeln betrachten, Farben und Muster ändern und natürlich kaufen.

[5] http://www.divxnetworks.com/ , http://www.igd.fhg.de/actual_divx.html (Video)
[6] http://www.iis.fhg.de/amm/techinf/mpeg4/index.html (Audio)
[7] Third Generation Partnership Project
[8] SoNG 3D Player http://www.song.blaxxun.de

Ebenfalls eine sehr interessante zukünftige Anwendung in Verbindung mit MPEG-7 ist das „MultiMedia Message Center – Die M³-Box“[9]. In diesem Projekt geht es um zukünftige asynchrone Multimediakommunikation zwischen mobilen Endgeräten verschiedenster Plattformen. Hierbei stehen die intuitive multimediale Nachrichtenerstellung und deren Anpassung an bestehende Netzwerkressourcen im Mittelpunkt der Entwicklung. Die auf MPEG-4 basierenden Mediadaten werden, ebenso wie die Metadaten der Nachricht, mittels MPEG-7 Profilen (*Description Schemes)* beschrieben.

Im virtuellen Studio erlaubt der MPEG-4 Standard eine effiziente Beschreibung von den genutzten Audio-, Video-, Grafik- und 3D-Daten. Die inhaltsbezogene Kodierung erlaubt neben der erhaltenen Originaltreue des Materials einen Transport der Mediadaten mit geringen Datenraten. Es ist ebenfalls möglich, neben dem Broadcasting, dem Anwender eines PCs oder dem Benutzer eines TV-Gerätes angepasste Applikationen zur Verfügung zu stellen.

Versucht man sich an einem noch weiteren Blick in die technologische Zukunft, so könnte man sich virtuelle 3D-Videokonferenzen mit Avataren vorstellen, die mittels personalisierten Körper- und Gesichtsanimationsparametern den realen menschlichen Benutzer nachbilden. Denkt man hierbei an internationale Gespräche, so könnten diese mit dem entfernten Gegenüber in der jeweiligen Landessprache über eine Echtzeitübersetzungsschnittstelle kommunizieren.
MPEG-4 Technologien wie *Text-To-Speech* und *Facial & Body animation parameters* rücken das Science-Fiction Szenario in greifbare Nähe.

Im Bereich Multimediabibliotheken wird das Suchen und gezielte Filtern bald der schwierigste Teil sein, um an Informationen zu gelangen. Mittels objektbezogenen und vor allem bewegungsbezogenen Szenenbeschreibungen eröffnen sich neue Wege der Informationsrecherche.

[9] Elektronische Medien: Technologien, Systeme, Anwendungen, ITG Fachbericht 167, S.157-162, 2001

2.1 MPEG7

MPEG-7 wurde ursprünglich als *Multimedia Content Description Interface* entwickelt und sollte Metadaten zu multimedialen Inhalten hinzufügen. Unter MPEG-7 versteht man die standardisierte Beschreibung von verschiedenen Arten multimedialer Informationen. Diese Beschreibungen sind mit dem Inhalt verknüpft und ermöglichen eine schnelle und effiziente Suche nach dem entsprechenden Material. Nicht im Standard festgeschrieben ist die (automatische) Merkmalsextraktion der Deskriptoren. Sehr vereinfacht kann man sagen, die vorhergehenden MPEG Projekte MPEG-1,-2 und -4 wurden entwickelt, um die Information an sich darzustellen. MPEG-7 aber wurde entwickelt, um Informationen über Informationen (Metadaten) zu repräsentieren[10]. MPEG-7 spezifiziert somit ein Standardset von Deskriptoren, welche für die Beschreibung von multimedialen Inhalten geeignet sind. Ebenfalls spezifiziert sind die Möglichkeiten eigene Deskriptoren als auch übergeordnete Strukturen (*Description Schemes (DSs)*) zu erzeugen. MPEG-7 definiert eine Beschreibungssprache zur Erstellung von *Description Schemes,* die *Description Definition Language (DDL).*

Anwendungen die MPEG-7 Beschreibungen verwenden, könnten inhaltsbasierte Bild- und Grafikdatenbanken, Suchmaschinen oder Contentmanagement Software für 3D Szenen sein. Verwirklicht wurde inzwischen eine Audiodatenbank, die Musikstücke über einen eindeutigen „Fingerabdruck“ identifiziert[11]. Durch die MPEG-7 Deskriptoren für z.B. Farbe, Texturen, Shapes, Bewegungen und Inhalt sind komplexe Suchanfragen z.B. nach Bewegungsabläufen (bestimmte Kamerafahrt) oder dem Verhalten von Objekten innerhalb der Szene, möglich (Bsp: Suche nach allen Szenen in denen der Moderator stehend berichtet).

Eine interessante Anwendung besteht in der Programmierung von personalisierten, intelligenten Agents, die das Internet oder spezifische Datenbanken selbstständig nach bestimmten Merkmalen durchsuchen und die Ergebnisse aufbereiten.

[10] Natürlich kann MPEG-7 ergänzend zu den bestehenden MPEG Standards verwendet werden. MPEG-4 und MPEG-7 haben z.B. gemeinsame Bereiche (Repräsentation der audiovisuellen Daten über Objekte). Es ist aber auch möglich MPEG-7 Deskriptoren auf z.B. analoges Video anzuwenden.

[11] Projekt AudioID, Fraunhofer Institut, http://www.emt.iis.fhg.de/PM_AudioID_Thomson.htm

Mittels der *Description Definition Language (DDL)* ist es möglich neue *Description Schemes (DSs)* und *Descriptors (Ds)* zu erzeugen sowie vorhandene DSs und Ds zu erweitern und zu modifizieren. Die DDL definiert syntaktische Regeln und kombiniert *Description Schemes* und *Descriptors*.
Eine DDL Datei ist ein Schema, das ein Set vom Einschränkungen definiert, welche eine gültige MPEG-7 Beschreibung erfüllen muss. Die DDL Datei ist in XML enkodiert und benutzt die in *W3C's XML Schema Language*[12] festgelegten Konstrukte.

W3C's XML Schema Language bildet die Basis für DDL. Da aber *XML Schema Language* nicht mir der Absicht entwickelt wurde, audiovisuelle Inhalte zu beschreiben, sind einige notwendige Erweitungen in DDL implementiert, um den Anforderungen von MPEG-7 zu genügen.

MPEG Multimedia Description Scheme (MDS) beinhalten spezielle Metadatenstrukturen, die für MPEG-7 entwickelt wurden, um audiovisuelle Daten zu beschreiben. Ein MDS kann zu einer visuellen (z.B. 3D Shape), auditiven (z.B. Melodiebeschreibung) oder generischen Beschreibung (z.B. Navigation) gehören. Auch hier können zusätzlich zu den spezifizierten Beschreibungen eigene Erweiterungen eingebracht werden.
MPEG Multimedia Description Scheme beschreiben in XML die Konzepte die hinter Suchen, Indizieren und Filtern von audiovisuellen Daten stehen. Die *Description Schemes* beschreiben die generischen Daten der A/V-Daten und über Contentmanagement deren spezifische inhaltliche Zugehörigkeiten. Hierbei werden verschiedene Ebenen, wie z.B. Semantik, Struktur, Models, Information und Navigation, betrachtet.
Sind so möglichst viele Metadaten erfasst, kann das generierte XML Dokument z.B. in einer Datenbank gespeichert werden.

[12] http://www.w3.org/XML/Schema

2.2 Extensible Markup Language (XML)

XML[13] wurde als Kompromiss zwischen der Einfachheit von HTML und der Funktionsvielfalt von SGML entwickelt. XML ist eine Teilmenge von SGML, die bei vertretbaren Implementationsaufwand, eine maximale Flexibilität in der Definition von Datenformaten gewährleisten soll.

Formatvorgaben wie CSS können sowohl von HTML, als auch von XML verwendet werden. Dagegen ist die Extensible Style Sheet Language (XSL) eine reine Implementation für XML.

Der XML Standard ist kein einzelner, komplett abgeschlossener Standard, sondern besteht aus vielen Einzelstandards. Man spricht deshalb von der XML Standardfamilie.

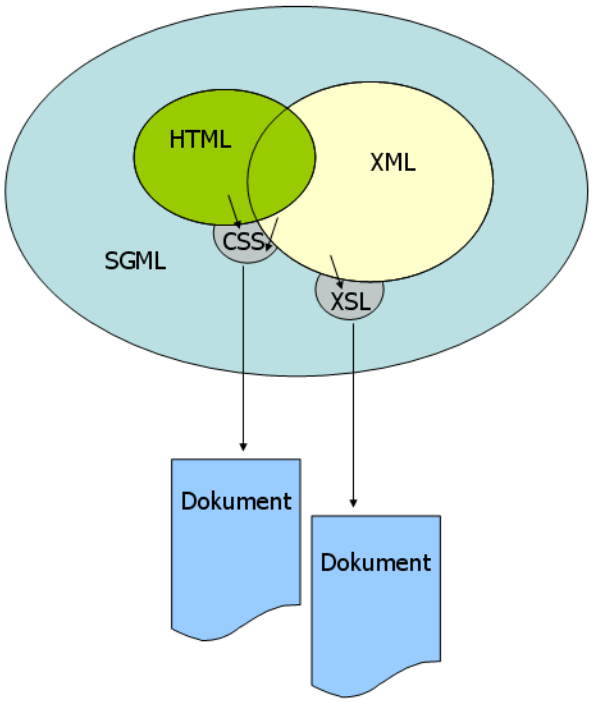

Abbildung 3 (2.3): Zusammenhang von SGML, HTML und XML

XML v1.0 legt die syntaktischen Grundlagen für die weiteren Standards. Für inhaltliche Aufgaben sowie deren Struktur kommt XML und XML Namespaces zum Einsatz. Mit XQL und der schon erwähnten XML Schema Language stehen Instrumente für das Contentmanagement, wie Anfragen und Typisierung, zur Verfügung. Mittels XSLT und XSL können Konvertierungen von XML nach XML oder mit entsprechenden Transformationsregeln in andere Formate wie HTML und PDF vorgenommen werden. Für die Verlinkung von Inhalten stehen gleich mehrere Standards zur Verfügung. Mit Hypermedia beschäftigen sich XPointer, XLink und XPath. Weiterhin wären da DOM und SAX, welche direkte Unterstützung für viele Programmiersprachen bieten. SMIL, SVG und CGM bieten Multimediafunktionalitäten und RDF ein semantisches Datenmodell für XML.

[13] Details zu den einzelnen Standards unter http://www.w3.org/

Die Ziele die mit XML verfolgt werden, sind sehr genau im XML Standard in Abs. 1.1 definiert.

„Die Entwurfsziele für XML sind:

1. XML soll sich im Internet auf einfache Weise nutzen lassen.
2. XML soll ein breites Spektrum von Anwendungen unterstützen.
3. XML soll zu SGML kompatibel sein.
4. Es soll einfach sein, Programme zu schreiben, die XML-Dokumente verarbeiten.
5. Die Zahl optionaler Merkmale in XML soll minimal sein, idealerweise Null.
6. XML-Dokumente sollten für Menschen lesbar und angemessen verständlich sein.
7. Der XML-Entwurf sollte zügig abgefasst sein.
8. Der Entwurf von XML soll formal und präzise sein.
9. XML-Dokumente sollen leicht zu erstellen sein.
10. Knappheit von XML-Markup ist von minimaler Bedeutung.“

2.3 Content Management mit XML

Im Allgemeinen versteht man unter Content Management (CM) eine systematische und strukturierte Beschaffung, Erzeugung, Aufbereitung, Verwaltung, Präsentation, Verarbeitung, Publikation und Wiederverwendung von Inhalten.[14]
An dieser Stelle muss jetzt untersucht werden, ob diese eher aus dem klassischen Dokumentenmanagement stammende Definition des CM auch auf eine Verarbeitungskette von MPEG-4 konformen Inhalten, wie z.B. 3D-Szenen anwendbar ist. Dies soll von Abbildung 4 illustriert werden, in der als erstes der übliche Weg des Dokumentenbasierten CM gezeigt wird und parallel dazu der Produktionsweg im virtuellen Studio, wie er mit diesem Schema des CM realisiert werden könnte. Zusätzlich wurden Bereiche markiert, an dessen Schnittstellen XML Anwendungen realisiert werden können.

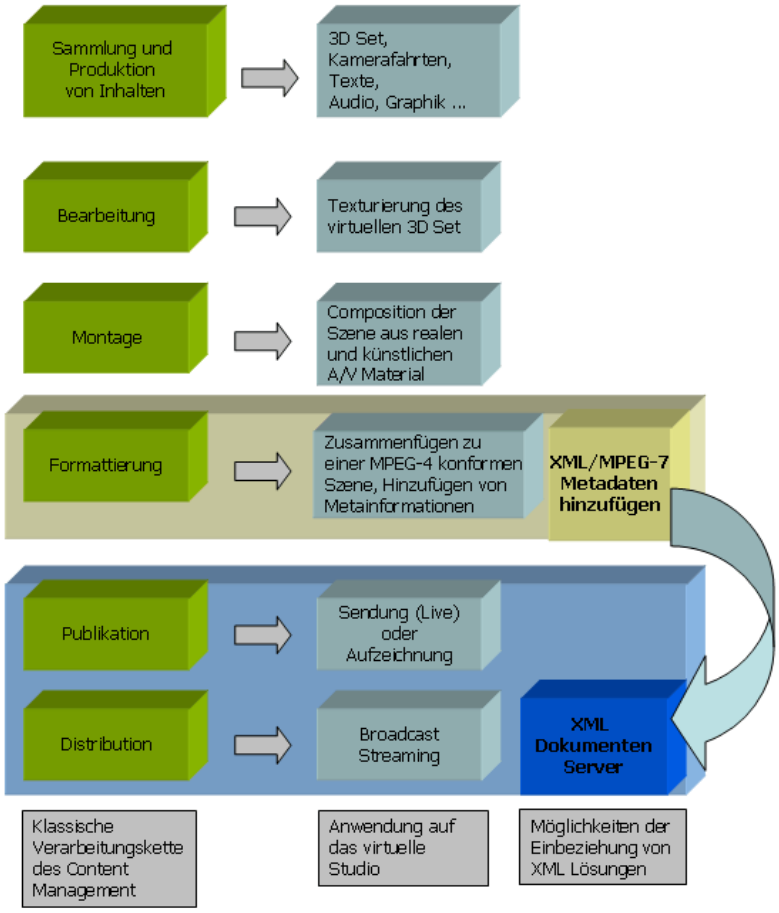

Abbildung 4 (2.4): Content Management

[14] Günther Rothfuss, Christian Ried, Content Management mit XML

2.4 Datenstrukturen in XML-Dokumenten

Was verbindet MPEG-4 mit XML, so dass an dieser Stelle tiefer in die Struktur von XML Dokumenten eingegangen werden soll?
Die Antwort liefert der MPEG-4 Standard selbst, da in diesem die Beschreibung von 3D-Szenen auf dem VRML Standard basiert. Gewisse Einschränkungen von VRML wurden mit MPEG-4/BIFS[15] behoben. Eine Weiterentwicklung bzw. eine Erweitung des VRML Standards, stellt die Entwicklung von Extensible 3D (X3D) dar. Wie die Abkürzung X3D schon vermuten lässt, handelt es sich dabei um eine Erweiterung mittels XML Tags. Die Integration von XML erlaubt nun, 3D Content mittels XML Tags zu erzeugen und zu bearbeiten.[16]

Zwei prinzipiell verschiedene Möglichkeiten, Datenstrukturen zu speichern, sind Aufzählungen und Hierarchien. Bilden bei Aufzählungen Listen von gleichartigen Objekten ohne gegenseitige Abhängigkeiten die Datengrundlage, so sind es bei Hierarchien Unterordnungsbeziehungen zwischen den einzelnen Objekten.
Ein typisches Beispiel für Listen sind Tabellen. Die relationalen Datenbanken[17] fungieren als „Navigationsaufsatz“ für diese Art der Datenstruktur. In einer Zeile befindet sich ein Datensatz (Record, Tupel) und in den Spalten die Attribute. Eine Tabelle ist somit eine Menge von n-Tupeln und jedes Tupelelement Spalte enthält genau einen Wert.
Typische Operationen, die auf Tabellen angewendet werden, sind die Sortierung oder die Auswahl von Teilmengen nach bestimmten Attributen.

Sucht man dagegen die typischen Operationen für Hierarchien, stellt man fest, dass dies vor allem die Suche nach Verwandschaftsbeziehungen und das Verpflanzen von Teilhierarchien sind (*pruning and grafting*[18]).

[15] für detaillierte Informationen über MPEG-4/BIFS, siehe Aron E. Walsh, CoreWeb 3D, Kapitel 17
[16] IBM Xeena v1.2E + X3D Extension, http://www.web3D.org/TaskGroups/x3d/translation/X3D-Edit-sdk.html
[17] O' Reilly, MySQL & mSQL, Kapitel 2 Datenbank Design
[18] engl. pruning: beschneiden, engl. grafting: einimpfen, verpflanzen

2.4.1 Struktureller Aufbau von Hierarchien

Grafisch darstellbar sind sowohl der Dokumenttyp *(Document Type Defintion DTD),* als auch das darüber ausgezeichnete Dokument, die Instanz des Dokumententyps. Dies ist wichtig, wenn man sich während des Designs einer eigenen DTD für seine speziellen Bedürfnisse die Struktur des entstehenden Dokumentes visualisieren möchte. Für XML Strukturen sind das Baumdiagramm, das Flussdiagramm und das Schachteldiagramm übliche grafische Notationen.
Dem Baumdiagramm soll hier der Vorzug gegeben werden, weil diese Darstellungsweise auch im VRML bzw. MPEG-4 Standard als Mittel zur Visualisierung von 3D-Szenen und Konstruktion von 3D-Objekten gebraucht wird. Ein Beispiel ist in Abbildung.5 und 6 zu sehen, wobei das Baumdiagramm im Allgemeinen der Übersicht halber vertikal dargestellt wird.

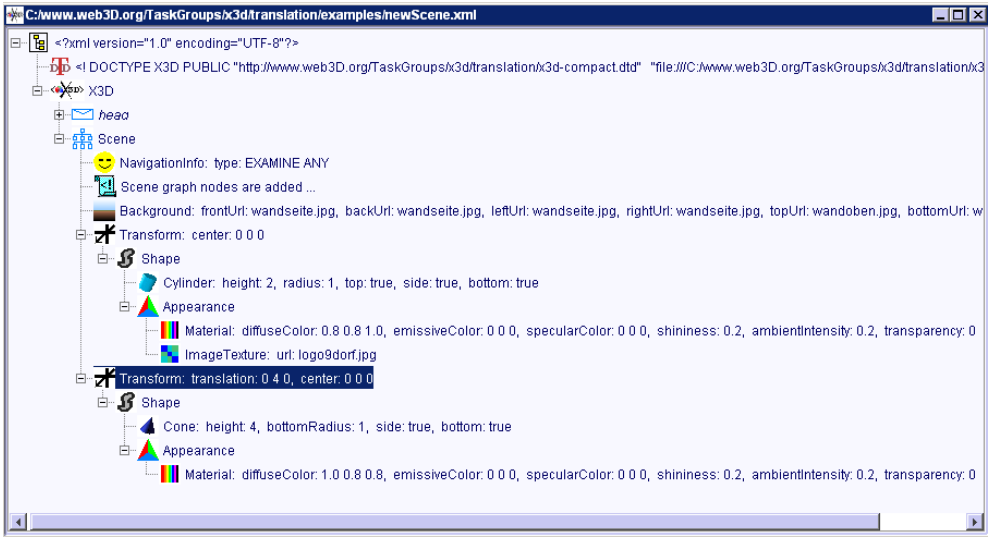

Abbildung 5 (2.5): Grafische Darstellung einer einfachen X3D Szene

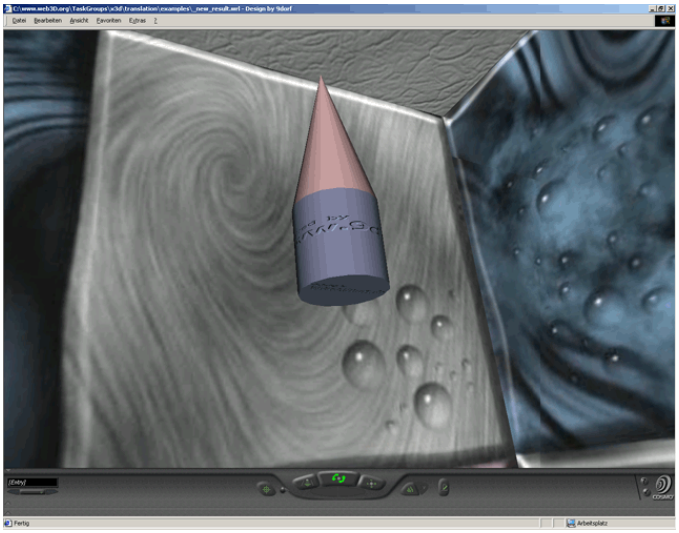

Abbildung 6 (2.6): Gerenderte Darstellung der Szene (VRML Datei) im Browser

2.4.2 Bäume und Graphen

Bäume sind Sonderformen von Graphen, wobei hier nur auf die gerichteten Graphen eingegangen werden soll. Speziell ist wiederum der so genannte azyklische Graph (DAG[19]) interessant. Dieser enthält wie der Name schon aussagt keine Zyklen (Selbstbezüglichkeiten) innerhalb des Graphen.

Ein gerichteter Graph besteht aus einer Menge von Knoten, zwischen denen Beziehungen bestehen, die gerichteten Kanten. Eine Kante (*Edge*) ist ein vollständiges 2-Tupel, welches an einem Knoten beginnt und an einem anderen Knoten endet. Diese Kombination heißt geordnetes Knotenpaar. Für den Startknoten und den Endknoten gibt es verschiedene Namen, welche immer das Abhängigkeitsverhältnis beschreiben (Vater, *parent* – Sohn, Kind, *child*; Vorgänger – Nachfolger; Knoten – Subknoten; *Rootnode* – *Subnode*). Dem Beispiel einer Familie zu Folge sind Knoten mit gleichem Startknoten Geschwister. Sind die Knoten und/oder Kanten mit Zusatzinformationen versehen, so heißt der Graph attributiert bzw. gefärbt. Findet man eine Folge von zusammenhängenden Kanten durch den Graph von einem Startknoten zu einem Endknoten, so heißt dieser Kantenweg Pfad (*Path*). Der Grad eines Knoten wird über die Zahl der Kanten im Pfad angegeben. Dieser Endknoten ist somit von dem Startknoten aus zugänglich. Können unter einem Knoten mehrere zusammengehörige Knoten gebündelt (gruppiert) werden, so spricht man von einem Gruppenknoten (*Groupnode*). Zur Veranschaulichung siehe Abbildung 7 .

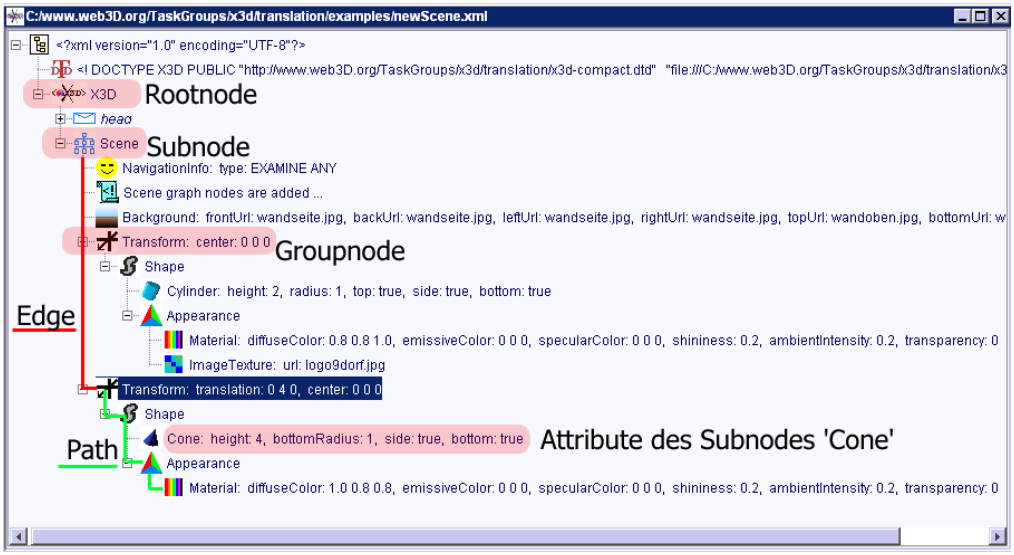

Abbildung 7 (2.7): Kommentiertes Beispiel der X3D Szene

[19] DAG: directed acyclic graph

2.4.3 Bäume und Hierarchien

Ein Baum T (*Tree*) ist ein spezieller gerichteter Graph mit einem ausgezeichneten Wurzelknoten R (*Rootnode*). R ist kein Endknoten, d.h. es enden keine Kanten in R. Alle Knoten von T außer R sind Endknoten von genau einer Kante und jeder Knoten ist von R aus zugänglich.
Knoten ohne Nachkommen heißen Blätter eines Baumes. Ein Teilbaum T' besteht aus einem Teil der Knoten des Originalbaumes T und erfüllt ebenso die Definition. Ein Teilbaum ist z.B. in Abbildung. 7 jeweils der Subnode ‚Transform'.

XML Dokumente haben den Aufbau von geordneten Bäumen, d.h. Kanten mit gleichem Startknoten sind linear geordnet. Diese Ordnung ist in der jeweiligen DTD festgeschrieben, die in serieller Schreibweise den Aufbau des Baumes und damit den Aufbau der Dokumentinstanz definiert.
Soll ein Baum in einem Dokument sequentiell gespeichert werden, muss er zuerst geplättet und dann mit einer Beschreibung der hierarchischen Abhängigkeit in der Reihenfolge der Plättung in das Dokument geschrieben werden. Dieser Prozess der Serialisierung kann mittels der inversen Operation der Deserialisierung umgekehrt werden. Über die Informationen der hierarischen Struktur wird der Baum rekonstruiert und dann die Sortierung der Knoten aus ihren relativen Speicherort bestimmt. Da es sich um eine inverse Operation handelt, wird exakt der Ursprungsbaum rekonstruiert. Diese Exaktheit der Rekonstruktion des Originals wird zu einem späteren Zeitpunkt, nämlich bei der Wiederherstellung von Bäumen aus Tabellen, noch eine entscheidende Rolle spielen.
Wesentliche Unterschiede zwischen Aufzählungen und Hierarchien bestehen ebenfalls in der Navigation von einem Element zum Vorgänger bzw. Nachfolger.
Innerhalb von Tabellen genügt das Auf – oder Abwärtsbewegen bis zum gefundenen Datensatz. Innerhalb von Bäumen ist das Bewegen von Knoten zu Nachbarknoten etwas komplizierter.
Von Traversieren spricht man, wenn ein Baum oder Teilbaum so durchquert wird, dass alle Knoten mit einem bestimmten Merkmal nacheinander aufgesucht werden. Eine schnelle Traversierung von Teilbäumen kann man durch vorhergehende Plättung des Baumes und die damit erfolgende Sortierung erreichen.

2.4.4 Übersetzbarkeit von XML Strukturen

Probleme ergeben sich bei einer Überführung von strukturierten XML-Inhalten von einer DTD in eine andere. Ein einfaches Beispiel ist dir Rückführung von einer X3D Szene in eine VRML Datei, um diese mittels Browser(plugin) darstellen zu können. Einige spezielle X3D Beschreibungsdimensionen (*Extensions*) wie Metatags, multiple Texturen, Partikel Sets, NURBS gehen dabei verloren.
Die vollständige Übertragung gelingt nur von der weniger detaillierten in die detailliertere DTD, in die ungekehrte Richtung würde Struktur verloren gehen.

3. Szenengraphstruktur am Modell eines VRML Dokuments

Die Szenengraph Technologie, welche von VRML[20,21] angewendet wird, geht auf eine Entwicklung von SGI zurück. In dem 3D Authoring Tool Open Inventor[22] von SGI wurde erstmals diese visuelle Art der Szenenbeschreibung eingesetzt.
Der Sourcecode von VRML entspricht in seiner Art der textuellen serialisierten Beschreibung der Szenengraph Datenstruktur. In dieser ist die Charakteristik der Objekte und die Beziehungen zwischen den Objekten gespeichert. Der Aufbau des Graphen entspricht der Struktur eines direkten azyklischen Graphen (DAG, siehe 2.4.2 Bäume und Graphen). Mittels der Szenegraphen ist eine Konstruktion von 3D Welten auf einer sehr hohen Abstraktionsebene möglich. (Abbildung 8)

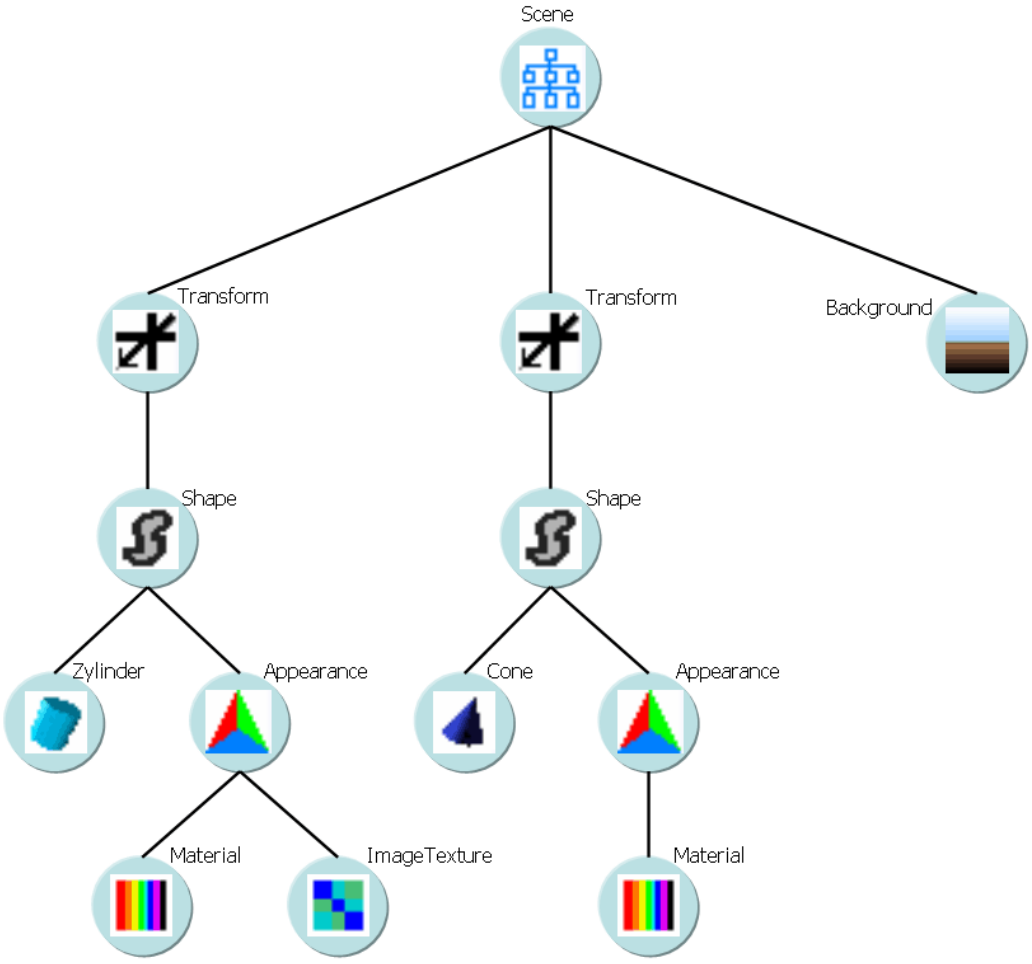

Abbildung 8 (3.1): VRML Szenengraph (ohne Attribute)

[20] Zur Entstehung von VRML, siehe Aron E. Walsh, CoreWeb 3D, Kapitel 4, VRML Overview
[21] VRML97 Specification, ISO-IEC 14772-11997,http://www.web3d.org/Specifications/VRML97/
[22] Open Inventor Mentor Manual, Chapter 3, Nodes and Groups

Der 3D Entwickler braucht sich nicht mehr um das „wie“ beim Renderprozess zu kümmern, sondern hauptsächlich um das, „was“ gerendert werden soll. Möchte man eine VRML Welt entwerfen sind OpenGL Kenntnisse durchaus von Vorteil, aber eben nicht Bedingung. Die Interpretation, Darstellung und das Rendering des VRML Codes übernimmt der VRML Browser[23].

3.1 Nodes, Fields, Events & Routes

3.1.1 Nodes

Szenegraphen sind aus Elementen aufgebaut. Diese Objekte werden als Knoten (Nodes) bezeichnet. Diese Nodes sind, ähnlich wie Klassen in Java oder C++, übergeordnete Objekte, die bestimmte Eigenschaften wie Textur, Licht oder Geometrie beschreiben. Szeneentwickler können auf 54 vordefinierte Knotentypen zugreifen. Es können ebenfalls neue oder von den bestehenden Knoten abgeleitete, eigene Knoten definiert werden. Dieses Prinzip des Weitergebens von Objekteigenschaften ist aus den OO Programmiersprachen als Vererbung bekannt.
Die Elternknoten (*parent nodes*) sind verantwortlich für die Verwaltung ihrer Kinderknoten (*child nodes*), wobei die *child nodes* wiederum eigene *child nodes* enthalten können usw. Dieses Konzept der prinzipiellen unendlichen Verzweigungstiefe von Knoten zeigt bei einer Visualisierung das typische (auf dem Kopf stehende) Abbild eines Baumes mit Wurzel, Ästen und Blättern.

Eine besondere Stellung nehmen die *Grouping Nodes* ein. Sie beinhalten und verwalten ebenfalls andere Knoten, aber auf einer noch höheren Abstraktionsebene als die normalen *parent nodes*. Geometrische Formen können zusammengefasst werden, um in dieser Gruppierung ein komplexeren Gegenstand darzustellen. VRML *Group Nodes* sind z.B. *Anchor*, *Group* oder wie im Beispiel in Abbildung 8 *Transform*. Der Anchor Knoten gibt der VRML Szene die Möglichkeit, Hyperlinking Funktionen einzubetten. Der Knoten Transform verändert die Position und Größe des geometrischen Objektes. So ist der Kegel aus dem Beispiel um 4 positive Höheneinheiten (siehe Abb.5, markierter Bereich) in Richtung y-Achse verschoben um so direkt auf dem Zylinder zu sitzen.

[23] Cosmo Player by Cosmo Software, http://ca.com/cosmo/ oder
Contact by blaxxun, http://www.blaxxun.de/services/support/download/install.shtml

3.1.2 Fields

Jeder Knoten enthält ein oder mehrere Felder (*fields*), die Attribute wie Größe, Farbe oder Position im Raum definieren und kontrollieren. Der Zylinder im Beispiel Abbildung 5 ist aus dem Knoten ‚Cylinder' konstruiert, der mit den Attributen für Größe, Farbe, Material und einer Textur definiert wird.
Jeder Knotentyp hat eigene vordefinierte fields, die mit Standardwerten belegt sind. Es gibt zwei Datentypen für Felder. SF (Single-value Field) enthält genau einen Wert, wogegen MF (Multiple-value Field) einen oder mehrere Werte enthalten kann. Für die Werte selbst stehen die Datentypen Boolean, Ganzzahl, Fließkommazahl, Node und Zeichenketten zur Verfügung. Boolean z.B. wird angewendet, um zu definieren, ob die obere, untere und umschließende Seite eines Zylinders angezeigt (true) werden soll oder nicht (false).
Grundsätzlich sind alle Felder in VRML, die URLs enthalten, MF. Sollte die erste URL nicht erreichbar sein, wird die zweite usw. versucht, bis eine URL aufgelöst werden konnte. Liegen die verlinkten Dateien lokal vor (z.B. eine Offline CDROM Produktion mit 3D Szenarien), braucht es keiner alternativen URLs. Ebenfalls Anwendung für MFs findet bei dem Knoten ‚*Background*' statt. Hier können je nach Betrachterwinkel unterschiedliche Farben definiert werden. So können leicht Farbverläufe (z.B. für Himmel oder Boden) in VRML realisiert werden.
Neben den Datentypen für Werte in den Feldern gibt es weiterhin vier verschiedene Feldtypen. Die Typen *eventIn* und *eventOut* sind für die Kommunikation von Knoten untereinander definiert. Sie managen hierbei das Empfangen und Senden von *Events*. Ein als *field* klassifizierter Feldtyp kann nicht von anderen Knoten aus veranlasst werden, Eigenschaften zu ändern (man sagt dieses Feld ist *private*). Möchte man dies erreichen, sollte der Feldtyp exposedField verwendet werden. Hierauf kann nun von anderen Knoten über *Events* oder per Skript auf die Eigenschaften Einfluss genommen werden, der Knoten ist öffentlich (*public*) zugänglich.

3.1.3 Events & Routes

Über das Management von *Events* können Knoten Ereignisse senden und/oder empfangen. Events entstehen nicht zufällig, sie werden getriggert, also durch bestimmte Aktionen ausgelöst. Sind in die Szene Trigger eingebaut, die auf Ereignisse reagieren, können Events ausgelöst werden. Typische Trigger sind z.B. *mouseOver*, *mouseClick*, zeitgesteuerte Auslöser oder Sensoren für Annäherung und Kollision.
Nicht alle Knoten unterstützen *Events* direkt. Solche Knoten können trotzdem über Events kontrolliert werden, indem ein eventfähiger Elternknoten „vorgeschaltet“ wird. In der VRML Beispiel-Szene ist der Knoten ‚Cylinder’ selbst nicht in der Lage mit Events umzugehen, wohl aber sein Elternknoten ‚Transform’. Über diesen kann der Zylinder mittels Events verformt, verschoben oder rotiert werden. Eng mit Events ist das *Routing* verbunden. Die *Routing* Technologie erlaubt es Knoten mit *eventOut* Feld, mit einem *eventIn* Feld eines anderen Knoten zu verbinden. Im Prinzip entspricht dies der virtuellen Variante eines Audiosteckfeldes. Auch dort werden Ein – und Ausgänge durch Kabel und Steckfeld verbunden.

3.2 Extensible 3D (X3D)

X3D[24] als konsequente, offene Weiterentwicklung von VRML97 wurde anfangs auch als VRML Next Generation bezeichnet. Für X3D entschied man sich, um das enge Verhältnis zu XML herauszustellen. X3D wird mittelfristig VRML97 ersetzen, aber dennoch kompatibel zu bestehenden VRML Inhalten und Browsern sein. Extensible bedeutet in Verbindung mit 3D, dass die Beschränkungen von VRML aufgehoben werden und zusätzliche APIs genutzt werden können.
Sehr vereinfacht gesagt, entspricht X3D einen in Profile zerlegten VRML97, mit den Fähigkeiten, beliebig um weitere Komponenten erweiterbar zu sein.
X3D versteht sich ebenfalls als der kommende 3D Standard für plattformübergreifende Applikationen wie Set Top Boxen, Spielkonsolen und Informationsanwendungen. Aufschluss über die eingeschlagene Richtung gibt ein genauerer Blick auf integrierte Web- und Mediastandards und Technologien.
X3D setzt auf XML und DOM zur Beschreibung von knotenbasierten Dokumenten. Formate für Mediadaten (GIF, JPG, PNG, WAV, MP3, MPEG-2, QT), MPEG-4 Technologien (Kompression, 3D, Streaming) und Seitenbeschreibungstechniken (HTML+TIME, SMIL) zur Darstellung von 3D Inhalten runden den Standard ab.

[24] http://www.web3d.org/x3d.html , siehe auch die X3D FAQ für einen ersten Überblick

3.2.1 Erweiterungen zu VRML

Am interessantesten im Zusammenhang mit dem Thema dieser Arbeit ist die erweiterte Traversierung, Modifizierung und Erzeugung von Szenegraphen. In der weiteren Verknüpfung mit Datenbanken ist die Fähigkeit von X3D, Metadaten einzubinden, von Bedeutung. Darüber werden die Indizierung und das Auffinden von Szenen vereinfacht.
Metadaten können für die gesamte Szene im Header der Datei definiert und ebenfalls für einzelne Objekte vergeben werden. Ähnlich den Profilen des MPEG-4 Standards gibt es in X3D Profile, die spezifische Anwendungsfälle abdecken. Die Profile sind gemäß der Anlehnung an den XML Standard in DTDs definiert. Wichtige DTDs[25] sind das X3D Core Profile, X3D Compact Profile, X3D Compromise Profile[26] und das SUN Profile[27].
Weitere spezielle X3D Extensions betreffen Techniken wie Multitexturing, Partikel Sets und NURBS.

3.2.2 X3D Markup Language (X3DML)

Das X3D XML Tagset ist ein „leichtes“ 3D Format, das eine API und eine Runtimeumgebung unterstützt, welche sich einfach in bestehende 2D Webstandards einbinden lässt. Das XML Tagset für X3D ist untergliedert in funktionelle Komponenten, die als Komponentensammlung wiederum Profile bilden. Es lässt sich durch direktes Mapping in das X3D Dateiformat umwandeln.

Im Gegensatz zu dem einen Standard repräsentierenden VRML97, ist X3DML eine Anwendung (im Sinne von Applikation) der Metasprache XML.
X3DML unterstützt ein Set von XML Entities und Element Type Deklarationen, um XML kompatiblen Code von X3D zu erstellen. Die Attribute der XML Element Typen sind exakt in einer DTD definiert. Das XML Vokabular besteht aus einem Baum von Knoten. Diese Knoten werden bei der Erstellung einer DTD zu Unterelementen (*child elements*) und Felder zu XML Attributen gemappt.

[25] http://www.web3d.org/TaskGroups/x3d/translation/
[26] Das X3D Compromise Profile ist die von Web3D empfohlene Variante.
[27] Das SUN Profile wurde verwendet, weil in diesem alle VRML97 Nodes definiert sind.

3.2.3 Rücktransformation von X3D mittels XSLT

Werden X3D Daten direkt mittels eines X3D Editors erstellt, gibt es verschiedene Möglichkeiten der Weiterverarbeitung. So kann es gewünscht sein, aus der X3D Datei eine HTML Seite für die Webpublikation zu erstellen oder X3D nach VRML zu transformieren.

Diese strukturelle Umformung von XML Dokumenten ist mittels Style Sheets möglich. Über XSL (Extensible Stylesheet Language) können, vergleichbar mit CSS für HTML, Objekte formatiert (*FO = Formatting Objects*) werden.[28] Die Auswahlpfade und Möglichkeiten von XSL sind hingegen wesentlich komplexer als die von CSS. FO entspricht momentan dem Funktionsumfang von CSS, soll aber nicht zwangsläufig darauf beschränkt bleiben.

Über XSL Transformations (XSLT) wird mittels Schablonen (*Template Rules*) beginnend an der Wurzel des Dokumentes nach Elementen des XML Baumes gesucht. Findet der XSL Parser den entsprechenden Knoten, so wird die definierte Regel angewendet. Regeln können von Sortieren eines XML Dokumentes über Erzeugung eines Inhaltsverzeichnisses bis zur Umformung in ein anderes Format reichen.

[28] c't Ausgabe 6/2000, Seite 244 ff XSL im Detail, Seite 255 ff XML mit Serviervorschlag

Die XSLT Technik erweist sich dann als effektiv, wenn über eine automatische Erkennung der Systemumgebung dem Server, auf dem XML Dokumente verwaltet werden, mitgeteilt wird, welches Style Sheet nötig ist. Der Server liefert dann die angeforderten Daten über die Definition des Style Sheets und XSLT die transformierten XML Daten an das Endgerät aus (Abbildung 9).
Entwicklungsbedarf besteht vor allem an der Verbindung von XML-Server – bzw. Datenbank und XSLT.

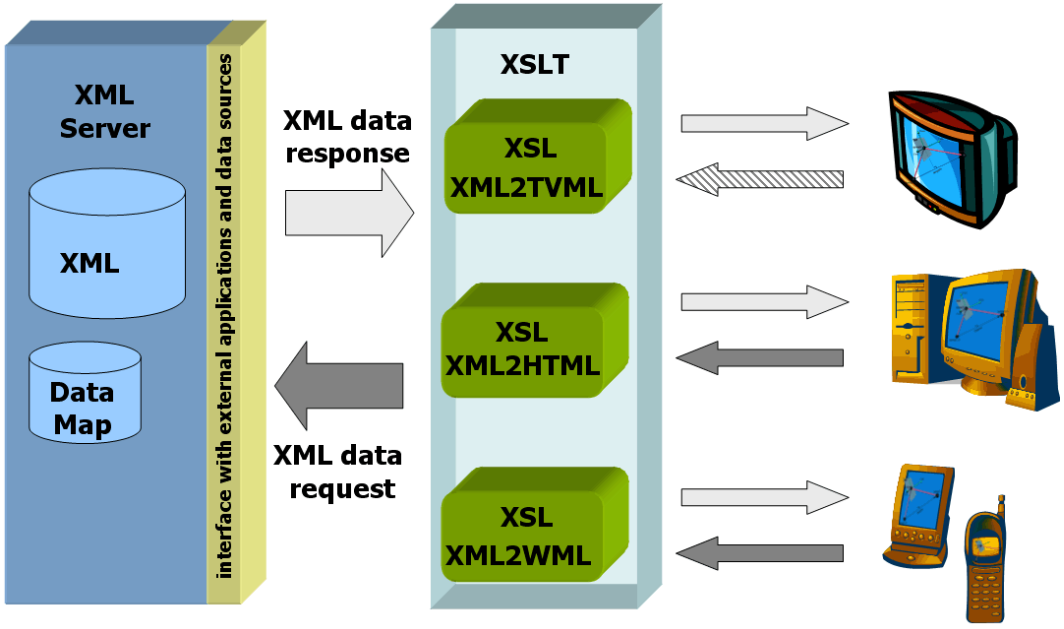

Abbildung 9 (3.2): XML Server + XSLT Szenario

4. Untersuchung von DBMS und ihre Eignung zur Speicherung von komplexen Knotenstrukturen

Ein Datenbanksystem besteht aus der Datenbasis, in der die Daten abgelegt werden, und den Datenverwaltungsprogrammen (Datenbanksoftware, Datenbankmanagementsystem (DBMS)), die die Daten entsprechend den vorgegebenen Beschreibungen abspeichern, auffinden oder weitere Operationen mit den Daten durchführen (Duden Informatik).

Die Notwendigkeit, audiovisuelle Szenen über ein DBMS zu verwalten, ergibt sich aus den Anforderungen, die an die weitere Verarbeitung der Szene gestellt werden. Benutzerrechte, die an einzelne Szenen oder Szenenkomponenten gekoppelt sind, verlangen nach definierbaren Zugriffskontrollen. Beispielsweise kann es bei Szenen größeren Umfangs erforderlich sein, Rechte für verschiedene Arten von zugriffsberechtigten Personen zu vergeben. Denkbar wäre eine Unterteilung in einen Szenen Administrator mit vollen Zugriffsrechten, einen Modeler für 3D-Konstruktionen oder um bei dem Beispiel virtuelles Studio zu bleiben einen Verantwortlichen nur für Kamerafahrten. Dies lässt sich dann aber nicht mehr alleine mit Schreib-, Lese- und Änderungsrechten für eine komplette Szene (in diesem Sinne Szene = Dokument) realisieren. Die Dokumentenstruktur muss in ihre elementaren Objekte zerlegt und diese dann mit Zugriffsrechten versehen werden. Hier sieht man schon, dass die Mittel der reinen dokumentenbasierten Speicherung nicht hinreichend sind. Ein weiteres Kriterium ist das Authoring von Szenen, bei dem unsere eben definierten Nutzer gleichzeitig an einer Szene arbeiten. Leserechte gewähren den Blick auf die gesamte Szene, Änderungs- und Schreibrechte die Bearbeitung von Szenenteilen.

Über ein DBMS lässt sich eine Content Base für Szenen schaffen, ohne dass der Anwender bei einer Suche wissen muss, wo und wie diese Szenen gespeichert sind. Das organisierte Speichern der Szenen und der enthaltenen Objekte inklusive deren Metadaten übernimmt vollständig das DBMS. Ebenso verhält es sich beim Suchen nach bestimmten Szenen und, was in diesem Falle besonders wichtig ist, einzelnen Objekten.
Ein weiteres Problem, welches als Ausschlusskriterium für dokumentenbasierte Speicherung gelten kann, ist das Problem des *Inlining*. Das bedeutet eine Szene kann andere Szenen enthalten, welche in externen Dateien gespeichert sind.
Erst in der Zusammenarbeit mit Content Management Systemen (CMS) können Funktionen, die heutigen DBMS fehlen, die aber für den Anwender nötig sind, erfüllt werden. Die wichtigste Forderung im Zusammenhang mit dem Szenario der gleichzeitigen Bearbeitung von audiovisuellen Dokumenten, ist die Versionskontrolle. Nur mit ihr kann sichergestellt werden, dass Änderungen an einem Teil der Szene erstens vermerkt und zweitens nachvollziehbar sind. Um dem Charakter von 3D-Szenen gerecht zu werden, müsste ein DBMS + CMS die Versionskontrolle auf Objektebene anbieten. Erste Ansätze für die Versionskontrolle für Dokumente bietet die Technik WebDAV[29] (Web Document Authoring and Versioning), welche nicht auf bestimmte Mediatypen beschränkt ist.
Eng damit verbunden ist die Funktion des Rollback, d.h. die Möglichkeit, beliebige Bearbeitungsstufen zurückzusetzen. Vergleichbar mit dem ‚UNDO' Befehl in herkömmlichen Programmen sollte das Rollback für knotenbasierte Dokumente Funktionen bieten, die auf einzelne Objekte, Objektgruppen und Teilbäume angewendet werden können.

Um feststellen zu können, welches DBMS geeignet ist, um knotenbasierte Dokumente zu verarbeiten, folgt als nächstes eine kurze Analyse der derzeit verbreitenden DBMS. Die Beschreibung ihrer grundlegenden Eigenschaften und die Eignung für dieses Projekt stehen hierbei im Vordergrund.

[29] http://www.webdav.org/

4.1 Relationale Datenbanken

In vielen Bereichen hat sich das relationale Datenbankmodell als Standard etabliert. Man denke nur an die DB von Orcacle, DB2, Informix oder das OpenSource Projekt MySQL/mSQL. Da aber in der Datenhaltung ein Wechsel von einfachen Listen zu komplexen Objekten stattfindet, stellt sich die Frage, ob dieses Modell den heutigen Anforderungen gewachsen ist.
Zur Datenbeschreibung wird die Data Definition Language verwendet (DDL). Mittels der Constaint Definition Language (CDL) werden Integritätsbedingungen festgelegt und die Data Manipulation Language (DML) dient schließlich zur Handhabung der Daten.
Um Daten in ein relationales Modell abzubilden, gibt es einige Regeln zu beachten. Entitäten werden zu Tabellen und ihre Attribute zu Spalten. Entität und Attribute stehen also über die Tabelle in Relation. Jede Spalte muss einem geeigneten Datentyp entsprechen. Eindeutige IDs werden zu Spalten, die keine Nullwerte enthalten dürfen (Primärschlüssel). Zwischen genau zwei Relationen können über Fremdschlüssel eigenschaftslose Beziehungen definiert werden.
Für Beziehungen zwischen mehr als zwei Tabellen wird eine Hilfstabelle benötigt. Eine solche ist ebenfalls nötig, wenn die Beziehung zwischen zwei Relationen nicht eigenschaftslos sein soll und wenn so genannte N:M Relationen bestehen.
Hierarchisch strukturierte Eigenschaften mit mehr als einem Wert je Datensatz (Zeile) können im relationalen Modell nicht direkt dargestellt werden. Ein Beispiel sind die Eigenschaften die den Zylinder aus Abbildung 5 beschreiben.
Aus dieser Betrachtungsweise leitet sich der größte Nachteil der RDBMS in Bezug auf die Speicherung von hierarchischen Strukturen ab. Die Daten einer Entität sind wie eben ausgeführt auf mehrere Tabellen verteilt und müssen wieder zusammengefügt werden, um das Originaldokument zu erzeugen. Diese *Join* Funktionen sind sehr zeitaufwändige Datenbankoperationen, vor allem wenn man sich vorstellt, welche Ausmaße die Verteilung von Daten auf mehrere Tabellen annehmen kann, wenn stark strukturierte Dokumente verarbeitet werden sollen.
Anderseits sind RDBMS auf einem ausgereiften Entwicklungsstand und auch in punkto Performance optimiert. Eine große Rolle spielen hier auch die vorhandenen Systeme und Kenntnisse von RDBMS und die Verfügbarkeit von technisch

ausgereiften OpenSource Varianten. Dadurch gibt es eine ganze Reihe von Versuchen und Techniken, wie man hierarisch strukturierte Daten in RDBS mappen kann (Details unter 4.6.1 – 4.6.3). Unter Mapping bzw. mappen versteht man die Abbildung von einer Struktur in eine andere, in diesem Fall von einer knotenbasierten Form in ein relationales Modell.

Einer der Hauptgründe, warum später noch sehr genau in die Techniken des Mapping eingegangen wird, obwohl schlussendlich dennoch eine native XML Datenbank verwendet wird, soll an dieser Stelle kurz erläutert werden.
Ausgehend von den Erfahrungen und den technischen Vorrausetzungen[30] bot sich eine Konzeption eines Datenbanksystemes zur Organisation audiovisueller Szenen mittels RDBMS an. So ist das Endergebnis auch als Erkenntnis aus der Entwicklung einer Idee und eines Konzepts über die Umsetzung und die Sammlung von Erfahrungen zu verstehen. Ein berechtigter Zweittitel des 4. Kapitels könnte „Evolution der Speicherung von knotenbasierten Dokumenten in relationalen Modellen bis hin zu nativen XML Umgebungen“ lauten.

4.2 Post-relationale Datenbanken (NF2)

Das postrelationale Datenmodell ermöglicht es, hierarchisch geschachtelte Strukturen zu beinhalten. Tabellen dürfen selbst Tabellen enthalten. Dies geht über die erste Normalform des relationalen Datenmodells hinaus.[31] Die Abkürzung NF2 bedeutet demzufolge Non-First-Normal-Form-Systeme. Geschachtelte Tabellen erlauben eine größere Flexibilität für komplexe Entitäten und schaffen Lokalitäten, was sich wiederum in gesteigerter Verarbeitungsgeschwindigkeit ausdrückt. So kann z.B. eine 1:N Beziehung mit nur einer Tabelle realisiert werden und auch wiederholte Schachtelungen stellen kein Problem dar. Für flache Strukturen wie in Adressdatenbanken, Bestellsystemen oder E-Commerce Anwendungen mag sich diese Lösung anbieten, aber für hochkomplexe 3D-Szenen ist auch sie nicht geeignet.

[30] 1. Es wurden schon einige Projekte mittels relationaler DB durchgeführt.
http://www.tu-ilmenau.de/~getsoft/ → Datenbank und
Multimediaprojekt Digitale Audiorestauration (TU Ilmenau)
2. Die technischen Vorraussetzungen zur Entwicklung waren komplett vorhanden.

[31] „Eine Entität befindet sich danach in erster Normalform, wenn alle Attribute nur einen einzigen Wert besitzen.“ MySQL &mSQL; King/Reese; S.17

4.3 Objekt relationale Datenbanken

Das objektrelationale Datenmodell bietet die Möglichkeit, selbst definierte Datentypen und Funktionen, die diese Datentypen verarbeiten, zu erzeugen. Ein weiteres Mal erhöhen sich die Möglichkeiten, Strukturierungen einzubinden. Tabellen sind hier Ausprägungen von Datentypen und im objektrelationalen Modell noch nicht gekapselt.
Erst durch die Übernahme von Techniken, wie sie in der objektorientierten Programmierung angewendet werden, entstanden gekapselte Datenstrukturen.
Auch als Klassen bezeichnete abstrakte Datentypen (ADT) können definiert werden. Tabellen sind dann als Ausprägung dieser Klassen Objekte, die Funktionen (Methoden) enthalten. In diesem Bereich spricht man von objektorientierten DBMS (OODBMS). Weitere Merkmale sind die Vererbung von Eigenschaften und die zusätzlichen hierarchischen Beziehungen zwischen den einzelnen Klassen.

Der größte Vorteil der OODMBS liegt darin, dass sie aufgrund ihres OO Datenmodells in ein einheitliches Modell von DBMS und zugehöriger Programmierschnittstelle eingebunden werden können. So können Entwickler die mit C++ oder Java arbeiten, auf Klassen mit definierten Schnittstellen zurückgreifen.

OODBMS sind zwar in der Lage, Strukturen besser als RDBMS abzubilden, aber die Überführung von vorhandenen Daten in das neue Modell produziert mehr Probleme als Nutzen. Dies ist einer der Gründe, warum sich diese Technik noch nicht im großen Maßstab, sprich auf kommerzieller und kaufmännischer Ebene durchgesetzt hat, da dort seit den Beginn der EDV-Welt mit RDBMS gearbeitet wird. Ebenfalls in Hinblick auf die Performance gibt es Verbesserungswünsche.

Bei einer Neuentwicklung eines kompletten Systems wie hier hingegen, sind die OODBMS eine genauere Untersuchung wert. (4.6.4)

4.4 XML als universelle Beschreibungssprache

Eine Sprache wie SQL3 speichert in einem Datenschema für jede Tabelle die dazugehörige Satzstruktur. Damit dies effektiv funktioniert, geht man davon aus, dass die Anzahl der Datensätze gegenüber der Anzahl der Tabellen wesentlich größer ist. Bei stark strukturierten Dokumenten kann dies beim Mapping aber genau in das Gegenteil umschlagen, viele Tabellen (Entities) mit je wenigen Datensätzen (Attributen).

Bei den genauen Betrachtungen eines XML Dokumentes und der Beschreibung eines DBMS könnte man zu dem Schluss gelangen, dass jedes XML Dokument in sich selbst schon eine Art Datenbank darstellt.

Wie in einer Datenbank enthält es strukturiert beschriebene Daten und bringt zusätzlich die Beschreibung der Daten mit den Metatags auch noch selbst mit.

Aus dem Strukturaufbau des Dokumentes lässt sich der dazugehörige Szenengraph generieren. Die Portabilität ist ebenfalls gewährleistet, da XML Dateien in Unicode gespeichert sind.

Ganz so einfach kann man es sich nicht machen. Es fehlen Funktionen wie strukturierte Suche, denn bei einer Volltextindizierung würden Metatags genauso behandelt wie Daten und wären so nicht mehr von einander unterscheidbar. Auch gibt es in XML keine definierten Datentypen für z.B. Datum, Zeit oder Währung. Operationen sind somit nur sehr eingeschränkt möglich. Die Speicherung von reinen textbasierten Dokumenten ist nicht sehr effektiv und sehr speicherhungrig.

Es zeigen sich also beim näheren Hinsehen einige Nachteile. Zur Behebung dieser kann man die XML Dokumente als Datenbasis (Content Base) nehmen und darauf zur Verwaltung ein XML fähiges DBMS aufbauen.

Dieses XML fähige DBMS muss in der Lage sein, XML Dokumente zu speichern und aus Schemas neue Daten zu generieren, über eine Abfragesprache verfügen und eine Schnittstelle zu anderen Programmen mitbringen.

Die Antworten auf diese Anforderungen heißen XML, DTD, XQL/XPath und DOM.

XML als Dokumentenformat ist schon bekannt, ebenso was eine DTD ist. XQL (XML Query Language) ist eine Abfragesprache für XML Dokumente. XQL ist völlig anders in Syntax und Verwendung als SQL. Mit XQL wird das Dokument über seine Knotenstruktur angesprochen und nicht über seine physikalische Struk-

tur. XQL ist eine Erweiterung zu XSL und unterstützt Knotenidentifizierung, Boolesche Logik, Filter und Indizierung in so genannte ‚Collections'.
XPath selbst ist keine direkte Abfragesprache, es unterstützt ein Datenmodell, das es erlaubt, Teile eines XML Dokumentes anzusprechen.

Das Document Object Model (DOM) wurde entwickelt, um mithilfe einer objektorientierten Programmiersprache wie beispielsweise C++ oder Java auf Elemente eines Dokuments zugreifen zu können. Es schafft eine einheitliche Schnittstelle (API) für den Zugriff auf die Elemente eines Dokumentes. Es ist vorgesehen, auch Objekte zu unterstützen, die auf DTDs bezogen sind. Die entsprechenden Definitionen sind jeweils in der Interface Definition Language (IDL) geschrieben. Definitionen sind hier jeweils als Interface realisiert.

4.6 Mapping von XML Dokumenten in ein DBMS

Mit Mapping bezeichnet man die Transformation mittels bestimmter Vorschriften von einem Datenmodell in ein anderes. Dies kann von XML Dokumenten in ein relationales oder objektrelationales Datenbanksystem geschehen oder in umgekehrter Weise, je nachdem in welchen Format die Daten vorliegen und welches Format für die weitere Verarbeitung gewünscht wird.
Bestimmte Voraussetzungen sind in diesem Fall schon gegeben, so haben wir es bei den 3D Szenen grundsätzlich mit tiefverschachtelten XML Dokumenten zu tun.
Das Mapping richtet sich u.a. auch danach, ob die Informationen daten-spezifisch oder dokument-spezifisch vorliegen. Im ersten Fall sind die Daten durch einen hohen Strukturiertheitsgrad und stark verteilte Informationseinheiten gekennzeichnet. Dies kann beim Ansehen des Quelltextes einer VRML97 Szene bzw. ihrer XML kompatiblen X3D Variante nur bestätigt werden(siehe Anhang 1).
Dokument-spezifische Daten benötigen das Dokument als Gesamtheit, da sonst Informationen beim Mapping verloren gehen könnten. Im aktuellen Anwendungsfall wird das gesamte Dokument benötigt, um den Szenengraphen exakt rekonstruieren zu können. Ebenso wichtig ist die Erhaltung der Metadaten. Es ist zu sehen, dass die hier verwendeten 3D-Szenen nicht korrekt nach Definition in genau eine der aufgeführten Klassen von XML Dokumenten passen. Je nachdem, welches Verfahren zum Einsatz kommt, müsste immer mit Einschränkungen gerechnet werden.

Mappingverfahren unterscheiden sich in uni- und bidirektionale Varianten. Ein Verfahren welches nur in eine Richtung angewendet wird, ist das *Template Driven Mapping*. Aus einer Datenbank heraus werden XML konforme Dokumente erzeugt.

Unter dem Begriff *Model Driven Mapping* fallen Methoden, die in beide Richtungen funktionieren. Dies sind das *Table Based Mapping* und das *objektrelationale Mapping* (Abbildung 10).

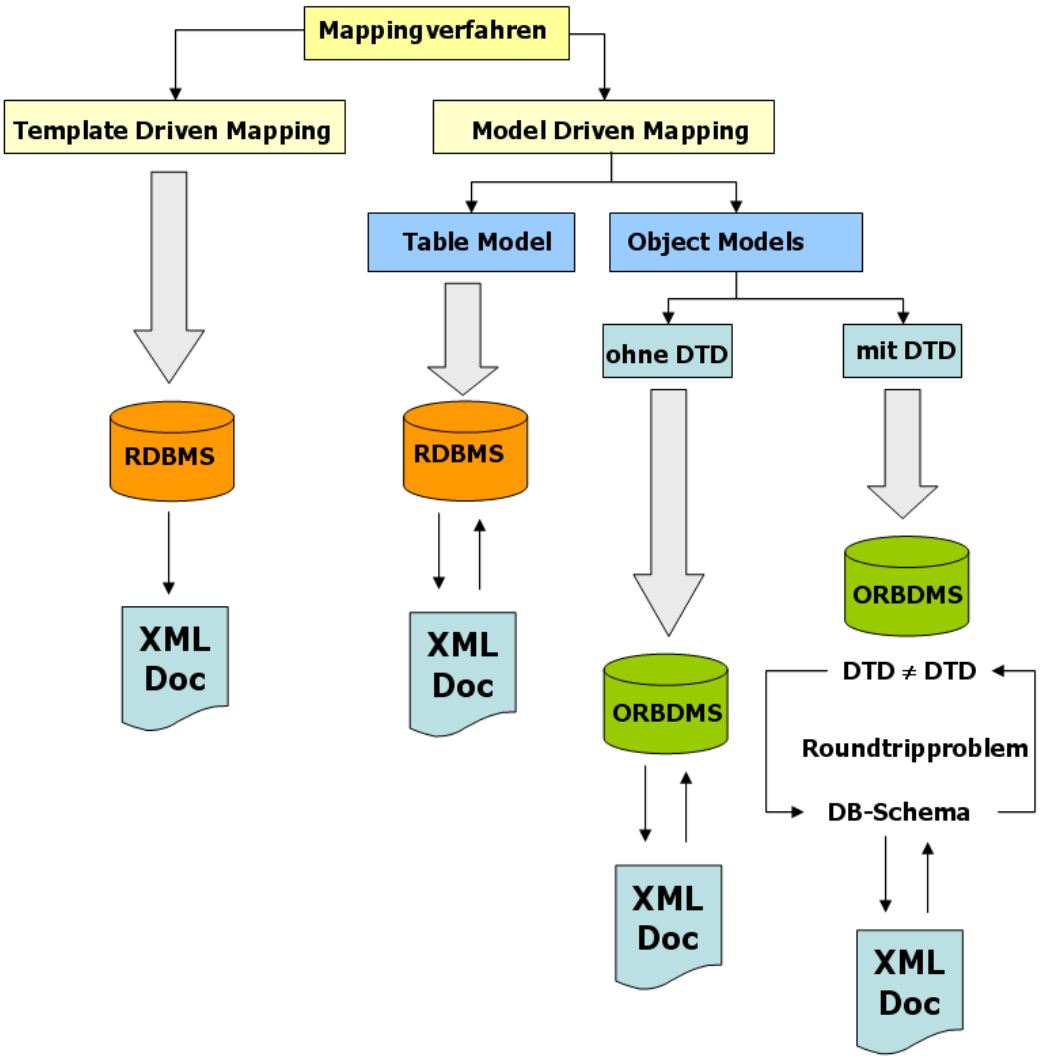

Abbildung 10 (4.1): Einordnung von Mappingverfahren

4.6.1 Template Driven Mapping

Das unidirektionale Template Driven Mapping erlaubt es, aus einer DB heraus mittels geeigneter Schablonen (Templates) XML Dateien zu erzeugen.
Als Technik kommt hier häufig XSL zum Einsatz, mit der dann aus einer relationalen Datenbank heraus dynamisch strukturierte Webseiten erzeugt werden können.
In den Templates sind Datenbankanweisungen in eigenen XML Tags eingebettet, welche nach der Ausführung XML konforme Daten beinhalten.
So können sehr einfach und flexibel Informationen aus SQL Datenbanken in XML Dokumente eingefügt werden. Die Templates sind leicht editierbar und lassen sich über Style Sheets anpassen.

4.6.2 Model Driven Mappings

Model Driven Mappings sind bidirektionale Mappingverfahren. Beim Model Driven Mappings muss man Flexibilitätseinbußen zu Gunsten eines besseren Modells in Kauf nehmen. Es existiert ein festes Modell für die Struktur der Daten des XML Dokumentes in die Datenbank und von der Datenbank zurück.

4.6.3 Table Based Mapping

Das Table Based Mapping ist eine Variante, die eher auf die Eigenschaften der Datenbank zugeschnitten ist. Sie ist zwar unflexibel, aber sie bildet sehr einfach und intuitiv XML Dokumente bzw. Tabellen ab. Das Lesen aus einer relationalen Datenbank ist momentan wahrscheinlich noch der häufigste Anwendungsfall in Zusammenhang mit XML Dokumenten. Optimal anwendbar ist das Table Based Mapping bei Dokumenten mit flacher Struktur und gleichförmigen Inhalten, wie z.B. Bestell- oder Buchungssysteme, die immer nach dem selben Schema ablaufen und sehr ähnliche Daten transportieren. XML dient hier meist als unabhängiges Zwischenformat für relationale Strukturen.
Enthält das Dokument unvollständige Datensätze, d.h. sind nicht alle Spalten einer Tabelle ausgefüllt, entspricht dies einer besondern Sicht (*view*) auf das Dokument. Die Abbildung erfolgt auf Metaelemente, die Tabellen, Spalten und Views entsprechen. Die Zuordnung der Elemente vom Dokument zur Datenbank erfolgt über ihre originären Namen.
Diese Abbildungsform ermöglicht einen schnellen effektiven Datenaustausch zwischen relationalen Datenmodellen und ist leicht implementierbar.
Nicht geeignet ist es hingegen für tiefverschachtelte XML Dokumente und für komplexe Datenmodelle.

4.6.4 Object-Relational Mapping

Über Table Based Mapping wird die allgemeine Struktur von Datenbanken und XML Dokumenten nachgebildet. Dagegen erfolgt beim objekt-relationalen Mapping eine Modellierung der Daten. Die modellierten Daten werden auf Objekte abgebildet und diese Objekte in einer Baumstruktur angeordnet. So entsteht ein datenspezifischer Objektbaum, in dem Elemente Objekte darstellen und Attribute (bzw. Subelemente) die Eigenschaften dieser Objekte repräsentieren. Über diese Organisation der Daten wird eine direkte Abbildung in ein OODBMS möglich. Soll eine relationale DB verwendet werden, so muss ein objekt-relationales Mapping durchgeführt werden. Eine eindeutige Eigenschaft eines Objektes setzt den Primärschlüssel oder es wird ein neues Attribut hierfür definiert.

Über diese Modellierung können alle Arten von XML Dokumenten verarbeitet werden ohne das es Einschränkungen aufgrund von der Verschachtelungstiefe gibt. Ebenso wichtig ist das die logische Struktur in beide Richtungen erhalten bleibt. Diese Art von Mapping wird bei vorhandenen relationalen Datenbanken[32] angewendet um XML Fähigkeiten einzuflechten. Man spricht bei solchen DB dann von *XML enabled Databases*.

Was allerdings vorausgesetzt wird ist das die Struktur und das relationale Datenbankschema zusammenpassen. Ist ein DB Schema gegeben kann es sein das nicht jede beliebige XML Datei nach dem Objektmodell gespeichert bzw. erzeugt werden kann. Hier ist eine Anpassung über DTDs möglich.

Ein Problem ist das so genannte Roundtripping. Es ist nicht möglich bei einer Konvertierung einer DTD in ein DB Schema und zurück exakt die gleiche DTD zu erhalten! Dies entsteht durch Einschränkungen auf beiden Seiten. In einer DTD können keine Datentypen definiert werden. Ebenso gibt es keine Wertebereichseinschränkungen von Elementen, wohl aber sind die Längen für Objekte in der DB definiert.

Im relationalen Modell gibt es keine Aggregierungen. Mittels dieser „Besteht-aus“ Beziehungen kann einem Objekt direkt mehrere Objekte zugewiesen werden.

Eine Erweiterung im Umgang mit DTD ist die *Inlining*[33] Technik. Hier wird eine DTD schrittweise vereinfacht und die Darstellung der DTD als Baumdiagramm bildet die Grundlage zur Übersetzung in ein relationales Schema. Es wird versucht verschachtelte Elemente von der Wurzel ausgehend in ein Relation abzubilden. XML Hierarchien werden in flache Relationen umgewandelt, wodurch nun Anfragen einfacher durchgeführt werden können. Ein Nachteil ist die erzeugte Redundanz. Da eine DTD selbst keine Wurzel hat, muss für jedes Element dieser Abbildung eine Relation erzeugt werden.

[32] Beispiel für klassische relationale DB die um XML Fähigkeiten erweitert wurden sind u.a. DB2, Oracle, Informix und MS SQL Server.

[33] Nicht zu verwechseln mit der Inlining Technik von VRML97 die es erlaubt, externe Szenen einzubinden (siehe Kapiel 4)

4.7 Speichern von XML Dokumenten in einem Nativen XML DBMS

Alle besprochenen Mappingverfahren haben Nachteile, die für den Aufbau eines Managementsystems für 3D-Szenen nicht akzeptabel sind.
Der Verlust von Metadaten und Zusatzinformationen bei allen Mappingverfahren ist genauso wenig annehmbar, wie eine Anpassung über ein manuelles Mapping. Da es schon eine DTD gibt, mit der X3D Szenen XML fähig gemacht werden können, erübrigt sich ein manuelles Mapping. Die X3D DTD mapped Knoten zu Unterelementen und Felder zu XML Attributen.
Das die angesprochenen Verfahren Schwierigkeiten mit unstrukturierten Daten haben, spielt in diesen Szenario keine Rolle. Die Szenen die benutzt werden sind klar strukturiert, da sie den Standard von X3D entsprechen.
Weitere Ausschlusskriterien betreffen die verwendeten Datenbanksysteme. RDBMS und Post-RDBMS können nicht eingesetzt werden da hier unter anderen auch durch die anwendbaren Mappingtechniken, kein Roundtripping gewährleistet werden. Es ist eine unverzichtbare Anforderungen, dass die Daten die in das DBMS gemappt werden, exakt zu rekonstruieren sind. Ebenso darf die Verschachtelungstiefe keine Rolle spielen.
Einzig das OODBMS könnte alle diese Anforderungen erfüllen, wenn das Roundtrippingproblem gelöst werden könnte. Aber schon bei der nächsten Forderung kann das OODBMS kein Lösung anbieten. Einige OODBMS bieten Security Richtlinien für ihre Systeme an. Bei dem untersuchten OODBMS „ozone“[34] z.B. findet man ein dokumentenbasiertes Sicherheitssystem. Damit können Rechte vergeben werden, aber nur für ein komplettes Dokument. Auch dies entspricht nicht den Anforderungen. Da in dem zu konzeptionierenden System mehrere Benutzer gleichzeitig an einer Szene arbeiten sollen, ist ein Sicherheitskonzept welches sich bis auf die Knotenebene konfigurieren lässt erforderlich.
Nach dem Verlauf dieser Überlegungen, stößt man zwangsläufig auf die nativen XML DBMS. Native XML DBMS zeichnen sich dadurch aus das sie

[34] http://www.ozone-db.org/

XML in seiner ursprünglichen (nativen) Form belassen speichern. Weiterhin verwenden sie DOM basierte Modelle zur Abbildungen der Knotenstrukturen oder nutzen Text indizierende Methoden für eine serialisierte Speicherung.
Native XML DBMS eignen sich für daten-spezifisch und dokumentspezifisch strukturierte XML-Dokumente (siehe 4.6 Mapping von XML Dokumenten in ein DBMS).

5. Auswahl des XML DBMS (Anforderungskatalog)

Um das für diese Arbeit am besten passende XML DBMS zu finden, werden alle Anforderungen die im Laufe der Diskussion aufgetaucht sind und auch solche die noch nicht erwähnt wurden hier aufgeführt werden.

5.1 Anforderungen

Es wird vorausgesetzt, dass die 3D-Szenen in dem X3D Format vorliegen und eine entsprechende DTD vorhanden ist.

5.1.1 Allgemeine Anforderungen an ein XML DBMS

- ✓ Es soll möglich sein, in der DB mittels einer XML Abfragesprache Suchanfragen zu formulieren.
- ✓ Die Suchergebnisse sollen als eine strukturierte Ausgabe erfolgen, um sie z.B. über XSLT weiterverarbeiten zu können.
- ✓ Die XML-Dokumente sollen über Volltext und/oder Metatags indizierbar sein.
- ✓ Eine Anbindung über das http Protokoll muss verfügbar sein.
- ✓ Das XML DBMS sollte unter einem freien Betriebssystem (Linux) verfügbar sein.

5.1.2 Spezielle Anforderungen an ein XML DBMS

- ✓ Die XML-Dokumente sollen in dem DBMS nativ gespeichert werden.
- ✓ Das Roundtripping beim Speichern und Auslesen von XML-Dokumenten wird erfüllt.
- ✓ Das Einbinden von XML-Dokumenten über DTDs muss möglich sein.
- ✓ Dem Benutzer muss das Bearbeiten und Speichern von Teilbäumen ermöglicht werden.
  - o Es muss eine strukturierte Zugriffskontrolle möglich sein.
  - o Rechte müssen über Benutzer und Benutzergruppen administrierbar sein.
  - o Sicherheitsmechanismen müssen auf Knotenebene anwendbar sein.
- ✓ Es sollen APIs für gängige Programmiersprachen (C++ oder Java) verfügbar sein.
- ✓ Die ausgelesenen XML Daten sollen über XSLT auch in andere Formate transformiert werden können.

5.2 Tamino XML Server

Mit dem obigen Anforderungskatalog ausgestattet, konnte nun der Markt nach passenden Produkten untersucht werden. Eine große Hilfe bei der Übersicht über den XML DB Markt war die Sammlung von aktuellen XML Produkten des XML Experten Ronald Bourret[35].

Bei dem Vergleich von den nativen XML DBMS kristallisierte sich ein Produkt besonders heraus, Tamino XML Server von der deutschen Firma Software AG[36] (Abbildung 11).

Vor allem die weitreichenden Möglichkeiten des Securitymanagements konnten überzeugten. Des weiteren konnte es alle Anforderungen des aufgestellten Katalogs erfüllen und bietet zusätzlich noch einige interessante Funktionen mehr. So ist das System dezentral über eine Webschnittstelle administrierbar.

Laut Aussage der Software AG ist der Datenbankserver „als erster und bisher einziger in der Lage, Daten direkt in nativem XML-Format zu speichern."

Tamino ist ein erweitertes Datenbanksystem, das Daten "Web-nah" speichert und via HTTP und XML verfügbar macht.

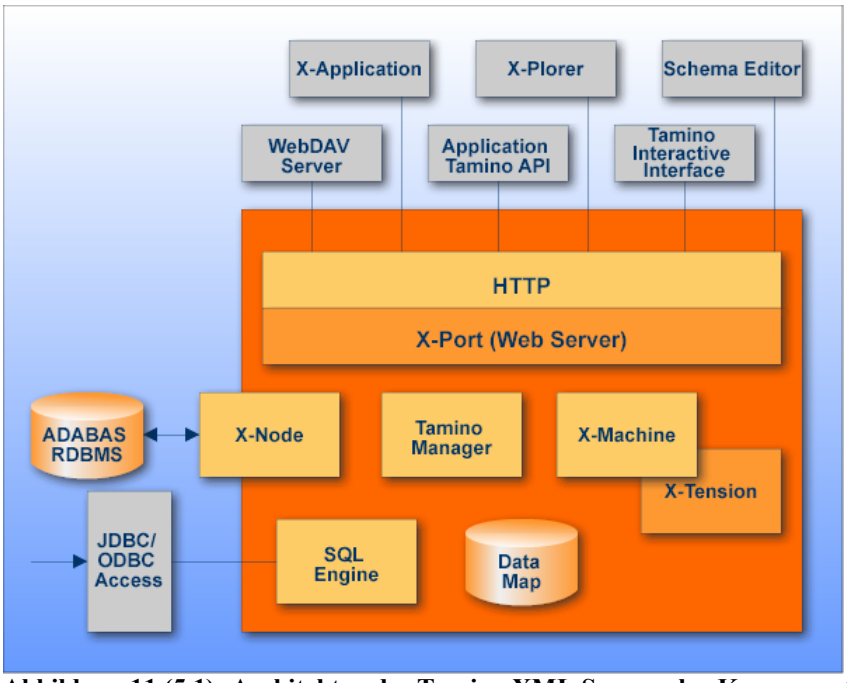

Abbildung 11 (5.1): Architektur des Tamino XML Server plus Komponenten

5.2.1 Hauptfunktionen von Tamino

[35] XML Database Products; http://www.rpbourret.com/xml/XMLDatabaseProds.htm
[36] Software AG; Tamino XML Server; http://www.softwareag.com/tamino/

- Die native Speicherung und Abfrage von XML Objekten mit *XML Store* und *X-Machine*.
- Eine Schnittstelle für externe Anwendungen und Datenquellen über *X-Node* (ODBC) .
- Interne Speicherung und Abfrage von SQL Daten mittels *SQL Store* und einer *SQL Engine*
- Über Datatracking werden Informationen wo Daten gespeichert und wie sie abgefragt werden können erzeugt (Data Map).
- Der Tamino Manager ermöglicht die zentrale Administration des Tamino Server über einen Internetbrowser.

5.2.2 Eigenschaften von Tamino

- Da Taminos XML verarbeitender Teil über das HTTP und TCP/IP Protokoll arbeitet, kann das DBMS leicht in bestehende Web Serverumgebungen eingebunden werden.
- Über dynamische Erweitungen kann ein Benutzer Elemente in ein XML-Dokument hinzufügen, ohne die zugrundeliegende Dokumentenstruktur ändern zu müssen.
- Über eine der Baumstruktur angepasste Abfragesprache (X-Query basierend auf X-Path), kann der Benutzer Anfragen an die Datenbank über das Internet stellen.
- Über die XML Schema Language werden die Fähigkeiten von DTDs erweitert.

5.3 Prinzipielle Arbeitsweise

Als erstes muss die Struktur der verwendeten Daten in einem Schema beschrieben werden. Dazu kann eine vorhandene DTD dienen. Diese wird dann automatisch in ein Tamino Schema konvertiert. Ein Schema kann auch mittels des graphischen Tamino Schema Editor erzeugt werden. Tamino basiert auf dem W3C Standard der XML Schema Language.

Dieses Schema wird auf die Tamino Data Map gemappt. Jetzt kann man Instanzen diese Schemas speichern. Über einen Browser können mittels X-Query Anfragen an XML Objekte gestellt werden und externe Anwendungen können die zurückgegebenen Daten weiter verarbeiten.

5.4 Das Sicherheitskonzept

Das Sicherheitskonzept von Tamino ist recht einfach zu überschauen und sehr effektiv. Dieses Konzept verspricht genau die Zugriffskontrollen, die für einen sinnvollen Mehrbenutzereinsatz notwendig sind.
Ein Benutzer (User) gehört einer Gruppe von Anwendern (Group) an, der in einer Liste die genauen Zugriffsberechtigungen beschrieben werden. In dieser Access *Control List* (ACL) stehen die einzelnen *Access Control Elements* (ACE) mit den dazugehörigen Rechten und Pfadangaben.

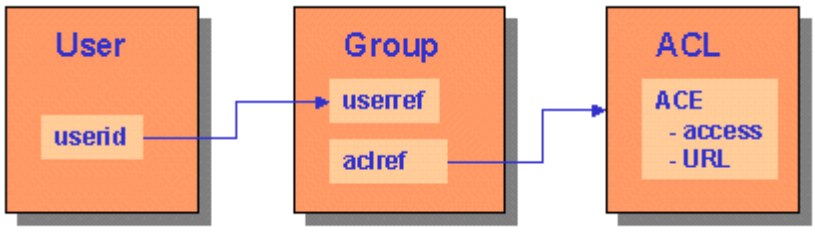

Abbildung 12 (5.2): Sicherheitskonzept von Tamino

Trotz der offensichtlichen Einfachheit dieses Systems lassen sich damit komplexe Regeln für geschlossene Benutzergruppen definieren. Ein Benutzer darf auch mehreren Gruppen angehören.

5.5 Die interne Verarbeitung (X-Machine)

Die sogenannte X-Machine (Abbildung 13) ist der Hauptbestandteil von Tamino. Die Grundfunktionen sind, XML Objekte nativ zu speichern und zu empfangen. Dies funktioniert nach den Schemas, die in der DB erzeugt oder geladen wurden. Tamino unterstützt ebenfalls andere Datenbanksysteme via ODBC und baut auf existierenden Web Standards auf.
In den Tests wurde als Webengine der Apache Webserver eingesetzt.

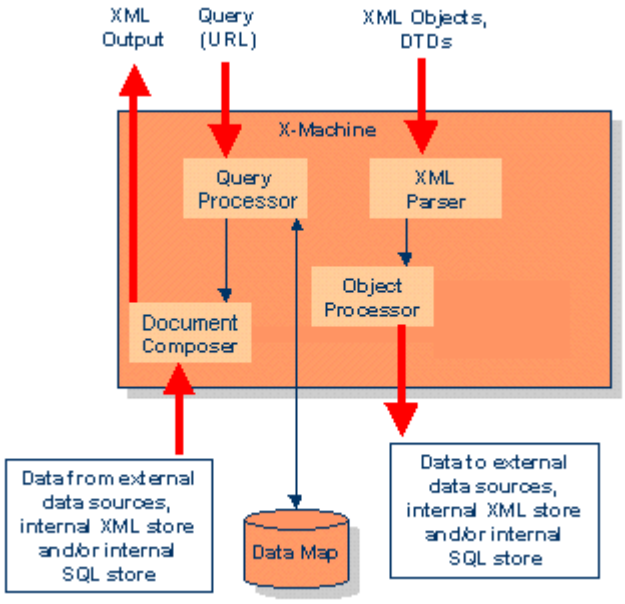

Abbildung 13 (5.3): The X-Machine

Der interne *XML Parser* überprüft den Syntax der Schemas und stellt sicher, dass die geladenen XML Objekte dem Prinzip der Wohlgeformtheit entsprechen. Der *Object Prozessor* wird benutzt, wenn Daten in Tamino abgespeichert werden. Die XML Instanzen des Schemas werden überprüft, ob sie den logischen Aufbau des Schemas entsprechen. Mit der Abfragesprache X-Query können in Tamino über den *Query Processor* Abfragen über das Schema gestellt werden.

Der Document Composer setzt die XML Informationen zusammen. Über die Speicher –und Abfrageregeln die in der Data Map definiert sind, konstruiert der Document Composer die Objekte und gibt ein XML-Dokument zurück.

6. Zusammenfassung

Im Laufe der Untersuchung wurde festgestellt und aufgezeigt, welche der aktuellen DBMS und Mappingverfahren für die Zwecke einer Administration und dem Authoring von audiovisuellen Szenen geeignet sind. So kann festgehalten werden, dass momentan nur native XML DBMS in Lage sind, alle Bedürfnisse abzudecken.

Es liegt ein lauffähiges System auf Basis von Windows 2000 und der Server Version von Tamino XML Server vor. Vorhandene VRML97 Szenen können, nach der Konvertierung in XML kompatible X3D Szenen, als Datengrundlage in die DB geladen werden. Über eine DTD, die mittels des Schema Editors in ein Tamino Schema umgewandelt wird, können *‚Collections'* und darin Instanzen von XML-Dokumenten abgelegt werde. Mehrere *Collections* können problemlos nebeneinander verwaltet werden.

Eine beliebige Abfrage an dieses System über den Browser ist ebenso möglich, wie der benutzerspezifische Zugriff nach definierten Regeln. Das Ergebnis der Abfrage ist XML kompatibler Code, der beliebig weiterverarbeitet werden kann.

Besonderer Dank gilt an dieser Stelle der Software AG, die es hier zum Zwecke der Forschung und Evaluierung ermöglichte, mit einer Original Tamino XML Server Lizenz zu arbeiten[37].
Es wurden der TU Ilmenau die Serverversionen für Windows 2000 und Linux kostenlos für einen Zeitraum von 8 Wochen überlassen. Diese vorübergehende Entwicklerlizenz, kann nach Absprache auch verlängert werden, um weitere universitäre Projekte durchzuführen.

Um eine Vorstellung davon zu haben, was mit diesem System machbar ist, sollte an dieser Stelle überlegt werden, an welchen Platz der Produktionskette im virtuellen Studio solch ein DBMS einsetzbar ist.

[37] Eine normale Kaufversion des Tamino XML Servers kostet ca. 40.000€.

Denkbar wäre eine Integration in das Authoring von audiovisuellen Szenen, mit einem nativen XML DBMS als Datengrundlage. Über die Möglichkeiten der Zugriffsbeschränkungen, kann die anwenderspezifische Bearbeitung der Szenen kontrolliert und administriert werden. Der Autor der 3D-Szene muss sich selbst keine Gedanken um die Speicherung und die Suche machen.

Über die Erweiterung mit XSLT ist es außerdem möglich, Abfrageergebnisse in andere Formate wie HTML oder VRML zu konvertieren. Dies Konvertierung hängt eng mit der Anbindung von verschiedenartigen Clients zusammen (3.2.3 Rücktransformation von X3D mittels XSLT).
Durch die Anbindung von APIs für C++ und Java, wäre es denkbar ein Tool zu entwickeln, das Änderungen an 3D Szenen inkrementell in die Datenbank, unter Berücksichtigung von Benutzerrechten, abspeichert.

Zukünftig wünschenswert wäre die Einbindung von Versionsmanagement. Mit WebDAV[38] stehen hier erste Instrumente zur Einbindung in ein DBMS zur Verfügung.

Wie die Situation von aktuellen Standards wie X3D oder MPEG-7 zeigt, geht die Entwicklung eindeutig in Richtung XML basiertes Datenmanagement. Mit dem vorgestellten Tamino XML Server steht ein sehr leitungsstarkes, neuartiges und leicht integrierbares Allround XML Werkzeug zur Verfügung, um diese Aufgaben zu bewältigen.

Man kann mit Sicherheit sagen, dass sich die weitere Integration von Content Management Funktionen in die Bearbeitungskette des virtuellen Studios mit diesem Werkzeug vereinfachen lässt. Es wäre wünschenswert Folgeprojekte zu initiieren, die die konkrete Implementierung zum Inhalt haben.

[38] http://www.ietf.org/rfc/rfc2518.txt

7. Quellenverzeichnis

Bücher:

Apache Web-Server; Eilebrecht; MITP; 2000; ISBN 3-8266-06112-4

Content Management mit XML; Rothfuss/Ried; Springer; 2001; ISBN 3-540-66594-3

Core Web3D; Walsh/Sevenier; 2001; ISBN 0-13-085728-9

Distributed Virtual Worlds; Diehl; Springer; 2001; ISBN 3-540-67624-4

MySQL & mSQL; Reese/King; O'Reilly; 2000; ISBN 3-89721-163-7

XML kompakt; Thomas Michel; Hanser, 1999; ISBN 3-446-21302-3

Artikel:

c't Ausgabe 6/2000, Seite 244 ff XSL im Detail,

c't Ausgabe 6/2000, Seite 255 ff XML mit Serviervorschlag

c't Ausgabe 2/2002, Seite 172 ff Report: Strukturierte Daten

c't Ausgabe 2/2002, Seite 176 ff Report: XML-Software

XML Bibliothek; Rainer Krause;2001; Von relational über NF2 zu SQL3 und XML – Datenstrukturen im Wandel

Spezifikationen:

MPEG Whitepapers

MPEG-4 Applications (2724)

MPEG-4 Content Based (2711)

MPEG-4 ISO (2201)

MPEG-4 Overview (2725)

MPEG-4 Requirements (2723)

MPEG-4 Systems (2201)

MPEG-4 Visuals (2502)

MPEG-7 Applications (N3934)
MPEG-7 Context, Objectives and Technical Roadmap (2861)
MPEG-7 Interoperability, Conformance testing and Profiling (4039)
MPEG-7 Introduction to MPEG 7 (N4325)
MPEG-7 Overview (N4031)
MPEG-7 Projects and Demos (N4034)
MPEG-7 Requirements (N4317)

Dissertationen und Diplomarbeiten:

Speicherung von XML-Dokumenten in objekt-relationalen Datenbanken; Jens Timm; Universität Rostock: Informatik; 1999

Internetverweise:

MPEG:

Die Offizielle MPEG Homepage
http://www.cselt.it/mpeg
MPEG Moving Picture Expert Group FAQ
http://www.crs4.it/~luigi/MPEG/mpegfaq.html
MPEG-7 Overview
http://www.cselt.it/mpeg/standards/mpeg-7/mpeg_7.htm
MPEG-7 Context & Objectives
http://www.cselt.it/mpeg/documents/mpeg-7_koenen/ppframe.htm
MPEG-7 - A scenario
http://www.cselt.it/mpeg/documents/mpeg-7_chiariglione/ppframe.htm
MPEG-7 Homepage
http://www.darmstadt.gmd.de/mobile/MPEG7/

DBMS:

Tamino XML Server - Home page

http://www.softwareag.com/tamino/

AmbySoft Inc. Mapping Objects to Relational Databases White Paper

http://www.ambysoft.com/mappingObjects.html

The Ozone DB Project

http://www.ozone-db.org/index.html

Object-Relational Mapping Articles

http://www.object relational.com/articles/object-relational_mapping.html

DoQ.de - Ihre Informationsquelle für DMS - Dokumentenmanagement aus dem Internet- und Entscheiderbereich

http://www.doq.de/

XML:

W3C Recommendations (http://www.w3c.org)

Associating Style Sheets with XML documents

http://www.w3.org/TR/xml-stylesheet

Document Object Model (DOM) Level 1

http://www.w3.org/TR/REC-DOM-Level-1

Extensible Markup Language (XML) 1.0 deutsch

http://www.mintert.com/xml/trans/REC-xml-19980210-de.html

Extensible Markup Language (XML) 1.0

http://www.w3.org/TR/REC-xml

Namespaces in XML

http://www.w3.org/TR/REC-xml-names

Resource Description Framework

http://www.w3.org/TR/REC-rdf-syntax

Synchronized Multimedia Integration Language (SMIL)

http://www.w3.org/TR/REC-smil

W3C Working Drafts

XML Path Language (XPath) Version 1.0

http://www.w3.org/TR/xpath

XSL Transformations (XSLT) Version 1.0

http://www.w3.org/TR/xslt

XML Query Language (XQL)

http://www.w3.org/TandS/QL/QL98/pp/xql.html

XML and Databases; Stand 11/2001

http://www.rpbourret.com/xml/XMLAndDatabases.htm;

Datenbanken und XML; Oliver Klein; Stand 02/2001;

http://www-is.informatik.uni-oldenburg.de/~grawund/Lehre/Seminare/WWD_2000_2001/Texte/

Daniela Florescu und Donald Kossmann; 09/1999;
Storing and Querying XML Data using an RDBMS. *Bulletin of the IEEE Computer Society Technical Commitee on Data Engineering*, 22(3):27–34,
http://www.research.microsoft.com/research/db/debull/99sept/issue.htm

Ronald Bourret; XML Database Products; 01/2002

http://www.rpbourret.com/xml/XMLDatabaseProds.htm

W3Schools Online Web Tutorials

http://www.w3schools.com/

VRML:

VRML97 Specification, ISO-IEC 14772-11997,
http://www.web3d.org/Specifications/VRML97/

3DSite VRML
http://www.3dsite.com/n/sites/3dsite/cgi/VRML-index.html

VRML Course References
http://www.stl.nps.navy.mil/~brutzman/vrml/#Course

vrmlbase.de - Das Virtual Reality Portal

http://www.vrmlbase.de/index.php

X3D:

Home of Web3D

http://www.web3d.org

X3D (Extensible 3D) FAQ; Stand 02/2002;

http://www.web3d.org/Taskgroups/x3d/faq

X3D International Standard ISO/IEC xxxxx:200x; Stand 10/2001

http://www.web3d.org/TaskGroups/x3d/specification-001october/main.html

X3D Edit (IBM Xeena v1.2E + X3D Extension);

http://www.web3D.org/TaskGroups/x3d/translation/X3D-Edit-sdk.html

8. Verzeichnis der verwendeten Abkürzungen

3GPP	Third Generation Partnership Project
A/V	Audio und Video
AAC-LD	Low Delay Advanced Audio Coding
ACE	Access *Control Elements*
ACL	Access *Control List*
ADT	abstrakte Datentypen
API	Application Programming Interface
ATM	Asynchronous Transfer Mode
BIFS	BInary Format for Scenes
CDL	Constaint Definition Language
CGM	Computer Graphics Metafile
CM	Content Management
CMS	Content Management System
CSS	Cascading Style Sheets
DAB	Digital Audio Broadcasting
DAG	Direkter Azyklischer Graph
DAI	Delivery Multimedia Interchange Format Application Interface
DB	Datenbank
DBMS	Datenbank Management System
DCT	Diskrete Cosinus Transformation
DDL	Description Definition Language
DML	Data Manipulation Language
DOM	Document Objekt Model
DS	Description Schemes
DTD	Document Type Defintion
ES Descriptor	Elementary Stream Descriptor
FO	Formatting Objects
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocoll
IAVAS	Interaktive Audiovisuelle Anwendungssysteme
IDL	Interface Definition Language
IPR	Intellectual Property Rights
ISO	International Standard Organisation
MDS	MPEG Multimedia Description Scheme
MF	Multiple-value Field
MP3	MPEG-1 Audio Layer-3
MPEG	Moving Pictures Expert Group
NF2	Non-First-Normal-Form-Systeme
OCI	Object Content Information
OD	Object Description Framework

OO	kurz für objektorientiert
OODBMS	objektorientierten DBMS
ORDBMS	objektrelationales DBMS
QT	Quicktime
RDBMS	Relationales Datenbank Management System
RDF	Resource Description Framework
SAX	Simple API for XML
SF	Single-value Field
SGML	Standard Generalized Markup Language
SMIL	Synchronized Multimedia Integration Language
SQL	Structured Query Language
SVG	Scalable Vector Graphics
Tamino	Transaction Architecture for the Management of INternet Objects
TCP/IP	Transport Control Protocol/Internet Protocol
TTS	Text To Speech
UMTS	Universal Mobile Telecommunications Systems
VRML	Virtual Reality Markup Language
WebDAV	Web Document Authoring and Versioning
X3D	Extensible 3D Format
XML	Extensible Markup Language
XQL	XML Query Language
XSL	Extensible Style Sheets Language
XSLT	Extensible Style Sheets Language Transformations

Anhang

1. Quelltext der Beispielszene aus Abb. 5

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE X3D PUBLIC "http://www.web3D.org/TaskGroups/x3d/translation/x3d-compact.dtd"
                                "file:///C:/www.web3D.org/TaskGroups/x3d/translation/x3d-compact.dtd">
<X3D>
  <head>
    <meta content="*enter FileNameWithNoAbbreviations.xml here*" name="filename"/>
    <meta
      content="*enter description here, short-sentence summaries preferred*" name="description"/>
    <meta content="*enter name of original author here*" name="author"/>
    <meta content="*if manually translating VRML-to-X3D, enter name of person translating here*"
name="translator"/>
    <meta content="*enter date of initial version here*" name="created"/>
    <meta content="*enter date of latest revision here*" name="revised"/>
    <meta content="*enter version here*" name="version"/>
    <meta content="*enter reference citation or relative/online url here*" name="reference"/>
    <meta content="*enter additional url/bibliographic reference information here*" name="reference"/>
    <meta content="*enter copyright information here* Example:  Copyright (c) Web3D Consortium Inc. 2001"
name="copyright"/>
    <meta content="*enter drawing filename/url here*" name="drawing"/>
    <meta content="*enter image filename/url here*" name="image"/>
    <meta content="*enter movie filename/url here*" name="movie"/>
    <meta content="*enter photo filename/url here*" name="photo"/>
    <meta content="*enter keywords here*" name="keywords"/>
    <meta content="*enter online url address for this file here*" name="url"/>
    <meta content="X3D-Edit, http://www.web3D.org/TaskGroups/x3d/translation/README.X3D-Edit.html"
name="generator"/>
<!--Additional authoring resources for  meta-tags:
      http://www.w3.org/TR/html4/struct/global.html#h-7.4.4
      http://dublincore.org/documents/dces
      http://vancouver-webpages.com/META
      http://vancouver-webpages.com/META/about-mk-metas2.html
      http://www.statelib.wa.gov:80/info_rscrs/dbs_tools/find-it/metadesc.htm
      Additional authoring resources for language codes:
      ftp://ftp.isi.edu/in-notes/bcp/bcp47.txt [IETF RFC3066/BCP47]
      http://www.loc.gov/standards/iso639-2/langhome.html
      http://www.iana.org/numbers.html#L Additional authoring resources
      for country names:  http://www.din.de/gremien/nas/nabd/iso3166ma
-->
</head>
  <Scene>
    <NavigationInfo type="EXAMINE ANY"/><!--Scene graph nodes are added
      here --><Background backUrl="wandseite.jpg"
      bottomUrl="wandunten.jpg" frontUrl="wandseite.jpg"
      leftUrl="wandseite.jpg" rightUrl="wandseite.jpg" topUrl="wandoben.jpg"/>
    <Transform center="0 0 0">
      <Shape>
        <Cylinder bottom="true" height="2" radius="1" side="true" top="true"/>
        <Appearance>
          <Material ambientIntensity="0.2" diffuseColor="0.8 0.8 1.0"
            emissiveColor="0 0 0" shininess="0.2" specularColor="0 0 0" transparency="0"/>
          <ImageTexture url="logo9dorf.jpg"/>
        </Appearance>
      </Shape>
    </Transform>
    <Transform center="0 0 0" translation="0 4 0">
      <Shape>
        <Cone bottom="true" bottomRadius="1" height="4" side="true"/>
        <Appearance>
          <Material ambientIntensity="0.2" diffuseColor="1.0 0.8 0.8"
            emissiveColor="0 0 0" shininess="0.2" specularColor="0 0 0" transparency="0"/>
        </Appearance>
      </Shape>
    </Transform>
  </Scene>
</X3D>
```

Abbildungsverzeichnis

Abbildung 11, 12, 13 entnommen aus der Tamino Documentation Version 3.1.1

Erklärung

Hiermit erkläre ich, dass ich diese Arbeit selbstständig durchgeführt und abgefasst habe. Quellen, Literatur und Hilfsmittel, die von mir benutzt wurden, sind als solche gekennzeichnet.

Unterschrift

Ilmenau, den 18. Feb. 2002

Thesen

I. **Traditionelle DBMS sind nicht geeignet, komplexe audiovisuelle Inhalte zu speichern.**
II. **Nur native XML DBMS können audiovisuelle Inhalte optimal verwalten.**
III. **Native XML DBMS sind roundtrip-fähig.**
IV. **Um XML-Dokumente strukturiert in Collections speichern zu können, muss eine entsprechende DTD vorhanden sein.**
V. **Zugriffsbeschränkungen müssen auf Knotenebene realisierbar sein.**
VI. **Versionsmanagement von Szenen ist ein Feature, welches integrierbar sein sollte.**