3D-Informationssysteme zur Visualisierung historischer Stadtlandschaften am Beispiel vom Berlin des 18.Jahrhunderts

Diplomarbeit

zur Erlangung des akademischen Grades
Diplom-Informatiker

an der
Fachhochschule für Technik und Wirtschaft Berlin

Fachbereich Wirtschaftswissenschaften II
Studiengang Angewandte Informatik

1. Betreuer: Prof. Dr. Thomas Jung
2. Betreuer: Prof. Dr. Elke Naumann

Eingereicht von Andreas Vogel
Berlin, 7.Mai 2002

Inhaltsverzeichnis

Abbildungsverzeichnis **4**
Tabellenverzeichnis **6**
0 Einleitung **7**
1 Informationssysteme **9**
1.1 Geografische Informationssysteme 9
1.2 Historische Informationssysteme 10
1.3 Vorstellung existierender Informationssysteme 11
1.3.1 Cyber Chicago (USA) *11*
1.3.2 Virtual Museum an der MMU (Großbritannien) *12*
1.3.3 Mdina Heritage Management System (Malta) *14*
1.3.4 VR Seattle (USA) *15*
1.3.5 Vergleich der Systeme *16*
2 Multimedia-Technologien zur Implementierung von 3D-Informationssystemen **18**
2.1 Web3D-Consortium 18
2.2 Virtual Reality 18
2.3 VRML 19
2.3.1 Erstellen von VRML-Welten *23*
2.4 Quicktime VR 26
2.4.1 Erstellen von QTVR-Welten *28*
2.5 Vergleich VRML Quicktime VR 31
2.6 Fazit 31
2.7 VRML-Konzepte in Bezug auf das 3D-Informationssystem 32
3 Realisierung des 3D-Informationssystems **41**
3.1 Anforderungsanalyse 41
3.2 Modellieren der 3D-Modelle 41
3.2.1 Recherche nach Bildquellen *42*
3.2.2 Modellieren der VRML-Modelle *43*
3.3 Erstellen des Gesamtmodells 53
3.3.1 Nachbearbeiten der Modelle *55*
3.4 Benutzerschnittstelle 60
3.4.1 Integration in VRML und Einbettung in HTML *61*
3.4.2 Funktionen *62*
3.4.3 Head Up Display *73*
3.4.4 Erstellen der Testumgebung *75*

3.5 Anforderungskatalog für Erweiterungen 75
3.5.1 Vorgehensweise zum Einfügen neuer Modelle 77
4 Test und Bewertung 79
4.1 Allgemeines 79
4.2 Bedienung des Systems 79
4.3 Performance-Tests 80
4.4 Bewertung und Ausblick 81
5 Abkürzungsverzeichnis 83
6 Quellenverzeichnis 84
6.1 Literaturverzeichnis 84
6.2 Informationsquellen im Internet 85
7 Eigenständigkeitserklärung 87

Abbildungsverzeichnis

Abbildung 1 – Cyber Chicago (links: Überblick, rechts: Webcam) 12
Abbildung 2 – Tomb of Menna 13
Abbildung 3 – City of Kahun 14
Abbildung 4 – Mdina Heritage Managment System 15
Abbildung 5 – VR Seattle 16
Abbildung 6 – Kugel, Zylinder, Kegel, Quader (v.l.) 21
Abbildung 7 – Internet Explorer mit Cosmoplayer-Plugin 22
Abbildung 8 – Cosmoworlds, Spazz3D (v.l.) 23
Abbildung 9 – 3D Studio MAX R4, Maya 4.0.1 PLE (v.l.) 25
Abbildung 10 – Grundaufbau von zylindrischen QTVR-Panoramen 27
Abbildung 11 – Quicktime Player 27
Abbildung 12 – Panorama Stitcher, Objekt Maker (v.l.) 29
Abbildung 13 – PointSet, IndexedLineSet, IndexedFaceSet (v.l.) 33
Abbildung 14 – Extrusion, Elevation Grid (v.l.) 34
Abbildung 15 – Text 34
Abbildung 16 – Scene Graph 35
Abbildung 17 – Definition einer Kugel in VRML 35
Abbildung 18 – DEF/USE-Konstrukt 36
Abbildung 19 – PROTO-Definition 37
Abbildung 20 – LOD's in INLINE-Knoten 40
Abbildung 21 – 3DSMAX: Keyboard Entry 45
Abbildung 22 – 3DS MAX: Objekt nach Anwendung der Modifikatoren EditMesh und Bend 45
Abbildung 23 – 3DS MAX: Modifikator-Stapel (Objekt Rampe_Links) 46
Abbildung 24 – PSP: Color Balance, Skew (v.l.) 47
Abbildung 25 – PSP: Einstellen der Transparenzwerte 48
Abbildung 26 – PSP: Reduzieren von Auflösung und Größe 49
Abbildung 27 – 3DS MAX: Material Editor 50
Abbildung 28 – 3DS MAX: VRML-Helper, LOD, VRML-Exportfilter (v.l.) 52
Abbildung 29 – Grundstruktur des 3D-Informationssystems 53
Abbildung 30 – Ausschnitt aus dem Gesamtmodell (Unter den Linden) 54
Abbildung 31 – Polygon Reduction Editor 56
Abbildung 32 – Chisel für Windows 58
Abbildung 33 – LOD unter Verwendung von Inlines 59
Abbildung 34 – Definition eines INLINE-Knotens mit Bounding Box 59
Abbildung 35 – Beispiel für textbasierte Benutzerschnittstelle (gzip unter Unix) 60

Abbildung 36 – Beispiel für grafische Benutzerschnittstelle (Windows Explorer) 60
Abbildung 37 – Menüleiste mit Bedienelementen 63
Abbildung 38 – Definition eines OrientationInterpolator 63
Abbildung 39 – Animation der Menüleiste 64
Abbildung 40 – Struktur des Zeitleisten-Switch 65
Abbildung 41 – Zeitleisten-Script 66
Abbildung 42 – Zeitleiste: 18.Jh.-alter Stadtplan, 21.Jh.-neuer Stadtplan (v.l.) 66
Abbildung 43 – Routing der Messlatte 67
Abbildung 44 – Slider-Script 68
Abbildung 45 – Verschachtelung der PlaneSensoren 69
Abbildung 46 – Menüleiste mit aktivierter Messlatte 69
Abbildung 47 – Infoterminal-Script 70
Abbildung 48 – Infoterminal 71
Abbildung 49 – Anchor-Knoten 71
Abbildung 50 – Schematische Darstellung des Infoterminals 72
Abbildung 51 – Schema des Head Up Display 74
Abbildung 52 – Prototyp des HUD 74
Abbildung 53 – Ausschnitt aus dem Gesamtmodell (Nikolaikirche) 75
Abbildung 54 – vorgegebene Verzeichnisstruktur des Anforderungskatalogs 76

Tabellenverzeichnis

Tabelle 1 – Systemvergleich 17
Tabelle 2 – VRML-Modellierer 26
Tabelle 3 – 3D-Modellierer 26
Tabelle 4 – Systemanforderungen Quicktime VR 28
Tabelle 5 – Quicktime VR Authoring Studio, Stitcher 3 30
Tabelle 6 – Vergleich VRML / Quicktime VR 31
Tabelle 7 – Dateigrößenvergleich GIF, JPG, PNG, TIFF 49
Tabelle 8 – Kompressionsraten des ZIP-Algorithmus 56
Tabelle 9 – Polygonzahl vor und nach Bearbeitung mit Polygon Reduction Editor 57
Tabelle 10 – Ergebnisse mit Chisel 58
Tabelle 11 – Ladezeiten des 3D-Informationssystems 80

0 Einleitung

„Wer ein Problem definiert,
hat es schon halb gelöst.“
Sir Julian Huxley

Diese Diplomarbeit beschreibt die Entwicklung und Realisierung eines Informationssystems, dessen Zweck es ist, historische Stadtlandschaften in einer dreidimensionalen Umgebung darzustellen. Seit dem Start als explizites 2D-Medium vor ungefähr 10 Jahren hat sich das Internet neben seinem quantitativen Exponentialwachstum auch in seinen multimedialen Fähigkeiten enorm weiterentwickelt. Aufgrund der stetig steigenden Leistungsfähigkeit der Hardware und verbesserter Bandbreiten in Netzwerken entwickelte sich das Internet zunehmend zu einer Informationsquelle, die nicht mehr nur Texte und Bilder als Daten bietet, sondern auch multimediale Komponenten. Darunter fallen nicht nur Audio- und Videodaten, sondern insbesondere auch die Darstellung und Präsentation von Informationsinhalten mittels 3D-Technologien, die in dieser Arbeit näher erörtert werden. Damit kommen neben lokalen Datenträgern, wie z. B. der CD-ROM, auch Netzwerke als Übertragungsmedium eines 3D-Informationssystems in Betracht.

Mit dieser Arbeit soll ein solches 3D-Informationssystem entwickelt werden. Dabei hat es den Anspruch, Informationen auf eine, in diesem Bereich, neue Art und Weise zu präsentieren. Zusätzlich zur Informationsbeschaffung sollen dem Anwender erweiterte Möglichkeiten geboten werden, mit dem System zu arbeiten. Als Zielgruppe dieses Systems kommen drei unterschiedliche Anwenderkreise in Frage.

Der erste betrifft interessierte Laien, die sich mittels des Informationssystems ein Bild vergangener Zeiten machen wollen. Sie bekommen die Möglichkeit durch die dargestellte 3D-Welt zu wandern und Informationen zu den gezeigten Modellen zu erhalten. Zur besseren Anschauung werden ihnen Funktionen geboten, die den Wechsel zwischen verschiedenen Zeitepochen erlauben, um den Wandel der Zeiten in der Architektur zu verdeutlichen. Des Weiteren werden zu jedem Gebäude Informationstafeln angezeigt, auf denen der Anwender Bilder des dargestellten Gebäudes betrachten kann.

Der zweite Anwenderkreis sind professionelle Anwender. Dazu gehören unter anderem Historiker. Neben den Standardfunktionen des ersten Anwenderkreises, werden ihnen zusätzliche Werkzeuge zur Verfügung gestellt, mit deren Hilfe es möglich ist, an den dargestellten Modellen einfache Untersuchungen vorzunehmen. Sie können beispielsweise mittels einer virtuellen Messlatte die Maße der 3D-Modelle bestimmen. Die so gewonnenen Ergebnisse können mit historischen Aufzeichnungen abgeglichen werden, um die Authentizität des Informationssystems zu überprüfen.

Den dritten Anwenderkreis stellen die 3D-Modellierer, deren Aufgabe es ist, weitere Modelle, zum Beispiel Gebäude oder Infrastrukturen, zu erstellen und so das System zu erweitern. Um eine homogene Struktur des Gesamtsystems zu ermöglichen, wird ihnen ein Format vorgegeben werden, das eine einfache Integration der Modelle in das Gesamtsystem ermöglicht.

Die Arbeit ist folgendermaßen gegliedert: Das erste Kapitel widmet sich dem Begriff Informationssystem. Es werden unterschiedliche Ansätze für Informationssysteme exemplarisch erläutert, bereits existierende Systeme vorgestellt und unter verschiedenen Gesichtspunkten analysiert und bewertet. Kapitel 2 beschäftigt sich mit der Diskussion unterschiedlicher 3D-Technologien, die im Zusammenhang mit 3D-Informationssystemen relevant sind. Dabei werden die Grundkonzepte und entsprechende Softwarelösungen vorgestellt und kritisch betrachtet. Zu der für das Informationssystem verwendeten Technologie werden im Weiteren einige theoretische Grundlagen beleuchtet. Die eigentliche Entwicklungsarbeit wird in Kapitel 3 erläutert. Dieser Abschnitt beinhaltet Funktionalität und Implementation. Den Abschluss bildet Kapitel 4 mit einem Test und der Bewertung des entstandenen Informationssystems. Des Weiteren werden Tendenzen für die künftige Entwicklung solcher Informationssysteme und Möglichkeiten für weitere Ausbaustufen aufgezeigt.

1 Informationssysteme

In der Anfangsphase wurde das Internet überwiegend vom Militär, von Forschungseinrichtungen und von Universitäten genutzt. Haupteinsatzzweck war das verteilte Rechnen und der schnelle Nachrichtenaustausch. Mit Entwicklung des WWW wurde das Internet einer breiten Öffentlichkeit zugänglich gemacht und entwickelte sich in rasantem Tempo zu einem enormen Wissensspeicher. Da das Angebot kaum noch zu überblicken ist und selbst eine Suche nach spezifischen Informationen mittels Suchmaschinen oftmals ins Leere läuft, bedarf es Strukturen, um dem Anwender Informationen thematisch sortiert im Internet anzubieten. Eine Möglichkeit einer solchen Strukturierung sind Informationssysteme. Diese gestatten den gezielten Zugriff auf Daten zu einem bestimmten Themengebiet. Solche Systeme können nahezu für jeden Zweck erstellt werden. Im Folgenden werden die Konzepte von geografischen und historischen Informationssystemen näher erläutert. Dabei werden insbesondere die verschiedenen Anwendungsmöglichkeiten, unterschiedliche Darstellungsarten und dabei verwendete Techniken besprochen. Anschließend werden vier Systeme vorgestellt, die dem Anwender eine bequeme Möglichkeit bieten, auf umfangreiches Datenmaterial zu spezifischen Themengebieten zuzugreifen. Dabei werden insbesondere der Datengehalt, die verwendete Technik und die Benutzerführung näher betrachtet und analysiert.

1.1 Geografische Informationssysteme

Im Zusammenhang mit der Erstellung dreidimensionaler historischer Stadtlandschaften ist eine spezielle Form von Informationssystem zu nennen: geografische Informationssysteme (GIS). Sie dienen der Visualisierung geografischer Daten. Ursprünglich kommen diese Systeme aus dem Bereich der Vermessung. Sie bieten die Möglichkeit, kartografische Daten mit Informationen, die in einer Datenbank gespeichert sind, zu verknüpfen. Als kartografische Datenbasis kommt dabei jegliche Art von Kartenmaterial in Frage. Bei Verwendung von historischen Karten lassen sich auf dieser Basis entsprechende Stadtmodelle aufbauen. Informationen zu einzelnen Gebäuden des Stadtmodells können in einer angeschlossenen Datenbank gespeichert werden. Die Art der gespeicherten Zusatzinformationen wiederum hängt ganz vom Anwendungsgebiet des GIS ab. Bei Einsatz eines geografischen Informationssystems in der Geschichtsforschung haben beispielsweise das Datum der Errichtung oder Informationen zur Nutzung eines Gebäudes besondere Bedeutung. Hauptsächlich finden geografische Informationssysteme jedoch überall dort Verwendung, wo Sachdaten mit einer räumlichen Komponente verknüpft werden müssen. Als Beispiel seien hier Liegenschaftskataster, Verkehrsplanung oder auch Leitungsnetzdokumentation genannt. Als Technologie kommen aufgrund der hohen Informationsdichte häufig mächtige

Datenbanksysteme wie Oracle oder Sybase zum Einsatz. Als Benutzerschnittstellen eignen sich proprietäre Oberflächen oder auch eine Einbindung in HTML als Web-Frontend.

1.2 Historische Informationssysteme

Die Einsatzmöglichkeiten von Informationssystemen in der Geschichtswissenschaft sind außerordentlich vielfältig, werden jedoch bisher kaum genutzt. Derzeit werden historische Informationen eher auf traditionelle Weise vermittelt. In Geschichtsbüchern findet man reichhaltige Informationen in Form von Texten und Abbildungen. Museen bieten inzwischen mit multimedialen Anwendungen wie computergenerierten Filmen oder Informations-Terminals bereits mehr. Sogar für den Heimanwender erscheinen Multimedia-CD-ROMs mit geschichtlichen Inhalten, um ihm die Vergangenheit näher zu bringen. Der Informationsgehalt der genannten Medien ist in der Regel sehr gut, jedoch ist zumindest bei Büchern die Darbietung eher trocken und eintönig. Die Präsentationen in Museen und auf CD-ROM sind zwar modern gestaltet und sehr informativ, jedoch sind sie oft nur bedingt interaktiv, das heißt der Besucher kann nur betrachten und wenig interagieren. Oftmals ist er bei der Suche nach Informationen an einen bestimmten Ablauf gebunden.

Ein historisches 3D-Informationssystem setzt sich die dreidimensionale Visualisierung historischer Informationen als Schwerpunkt. Wie der Name bereits andeutet, werden solche Systeme eingesetzt, um geschichtliche Fakten und Informationen thematisch sortiert darzubieten. Dafür wird neben den bereits genannten Medien (Text, Bild, Video und Animation) zusätzlich die dreidimensionale Darstellung genutzt. Mittels geeigneter Technologien wird eine anschauliche Nachbildung der Vergangenheit erzeugt, welche vom Anwender erforscht werden kann. Ziel der Nachbildung können einzelne Artefakte, einzelne Gebäude oder ganze Stadtlandschaften sein. Durch die dreidimensionale Umgebung wird dem Anwender ein anschaulicher Eindruck vergangener Zeiten oder der damaligen Architektur vermittelt. Aktuelle 3D-Standards erlauben zusätzlich die Integration von interaktiven Elementen. Dem Anwender wird damit die Möglichkeit gegeben, Geschichte auf eine neue Art zu erleben. Im Gegensatz zu klassischen Medien wie Buch oder CD-ROM, kann der Benutzer in gewisser Hinsicht selbst bestimmen, auf welche Art und Weise er das System benutzen möchte. Abhängig von der verwendeten Technologie kann er sich absolut frei in der virtuellen Welt bewegen und verschiedene Aktionen ausführen. So lassen sich die dargestellten Objekte aus allen möglichen Perspektiven betrachten. Um einen ganz neuen Eindruck der Umgebung zu erlangen, besteht die Möglichkeit, über die Szenerie hinwegzufliegen. Verschiedene Funktionen erlauben dem Anwender, Gegenstände zu bewegen oder zu verändern. Er kann historische und Modelle der Gegenwart vergleichend gegenüberstellen. Ebenso können Animationen ausgelöst werden, um beispielsweise Veränderungen in der Geschichte zu visualisieren. Die zahlreichen Navigationsmodi erlauben im Zusammenspiel mit Animation die Durchführung geführter Touren (Guided

Tours) durch die Szenerie. Da der Informationsgehalt von Daten in Textform ungeschlagen ist, werden neben den dreidimensionalen Modellen Informationen auch in traditioneller Weise mit Text und Bildern geboten. Die Möglichkeiten der Interaktivität erlauben ein Gestalten des Systems für zahlreiche Anwendungszwecke. Als Benutzer kommen sowohl Privatanwender als auch Fachleute in Frage. Professionellen Historikern bietet ein 3D-Informationssystem Gelegenheit, einfache Untersuchungen an den Modellen vorzunehmen und mit Bildern der damaligen Zeit abzugleichen.

Zur Erstellung eines dreidimensionalen Informationssystems bieten sich unterschiedliche Technologien an. Die bekanntesten sind VRML und Quicktime VR. VRML ist eine Beschreibungssprache für dreidimensionale Welten, Quicktime VR ein von der Firma Apple entwickeltes Format, welches aus Einzelbildern Panoramen generiert. Vor- und Nachteile beider Technologien werden in Kapitel 2 näher erläutert.

1.3 Vorstellung existierender Informationssysteme

In diesem Abschnitt werden vier bereits existierende Informationssysteme vorgestellt. Bei der Auswahl wurde Wert auf die Verwendung unterschiedlicher Technologien gelegt. Damit wird ein guter Querschnitt der verfügbaren Systeme beschrieben. Im Anschluss werden die Vor- und Nachteile der verschiedenen Systeme vergleichend gegenübergestellt.

1.3.1 Cyber Chicago (USA)

Unter [6] findet sich ein 3D-Modell der amerikanischen Millionenstadt Chicago, das mit VRML 97 realisiert wurde. Der Benutzer hat auf der Startseite die Möglichkeit zwischen verschiedenen Varianten, die sich in ihrem Detailreichtum unterscheiden, zu wählen. Dies gibt dem Benutzer die Möglichkeit, sich das für seinen PC am besten geeignete Modell auszuwählen. So bietet die „XL-Version“ Anchor-Links, Texturen, Sound-Effekte, Animation (Sonne, Mond, Tageslicht), Rundflüge und Links zu Live Webcams. In der Version mit den geringsten Anforderungen, der „S-Version“, beschränkt sich die Darstellung auf einfache Geometrien und einige Sound-Effekte. Die Ladezeit für dieses Modell ist entsprechend kürzer, was langsamere Rechner nicht überfordert und auch dem Gelegenheitssurfer mit geringer Bandbreite bei der Datenübertragung entgegenkommt. Das Modell ist aus der Idee entstanden, einen zweidimensionalen Stadtplan in die dritte Dimension zu erweitern und eine virtuelle Realität zu erschaffen.

Als Ergebnis dieser Idee entstand ein sehr interessantes System. Die Stadt Chicago wurde mit unzähligen Gebäuden, Strassen, Flüssen und Parks nahezu komplett in eine dreidimensionale Welt umgesetzt. Der Benutzer hat die Möglichkeit, durch die Strassen zu wandern oder einen Rundflug zu wagen. Ein einfacher Weg zur Erkundung des Modells wird durch die vielen bereits vordefinierten Navigationspunkte möglich. Durch einen Mausklick wird der Benutzer von einem Punkt zum nächsten geführt. An einigen Stellen können über

Hyperlinks weitere Informationen abgerufen werden. So werden aktuelle Fotos von Chicago angeboten, um einen Vergleich zwischen der Realität und dem virtuellen Modell zu bekommen. Des Weiteren gibt es bei einigen Gebäuden weiterführende Verweise zu architektonischen Informationen. Auf Aussichtsplattformen von Hochhäusern gibt es virtuelle Pulte, auf denen Bilder einer Live-Kamera als Texturen dienen und so dem Benutzer einen realistischen Ausblick bieten (Abb. 1). Noch wirklichkeitsgetreuer wirkt der Spaziergang durch den Einsatz von Audiodateien. Mit Verkehrslärm, heulendem Wind und Vogelgezwitscher wird das Leben in einer Großstadt nachempfunden. Der Eindruck der Realität wird durch die Tag/Nacht-Animation noch verstärkt.

Abbildung 1 – Cyber Chicago (links: Überblick, rechts: Webcam)

Insgesamt gesehen ist Cyber Chicago ein sehr gelungenes VRML-System mit vielen interessanten Features. Wünschenswert wäre jedoch, mehr Informationen über die Stadt selbst oder einzelne Gebäude, Parks oder Flüsse anzubieten. Da das Modell neben dem Internet noch kommerziell auf CD-ROM vertrieben wird, ist fraglich, ob eine solche Erweiterung geplant ist, da die CD-ROM als Medium einer ständigen Aktualisierung eher entgegensteht.

1.3.2 Virtual Museum an der MMU (Großbritannien)

An der Manchester Metropolitan University (MMU) arbeitet man an einem Projekt, welches sich mit einem virtuellen Museum beschäftigt [7]. Ziel des Projektes ist es zu erforschen, inwieweit sich neue Medien wie beispielsweise das Internet oder virtuelle Umgebungen für den Einsatz in Museen nutzen lassen. Das Projekt besteht aus mehreren Teilen. Davon werden im Folgenden ‚Tomb Of Menna' (Grabmal von Menna) und die Stadt ‚Kahun' näher betrachtet.

Tomb of Menna ist die virtuelle Nachbildung eines alten ägyptischen Grabmals (Abb. 2). Das Modell ist recht einfach gehalten. Einfache Geometrien wurden mit Texturen versehen

und vermitteln dem Benutzer einen Eindruck vom Aussehen des Grabmals zum Zeitpunkt der Ausgrabung im Jahre 1916. Jede einzelne Wand lässt sich anklicken, wodurch der Benutzer Informationen zu den dargestellten Zeichnungen und Hieroglyphen erhält. Die Steuerung ist etwas benutzerunfreundlich gelöst. Nach jedem Klick auf eine der Wände gelangt man nur durch Betätigung des Zurück-Buttons des Web-Browsers auf die vorherige Seite. Da das Modell jedes Mal neu geladen wird, steht der Besucher immer wieder am Eingang des Grabmals und muss die Tour von vorn beginnen.

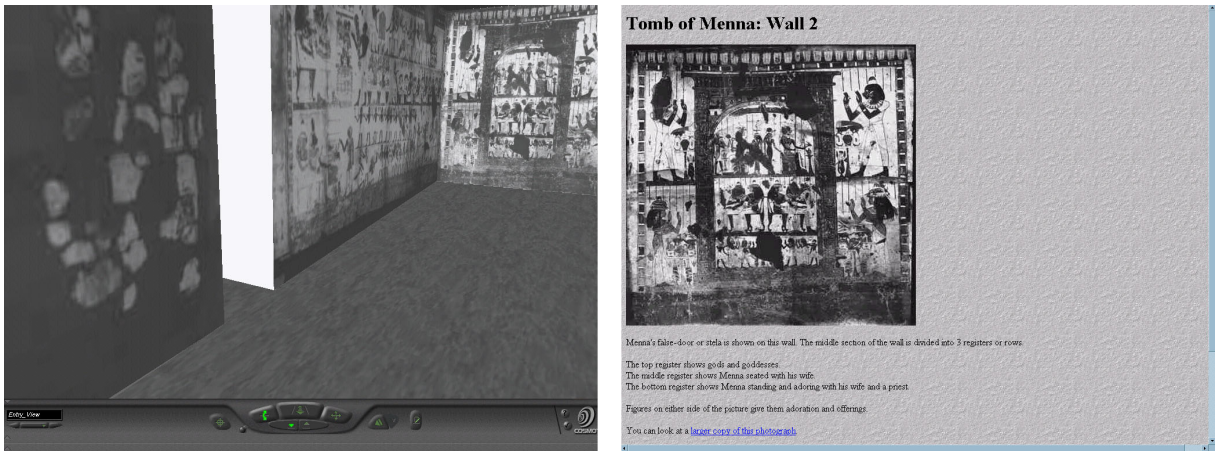

Abbildung 2 – Tomb of Menna

Im zweiten Teilprojekt wurde die Stadt Kahun modelliert. Es wurden verschiedene Prototypen für das Stadtmodell und einzelne Artefakte erstellt. Beim Stadtmodell wird die Auswahl zwischen einer zwei- und einer dreidimensionalen Variante angeboten. Die erste Variante basiert auf einem einfachen Grundriss, auf dem mittels einer Imagemap einzelne Gebäudegruppen angeklickt werden können, um weitere Informationen zu erhalten. Da lediglich eine Schwarz-Weiß-Grafik und Informationen in Textform übertragen werden müssen, ist diese Version des Informationssystems zwar sehr ressourcensparend, jedoch auch sehr spartanisch. Bei dem dreidimensionalen VRML-Modell kann man zwischen einem Überblick oder einem virtuellen Spaziergang wählen (Abb. 3). Der Überblick ist nur ein einfaches Modell, in dem alle Gebäude ohne Texturen und Links dargestellt sind. Der virtuelle Spaziergang war zum Zeitpunkt der Recherche leider noch nicht zugänglich, so dass nur drei Screenshots abgebildet waren. Diese versprechen jedoch eine reichhaltige Texturierung und detaillierte Darstellung der Szenerie. Ein kleineres Einzelmodell einer Arbeiterbehausung zeigt dem Benutzer dann aber doch noch einige Einzelheiten der Stadt. Im Bereich Artefakte wurde eine VRML-Szene entwickelt, die eine altertümliche Wasserschöpfmaschine darstellt. Ein zweites Modell stellt eine Art Brettspiel dar. Zu bemängeln ist das Fehlen der Möglichkeit, innerhalb des 3D-Modells Informationen über die

dargestellten Objekte abzufragen. Lediglich auf der Hauptseite des Projektes und den Seiten der einzelnen Teile werden Einzelheiten und Hintergrundinformationen angeboten.

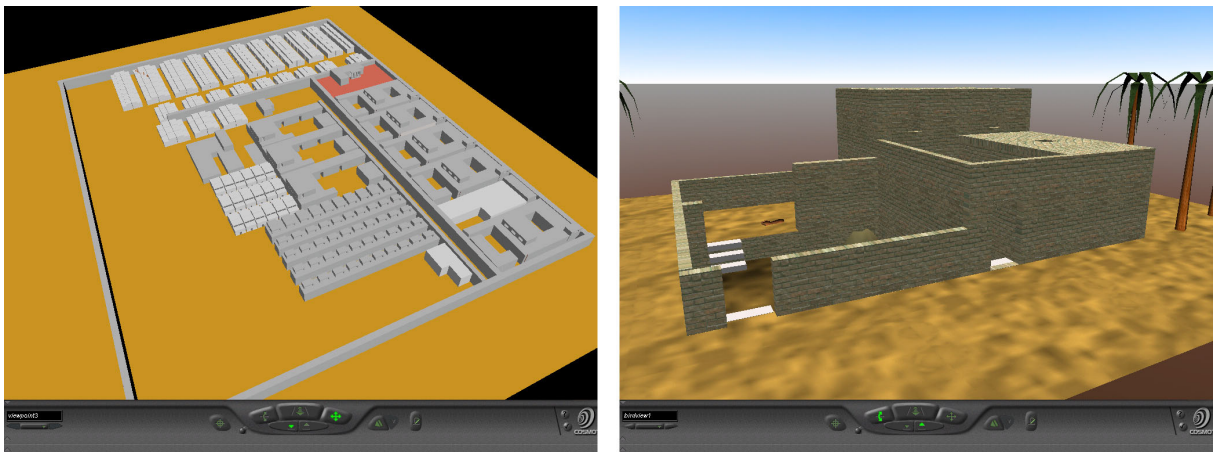

Abbildung 3 – City of Kahun

Das Projekt befindet sich noch in der Entwicklung, weshalb viele Details und Funktionen noch nicht implementiert sind. Der Ansatz ist jedoch vielversprechend und zeigt, dass eine Kombination von Internet und VRML als Medium durchaus zur Vermittlung historischer Informationen dienen kann.

1.3.3 Mdina Heritage Management System (Malta)

Die Planning Authority in Malta [8] verfolgt mit dem Heritage Management System einen interessanten Ansatz zur Entwicklung eines Informationssystems. Basierend auf GIS-Daten wird für die Stadt Mdina eine Datenbank aufgebaut, in der jede einzelne Strasse und jedes Gebäude verzeichnet sind. Zu jedem Eintrag lassen sich Zusatzinformationen abrufen. Zu Strassen sind der Name und eine Kategorie gespeichert, bei Gebäuden sind außerdem Informationen über Alter und Nutzung des Gebäudes sowie Bilder und allgemeine Informationen vorhanden. Zusätzlich gibt es noch eine Datenbank, in der sämtliche Flurstücke der Stadt verzeichnet sind. Verschiedene Farben entsprechen dabei unterschiedlichen Nutzungstypen (zivil, militärisch, Parks etc.). Der Umfang des Modells schließt im Augenblick nur die Stadt Mdina und einige wenige Gebäude außerhalb der Stadtgrenzen ein. Momentan ist zwar noch kein 3D-Modell der Stadt vorhanden, der Plan für die Umsetzung in VRML besteht jedoch bereits. Das Potenzial der Datenbank ist enorm, so dass sich mit Hilfe der gespeicherten Daten in einem zukünftigen 3D-Modell der Stadt ein komplettes Informationssystem erstellen ließe. Technisch gesehen, ist das System sehr einfach gehalten. Die Informationen werden mittels normaler HTML-Dokumente präsentiert. Für einige Funktionen wurde JavaScript verwandt. Der große Vorteil dieser Darstellungsart liegt in der schnellen Übertragung der Daten. Angesichts der Mächtigkeit der Datenbank ist

anzunehmen, dass das entstehende 3D-Modell in Bezug auf die Systemanforderungen wesentlich anspruchsvoller sein wird. Die Navigation gestaltet sich recht einfach. Der Bildschirm ist zweigeteilt. Auf der linken Seite hat man die Karte im Blick, auf der rechten Seite werden zu den angeklickten Objekten die Informationen in Tabellenform präsentiert (Abb. 4). Da die Anzeige mit Frames realisiert wurde, entfällt lästiges Zurückblättern, denn die Karte bleibt stets im Blickfeld.

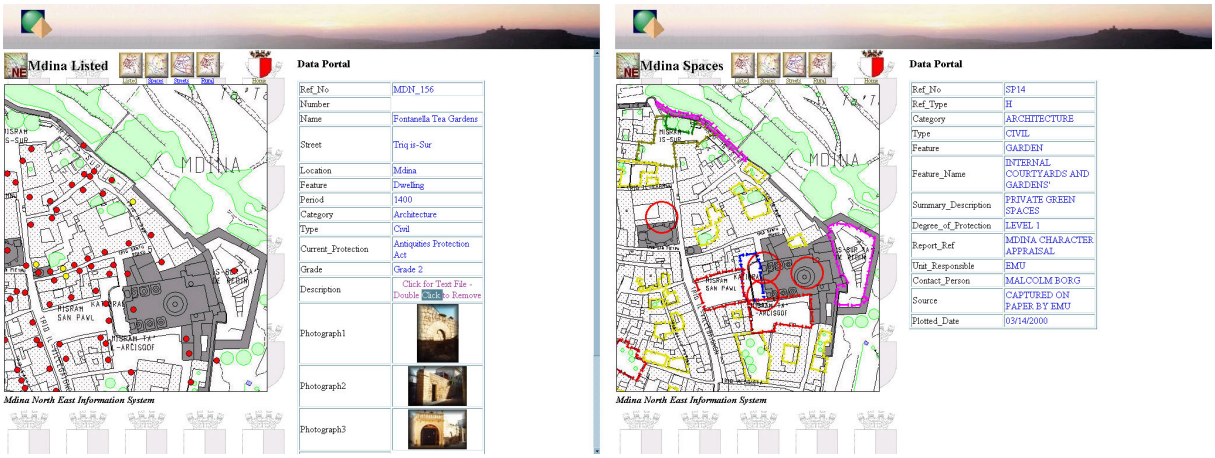

Abbildung 4 – Mdina Heritage Managment System

1.3.4 VR Seattle (USA)

Unter [9] findet sich ein von einer Privatperson entwickeltes Projekt. Dabei wird versucht, dem Besucher den US-Bundesstaat Washington und die dort befindliche Stadt Seattle näher zu bringen. Über Menüs kann man zwischen dem Bundesstaat und der Stadt Seattle wählen. Entscheidet man sich für Seattle, erscheint eine Liste von Orten, die vom Besucher betrachtet werden können. Wird dagegen ein Ort im Bundesstaat Washington ausgewählt, zeigt eine Karte die Lage und bietet ein oder mehrere Panoramen zur Auswahl. Hinter den einzelnen Orten ist die Anzahl der verfügbaren Panoramen in Klammern angegeben. Wird ein Ort gewählt, so erscheint eine weitere Liste, die Vorschaubilder der Panoramen zeigt (Abb. 5). Das verschafft einen schnellen Überblick über die gezeigten Motive. Die Panoramen basieren auf der Quicktime VR Technologie, weshalb zum Betrachten ein entsprechender Quicktime-Player benötigt wird. Die Qualität der Panoramen ist sehr gut. Der Besucher erhält einen interessanten Einblick und detaillierte Bilder der Stadt. Viele Orte bieten mehrere Panoramen an, um größere Plätze abzudecken. Zu einigen Orten wird parallel ein Stadtplan eingeblendet, auf dem der augenblickliche Standort gekennzeichnet ist.

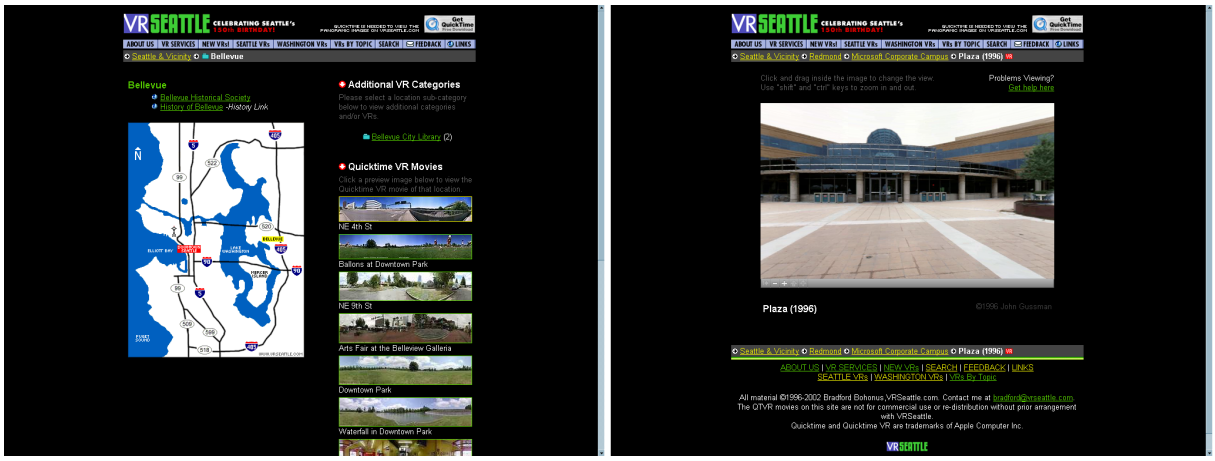

Abbildung 5 – VR Seattle

Insgesamt sind in dem System über 700 unterschiedliche Panoramen gespeichert. Eine Suchfunktion erleichtert das Auffinden spezieller Orte. Über eine Liste mit weiterführenden Links kann der Besucher schnell zu weiteren Internetseiten gelangen, die ähnliche Panoramen anbieten. Das Angebot ist sehr umfangreich und von ausgesprochen guter Qualität. Aufgrund der Beschränkungen des Quicktime-Formats, ist eine Navigation innerhalb der Panoramen allerdings nicht möglich. Ebenso fehlt die Möglichkeit, in den Panoramen interaktiv zu werden. Die Qualität der dargebotenen Panoramen macht das System zu einem sehr sehenswerten Projekt. Zum Zeitpunkt der Recherche war die Entwicklung noch nicht abgeschlossen, so dass künftig mit weiteren Panoramen zu rechnen ist.

1.3.5 Vergleich der Systeme

Der Vergleich der Systeme lässt erkennen, dass es sehr unterschiedliche Ansätze für Informationssysteme gibt. Einerseits jene, die auf 3D-Modelle setzen (Cyber Chicago, Virtual Museum), andere, die mittels klassischem HTML implementiert wurden (Heritage Management System). Einen Mittelweg aus beiden geht das System VR Seattle. Zudem verfolgen alle Informationssysteme ein anderes Ziel.

Cyber Chicago und VR Seattle richten sich vor allem an den Privatanwender, der sich einen Überblick über die beiden Städte verschaffen will. Dabei beschränken sich beide auf die visuelle Darstellung. Die 3D-Darstellung bzw. die Panoramen sind sehr gelungen. Die in Cyber Chicago verwendeten Effekte zur Erzeugung der Großstadtatmosphäre verbessern die Anschaulichkeit. VR Seattle dagegen besticht durch seinen enormen Umfang an Orten, die besucht werden können. Cyber Chicago bietet keinerlei weiterführende Information, VR Seattle wenigstens zu einigen Orten.

Das Virtual Museum der MMU hat zu dem visuellen Aspekt noch den Anspruch, Informationen zu vermitteln. Dabei greift es auch auf Informationen in Textform zurück. Das

Modell beschränkt sich außerdem nicht nur auf die Stadt, sondern geht auch auf Details ein und bietet dadurch dem Benutzer umfassende Inhalte. Da sich das System als Studie versteht, wurde mehr Wert auf Funktionalität denn auf Design gelegt. Ein Einsatz in ‚echten' Museen ist definitiv denkbar.

Das Mdina Heritage Mangement System geht noch einen Schritt weiter und beschränkt sich fast hauptsächlich auf die Darbietung von Information in Textform. Die Informationen sind in der angeschlossenen Inventar-Datenbank gespeichert. Dabei richtet sich dieses ganz klar an den professionellen Benutzer wie beispielsweise Historiker oder Stadtplaner.

Die 3D-Technik gibt dem Benutzer grundsätzlich einen besseren Eindruck von den darzustellenden Modellen. Im Zusammenspiel mit gezielten Informationen zu den Modellen ist diese Technologie sicherlich die zukunftsträchtigere. In Anbetracht der steigenden Leistung der Computer und der Bandbreite bei der Datenübertragung müssen hier bald keine Einbußen bei der Performance mehr in Kauf genommen werden. Die untersuchten Systeme bestätigen diesen Trend. Sowohl das Virtual Museum als auch das Mdina Heritage Mangement System sollen in weiteren Ausbaustufen zusätzliche 3D-Komponenten erhalten.

Die untersuchten Systeme unterscheiden sich also sowohl in ihrer technischen Umsetzung als auch in ihrem Datengehalt und der anvisierten Zielgruppe.

	Cyber Chicago	Virtual Museum	Mdina Heritage Management System	VR Seattle
Zielgruppe	Amateure	Amateure	Profis	Amateure
Technologie	VRML	VRML, HTML	HTML	QTVR
Informationsgehalt	sehr gering	hoch	sehr hoch	gering

Tabelle 1 – Systemvergleich

Dabei haben alle Systeme in ihrer Umsetzung noch Potenzial für Erweiterungen, sei es in technischer oder inhaltlicher Hinsicht. Alle zusammen zeigen jedoch, was im Bereich Informationssysteme möglich ist. Einen technischen Favoriten gibt es nicht, da jedes der Systeme die für seinen spezifischen Einsatzzweck nötige Technologie verwendet. Dabei bietet sich sowohl die VRML-Technologie als auch Quicktime VR als Basis für die dreidimensionale Darbietung von Informationen im Internet hervorragend an. Die 3D-Technologie dagegen erlaubt eine sehr anschauliche Darstellung von Gegenwart oder Vergangenheit. Um tiefergehende Informationen zu präsentieren sind nach wie vor Bilder und Text das richtige Medium. Es wird jedoch auch deutlich, dass die Umsetzung der Ideen von entscheidender Bedeutung ist. Wichtig ist eine ausgewogene Mischung aus Informationsgehalt und grafischen Feinheiten.

2 Multimedia-Technologien zur Implementierung von 3D-Informationssystemen

2.1 Web3D-Consortium

Das Web3D-Consortium (W3C) wurde 1994 als VRML-Consortium gegründet und ist ein Zusammenschluss verschiedener Interessengruppen, deren Ziel es war, die Entwicklung offener 3D-Standards voranzutreiben. Man erhoffte sich von der Gründung eines Konsortiums, welches sich ausschließlich auf Web3D-Anwendungen konzentriert, die nötige Struktur für eine solche Entwicklung bieten zu können. Zu Beginn der Arbeit lag das Hauptaugenmerk auf Virtual Reality im Internet. Bald traten dem W3C die Erfinder von VRML bei und man arbeitete gemeinsam an der Spezifikation des VRML 2.0-Standards, der heute die Grundlage vieler dreidimensionaler Web-Anwendungen ist. Größter Erfolg dieser Zusammenarbeit war die Verabschiedung von VRML 97 als ISO/IEC-Standard 14772. Da sich die Gruppe nicht mehr ausschließlich mit VRML beschäftigen wollte, erweiterte man den Fokus auf jegliche dreidimensionalen Internet-Technologien und änderte 1999 den Namen in Web3D-Consortium.

Heute zählt das Konsortium mehr als 1500 Mitglieder (3D-Labs, Sony, Sun u.a.), von denen viele an der Entwicklung von Internet-Technologien beteiligt sind. Aufgrund dessen und der damit verbundenen breiten Unterstützung durch namhafte Firmen wird unter anderem die Entwicklung des VRML-Standards immer weiter vorangetrieben. Neuestes Beispiel dieser Arbeit ist die Extensible 3D Specification (X3D), welche VRML 97 um die Extensible Markup Language (XML) erweitert. Mit XML lassen sich Daten beliebig strukturieren. Des Weiteren sind damit einfache Möglichkeiten der Erweiterung gegeben. Im August 2001 wurde dieser Standard verabschiedet. In der weiteren Entwicklung soll X3D die Integration dreidimensionaler Inhalte in den neuen MPEG-4-Standard ermöglichen.

2.2 Virtual Reality

Im Allgemeinen spricht man von Virtual Reality (VR) oder virtueller Realität wenn es um die dreidimensionale Darstellung realer oder abstrakter Umgebungen im Computer geht. Dabei kann es sich um Landschaften jeglicher Art oder auch um komplexe Systeme handeln. Eine allgemeingültige Definition gibt es nicht. Die Schweizer Firma ViewTec AG, die sich mit Softwarelösungen in den Bereichen VR-Simulation beschäftigt, definiert VR in [10] folgendermaßen:

„Virtual Reality ist eine vom Computer geschaffene, interaktive, dreidimensionale Umwelt, in die eine Person eintaucht.“

Eine weitere Definition für VR findet sich in [11]:

„Unter Virtual Reality - Virtuelle Realität - wird eine durch eine Computersimulation erzeugte Scheinwelt mit audiovisueller und taktiler Ausgestaltung verstanden. Die Generierung dieser Welten erfolgt unter unmittelbarer Einbeziehung des Benutzers und zugrundeliegenden (häufig physikalischen) Gesetzmäßigkeiten.“

Beiden Definitionen ist gemein, dass es sich bei Virtual Reality um eine computergenerierte Umgebung handelt, in welcher der Anwender mit den dargestellten Objekten interagieren kann.

Virtual Reality wird unter anderem in Situationen eingesetzt, in denen eine reale Darstellung unmöglich ist, sei es um Vorgänge im Mikrokosmos darzustellen, historische Architektur erlebbar zu machen oder auch zu Trainingszwecken. VR bildet dafür eine Art Schnittstelle zwischen dem System und dem Menschen. Mit Hilfe einer Benutzerschnittstelle wird dem Anwender die Möglichkeit gegeben, mit dem System und den darin vorkommenden Elementen zu interagieren. In einem 3D-Informationssystem, welches sowohl für Heimanwender als auch für professionelle Anwender konzipiert ist, ist die Beschränkung auf Maus- oder Tastaturbedienung sinnvoll. In Bereichen, in denen Vorgänge naturgetreu, bis hin zu Schwerkraft oder Wettereinflüssen, simuliert werden müssen, können jedoch fortgeschrittenere Eingabegeräte verwendet werden. Über VR-Helme und Datenhandschuhe taucht der Anwender viel intensiver in die virtuelle Realität ein oder kann virtuelle Werkzeuge bedienen. Bei dieser speziellen Form von VR spricht man auch von Immersion (engl.: eintauchen). Der Einsatz dieser Technik bietet sich jedoch nur bei einigen spezifischen Anwendungsgebieten an, beispielsweise im wissenschaftlichen Sektor zur Visualisierung medizinischer Vorgänge oder im Bereich der Technik & Forschung (Raumfahrt). Aufgrund der relativ hohen Kosten der nötigen Hardware, ist sie für den Heimgebrauch nicht geeignet.

2.3 VRML

VRML steht für Virtual Reality Modeling Language, was soviel heißt wie Modelliersprache für virtuelle Realität. Genauer gesagt handelt es sich um eine Beschreibungssprache zum Erstellen und Definieren dreidimensionaler Umgebungen. VRML liegt mittlerweile in der Version 2.0, auch als VRML 97 bekannt, vor. Mit VRML wurde des Weiteren ein Format entwickelt, welches zum Austausch von 3D-Daten zwischen verschiedenen Applikationen benutzt werden kann. Es enthält alle Semantiken, die auch in modernen 3D-Anwendungen zu finden sind, beispielsweise Hierarchie, Transformation, Lichtquellen, Animation etc. Ein großer Vorteil von VRML ist seine Einfachheit. Den Anstoß zu dieser Entwicklung gab die Notwendigkeit eines neuen Formates, welches dem Anwender durch den Einsatz von 3D-

Technologien bestimmte Informationen besser erfahrbar gestaltet, beispielsweise 3D-Spiele, wissenschaftliche Aspekte oder Architektur. Eines der Ziele war es, das bisher zweidimensionale Internet um die dritte Dimension zu erweitern, um auf diese Weise neue Anwendungsmöglichkeiten zu eröffnen. VRML ist sozusagen eine 3D-Erweiterung zu HTML, was VRML als plattformübergreifende Möglichkeit zur Erstellung dreidimensionaler Webseiten anbietet. VRML bietet Interaktion, Animation und Möglichkeiten, das System zu erkunden, also Dinge, die mit einem text- oder grafikbasierten Format wie HTML nur beschränkt möglich sind. Der Nutzer kann sich innerhalb einer VRML-Welt absolut frei bewegen. Da VRML jedoch problemlos in HTML eingebettet werden kann und auch mit Scriptsprachen zusammenarbeitet, entsteht ein kohärentes Modell, das zwei- und dreidimensionale Inhalte, Text und multimediale Komponenten vereinigt. Bei der Entwicklung wurde des Weiteren Wert darauf gelegt, vorhandene Infrastrukturen und Technologien in Bezug auf Internet und WWW weiter zu nutzen. So wird die Übertragung der Daten über das HTTP-Protokoll abgewickelt, um Inkompatibilitäten zu vermeiden. Durch die Möglichkeit, VRML-Welten über das Internet zu publizieren, werden die 3D-Szenarien einem breiten Publikum zugänglich gemacht.

Das Grundkonzept von VRML besteht aus drei Teilen:

- Beschreibung von Strukturen
- Beschreibung von Verhalten
- Beschreibung von Kommunikation

Mit Punkt eins ist gemeint, wie man mittels möglichst weniger Parameter das Aussehen dreidimensionaler Objekte beschreibt. In VRML gibt es nur vier sogenannte geometrische Primitive: Quader, Kegel, Zylinder und Kugel (Abb. 6). Diese Primitive lassen sich mit beliebigen Parametern generieren und erlauben es so, praktisch jedes Modell auf Basis dieser Objekte zu erstellen. Durch Kombination dieser Primitive lassen sich beliebig komplexe Objekte erzeugen, um auch die verwinkeltsten Geometrien darstellen zu können. Zudem existieren Mechanismen, die eine Erzeugung komplizierter Gegenstände aus einzelnen Polygonen ermöglichen, falls die Kombination der Grundprimitive zu aufwändig ist.

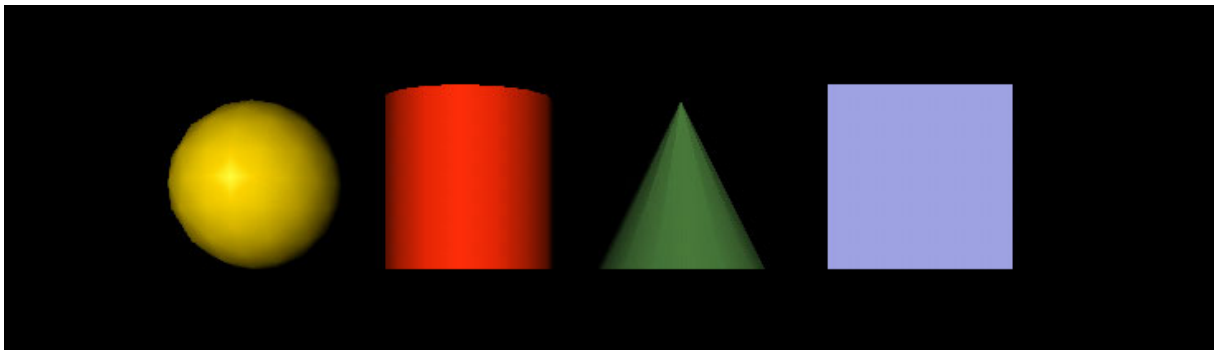

Abbildung 6 – Kugel, Zylinder, Kegel, Quader (v.l.)

In VRML lassen sich zu Objekten bestimmte Verhalten definieren. Der zweite Punkt des Grundkonzepts befasst sich mit dieser Definition. Durch sogenannte Sensoren lässt sich das Verhalten von Objekten beeinflussen. Beispielsweise können Objekte in Drehung versetzt werden, sie können selbst Aktionen auslösen (Wiedergabe von Audio- oder Videodateien) oder den Zustand anderer Objekte ändern. Nicht zuletzt lassen sich über die Verhaltensdefinitionen Hyperlinks erzeugen, die dem Nutzer die Möglichkeit bieten, weitere Welten oder Internetseiten zu erreichen. Über so genannte Sensoren kann der Nutzer direkt mit den Objekten und der Umgebung interaktiv agieren. Durch Kopplung von Sensoren an Objekte lassen sich Ereignisse auslösen, die sich auf Objekte oder die VRML-Welt auswirken.

In Punkt drei geht es um das Verteilen der dreidimensionalen Welten über das Netz. VRML bietet die Möglichkeit, Modelle in einzelne Teile aufzusplitten und diese Einzelmodelle verteilt über das Netz zu speichern. Bei der Darstellung werden diese Einzelmodelle dann geladen und in das Gesamtmodell eingefügt. Auf diese Weise lässt sich eine deutlich performantere Darstellung erreichen, da einzelne Teile des Modells erst geladen werden müssen, wenn sie in Sichtweite des Anwenders gelangen oder wenn bestimmte Ereignisse durch den Anwender ausgelöst werden.

Um die VRML-Welten zu betrachten, benötigt der Anwender entweder einen speziellen VRML-Browser oder ein Plugin, welches den vorhandenen Internet-Browser um die VRML-Funktionalität erweitert. Als Beispiel wären hier der Cosmoplayer oder Blaxxun Contact zu nennen. Als Quelle für VRML-Welten sind sowohl das Inter- oder Intranet geeignet, als auch lokale Datenträger oder CD-ROMs.

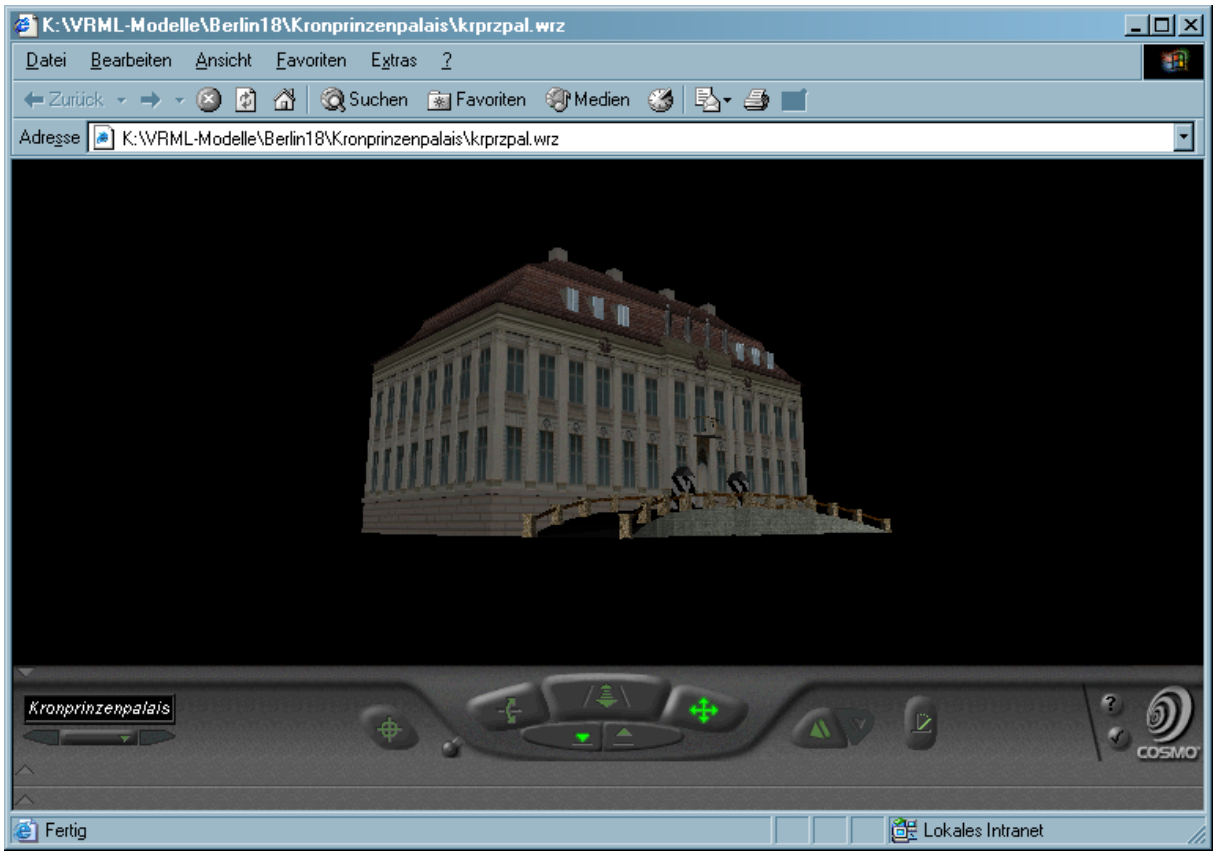

Abbildung 7 – Internet Explorer mit Cosmoplayer-Plugin

Der große Vorteil von VRML besteht darin, dass es sich bei dieser Technologie um ein echtes 3D-Format handelt. Es bietet die Möglichkeit, sich in einer virtuellen 3D-Umgebung zu bewegen, Objekte zu betrachten und zu benutzen. Dabei ist der Anwender in keiner Weise in seiner Bewegungsfreiheit eingeschränkt. Neben der Möglichkeit zu laufen, kann er auch fliegen oder einfach nur in der Luft schweben und die Szenerie von oben betrachten. Damit erlangt er von den dargestellten Objekten einen sehr umfassenden und anschaulichen Eindruck. Ihm eröffnen sich Einblicke und Perspektiven, die in der Realität unmöglich oder sehr aufwändig zu realisieren wären. Da es sich bei VRML-Szenen um modellierte Welten handelt, sind neben der Darstellung der Wirklichkeit auch Fiktionen möglich. VRML ist des Weiteren ein plattformübergreifendes Format. Einen entsprechenden Player vorrausgesetzt, sind VRML-Welten auf allen Systemen benutzbar (Abb. 7). Als Nachteil ist der recht große Modellieraufwand zu sehen. Will man Objekte naturgetreu nachbilden, ist der Einsatz eines professionellen 3D-Modellier-Programmes unumgänglich. Das bedeutet einen erheblichen technischen und auch finanziellen Aufwand. Mit zunehmendem Umfang und steigender Komplexität der VRML-Welten werden zudem die Anforderungen an die Hardware des Endanwenders in die Höhe getrieben. Konkrete Aussagen dazu sind jedoch nicht möglich, da die Anforderungen in hohem Maße von der darzustellenden VRML-Welt abhängig sind.

2.3.1 Erstellen von VRML-Welten

Eine Devise im Zusammenhang mit VRML lautet, dass sowohl das Entwickeln als auch das Abrufen von Webdokumenten von jedem an das Internet angeschlossen Rechner möglich sein sollte. Angesichts heutiger Standard-PCs steht dem grundsätzlich nichts entgegen, da die Leistungsfähigkeit zum Erstellen von VRML-Welten mehr als ausreichend ist. Theoretisch genügt ein einfacher Texteditor, um beliebig komplexe VRML-Umgebungen zu beschreiben. Jedoch wäre dafür ein extrem ausgeprägtes dreidimensionales Vorstellungsvermögen nötig. Die anfallende Datenmenge ist selbst für professionelle Modellierer nicht zu überschauen. Aus diesem Grund bieten sich diverse grafische Werkzeuge an. Es gibt sowohl unter Windows als auch unter UNIX-basierten Betriebssystemen umfangreiche und mächtige 3D-Anwendungen, die das Gestalten der 3D-Szenen enorm vereinfachen. Dabei muss zwischen reinen VRML-Programmen und 3D-Modellierern unterschieden werden.

In der Kategorie der VRML-Programme sind Cosmoworlds von Silicon Graphics und Spazz3D von Virtock Technologies zwei leistungsfähige Vertreter (Abb. 8).

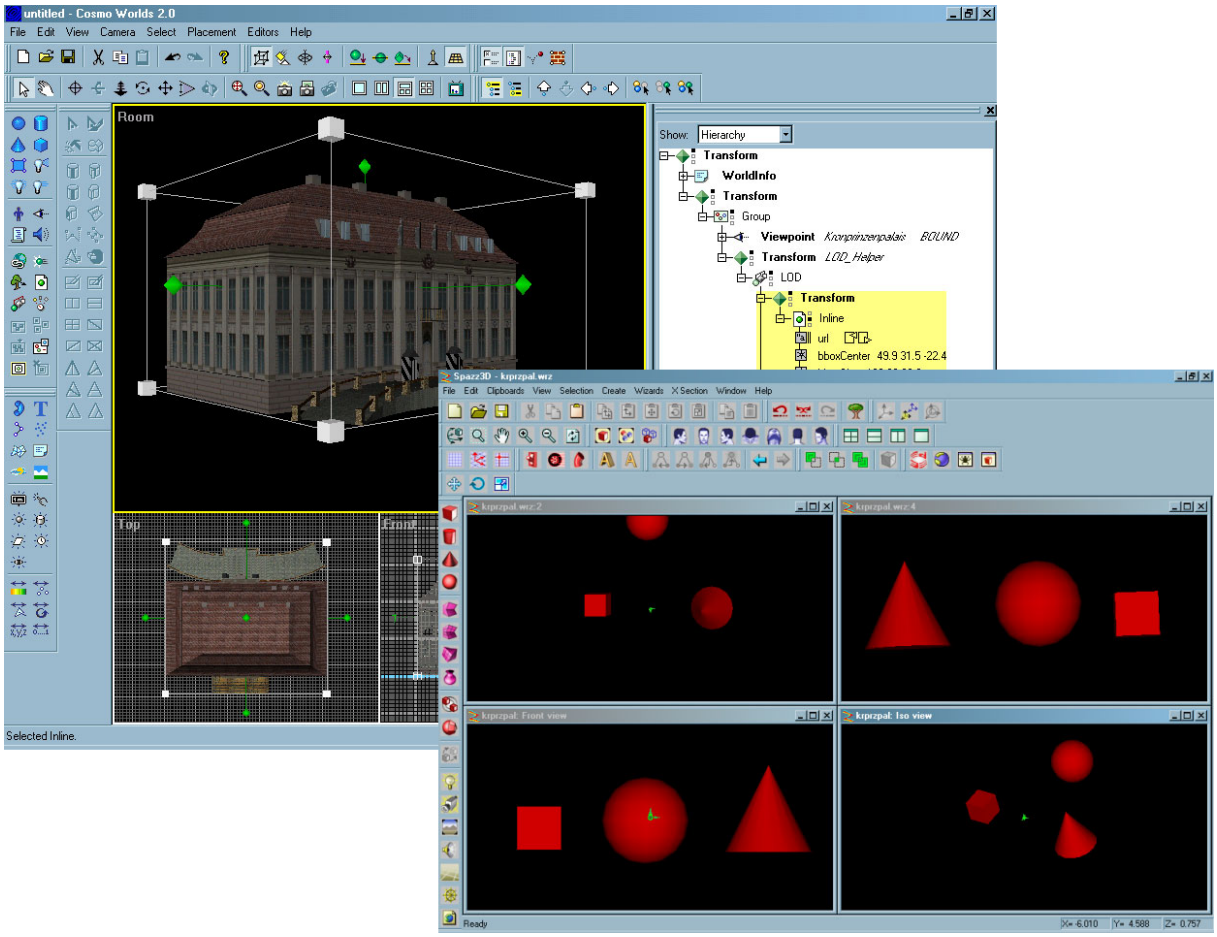

Abbildung 8 – Cosmoworlds, Spazz3D (v.l.)

Silicon Graphics war an der Entwicklung des VRML-Standards maßgeblich beteiligt. Cosmoworlds wurde demnach auch speziell zur Erstellung von VRML-Welten entwickelt und bietet sich deshalb besonders für deren Entwicklung an. Spazz3D eignet sich ebenfalls aufgrund der umfassenden Unterstützung des VRML97-Standards für deren Erstellung. Beide Anwendungen bieten dem Modellierer ein grafisches Frontend zum textbasierten Quellcode der VRML-Szene an, das es ermöglicht, die 3D-Szene in bis zu vier Fenstern (Viewports) in Echtzeit aus verschiedenen Blickwinkeln darzustellen. Über Werkzeugleisten lassen sich auf unkomplizierte Weise neue Knoten (Objekte, Lichtquellen, Viewpoints, Sensoren usw.) erstellen, in der Szene platzieren oder verändern. Da es sich um reine VRML-Modellierer handelt, beschränken sich die Möglichkeiten dabei auf die von VRML unterstützten Knoten (Grundprimitive, Text, Sensoren usw.). Parallel dazu lassen sich diverse Editoren öffnen, die dem Anwender weitere Möglichkeiten bieten. Am Beispiel von Cosmoworlds sollen einige Möglichkeiten erläutert werden. Der Outline-Editor zeigt zu jedem ausgewählten Knoten sämtliche verfügbaren Eigenschaftsfelder an. Auf einfache Art und Weise lassen sich so die Eigenschaften eines Objektes verändern. Des Weiteren erlaubt der Outline-Editor das Routing von Ereignissen von einem Knoten zum anderen. Der Keyframe Animator ist ein weiteres nützliches Tool, mit dessen Hilfe leicht Animationen erstellt werden können. Eine Reihe anderer Editoren erleichtern das Einbinden und Bearbeiten von Scripten (Script Editor) oder das Aufbringen von Texturen auf Objekte (Property Inspector, PEP Texture Applicator). Auswirkungen von Modifikationen werden in Echtzeit in den Viewports angezeigt, lassen sich aber auch über einen integrierten VRML-Viewer direkt in der Szene ausprobieren. Beide Programme bieten den vollen Umfang der VRML-Spezifikation. Auf einfache Art und Weise lassen sich Verhalten, Animationen und interaktive Elemente definieren.

In die zweite Kategorie fallen 3D-Modellierer wie 3D-Studio MAX von discreet oder Maya von Alias|Wavefront (Abb. 9). Bei beiden handelt es sich um professionelle 3D-Anwendungen, die neben der Funktionalität des eigentlichen Modellierens eine Reihe zusätzlicher Features aufweisen. Der Aufbau des grafischen Frontends ist ähnlich dem von Cosmoworlds. Mehrere Viewports erlauben die Ansicht der Objekte aus verschiedenen Blickwinkeln. Der Haupteinsatzzweck dieser Anwendungen ist das Erzeugen fotorealistischer Animationen. Zu diesem Zweck sind in die Programme etliche Elemente integriert, die das Erzeugen von Spezialeffekten ermöglichen. Da es sich bei beiden nicht um reine VRML-Programme handelt, bieten sie nicht nur wie Cosmoworlds die 4 Grundprimitive an, sondern stellen dem Modellierer zahlreiche weitere Objekte (z.B. Torus, Tube, Pyramide u.a.) zur Verfügung. Zusätzlich dazu bieten sie Modifikatoren, mit deren Hilfe Objekte auf vielfältige Art und Weise manipuliert werden können. Komplexe Geometrien lassen sich so deutlich schneller und einfacher erzeugen, als wenn man sie mühsam aus den VRML-

Grundprimitiven zusammensetzen würde. Die Schnittstelle zu VRML wird durch entsprechende Plugins gewährleistet. Durch diese werden einerseits VRML-Funktionen (Sensoren, LODs u.a.) in die Programme integriert. Andererseits erlauben sie dem Anwender, erstellte Objekte in das VRML-Format zu exportieren. Dabei werden beispielsweise erzeugte Kameras oder Lichtquellen in die entsprechenden VRML-Pendants (Navigationspunkte und Lichtquellen) umgewandelt. Objekte, die nicht den VRML-Primitiven entsprechen, werden als IndexedFaceSet exportiert.

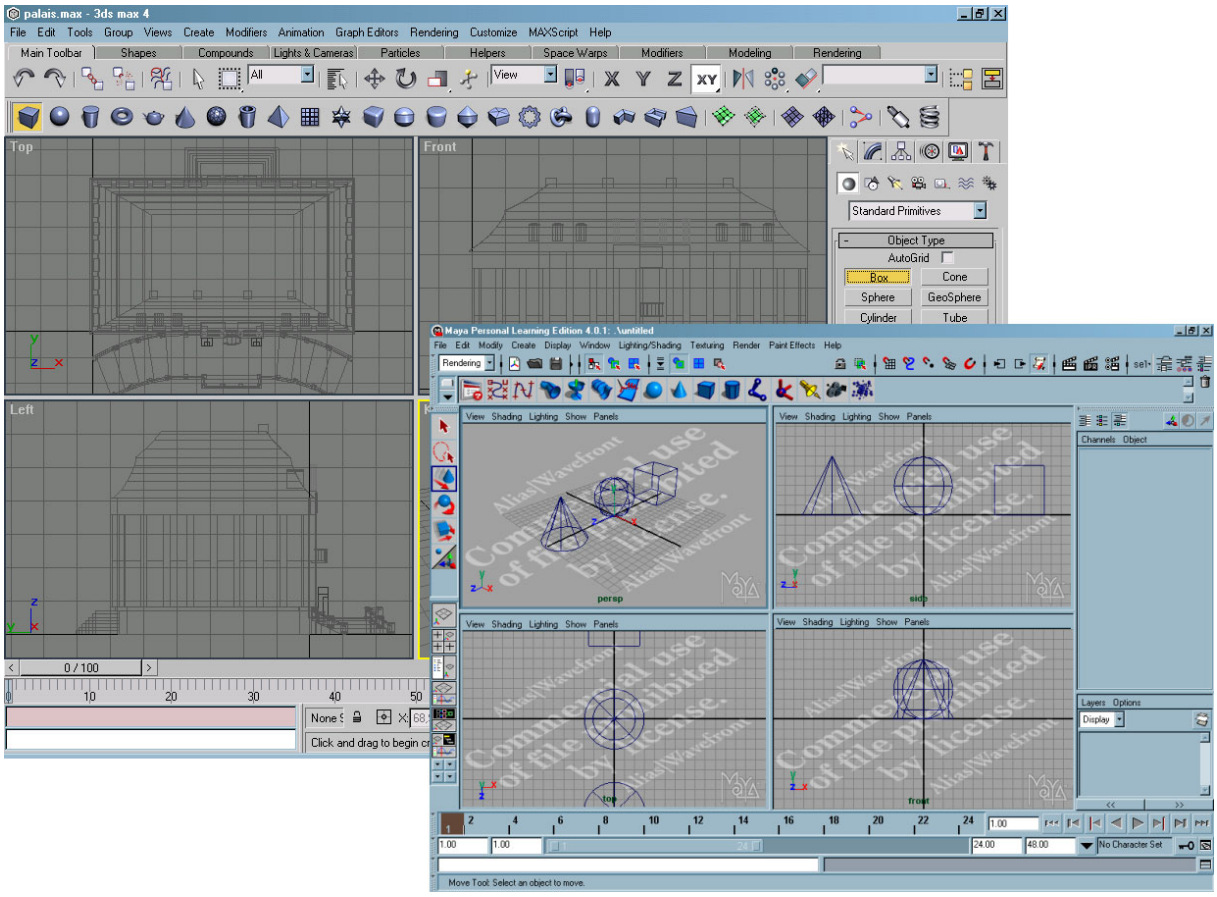

Abbildung 9 – 3D Studio MAX R4, Maya 4.0.1 PLE (v.l.)

Beide Kategorien von Programmen sind folglich sehr gut zum Modellieren von 3D-Szenen und zum Erstellen von VRML-Welten geeignet. Dabei sind Cosmoworlds und Spazz3D speziell auf VRML ausgerichtet, wogegen 3D-Studio MAX und Maya mehr Möglichkeiten bei der Geometrie bieten. Nicht unwesentlich ist auch, dass Cosmoworlds und Spazz3D jeweils in einer kostenlosen Testversion für 30 Tage uneingeschränkt verwendet werden können. Die professionellen 3D-Modellierer finden aufgrund ihrer Komplexität und Fähigkeiten selbst für Spezialeffekte bei Filmproduktionen Verwendung. Entsprechend hoch ist auch ihr Preis. Von Maya existiert zwar ebenfalls eine kostenlose Testversion (Personal Learning Edition), diese unterstützt jedoch nur ein proprietäres 3D-Format und ist somit für VRML nicht

geeignet. In den Tabellen 2 und 3 sind die Anforderungen der Modellierprogramme zusammengefasst.

	Cosmoworlds	Spazz 3D
Hersteller	Silicon Graphics	Virtock Technology
Betriebssystem	Windows 9x/2000,	Windows 9x/NT/2000
	Solaris, IRIX	
Testversion	Kostenlos	kostenlos
Vollversion	ca. 1000,- €	ca. 50,- €

Tabelle 2 – VRML-Modellierer

	3D-Studio Max 4	Maya
Hersteller	Discreet	Alias|Wavefont
Betriebssystem	Windows 98/NT/2000	Windows NT/2000 Pro,
		IRIX, Linux, Mac OS X
Testversion	-	Maya Personal Learning Edition
Vollversion	ca. 4930,- €	2725,- €

Tabelle 3 – 3D-Modellierer

2.4 Quicktime VR

Hinter Quicktime VR verbirgt sich ein von der Firma Apple entwickeltes Format, das es ermöglicht, aus Einzelbildern bewegte Panoramen zu generieren. Es kann den Übergang der flachen zweidimensionalen Welt der Fotos zu 3D-Szenarien herstellen. Dabei stehen begrenzte interaktive Elemente zur Verfügung. Das Format basiert auf der IBR-Technik (Image Based Rendering). Das heißt, dass Quicktime VR-Filme auf vorberechneten Bildern basieren und nicht wie VRML in Echtzeit generiert werden. Quicktime VR bietet zwei unterschiedliche Ansätze der 3D-Visualisierung, nämlich QTVR-Objekte und QTVR-Panoramen. Bei der ersten Variante werden Objekte frei drehbar im Raum dargestellt. Dadurch kann der Betrachter Gegenstände auch von hinten oder von unten betrachten. Diese Technik kommt vorwiegend in der Produktpräsentation zum Einsatz. Als Anwendung im Bereich der historischen Informationssysteme käme das für die Darstellung von Artefakten in Frage. Die QTVR-Panoramen bieten dem Benutzer ein anderes 3D-Erlebnis. Aus Einzelbildern werden 360°-Panoramen zusammengerechnet, in denen der Betrachter interaktiv zu jeder möglichen Ansicht des erfassten Raumes navigieren kann. Zusätzlich wird zwischen zylindrischen und sphärischen Panoramen unterschieden. Die Einzelbilder bilden

gewissermaßen einen Kreis oder eine Kugel in dessen Mittelpunkt der Anwender steht (Abb. 10).

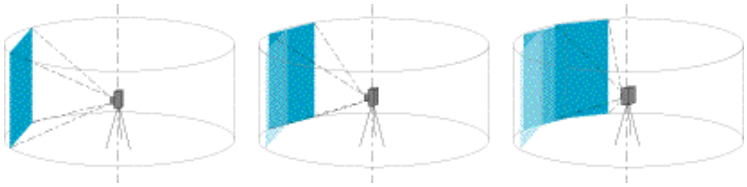

Abbildung 10 – Grundaufbau von zylindrischen QTVR-Panoramen

Dabei hat er die Möglichkeit, sich um die eigene Achse zu drehen (zylindrische Panoramen) oder zusätzlich nach oben und unten zu schauen (sphärische Panoramen) und in die Bilder hineinzuzoomen, um Details zu erkennen. In beiden QTVR-Varianten lassen sich so genannte Hot Spots definieren. Diese fungieren als Hyperlinks, mit denen der Anwender durch Anklicken zu weiteren QTVR-Panoramen oder -Objekten gelangt, weiterführende Informationen erhält oder einfach andere Internetseiten erreicht. Auf diese Weise lassen sich beliebig große Welten erstellen. Da QTVR meist auf Fotografien basiert, erweckt die Grafik einen sehr realistischen Eindruck. Über zusätzliche Werkzeuge lassen sich auch Soundeffekte einbinden oder Animationen erzeugen.

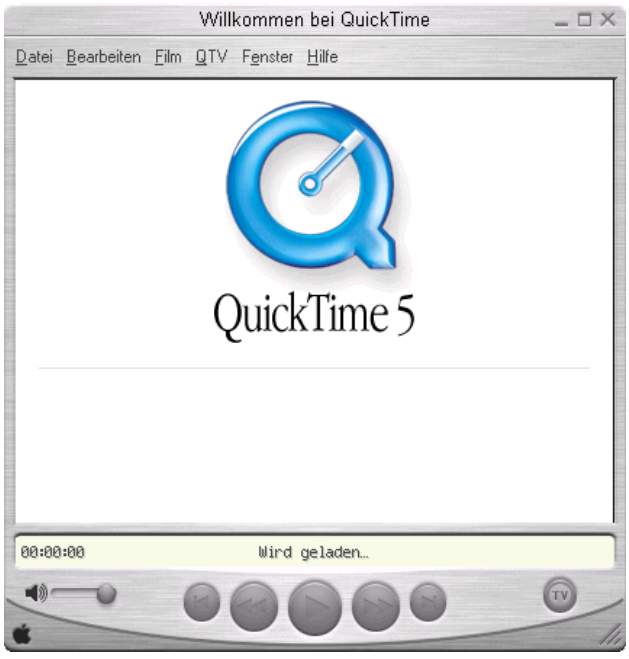

Abbildung 11 – Quicktime Player

Um die Panoramen zu betrachten ist entweder der Quicktime Player der Firma Apple (Abb. 11) oder ein entsprechendes Plugin für den Internet-Browser erforderlich. Mittlerweile gehört der Quicktime-Player jedoch schon fast zur Standardausstattung eines durchschnittlichen PCs, so dass das Abspielen keine Probleme darstellen sollte. Sowohl

Player als auch Plugin sind für jede Plattform (MacOS, Windows, Unix u.v.a.) verfügbar, wodurch Quicktime auf allen Plattformen wiedergegeben werden kann. Der große Vorteil ergibt sich aus der Verwendung der IBR-Technologie. Da die Szenen bereits vorberechnet sind, sind für die Darstellung selbst keinerlei komplexe Berechnungen mehr nötig. Entsprechend niedrig sind auch die Systemanforderungen die Apple für Quicktime VR angibt (Tab. 4).

MacOS	Windows
PowerPC basierter Macintosh-Computer	Pentium-PC oder kompatibel
mind. 32 MB RAM	mind. 32 MB RAM
MacOS 7.5.5 oder höher	Windows 95/98/NT/ME/2000
	SB-kompatible Soundkarte
	DirectX Version 3.0

Tabelle 4 – Systemanforderungen Quicktime VR

Aufgrund der Verwendung von Fotos als Grundlage für die Erstellung der QTVR-Filme wirkt die Darstellung sehr realitätsnah.

Als Nachteil von Quicktime VR ist anzusehen, dass es sich nicht um ein echtes 3D-Format handelt. Die QTVR-Panoramen oder QTVR-Objekte basieren auf zweidimensionalen Fotografien oder Zeichnungen. Der dreidimensionale Eindruck wird dem Anwender lediglich auf geschickte Art und Weise vorgetäuscht. Dazu kommt die eingeschränkte Interaktivität von Quicktime VR. Die Bewegungsfreiheit der Benutzer ist beim Erforschen der Welt stark begrenzt, da er die vorberechneten Wege nicht verlassen kann. Außer den angesprochenen Hot Spots, gibt es keine Möglichkeit, interaktive Elemente einzubinden. Da die Panoramen aus digitalisierten Fotos zusammengesetzt werden, können außerdem recht große Dateien entstehen, deren Übertragung in Abhängigkeit der verfügbaren Bandbreite viel Zeit in Anspruch nehmen kann. Dem gegenüber steht jedoch wiederum die Streaming-Fähigkeit von Quicktime. Dadurch müssen die Daten nicht erst komplett geladen werden, sondern werden nach und nach vervollständigt.

2.4.1 Erstellen von QTVR-Welten

Auch bei der Erstellung von QTVR-Welten muss zwischen QTVR-Panoramen und QTVR-Objekten unterschieden werden. Der Aufwand zur Erstellung von Objekten ist enorm hoch. Um eine flüssige Rotation um das Objekt zu ermöglichen, muss der Gegenstand in präzisen Drehstufen von allen Seiten fotografiert werden. Bei einer Beschränkung auf 10-Grad-Schritte, sind zur kompletten Erfassung 684 Einzelbilder nötig. Diese müssen anschließend noch zu dem QTVR-Objekt zusammengesetzt werden. Bei zylindrischen QTVR-Panoramen

ist der Aufwand deutlich geringer. Der Vorgang des Erstellens ist grundsätzlich in drei Schritten zusammenzufassen:

1. Es wird ein Standpunkt gewählt und ein Foto gemacht.
2. Die Kamera wird um einen festen Winkel gedreht und es wird ein weiteres Foto gemacht. Dabei ist zu beachten, dass sich die Bilder um 30-50 % überlappen sollten. Diese Prozedur wiederholt sich bis eine 360°-Umdrehung erfolgt ist. Auf diese Weise entstehen 12-18 Bilder, die anschließend digitalisiert und zu einem Panorama zusammengesetzt werden. Das Zusammensetzen der Einzelbilder wird Stitching genannt.
3. Um den Übergang zwischen den Fotos so flüssig wie möglich erscheinen zu lassen, werden die überlappenden Bildteile angepasst. Man spricht bei diesem Vorgang von Warping.

Bei kubischen QTVR-Panoramen muss im zweiten Schritt zusätzlich der Bereich ober- und unterhalb der normalen Bildebene fotografiert werden. Um Panoramen im Quicktime VR Format zu erstellen, ist eine entsprechende Authoring-Software unerlässlich. Die Referenz-Anwendung kommt von Apple und heißt Quicktime VR Authoring Studio. Das Programm besteht aus mehreren Modulen, u.a. dem Panorama-Sticher und dem Panorama-Maker, mit denen das angesprochene Stitching und Warping der Einzelbilder vorgenommen wird. Durch das Warping können unter Umständen Verzerrungen entstehen, weil eigentlich gerade Linien nach der Bearbeitung aufgrund der Übertragung auf Kugel oder Zylinder hinterher wie Kurven aussehen. In der Praxis stellt das jedoch kein Problem dar, da QTVR in der Lage ist, während des Abspielens die angezeigten Bilder zu entzerren und so einen realistischen Ablauf darzustellen. Mit dem Object Maker lassen sich QTVR-Objekte erzeugen (Abb. 12).

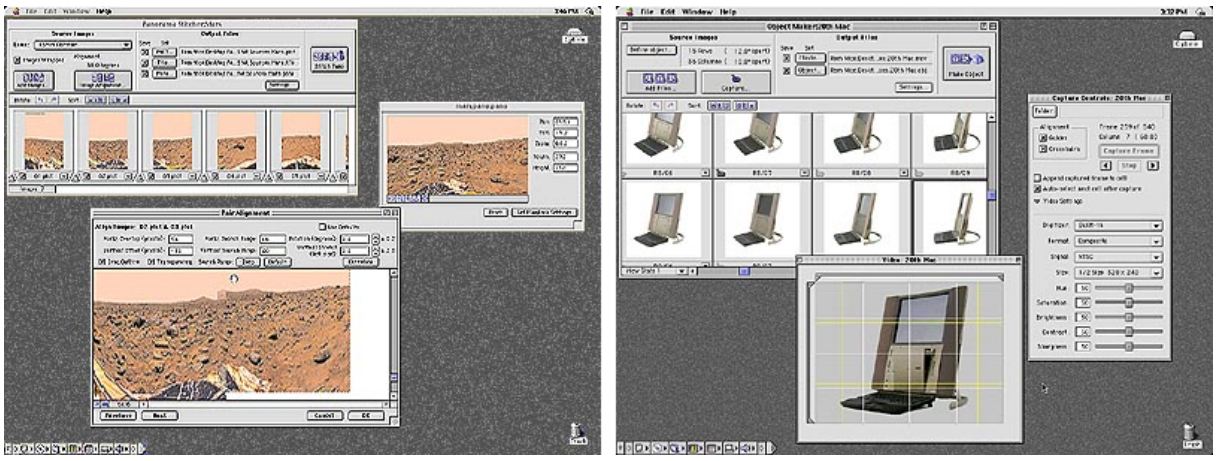

Abbildung 12 – Panorama Stitcher, Objekt Maker (v.l.)

Außerdem ist es mit Quicktime VR Authoring Studio möglich, die oben angesprochenen Hot Spots zu erzeugen. Dabei werden Bildbereiche gekennzeichnet und mit anderen Objekten verbunden. Diese Methode ist mit Imagemaps zu vergleichen, die in HTML Anwendung finden. Durch zusätzliche Tools können auch Soundeffekte (SoundsaVR) oder automatische Animationen (RevolVR) erzeugt werden. Nachteilig ist, dass Quicktime VR Authoring Studio nur unter MacOS läuft. Zudem hat Apple die Weiterentwicklung eingestellt. Eine QTVR-Anwendung die ebenfalls für Windows verfügbar ist, ist beispielsweise Stitcher 3 von RealViz (Tab. 5).

	Quicktime VR Authoring Studio	**Stitcher 3**
Hersteller	Apple	RealViz
Betriebssystem	Mac OS 7.5 oder höher	Windows 98SE, ME, 2000, NT4.0 SP3 Mac OS 9.1, X
Vollversion	439,64 €	561,- €

Tabelle 5 – Quicktime VR Authoring Studio, Stitcher 3

2.5 Vergleich VRML Quicktime VR

In der folgenden Tabelle (Tab. 6) sind noch einmal alle Eigenschaften von VRML und Quicktime VR vergleichend gegenübergestellt.

VRML	QTVR
- rechenintensive Fließkomma-Berechnungen erfordern schnelle PCs	- Abspielen auf Standard-PCs möglich
- Internet-Unterstützung mittels Browser-Plugins	- Internet-Unterstützung mittels Browser-Plugins
- plattformübergreifend	- plattformübergreifend
- dargestellte Objekte sind modelliert	- dargestellte Objekt sind real
- Computeranimation	- fotorealistische Darstellung
- Modellieren der 3D-Objekte erforderlich	- kein Modellieren erforderlich
- Entwicklung komplexer Szenerien erfordern professionelle 3D-Werkzeuge	- relativ einfache Entwicklung von QTVR-Panoramen
- Quellcode lässt sich gut komprimieren, da textbasiert	- große Dateien, da auf digitalisierten Bildern basierend
- anspruchsvolle Hardwareanfordungen mit steigender Komplexität	- schnelle Navigation komplexer Szenen auf Standard-PCs
- absolute Bewegungsfreiheit in alle Richtungen	- Bewegung nur von einem festen Punkt zu einem anderen
- Interaktivität durch zahlreiche Funktionen	- kaum Interaktivität

Tabelle 6 – Vergleich VRML / Quicktime VR

2.6 Fazit

Bei der Wahl des geeigneten Formates zur Erstellung eines historischen 3D-Informationssystems spielen vor allem die Möglichkeiten zur Umsetzung der geplanten Funktionalitäten eine wichtige Rolle.

Da es dem Anwender in dem zu entwickelnden Informationssystem möglich sein soll, Objekte zu benutzen und zu bewegen, kommt Quicktime VR angesichts der stark eingeschränkten Interaktivität nicht in Frage. Des Weiteren würde ein komplexes Informationssystem aufgrund der speicherintensiven Bilddaten der QTVR-Szene jeglichen Rahmen sprengen. Ein weiterer Nachteil ist, dass QTVR auf realen Bilddaten basiert. Für ein historisches Informationssystem, das auf alte Bilder oder Kupferstiche angewiesen ist, ist die Umsetzung mit QTVR schwer vorstellbar. Theoretisch ließen sich zwar auch solche Bilddaten zu Panoramen zusammenfügen, jedoch sind die Quellen der damaligen Zeit nicht

sehr umfangreich. Der Vorteil von QTVR liegt klar auf der Performance-Seite. Durch vorberechnete Szenen ist es VRML überlegen. Auch die Anfertigung von Panoramen ist dank einfach zu bedienender Software sehr unkompliziert. Und schließlich ist Quicktime VR kein echter 3D-Standard. Die Möglichkeit, direkt durch die Szenerie zu wandern, lässt sich so nicht realisieren.

VRML ist aufgrund der Möglichkeiten, den Anwender interaktiv in die Szene einzubeziehen, für ein historisches 3D-Informationssystem besser geeignet. Der Anspruch, dem Anwender vergangene Zeiten nahe zu bringen, erfordert auch die virtuelle Rekonstruktion nicht mehr existierender Gebäude. Eine Modellierung der Szene ist somit unumgänglich. Den fotorealistischen Eindruck kann man mittels guter Texturen, die sich anhand alter Bilddaten rekonstruieren lassen, erreichen. Das Problem der relativ komplexen Software zur Erstellung der Modelle stellt sich nicht, da ein professionelles Informationssystem auch von erfahrenen Fachleuten erstellt wird, die mit diesen Werkzeugen bestens vertraut sind. Auch die erhöhten Anforderungen an die Hardware des Anwenders können als Kriterium nicht gelten, da heute bereits Einsteiger-PCs mit leistungsfähigen Prozessoren und Grafikkarten ausgestattet sind, die eine performante Darstellung ermöglichen. Dieser Trend wird sich in den nächsten Jahren fortsetzen, so dass bald keinerlei Einschränkungen in dieser Hinsicht existieren werden.

2.7 VRML-Konzepte in Bezug auf das 3D-Informationssystem

Im folgenden Abschnitt werden einige VRML-Konzepte, die für die Implementierung des 3D-Informationssystems von Belang sind, näher betrachtet. In diesem Zusammenhang werden sowohl der Grundaufbau und die prinzipielle Struktur von VRML beschrieben, als auch die zur Umsetzung der geplanten Funktionalität nötigen Knoten und Konzepte beleuchtet.

Geometrie

Wie bereits angesprochen, existieren in VRML nur 4 geometrische Grundprimitive, mit denen Szenerien schon sehr detailliert modelliert werden können. Mit zunehmender Komplexität des zu erstellenden Objektes, steigt jedoch der Aufwand, dieses aus den Grundprimitiven zu modellieren, enorm an. Für diesen Fall stehen dem VRML-Entwickler weitere Geometrie-Knoten zur Verfügung.

3D-Objekte setzen sich aus vielen Eckpunkten zusammen, die mit Linien verbunden sind. Zwischen den Linien spannen sich Flächen auf, die als Polygon bezeichnet werden und die Oberfläche des 3D-Objektes bilden.

Mittels dreier verschiedener Knoten lassen sich durch Definition der einzelnen Eckpunkte komplexe Objekte definieren, ohne auf die Grundprimitive zurückzugreifen.

- PointSet
- IndexedLineSet
- IndexedFaceSet

Bei diesen drei Knoten handelt es sich um unterschiedliche Darstellungsmodi für Objekte (Abb. 13). PointSet wird verwendet, um eine Punktwolke zu erzeugen. Dabei werden lediglich die Eckpunkte des Objektes dargestellt. In dem Modus IndexedLineSet werden die Eckpunkte verbunden, die Flächen aber nicht ausgefüllt. Es entsteht ein sogenanntes Drahtgittermodell. Bei IndexedFaceSet wird das Objekt mit allen Flächen dargestellt.

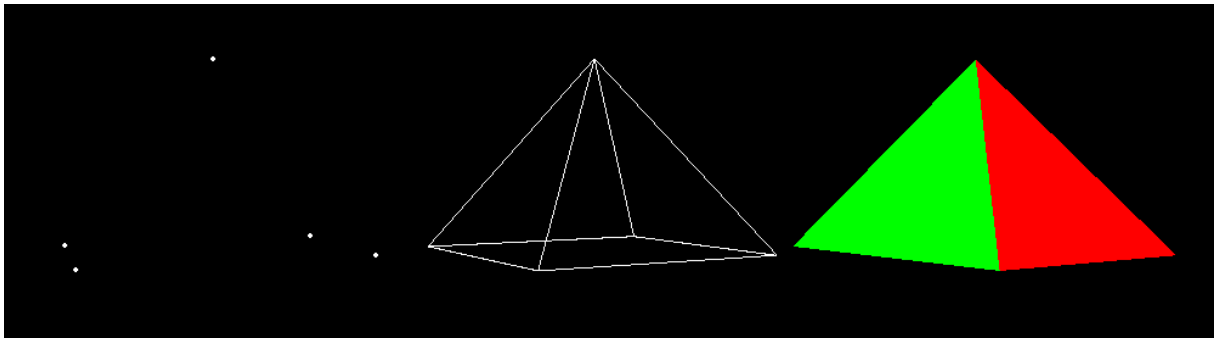

Abbildung 13 – PointSet, IndexedLineSet, IndexedFaceSet (v.l.)

Der Vorteil dieser Modi liegt in der einfacheren Definition komplexer Objekte. Eine Pyramide lässt sich auf diese Weise durch 5 Koordinatentripel definieren. Wollte man die Pyramide aus den Grundprimitiven zusammensetzen, wären nach traditioneller Bauweise etliche Quader vonnöten.

Eine weitere Möglichkeit komplexe Objekte zu erzeugen ist die Extrusion (Abb. 14), auch als Ziehen bezeichnet. Dabei wird eine zweidimensionale Geometrie entlang eines Extrusionspfades in die dritte Dimension erweitert. Der Extrusionspfad kann beliebig manipuliert werden, so dass das 2D-Ausgangsobjekt rotiert, skaliert oder anderweitig manipuliert werden kann. Auf diese Weise lässt sich beispielsweise aus einem Kreis auf einfache Art und Weise eine Flasche erzeugen, indem der Durchmesser des Kreises variiert wird, um Flaschenhals und Flaschenrumpf darzustellen.

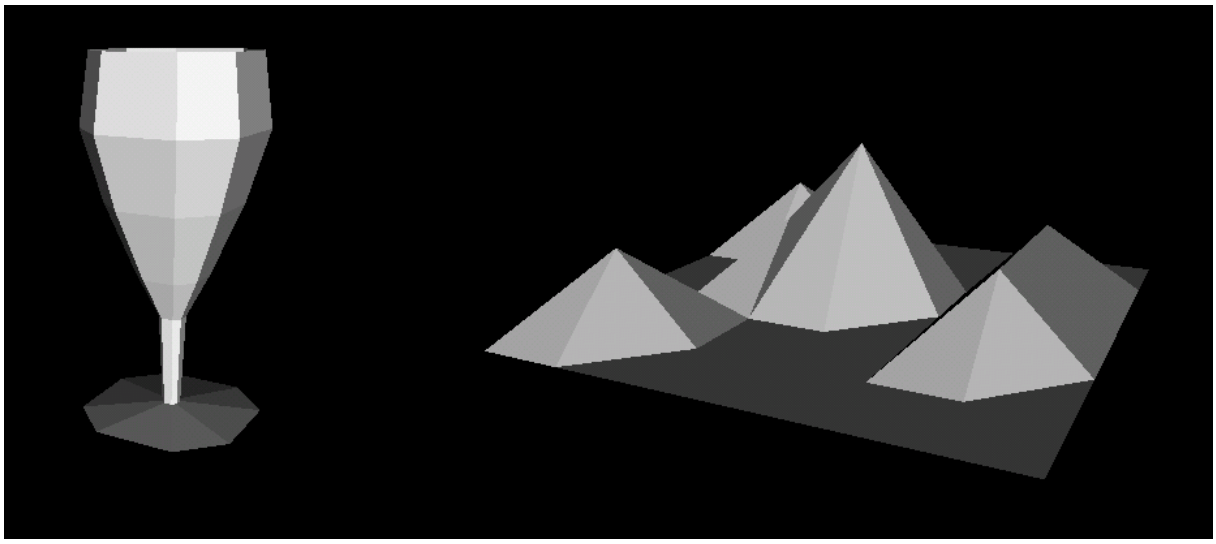

Abbildung 14 – Extrusion, Elevation Grid (v.l.)

Um Oberflächen zu gestalten, bietet sich das ElevationGrid oder auch Höhengitter an (Abb. 14). Dieses dient in besonderem Maße der Erzeugung von Landschaften. Ein ElevationGrid besteht aus einem Rechteck, welches wiederum in viele einzelne Rechtecke unterteilt ist. Ähnlich einem Drahtnetz lassen sich in einem ElevationGrid Höhenunterschiede darstellen, indem den einzelnen Rechtecken unterschiedliche Höhenkoordinaten zugeordnet werden. So können beispielsweise Berge und Täler modelliert werden. Durch dynamische Veränderung der Höhendaten eignet sich ein ElevationGrid sehr gut zur Darstellung von Wasseroberflächen. Je mehr Unterteilungen das ElevationGrid besitzt, desto feiner lassen sich Übergänge gestalten.

Abbildung 15 – Text

In einer VRML-Szene kann sich der Einsatz von Text als sehr nützlich erweisen. Vielerlei Beschriftungen (Straßennamen, Firmen) lassen sich so in der VRML-Welt realisieren. Die Darstellung von dreidimensionalem Text wird durch den TEXT-Knoten unterstützt (Abb. 15). Neben der darzustellenden Zeichenkette sind Angaben zu Formatierung (Schriftart u.ä.) und Ausrichtung gestattet. Wie jedes 3D-Objekt lassen sich TEXT-Knoten skalieren, rotieren oder anderweitig manipulieren.

Die angesprochenen Methoden bieten also vielfältige Möglichkeiten um komplexere Geometrien zu erstellen. Speziell bei PointSet, IndexedLineSet, IndexedFaceSet und ElevationGrid bietet sich jedoch der Einsatz eines 3D-Modellierungsprogrammes an, da die Definition ein hohes Maß an dreidimensionaler Vorstellungskraft erfordert.

Hierarchie

Eines der wohl wichtigsten Konzepte in VRML ist der hierarchische Aufbau der Sprache und die Unterteilung der einzelnen Sprachelemente in Knoten. Die hierarchische Struktur wird durch den sogenannten Scene Graph realisiert (Abb. 16), der sich aus Gruppen-, Kind- und Objektknoten zusammensetzt.

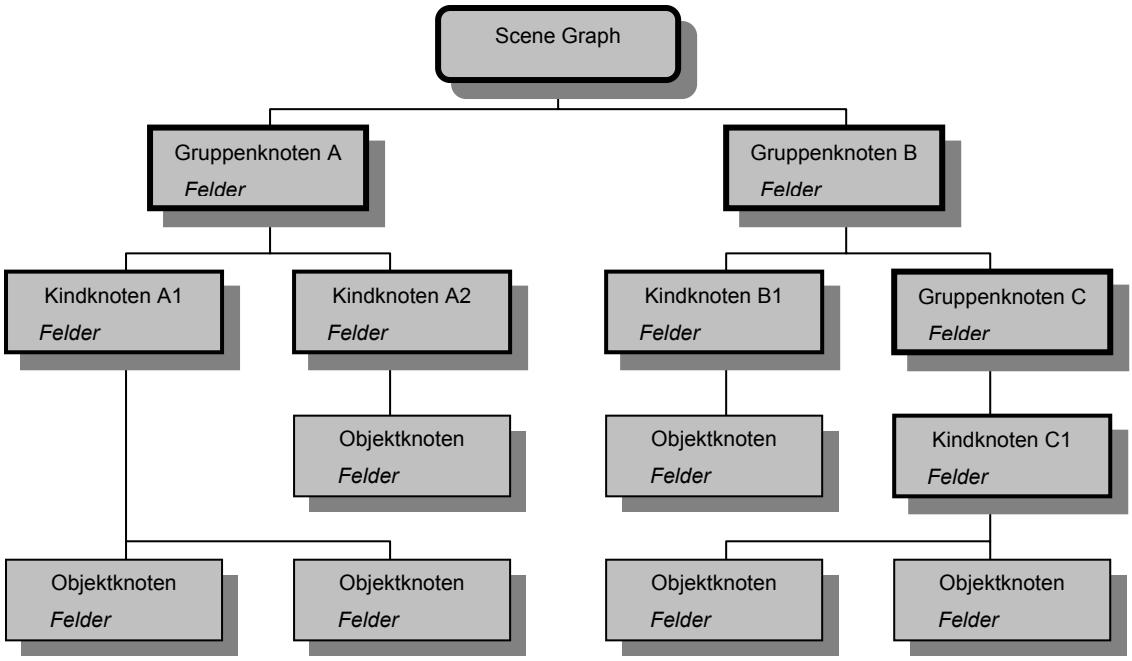

Abbildung 16 – Scene Graph

Gruppenknoten befinden sich auf der höchsten Ebene der Hierarchie und manipulieren alle untergeordneten Kindknoten, wobei Kindknoten ebenfalls Gruppenknoten sein können. Im folgenden Beispiel (Abb. 17) ist die Definition einer Kugel dargestellt. Über Transform wird das Objekt im dreidimensionalen Raum positioniert, appearance beschreibt das farbliche Aussehen und geometry die geometrische Form des Objekts.

```
Transform {
  children  Shape {
    appearance    Appearance { material Material { } }
    geometry      Sphere { radius 2 }
  }
  translation     0 0 0
}
```

Abbildung 17 – Definition einer Kugel in VRML

Kindknoten (Shape) enthalten Objektknoten (Sphere), die von dem übergeordneten Gruppenknoten (Transform) beeinflusst werden. Jeder Gruppen-, Kind- und Objektknoten besitzt Felder (children, material, radius, translation), die einen Datentyp (float, integer, character etc.) besitzen und verschiedene Informationen beinhalten. So sind sowohl Daten über untergeordnete Knoten als auch die Eigenschaften des Knotens selbst gespeichert. In den Feldern der Objektknoten wird das spezifische Aussehen und Verhalten bestimmt.

Vererbung

Alle in VRML definierten Knoten sind von ihren übergeordneten Knoten abhängig. Veränderungen, die an einem Gruppenknoten vorgenommen werden, wirken sich direkt auch auf seine untergeordneten Kindknoten und die wiederum untergeordneten Objektknoten aus. Als Beispiel wäre eine Verschiebung eines Gruppenknotens im Raum zu nennen: ein Objekt (Kindknoten) mit der Form einer Kugel (Objektknoten) verändert seine Position (Gruppenknoten).

Instanziierung und Prototypen

VRML bietet eine Reihe von Mechanismen, um einmal definierte Knoten mehrmals innerhalb einer Szenerie zu verwenden. Der erste wird Instanziierung genannt. Bei diesem Verfahren werden Objekte mittels des DEF-Kommandos definiert und mit einem Namen versehen. Auf diese Objekte kann dann beliebig oft mit der Anweisung USE referenziert werden (Abb. 18).

```
DEF Kasten Transform {                  # Kasten
  translation 1 1 1
  children  Shape {
    appearance   Appearance { material Material { } }
    geometry     Box { size 2 4 2 }
  }
}
DEF Kasten_2 Transform {                # Kasten_2
  translation 3 2 1
  scale 0.5 1 1
  children USE Kasten                   # Referenz auf Kasten
}
```

Abbildung 18 – DEF/USE-Konstrukt

Der Vorteil dieser Methode liegt darin, dass ein Objekt nur ein einziges Mal im Sourcecode beschrieben werden muss, aber trotzdem beliebig oft in der Welt eingesetzt werden kann. Die instanziierten Objekte müssen jedoch nicht identisch sein, sondern lassen sich beliebig skalieren und in der Oberflächenbeschaffenheit variieren. Durch Instanzen kann Speicherplatz gespart und damit die Übertragung der Daten beschleunigt werden.

Der zweite Mechanismus ist die Verwendung von Prototypen. Sie werden in der Regel eingesetzt, um den Sprachumfang von VRML mit eigenen, selbst definierten Knoten zu

erweitern. Komplexe Geometrien lassen sich aus mehreren Knoten einmal definieren und zu einem Prototypen zusammenfassen. Anschließend kann dieser beliebig oft in der Szene verwendet werden. Wie auch bei instanziierten Objekten lassen sich Ausmaße, Farben oder Texturen variieren. Ein Prototyp wird mit der Anweisung PROTO spezifiziert und setzt sich aus Deklaration und Definition zusammen (Abb. 19). Über die Deklaration werden die Eigenschaften des durch den Prototypen erzeugten Objekts festgelegt, die später manipuliert werden können. Über die Definition werden die zu erzeugenden Geometrien und deren Eigenschaften definiert.

```
PROTO Kasten [
  field SFVev3f Groesse 4 2 1 ] {      # Deklaration
  Box { size IS Groesse } }            # Definition
Group { children [
  Shape {
    appereance Appereance { material Material {
      diffuseColor 1 0 0 } }
  geometry Kasten { Groesse 5 3 2 }    # Instanz
  }
] }
```

Abbildung 19 – PROTO-Definition

Im Unterschied zum DEF/USE-Konstrukt wird durch Definition eines Prototyps kein Objekt erzeugt, sondern lediglich ein neuer Knotentyp bereitgestellt. Da bei komplexeren Prototyp-Definitionen die Übersichtlichkeit des Sourcecodes der VRML-Datei leidet, lassen sich solche Definitionen in externe Dateien auslagern. Um einen so ausgelagerten Prototypen zu verwenden, sieht VRML die Anweisung EXTERNPROTO vor.

Animation und Interaktion

Damit der Benutzer interaktiv agieren kann, werden Mechanismen benötigt, um Ereignisse abzufangen und zu behandeln. In VRML ist das durch die sogenannten Sensor-Knoten realisiert. Diese können benutzt werden, um durch den Anwender ausgelöste Ereignisse zu behandeln. Man unterscheidet drei Kategorien von Sensoren:

1. Generelle Sensoren (ProximitySensor, VisibilitySensor, TimeSensor, LOD, Collision)
2. Zeigegerät-Sensoren (Anchor, TouchSensor)
3. Drag-Sensoren (CylinderSensor, PlaneSensor, SphereSensor)

In der ersten Kategorie finden sich Sensoren, die automatisch Ereignisse auslösen. Der ProximitySensor (Annäherungssensor) reagiert, wenn der Anwender einen definierten Bereich um den Sensor betritt. Mit dem VisibilitySensor können Objekte ein- oder ausgeblendet werden, je nachdem ob diese Objekte im Sichtbereich des Benutzers liegen oder nicht. TimeSensoren eignen sich, um Aktionen zeitgesteuert ablaufen zu lassen.

Beispielsweise lassen sich so Simulationsabläufe oder Animationssequenzen steuern. Mit dem Collision-Knoten kann eine Kollision des Avatars mit Objekten erfasst werden, um reale physikalische Effekte zu erzeugen. Der LOD-Knoten dient der Optimierung der Szene und wird im nächsten Abschnitt erläutert.

Die Zeigegerät-Sensoren der zweiten Kategorie und die Drag-Sensoren der dritten Kategorie werden zur Erfassung von Ereignissen benutzt, die durch direkte Aktionen des Anwenders ausgelöst werden. Anchor und TouchSensor melden, wenn der Benutzer Objekte mit der Maus anklickt oder berührt. Dabei dient der Anchor-Knoten der Integration von Hyperlinks in die VRML-Welt, mit dem TouchSensor lassen sich zum Beispiel Schalter realisieren. Über den PlaneSensor können Objekte mit der Maus verschoben werden. CylinderSensor und SphereSensor werden benutzt, um Objekte zu drehen. Der CylinderSensor ist dabei auf die Drehung um eine Achse beschränkt, wogegen der SphereSensor die Rotation um alle Achsen ermöglicht.

Die meisten in VRML existierenden Knoten besitzen eventIn-Felder, mit denen Ereignisse empfangen werden können, beispielsweise um den aktuellen Wert eines Feldes zu ändern. Einige Knoten in VRML (z.B. Sensoren) besitzen außerdem noch eventOut-Felder, um Ereignisse zu erzeugen. Durch die Verbindung zwischen eventOut- und eventIn-Feldern, ROUTE genannt, werden Ereignisse zu den entsprechenden Knoten, die von dem Ereignis betroffen sein sollen, geleitet.

```
ROUTE NodeName.eventOut TO NodeName.eventIn
```

Für die Erstellung von Animationen stehen in VRML verschiedene Interpolationsmethoden zur Verfügung, die durch sogenannte Interpolatoren Animationen erzeugen. So verändert der ColorInterpolator die Farbwerte eines Knotens, mit dem CoordinateInterpolator lassen sich dagegen Koordinatenwerte linear interpolieren. Auf diese Weise lässt sich beispielsweise ein Morphing-Effekt erzeugen. Position und Orientierung können mittels des Position- und OrientationInterpolators interpoliert werden, um ein Objekt auf einer vordefinierten Bahn zu bewegen. Wenn einzelne Parameter zu interpolieren sind, findet der ScalarInterpolator Verwendung. Mit dem NormalInterpolator lassen sich Flächennormalen transformieren, beispielsweise wenn Reflexionseigenschaften von Oberflächen simuliert werden sollen.

Scripting

Im Zusammenhang mit Sensoren ist die Möglichkeit des Scriptings zu nennen. Mittels Scriptsprachen kann die Funktionalität von VRML erweitert werden, um beispielsweise Reaktionen auf Ereignisse zu implementieren. Der Script-Knoten enthält ausführbare Funktionen, die in einer bestimmten Programmiersprache (z.B. Java, JavaScript) geschrieben sind. Mit Hilfe von Scripten können von Sensoren abgefangene Ereignisse

verarbeitet und neue Ereignisse erzeugt werden. Am Beispiel eines TouchSensors wird das eventOut touchTime mittels ROUTE an das Script weitergeleitet, dort wird eine entsprechende Funktion ausgeführt und ein eventIn an den Zielknoten gesendet.

Optimierung

VRML wurde unter den Gesichtspunkten der einfachen Programmierung, der verteilten Datenhaltung und vor allem der optimierten Datenübertragung entwickelt. Zu der Zeit, als die Sprache entstand, waren multimediale Home-PCs noch nicht vorhanden und große Bandbreiten im Internet nur für Universitäten oder Forschungseinrichtungen zugänglich. Ziel der Optimierung ist es, die Downloadzeiten zu reduzieren und letztendlich die Performance auf dem Client-Rechner zu steigern. Zu den Möglichkeiten der Optimierung gehören vor allem die bereits oben beschriebenen Mechanismen zur Verwendung von Instanzen und Prototypen. Darüber hinaus lassen sich VRML-Welten mittels des ZIP-Algorithmus komprimieren. Die deutlich verringerten Dateigrößen beschleunigen die Übertragung der Daten enorm. VRML-Browser erkennen automatisch, ob es sich um eine komprimierte VRML-Datei handelt und dekomprimieren sie entsprechend, bevor sie angezeigt wird. Erwähnung verdienen auch verschiedene andere Möglichkeiten die Darstellung zu beschleunigen. So lassen sich Objekte der VRML-Welt in gesonderte Dateien auslagern. Mit Hilfe des INLINE-Knotens können sie unter Angabe einer URL später wieder in die Welt eingefügt werden.

```
Inline { url "http://www.dummy.com/object.wrl" }
```

Der Vorteil dieser Methode liegt vor allem bei Objekten, die beim Eintritt in die Welt noch nicht sichtbar sind, da sie zunächst noch nicht geladen werden müssen. Dadurch wird die Ladegeschwindigkeit insgesamt beschleunigt. Bewegt man sich anschließend in Sichtweite dieser Objekte, so werden sie nachträglich geladen. Das subjektive Zeitempfinden des Ladevorgangs ist für den Anwender angenehmer als wenn eine einzelne große Datei geladen werden müsste. Eine andere Option zur Verbesserung der Darstellungsgeschwindigkeit ist der Einsatz von Levels-of-Detail (LOD). Bei diesem Verfahren sind in der VRML-Datei mehrere, unterschiedlich detaillierte Versionen desselben Objekts gespeichert. Abhängig vom Abstand des Betrachters wird entweder das am meisten

detaillierte (kurze Entfernung) oder das einfachste Modell (große Entfernung) dargestellt. Alle Objekte, die zu derselben Levelstufe des LOD gehören, lassen sich wiederum in INLINE-Knoten zusammenfassen, werden also erst geladen, wenn sie sichtbar sind (Abb. 20).

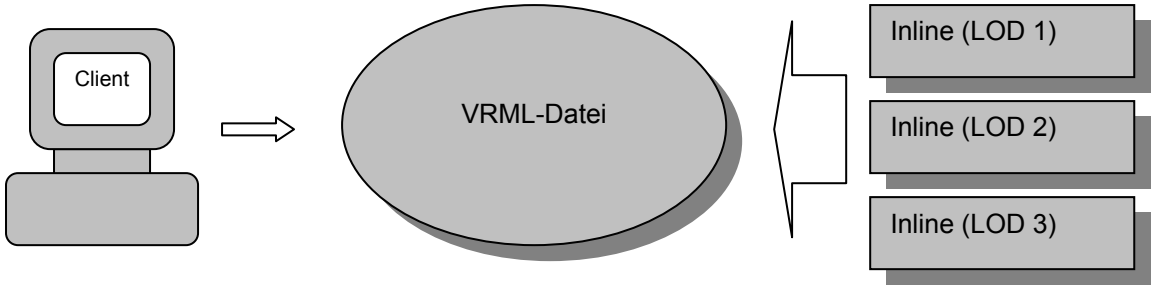

Abbildung 20 – LOD's in INLINE-Knoten

3 Realisierung des 3D-Informationssystems

3.1 Anforderungsanalyse

Eine Anforderungsanalyse muss in Anbetracht eines benutzerfreundlichen Systems aus der Sicht der zukünftigen Anwender vorgenommen werden. Besonderes Augenmerk liegt dabei auf den zu unterstützenden Funktionen, die für die unterschiedlichen Benutzergruppen (Besucher, Historiker, 3D-Modellierer) vorgesehen sind.

Dem Besucher soll die Möglichkeit gegeben werden, über eine geeignete Benutzerschnittstelle verschiedene Funktionen aufzurufen. Neben der Möglichkeit, interaktiv durch das 3D-Modell zu wandern, erhält er zu jedem Gebäude Informationen, die über spezielle Informationstafeln angezeigt werden. Des Weiteren kann er zwischen verschiedenen Zeitepochen wählen. Mit dieser Funktion sollen beispielsweise Veränderungen in der Architektur sichtbar gemacht werden. Der Historiker hat über die Informationstafeln die Möglichkeit, die modellierten Gebäude mit historischem Bildmaterial zu vergleichen. Zusätzlich wird ihm ein Werkzeug in die Hand gegeben, mit dem er einfache Längenmessungen vornehmen und so Höhe und Breite von Gebäuden bestimmen kann. Dem 3D-Modellierer wird ein Format geboten, mit dem auf einfache Weise weitere Modelle in das Gesamtsystem eingebunden werden können. Als Grundlage des Informationssystems dienen 3D-Modelle des historischen Berlins, die von Studenten im Zuge der Lehrveranstaltung Simulation / Virtual Reality erstellt wurden. Da das System aus vielen Einzelmodellen besteht, gilt es, alle Modelle in ein Gesamtmodell zu integrieren. Dazu gehört auch die Optimierung und Anpassung der Modelle, um ein einheitliches Format zu erhalten.

Nicht zuletzt muss eine geeignete Benutzerschnittstelle entworfen werden, um einen plattformübergreifenden Einsatz des Informationssystems zu gewährleisten. Als Einsatzort für das System sind sowohl das Internet und Intranet als auch die CD-ROM vorgesehen. Dabei kann die Präsentation auch in Museen oder an speziellen Infoterminals erfolgen.

3.2 Modellieren der 3D-Modelle

Das in dieser Arbeit beschriebene 3D-Informationssystem der Stadt Berlin, setzt sich aus vielen Einzelmodellen zusammen. Die Einzelmodelle umfassen in der Regel ein Gebäude, in einigen Fällen aber auch Gebäudegruppen. Jedes dieser Modelle muss getrennt modelliert werden. Hierbei ist darauf zu achten, dass das Modell möglichst detailgetreu und realistisch gestaltet wird. Speziell bei Gebäuden, die heute nicht mehr existieren oder in der Vergangenheit bauliche Veränderungen erfahren haben, muss der 3D-Modellierer besondere Sorgfalt walten lassen. Im folgenden Abschnitt werden Möglichkeiten zur Modellierung aufgezeigt. Dabei geht es um die Beschaffung von Bildmaterial, den

eigentlichen Vorgang des Modellierens und die Integration in das Gesamtmodell. Eine allgemeingültige Vorgehensweise lässt sich dabei jedoch nicht festlegen. Daher wird der Prozess des Modellierens beispielhaft am Kronprinzenpalais veranschaulicht.

3.2.1 Recherche nach Bildquellen

Das 3D-Informationssystem soll den Stand des 18.Jahrhunderts und die Gegenwart repräsentieren. Da es sich bei den zu modellierenden Modellen meist um historische Gebäude handelt, gilt es zuerst, Bildquellen ausfindig zu machen, die Informationen über das damalige Aussehen enthalten. Es gibt mehrere Abbildungsformen, die in den einzelnen Zeitepochen benutzt wurden. Dazu zählen vor allem:

- Gemälde, Zeichnungen
- Kupferstiche
- Fotografien
- Grundrisse, Pläne

Die Kunst des Malens und Zeichnens existiert bereits seit den Anfängen der Menschheit. Deshalb sind auch über den gesamten Zeitraum, den das Informationssystem abdecken soll, entsprechende Dokumente vorhanden. Der Kupferstich wurde um 1420 erfunden und damit gibt es auch aus der Zeit des 18. Jahrhunderts zahlreiche Werke, die mit dieser Technik geschaffen wurden. Die Fototechnik wurde demgegenüber erst 1844 erfunden. Daher existieren bis zu deren Entwicklung ausschließlich Gemälde, Zeichnungen und Kupferstiche. Seit Ende des 19. Jahrhunderts lösten Fotografien diese Abbildungsformen weitgehend ab. Ebenso wichtig wie Abbildungen der Gebäude sind Stadtpläne. Diese sind nötig, um die Position nicht mehr existenter Gebäude zu erhalten. In Archiven finden sich Exemplare die auf das Jahr 1650 datiert sind.

Als Quelle für historische Bilddokumente bieten sich Archive an. In alten Aufzeichnungen sind zahlreiche Abbildungen zu finden. Neben der Recherche im Archiv eignen sich insbesondere auch Bildbände der Stadt Berlin zur Informationssammlung. Die darin gezeigten Abbildungen sind meist den Archiven entnommen und bieten so eine kompakte Informationsquelle, die praktisch jedermann zugänglich ist. Darüber hinaus ist bei Bildbänden die Möglichkeit des Scannens der Vorlagen gegeben. Das ist ein großer Vorteil bei der Texturgewinnung. Das recherchierte Bildmaterial bietet dem Modellierer zahlreiche Anhaltspunkte, um das damalige Aussehen zu rekonstruieren. Allerdings muss darauf geachtet werden, dass speziell bei Kupferstichen, Gemälden und Zeichnungen nicht in jedem Falle gewährleistet ist, dass die Abbildungen wahrheitsgetreu sind. Viele Kunstwerke der damaligen Zeit sind von den Künstlern geschönt oder schlichtweg falsch dargestellt worden. Unter [12] ist im Internet ein Bilderarchiv für Kunst und Architektur zu finden. Die

angeschlossene Datenbank erlaubt die Suche nach verschiedenen Kriterien. Durch die Möglichkeit, Orte direkt auszuwählen, gelangt man schnell zu einer umfangreichen Sammlung von Bilddokumenten über Berlin. Über Untermenüs ist es möglich, recht einfach spezifische Informationen über einzelne Gebäude zu erhalten.

Die bis hier beschriebenen Möglichkeiten beschränken sich auf historisches Bildmaterial. Um einen Eindruck vom heutigen Aussehen historischer Gebäude zu erhalten, eignen sich vielfältige Medien. Neben aktuellen Ansichtskarten existieren auch hierfür Bildbände. Eine weitere Möglichkeit ist das Anfertigen eigener Fotos. Angesichts der inzwischen guten Qualität und der einfachen Weiterbearbeitung, bietet sich hierfür die Digitalfotografie an. Da die auf diese Weise gewonnen Bilddaten ebenso zum Erstellen von Texturen genutzt werden können, gilt es, einige Regeln zu beachten. Die Gebäude müssen möglichst ohne Verzerrungen fotografiert werden, da verzerrte Aufnahmen für die Berechnung von Abmessungen und Gewinnung von Texturen nachbearbeitet werden müssen. Des Weiteren muss darauf geachtet werden, dass Spiegelungen oder Schatten vermieden werden, weil diese später störend wirken. Fenster, Türen, Reliefs, Statuen und andere Details sollten zusätzlich in einer Nahaufnahme festgehalten werden. Ein wichtiges Kriterium für die Authentizität der Modelle ist die Einhaltung der richtigen Abmessungen. Diese können entweder aus Grundrissen entnommen werden, müssen am echten Gebäude nachgemessen werden oder, falls dies nicht möglich ist, anhand der Abbildungen und Fotos abgeschätzt werden.

Die Notwendigkeit der Recherche zeigt sich in besonderem Maße am Modell des Kronprinzenpalais. Das Gebäude existiert zwar heute noch, wurde jedoch im 19.Jahrhundert umgebaut, so dass heutige Aufnahmen ein verfälschtes Abbild darstellen. Die meisten historischen Abbildungen davon konnten in Bildbänden und dem Internet-Archiv gefunden werden. Darunter fanden sich zahlreiche alte Kupferstiche, Gemälde und Fotografien. Als Vergleich wurden eigene Fotografien angefertigt. Ein weiteres Beispiel ist das Berliner Stadtschloss, das im Zweiten Weltkrieg stark zerstört und Anfang der 50er Jahre gesprengt wurde. Hier ist man vollkommen auf alte Abbildungen angewiesen.

3.2.2 Modellieren der VRML-Modelle

Die Modellierung der VRML-Modelle kann in drei Etappen zusammengefasst werden:

1. Modellieren
2. Texturieren
3. Exportieren in VRML und Anpassen

Diese drei Etappen werden im Folgenden exemplarisch am Beispiel des Kronprinzenpalais erläutert. Dabei werden alternative Vorgehensweisen beleuchtet.

Modellieren

Die Etappe des Modellierens dient der Erstellung des eigentlichen 3D-Modells eines Gebäudes. Grundlage dafür sind die durch die Recherche gewonnen Bildinformationen. Auf diese Weise kann das Aussehen sehr gut rekonstruiert werden. Besonders wertvoll für diese Arbeit sind eventuell vorhandene Grundrisse, anhand derer die Abmessungen der Gebäude bestimmt werden können. Falls entsprechende Dokumente nicht verfügbar sind, muss die Größe anhand der Bilddokumente über Verhältnisgleichungen errechnet werden. Dabei werden unbekannte Abmessungen durch rekonstruierbare Maße ermittelt. Beim Kronprinzenpalais gehörten Säulen und Fenster zu den bekannten Maßen, da diese am Original abgenommen werden konnten. Aus der Breite kann dann die Höhe, oder die Abmessung des gesamten Gebäudes errechnet werden. Eine hundertprozentige Genauigkeit kann dadurch allerdings nicht erreicht werden. Als 3D-Modellierprogramm bieten sich mehrere Anwendungen an. In Kapitel 2.3.1 sind bereits unterschiedliche Programme beleuchtet worden. Aufgrund der erweiterten 3D-Fähigkeiten, sind 3D-Modellierer den reinen VRML-Programmen überlegen. Aus diesen Gründen wurde zur Erstellung des Modells des Kronprinzenpalais 3D-Studio MAX eingesetzt. Als Vorgehensweise zur Modellierung von Gebäuden bietet sich das Bauklötzchen-Prinzip an. Dabei wird das Modell aus vielen kleinen Teilen (Bauklötzen) zusammengefügt. Dieses Verfahren ist besonders bei Gebäuden geeignet, die sich aus einfachen Quadern zusammensetzen. Da das Kronprinzenpalais diese Charakteristik aufweist, war es für ein solches Verfahren gut geeignet. Als ungünstig erweisen sich dafür Gebäude, die eine runde Form aufweisen. Da der Quader einer der Grundprimitive von VRML ist, kann das Modell problemlos in VRML exportiert werden. Das hat zudem den Vorteil, dass ein Quader bei der späteren Darstellung im VRML-Browser dem darstellenden Rechner weniger Rechenleistung abfordert, als das komplexe Polygone tun. Die meisten 3D-Programme erlauben das Anlegen von Kopien, Referenzen und Instanzen von Objekten und ermöglichen so das mehrmalige Verwenden einmal definierter Objekte. Beispielsweise setzt sich die Fassade des Kronprinzenpalais aus vielen gleichen Quadern zusammen. Folglich ist es ausreichend einen einzelnen Quader zu definieren. Die Front des Gebäudes wird dann aus Referenzen desselben zusammengesetzt. Die meisten Gebäude weisen ähnliche Eigenheiten auf, so dass Referenzen sehr vielfältig genutzt werden können. Diese Vorgehensweise hat zudem den Vorteil, dass Referenzen auch beim späteren Export in VRML in DEF/USE-Konstrukte umgewandelt werden. Auf diese Art wird von vornherein ein optimiertes VRML-Modell ermöglicht. Das Zusammensetzen von 3D-Modellen aus vielen Teilobjekten setzt eine genaue Positionierung der Einzelteile voraus. Die Modellierprogramme erlauben die Eingabe der Koordinaten per Tastatur. In 3D-Studio MAX nennt sich die Funktion Keyboard-Entry (Abb. 21). Unter Angabe der exakten Position der Objekte lassen sich Lücken zwischen

einzelnen Teilen vermeiden, die durch Positionierung mit der Maus unweigerlich entstehen würden. Das Grundgerüst des Kronprinzenpalais wurde so aus über 150 Quadern zusammengesetzt.

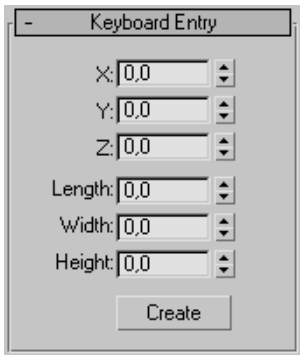

Abbildung 21 – 3DSMAX: Keyboard Entry

Für komplexere Objekte ist das Bauklötzchen-Prinzip jedoch nicht geeignet. Diese lassen sich einfacher mit Hilfe der speziellen Funktionen der 3D-Programme erstellen. 3D-Studio MAX bietet eine Reihe von Modifikatoren, mit denen Objekte auf unterschiedlichste Art und Weise manipuliert werden können. So lassen sich Objekte so stark verändern oder verformen, dass nahezu jede Form modellierbar ist. Bei der Modellierung der Gebäude des 3D-Informationssystems ist das speziell bei Dächern, Kuppeln, Statuen, Säulen oder ähnlichem erforderlich. Am Beispiel des Kronprinzenpalais wurden das Dach und die Rampe an der Frontseite des Gebäudes auf diese Weise modelliert.

Für die Modellierung der Rampe des Kronprinzenpalais kamen drei Modifikatoren zum Einsatz. Der Grundkörper der Rampe ist ein Quader, der in fünf Längensegmente unterteilt ist. Als erstes wurde der EditMesh-Modifikator zum Bearbeiten der einzelnen Oberflächen des Objektes angewandt. Die Eckpunkte der Längensegmente wurden markiert und in z-Richtung verschoben. Dadurch konnte eine Absenkung auf einer Seite erreicht werden. Im nächsten Schritt wurde der Bend-Modifikator zum Biegen der Rampe verwendet (Abb. 22).

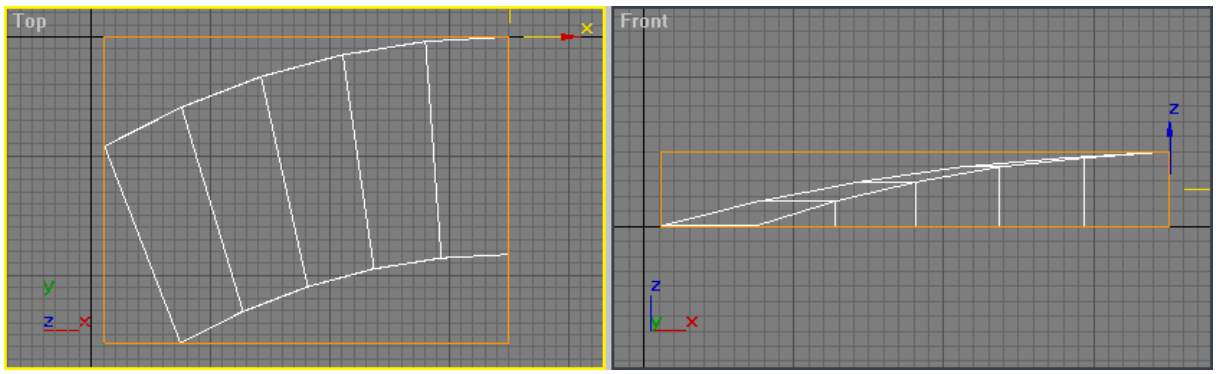

Abbildung 22 – 3DS MAX: Objekt nach Anwendung der Modifikatoren EditMesh und Bend

Dazu musste der Pivotpunkt des Objektes verschoben werden, damit die Biegung von einer Seite ausgehend erfolgen konnte. So konnte eine homogene Krümmung der Rampe modelliert werden. Als letztes kam der UVW-Mapping-Modifikator zum Einsatz. Dieser dient dem richtigen Platzieren einer Textur auf einem Objekt.

Die auf ein Objekt angewendeten Modifikatoren werden in der Reihenfolge ihrer Verwendung im sogenannten Modifikator-Stapel (Abb. 23) abgelegt.

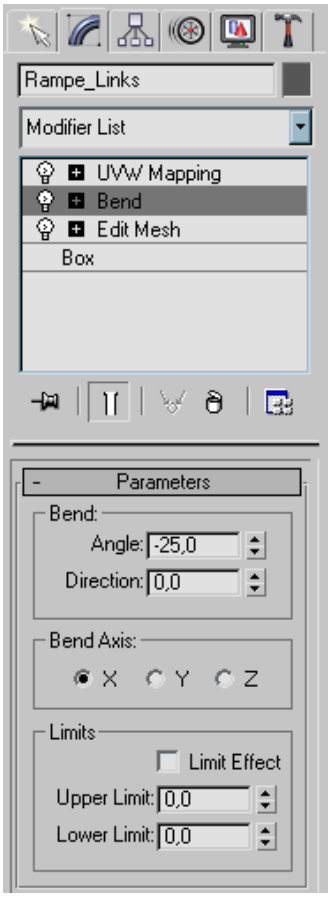

Abbildung 23 – 3DS MAX: Modifikator-Stapel (Objekt Rampe_Links)

Somit ist nachträglich der Zugriff auf jeden einzelnen Modifikator möglich. Allerdings wirken sich Veränderungen der Parameter eines weiter unten im Stapel liegenden Modifikators auch auf alle darüber liegenden aus, was oft zu unliebsamen Ergebnissen führt. Die Veränderungen sind im Voraus oft nicht abzusehen. Wenn die Modifikationen abgeschlossen sind, können die entsprechenden Objekte in der Szenerie platziert werden. Dabei muss ebenso auf Genauigkeit geachtet werden, wie beim Zusammenfügen der Fassade. Die eigentliche Modellierung ist damit abgeschlossen.

Texturieren

Abschnitt zwei beschäftigt sich mit dem Aufbringen von Texturen auf das 3D-Modell. Texturen sind Bilddateien, die auf Objekte ‚geklebt' werden, um einen realistischeren Eindruck zu erreichen. Normalerweise müssen die Texturen für die Gebäude des 3D-Informationssystems zunächst erstellt werden. Als Quelle dienen dabei Abbildungen aus Bildbänden und Archiven oder digitale Fotografien. Erstere müssen zunächst mit Hilfe eines Flachbettscanners digitalisiert werden. Da bei beiden des Weiteren ein Bearbeiten nötig ist, bieten sich verschiedene Bildbearbeitungsprogramme an. Typische Vertreter hierfür sind Adobe Photoshop 6.0 oder auch Paint Shop Pro 6.0. Beide enthalten ein TWAIN-Modul zum Scannen der Vorlagen, umfangreiche Bearbeitungsfunktionen und Filter zur Nachbearbeitung. Nach dem Digitalisieren aller Vorlagen müssen für alle Gebäudeteile (Gebäudefront, Fenster, Türen, Dach, Säulen usw.) entsprechende Texturen erstellt werden. Dafür werden die Freistellfunktionen der Bildbearbeitungsprogramme genutzt. Die entsprechenden Bildabschnitte werden zur Bearbeitung ausgeschnitten. Im weiteren Verlauf müssen verzerrte Bildteile entzerrt werden, damit sie als Texturen dienen können. Falls diese aus unterschiedlichen Abbildungen gewonnen werden, treten eventuell Abweichungen in der Farbgebung auf, die nachträglich beseitigt werden müssen, um einen gleichmäßigen Farbton aller Texturen zu erreichen. Das ist nötig, damit im späteren Modell die Texturen aller Gebäudeteile farblich harmonieren. Die Texturen des Kronprinzenpalais wurde mit Paint Shop Pro bearbeitet.

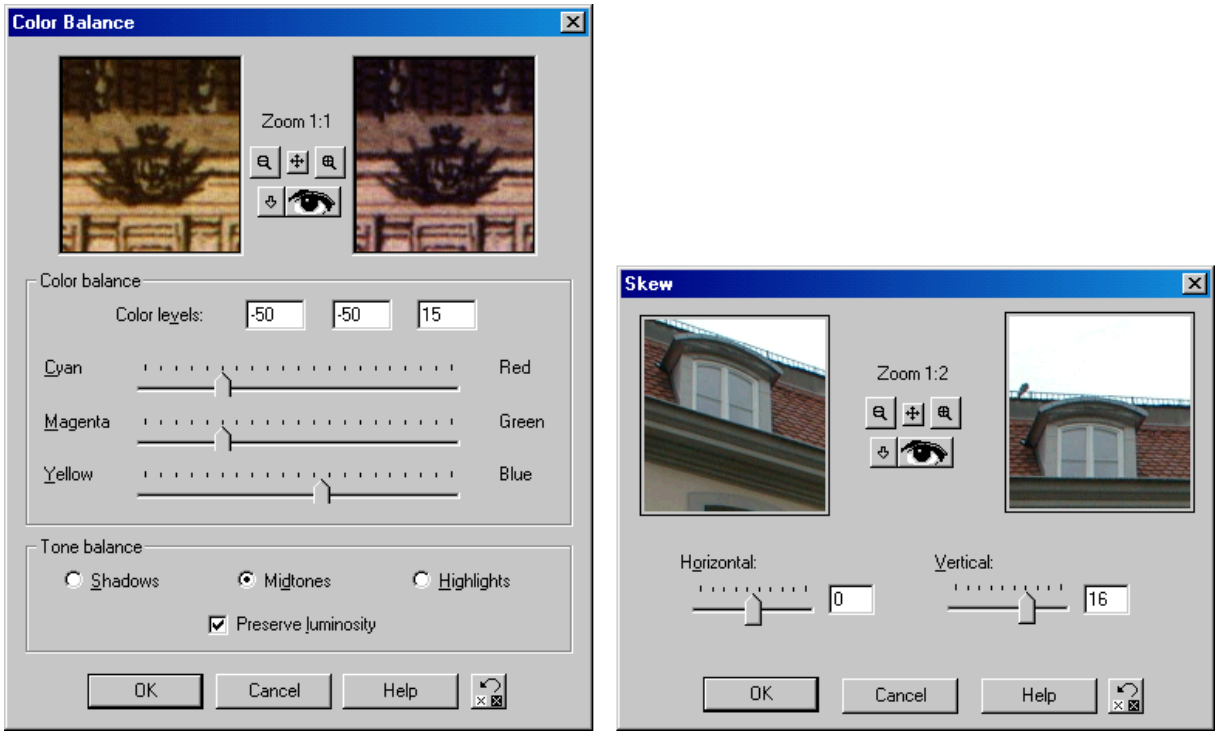

Abbildung 24 – PSP: Color Balance, Skew (v.l.)

Mit Hilfe der Skew-Funktion lässt sich ein Bildbereich in horizontaler und vertikaler Richtung verzerren. Damit kann der ursprünglichen Verzerrung entgegengewirkt und ein entzerrtes Bild erreicht werden. Mit der Funktion Color Balance besteht die Möglichkeit, die drei Farbkanäle Rot, Grün und Blau zu manipulieren und die Farbwerte anzupassen. In Abbildung 24 sind beide Funktionen dargestellt.

Bei der Modellierung historischer Gebäude sieht man sich häufig mit komplexen Statuen oder Objekten mit unregelmäßigen Umrissen (z.B. Bäume) konfrontiert. Eine Umsetzung in 3D wäre sehr aufwändig. Eine einfachere Methode ist die Verwendung transparenter Texturen. Für deren Erzeugung muss ein Farbwert aus der Palette ausgewählt werden. Dieser Wert wird in der Farbpalette als transparent gekennzeichnet. Bildbereiche mit diesem Farbwert sind in der späteren Darstellung durchsichtig. Statt nun einen komplexen Körper zu modellieren, wird im Falle einer Statue ein einfacher, zu den Grundprimitiven gehörender Zylinder definiert. Auf diesen werden die Texturen aufgetragen. Da die transparenten Bildteile nicht dargestellt werden, entsteht so ein räumlicher Eindruck der Statue. Abbildung 25 zeigt die beim Kronprinzenpalais für die Statuen verwendete Textur. Die gekachelten Bildteile sind transparent.

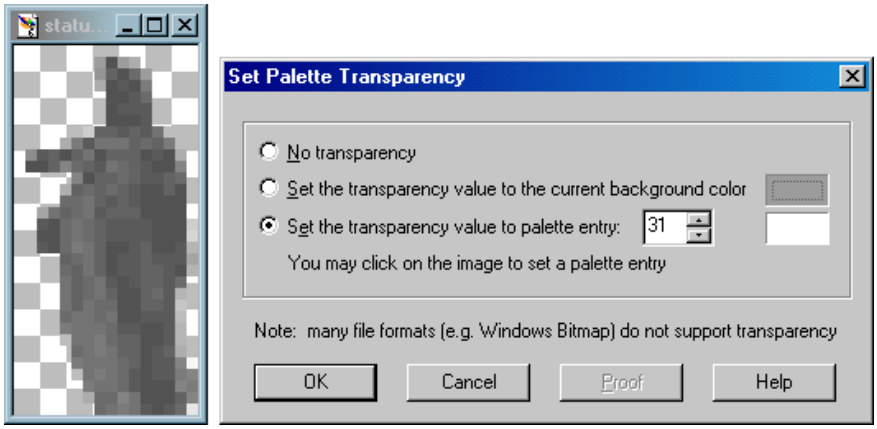

Abbildung 25 – PSP: Einstellen der Transparenzwerte

Vor dem Aufbringen der Texturen müssen unter Umständen noch die Auflösung und die Farbtiefe optimiert werden. Digitale Vorlagen (gescannte Bilder, Fotos) haben in der Regel eine Auflösung von 150 bis zu 1200 dpi. Daraus resultieren zwar sehr detaillierte aber meist recht große Bilddateien. Eine hohe Auflösung ist nur dann notwendig, wenn detaillierte Nahansichten geboten werden sollen. Dies ist jedoch im Falle des 3D-Informationssystems, in dem der Anwender meist aus genügendem Abstand die Modelle besichtigt nicht nötig. Da Monitore außerdem maximal 72 dpi anzeigen können, ist eine Reduzierung der Auflösung sinnvoll. Im gleichen Arbeitsgang sollte die Größe der Texturen angepasst werden (Abb. 26). Solange dabei keine wesentlichen Details verloren gehen, steht einer Verkleinerung nichts

entgegen. Durch diese Maßnahmen wird eine deutliche Reduzierung der Dateigröße erreicht, was eine gute Performance unterstützt.

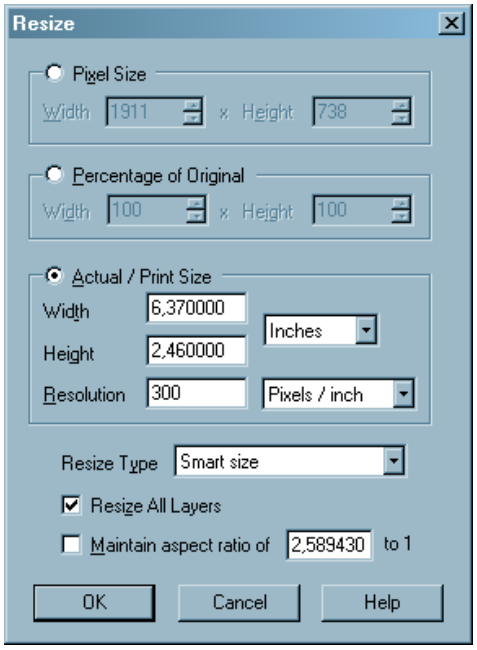

Abbildung 26 – PSP: Reduzieren von Auflösung und Größe

Zur weiteren Optimierung der Texturen sollte die Farbtiefe auf 8-Bit verringert werden. Das ergibt eine 256-Farben-Palette, die für Farbverläufe völlig ausreichend ist. Die Farbtiefe hängt jedoch auch vom verwendeten Dateiformat ab. Dafür kommen folgende Formate in Frage:

- GIF (Graphics Interchange Format)
- JPEG (Joint Photographic Expert Group)
- PNG (Portable Network Graphics)

Im GIF-Format gespeicherte Bilder haben eine Farbtiefe von 8-Bit, JPEG und PNG speichern mit 24-Bit Farbtiefe.

GIF	JPEG-10	JPEG-40	PNG	TIFF
18 KB	8 KB	4 KB	34 KB	44 KB

Tabelle 7 – Dateigrößenvergleich GIF, JPG, PNG, TIFF

Tabelle 7 zeigt die Dateigrößen einer 307 x 74 x 24 Bit Pixel großen Bilddatei in den verschiedenen Bildformaten. Qualitativ ist mit bloßem Auge kaum ein Unterschied zu erkennen. Die beim JPEG-Format verwendete Kompressionsmethode arbeitet

verlustbehaftet. Die Kompressionsrate lässt sich aber durch den Anwender selbst bestimmen, wodurch ein guter Mittelweg zwischen Qualität und Dateigröße gefunden werden kann. Das GIF-Format arbeitet mit verlustfreier Kompression und ist andererseits nötig um transparente Texturen zu speichern. Neben GIF und JPEG bietet sich noch das PNG-Format an, da es ebenfalls von allen gängigen Browsern unterstützt wird. Dieses Format ist eine Kombination aus GIF und JPEG. Es unterstützt verlustfreie Kompression, Transparenz und 24-Bit-Farbtiefe. Das Bearbeiten der Texturen dient vor allem der Optimierung der Dateigröße, wodurch eine schnellere Datenübertragung und vor allem eine schnellere Darstellung erreicht wird.

Um die Texturen letztendlich auf die 3D-Objekte aufzubringen, bedient man sich wiederum eines 3D- oder VRML-Modellierers. Diese Anwendungen bieten komfortable Oberflächen. Dabei existieren unterschiedliche Konzepte. Das VRML-Programm Cosmoworlds listet in einer Baumstruktur alle Objekte auf. Über den Eigenschaftseditor (Property Inspector) lässt sich jedes Objekt einzeln auswählen und mit einer Textur versehen. Diese können zusätzlich noch ausgerichtet werden. Einen anderen Weg geht das 3D-Programm 3D-Studio MAX. Hier wird zum Aufbringen der Texturen der Material Editor benutzt (Abb. 27).

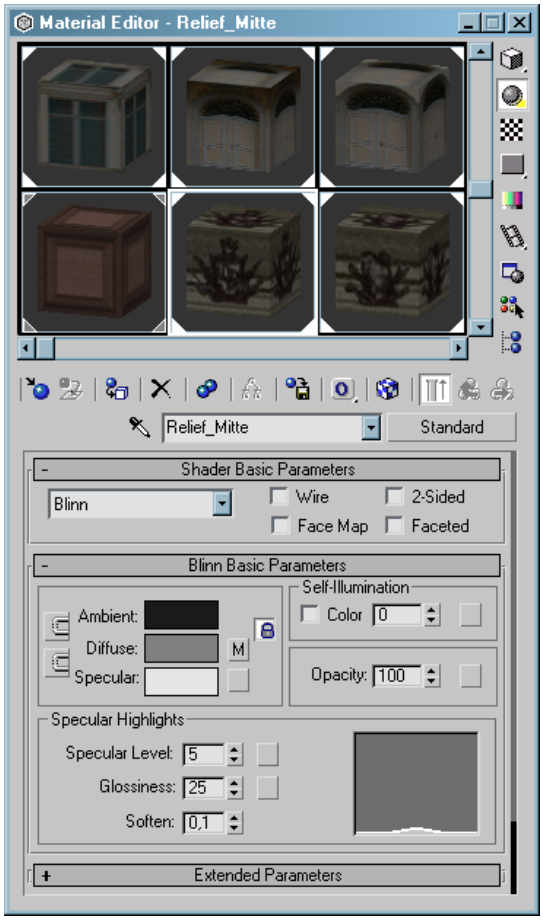

Abbildung 27 – 3DS MAX: Material Editor

Dieser erlaubt sowohl das Importieren von Texturen als auch das Erstellen neuer Texturen, die dann in der Materialbibliothek gespeichert werden. Bei der Modellierung des Kronprinzenpalais wurde eine eigene Materialbibliothek angelegt, die alle zur Texturierung nötigen Texturen enthält. Der Vorteil dieses Verfahrens liegt darin, dass man die Bibliothek exportieren und in anderen Projekten erneut verwenden kann. Um die Texturen auf Objekte zu übertragen, müssen diese ausgewählt werden. Dann können ihnen die in der Materialbibliothek gespeicherten Texturen zugewiesen werden. Bei Objekten, die durch Modifikatoren verändert wurden, kann es notwendig sein, eine UVW-Map auf die Objekte zu legen. Mit diesem Modifikator kann nachträglich die Ausrichtung der Textur beeinflusst werden, um sie richtig zu platzieren. Bei der Texturierung mit 3D-Studio MAX muss jedoch in Kauf genommen werden, dass alle als Quader definierten Körper beim späteren Export in VRML in IndexedFaceSets umgewandelt werden. Zu erklären ist dieses Verhalten damit, dass beim Auftragen der Texturen ein Quader in seine 6 Einzeloberflächen aufgeteilt wird. Sowohl Cosmoworlds als auch 3D-Studio MAX gestatten des Weiteren diverse Einstellungen, um eine Textur auf einem Objekt in x- oder y-Richtung mehrfach wiederholt aufzutragen. Damit kann dem Effekt des Verzerrens entgegengewirkt werden, der entsteht, wenn eine kleine Textur auf eine große Oberfläche aufgetragen wird.

VRML-Export und Anpassen

Der letzte Punkt der Modellierung befasst sich mit der Integration VRML-spezifischer Funktionen. Dazu gehören vor allem Level-of-Detail (LOD). Abhängig vom verwendeten Modellierprogramm muss die 3D-Szene zudem noch in das VRML-Format konvertiert werden. Die Definition von LOD ist sowohl mit Cosmoworlds als auch mit 3D-Studio MAX über entsprechende Funktionen möglich. Beiden ist gleich, dass bereits in der Modellierungsphase unterschiedlich detaillierte Modelle erstellt werden müssen. Da das Kronprinzenpalais mit 3D-Studio modelliert wurde, soll an diesem Beispiel das Vorgehen exemplarisch erläutert werden. Alle Objekte einer Detailstufe müssen zunächst in einer Gruppe zusammengefasst werden. Unter der Kategorie Helper findet sich in 3D-Studio auch VRML97. Hier muss nun ein LOD-Helper definiert werden. Über die Eigenschaften dieses Helpers können die unterschiedlich detaillierten Gruppen dem LOD zugeordnet werden. Des Weiteren werden hier die Sichtbarkeitsbereiche definiert (Abb. 28), wodurch jede Detailstufe nur in dem ihm zugeordneten Bereich sichtbar ist.

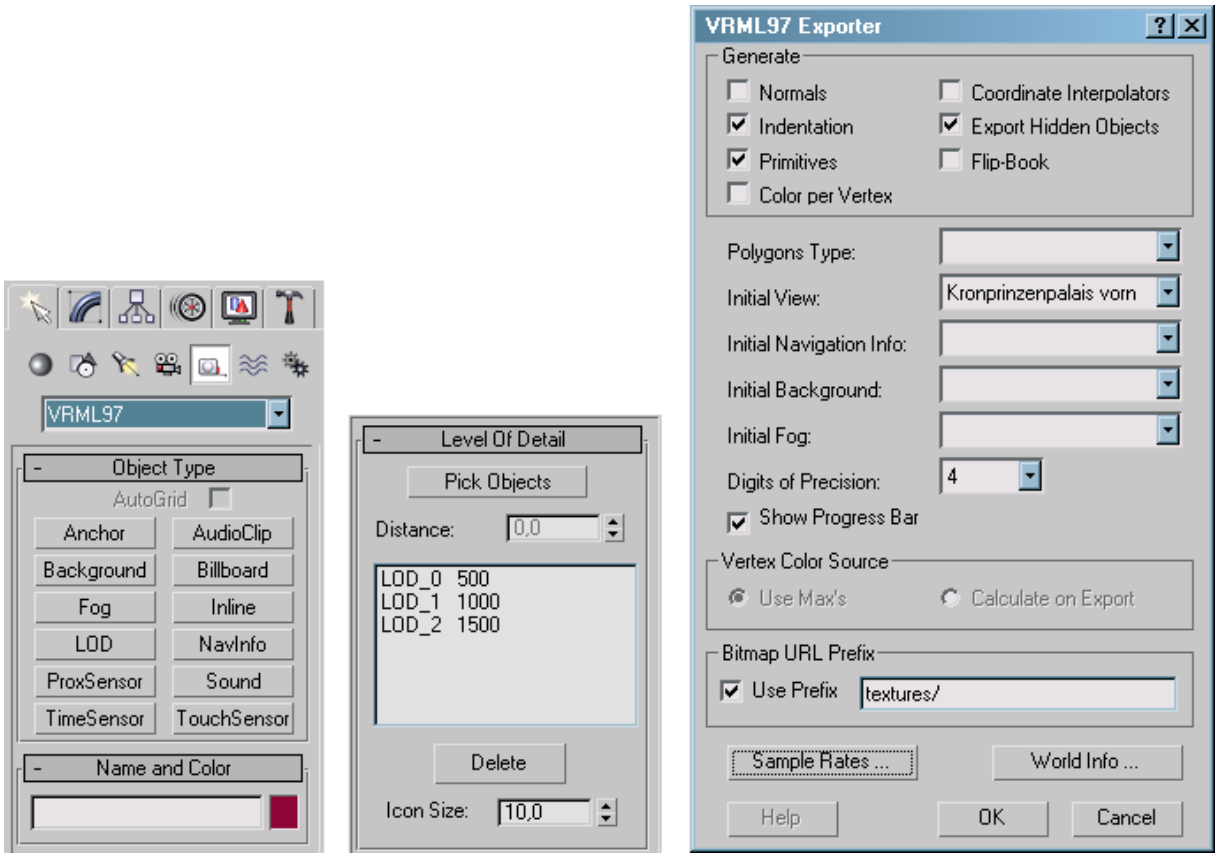

Abbildung 28 – 3DS MAX: VRML-Helper, LOD, VRML-Exportfilter (v.l.)

Da 3D-Studio MAX ein eigenes Dateiformat verwendet, müssen alle 3D-Modelle in das VRML-Format umgewandelt werden. Für diesen Zweck bietet das Programm einen Export-Filter an, mit dem in 3D-Studio erzeugte Modelle direkt im VRML97-Format gespeichert werden können. Der Exporter gestattet verschiedene Einstellungen, um beispielsweise den Startpunkt der VRML-Szene festzulegen. Dafür kann eine in 3D-Studio angelegte Kamera ausgewählt werden, die in der fertigen VRML-Szene als Navigationspunkt fungiert. Des Weiteren besteht die Möglichkeit, den Pfad für die Texturen anzugeben. Erstellte Lichtquellen werden ebenfalls exportiert und in entsprechende VRML-Lichtquellen umgewandelt. Auf Lichtquellen wurde jedoch bewusst verzichtet, da die Beleuchtung des entstehenden VRML-Gesamtmodells im Hauptmodell des Systems definiert wird. Eine zusätzliche Beleuchtung jedes Einzelmodells würde nur ungewollte Lichteffekte oder Überstrahlungen bewirken. Nach dem Überführen des Modells in das VRML97-Format sind einige weitere Anpassungen nötig. Wird beim Modellieren auf einen Meter-Maßstab verzichtet oder sind die Abmessungen nicht bekannt, ist mit hoher Wahrscheinlichkeit die Größe des Modells unter- oder überdimensioniert. Aus diesem Grund muss das Gebäude entsprechend skaliert werden. Des Weiteren muss es in die richtige Position gebracht werden. Für diese Anpassungen eignet sich Cosmoworlds am besten. Um die richtige Größe und Position zu bestimmen, wird das Hauptmodell, welches lediglich die Beleuchtung, das

Userinterface und einen Grundkörper (mit Stadtplan als Textur) enthält, temporär importiert. Anhand des Stadtplans kann das Modell nun an die richtige Position verschoben und seiner wirklichen Größe entsprechend skaliert werden. Anschließend wird das Hauptmodell wieder entfernt. Mit diesem letzten Schritt ist das Modell fertig modelliert und soweit vorbereitet, dass es in das Gesamtmodell eingefügt werden kann.

3.3 Erstellen des Gesamtmodells

Wie bereits angesprochen, wird das Gesamtmodell aus einzelnen Modellen der Gebäude zusammengesetzt. Um von vornherein eine optimierte Lösung anzubieten, werden die Daten der Einzelmodelle in separaten Dateien vorgehalten. Die Hauptdatei des Systems enthält lediglich die Grundplatte, auf der alle Gebäude Platz finden, die Benutzerschnittstelle, Beleuchtung, Informationen zur Navigation und Verweise auf die Einzelmodelle. Die Grundstruktur des 3D-Informationssystems ist in Abbildung 29 schematisch dargestellt.

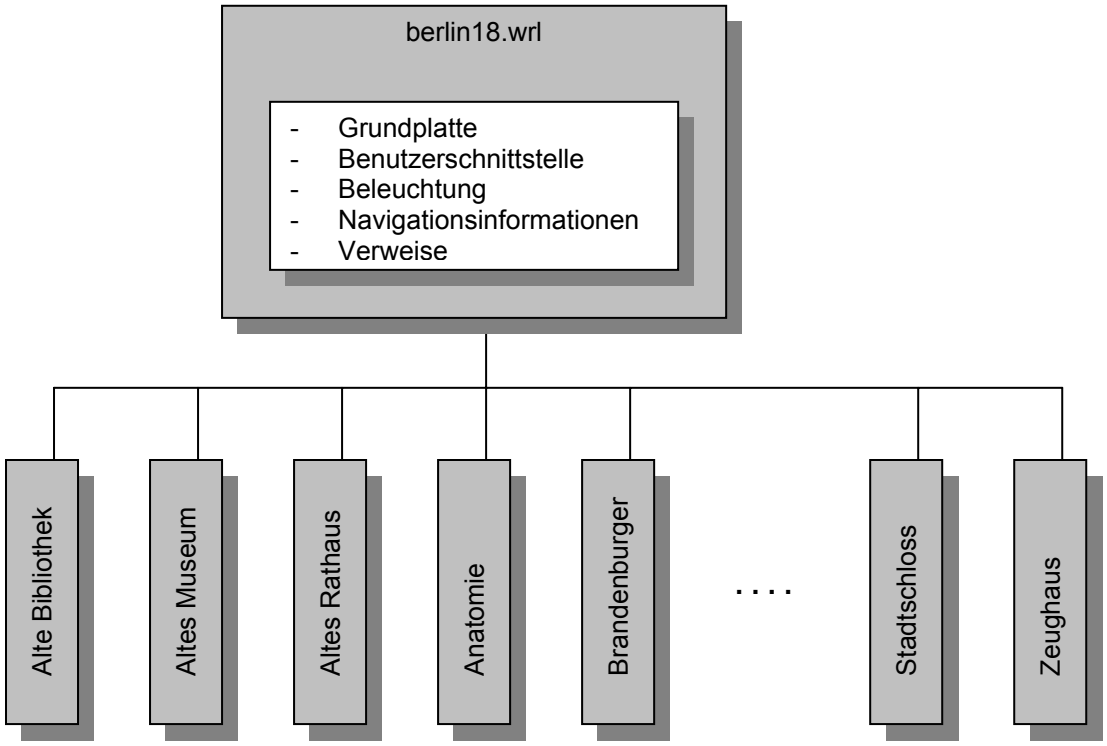

Abbildung 29 – Grundstruktur des 3D-Informationssystems

Dieser Aufbau hat mehrere Vorteile. Erstens muss beim Starten des Modells nicht das gesamte System mit allen Modellen geladen werden, sondern nur die Teile, die zu Beginn sichtbar sind. Des Weiteren hat man die Möglichkeit, die Modelle verteilt im Netz zu speichern. Das erleichtert im Zuge der Modellierung eines Gebäudes die Arbeit während der Testphase erheblich, da zur Integration des Modells lediglich einige wenige Codezeilen

einzufügen sind. Drittens wird auf diese Weise eine sehr einfache Möglichkeit geschaffen, das Modell zu erweitern, da nicht jedes Mal das Gesamtsystem neu bearbeitet werden muss.

Die einzelnen Modelle, die zur Entwicklung des 3D-Informationssystems nötig sind, standen zu Beginn der Entwicklungszeit bereits zur Verfügung. Sie wurden in einer übersichtlichen Dateistruktur zusammengefasst. In der Hauptdatei des Systems wurden sie dann an entsprechender Stelle mittels des INLINE-Knotens eingebunden. Abbildung 30 zeigt einen Ausschnitt des Gesamtmodells nach Integration aller Gebäude.

Zunächst mussten jedoch noch einige Anpassungen vorgenommen werden. Bei der Modellierung der einzelnen Gebäude konnte in Ermangelung eines entsprechenden Systems nicht getestet werden, wie sich das Zusammenführen aller Modelle auf die Performance auswirkt. Entsprechend langsam waren die Darstellung und die Ladezeit. Sowohl die Texturen als auch die VRML-Modelle wurden verschiedenen Optimierungsalgorithmen unterzogen. Um später solche Engpässe von vornherein auszuschließen, sollte dem 3D-Modellierer eine Möglichkeit gegeben werden entsprechende Tests durchzuführen. Dies könnte beispielsweise durch das Herunterladen des Gesamtsystems als ZIP-Datei erfolgen.

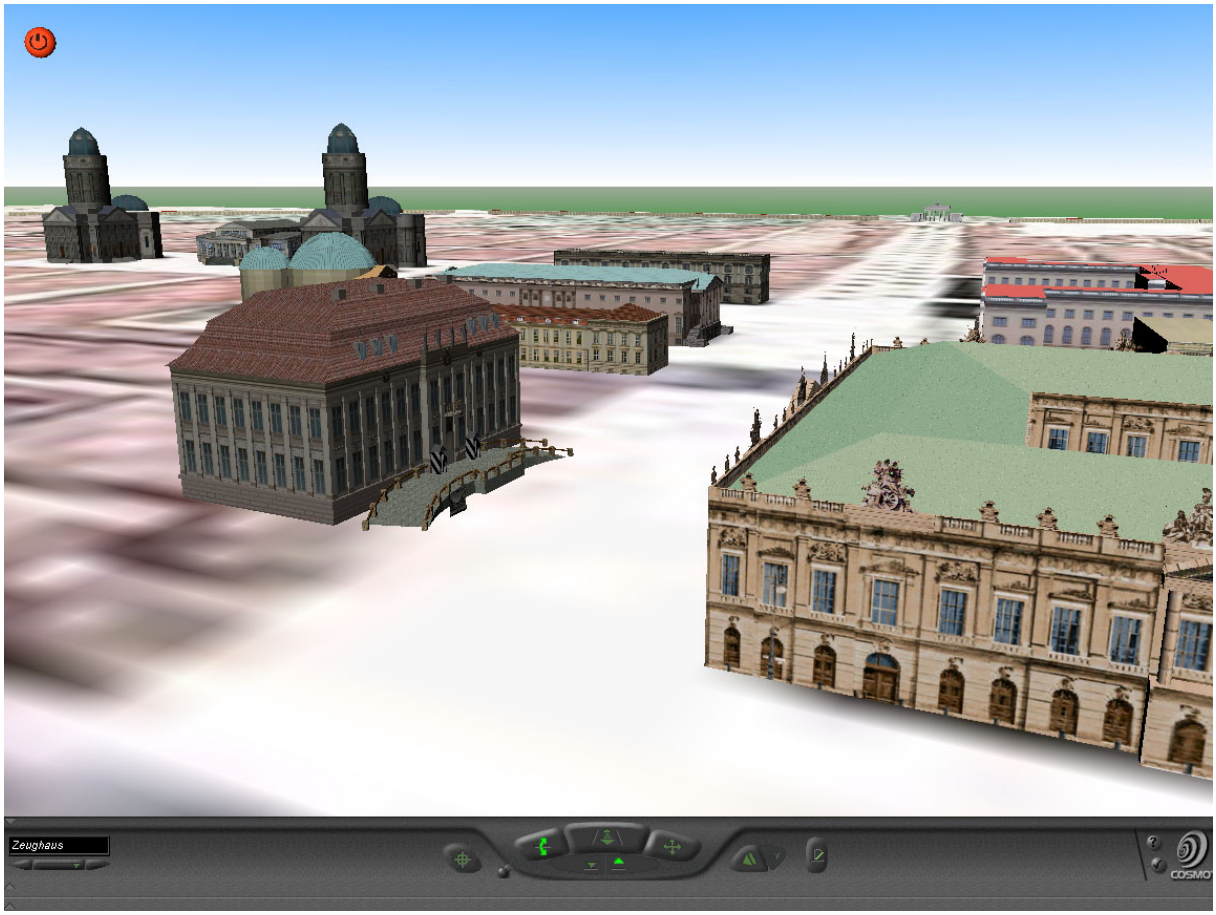

Abbildung 30 – Ausschnitt aus dem Gesamtmodell (Unter den Linden)

3.3.1 Nachbearbeiten der Modelle

Die Notwendigkeit des Nachbearbeitens der Einzelmodelle ergab sich aus der sehr schlechten Performance des Informationssystems. Um in dieser Hinsicht Fortschritte zu machen, galt es vor allem Texturen herunterzurechnen und die VRML-Modelle in ihrer Größe und der Zahl der verwendeten Polygone zu reduzieren.

Bearbeiten der Texturen

Bei der Optimierung der Texturen geht es vor allem um die Reduzierung der Ladezeit. Zu diesem Zweck ist eine Verringerung der Dateigröße zu erreichen. Die einfachste Möglichkeit dafür ist die Umwandlung von speicherintensiven Dateiformaten in platzsparendere. Die Formate JPEG, GIF und PNG sind komprimiert und eignen sich damit für den Einsatz im Informationssystem. Sie haben sich in den letzten Jahren zu einem de facto-Standard im Internet entwickelt und werden von jedem Webbrowser unterstützt. Das PNG-Format ist aber bei weitem nicht so weit verbreitet wie JPEG und GIF. Daher wurden die Texturen, sofern erforderlich, in eines dieser beiden Dateiformate umgewandelt. Neben der Umwandlung in andere Dateiformate gibt es auch andere Möglichkeiten eine Reduktion der Dateigröße zu erreichen. In Kapitel 3.2.2 sind unter der Überschrift Texturieren verschiedene Methoden aufgeführt, um sowohl Auflösung als auch Farbtiefe zu verringern. Die selben Routinen können auch hier angewendet werden.

Optimieren der VRML-Modelle

Neben den Texturen mussten auch die VRML-Modelle optimiert werden. Die Vorgabe bei der Modellierung war eine ruckelfreie Darstellung. Jedes Modell für sich ereichte diese auch. Nach dem Einbinden aller Gebäude in das Gesamtmodell, benötigte der Testrechner (PC, AMD Athlon 1000MHz) immer noch mehr als zwei Minuten um das System lokal von der Festplatte zu laden. Als Ursache der langen Ladezeit konnten die folgenden Faktoren ausgemacht werden:

1. unkomprimierte VRML-Dateien
2. hohe Anzahl von Polygonen
3. mangelnde Verwendung von DEF/USE-Konstrukten
4. wenige Inlines

Der erste Punkt konnte einfach behoben werden. Alle verwendeten VRML-Programme bieten eine Möglichkeit die 3D-Szenen in einer komprimierten Version zu speichern. In Cosmoworlds wird diese Funktion über den Menüpunkt Publish angeboten. Dabei wird der ASCII-Quelltext mit dem ZIP-Algorithmus komprimiert. Dieser Algorithmus erreicht speziell bei ASCII-Text imposante Kompressionsraten. Der VRML-Browser erkennt komprimierte

VRML-Dateien und dekomprimiert sie zur Laufzeit. Auf diese Weise lässt sich die zu übertragende Datenmenge bereits erheblich verringern. Im Durchschnitt ließen sich die VRML-Dateien etwa um den Faktor 8 verkleinern (Tabelle 8).

	unkomprimiert	komprimiert	Kompressionsrate
Brandenburger Tor	92 KB	11 KB	8 : 1
Domkirche	51 KB	6 KB	8,5 : 1
Kronprinzenpalais	178 KB	15,5 KB	11 : 1
Opernhaus	104 KB	15 KB	7 : 1
Zeughaus	48,7 KB	6 KB	8 : 1

Tabelle 8 – Kompressionsraten des ZIP-Algorithmus

Ein zweites Problem war die teilweise extrem hohe Anzahl der verwendeten Polygone. Diese ergibt sich unter anderem aus der häufigen Verwendung von IndexedFaceSets und Nichtverwendung von DEF/USE-Konstrukten. Speziell aus IndexedFaceSets resultieren viele Polygone, die dem darstellenden Computer einiges an Rechenkraft abfordern. Da deren Definition aus vielen Koordinatentripeln besteht werden auf diese Weise die VRML-Dateien zusätzlich unnötig groß.

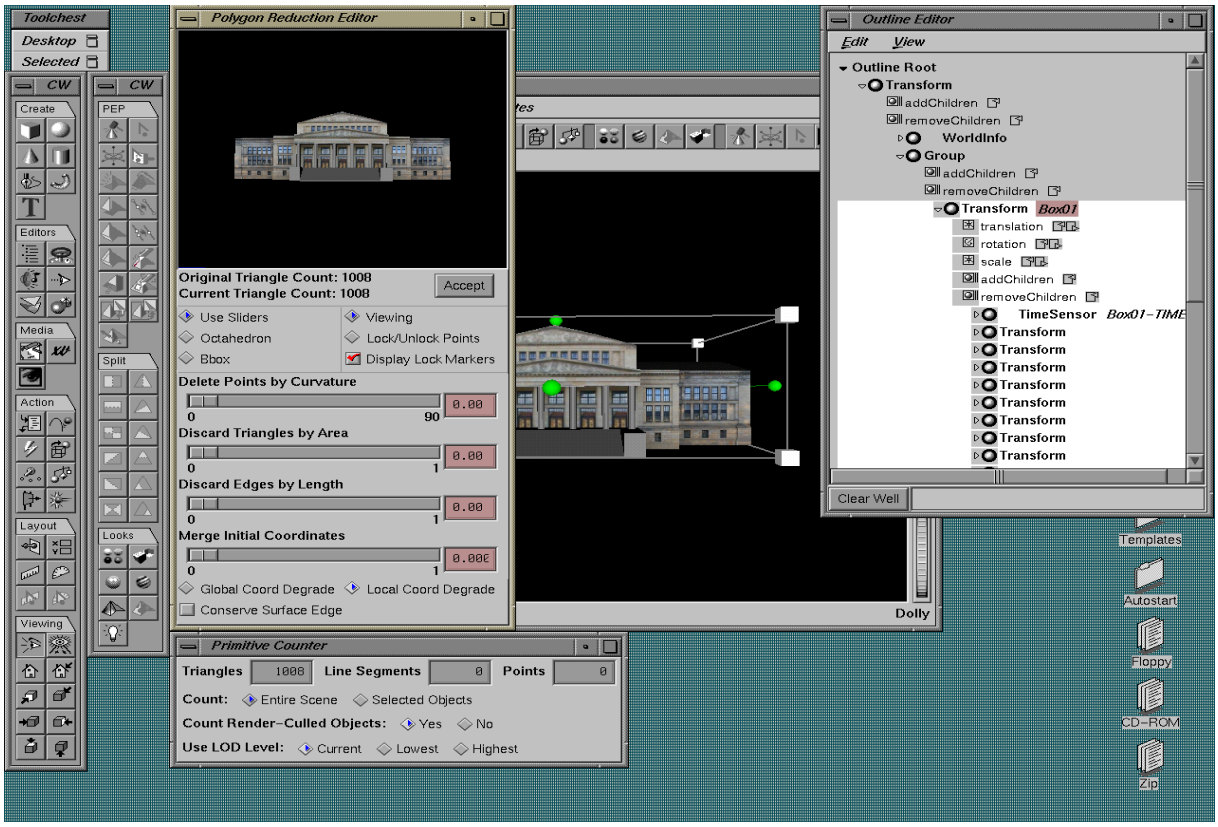

Abbildung 31 – Polygon Reduction Editor

Um die Anzahl der Polygone nachträglich zu reduzieren, existieren verschiedene Tools. Das Programmpaket WebSpace™ Author 1.0 - ein Zusatzprogramm für Cosmoworlds – enthält den Polygon Reduction Editor, ist aber leider nur für IRIX 5.3 oder höher erhältlich (Abb. 31). Dieser erlaubt dem Anwender, über Schieberegler verschiedene Parameter zu beeinflussen. So lassen sich einzelne Dreiecke oder Kanten aus dem Modelle entfernen. In einem Vorschaufenster werden die vorgenommenen Änderungen in Echtzeit dargestellt. Ein Zähler zeigt die Anzahl der verbleibenden Polygone an. Da das Vorschaufenster nur einen einfachen Überblick gibt, entstehen oft grobe Fehler beim Reduzieren. Gebäude haben plötzlich Löcher oder Kuppeln sind nicht mehr rund. Die Versuche, die Polygonzahl ohne sichtbare Veränderungen zu reduzieren, enden oft in einer Trial-and-Error-Prozedur. Daher sollte man das Ergebnis der Änderungen zwingend mit einem VRML-Browser kontrollieren. Trotzdem lassen sich mit dem Polygon Reduction Editor brauchbare Ergebnisse erzielen. Tabelle 9 zeigt einige repräsentative Beispiele.

	vorher	nachher	Differenz
Altes Museum	19968	10266	9702
Domkirche	1866	1386	480
Kronprinzenpalais	5176	3952	1224
Opernhaus	1562	72	1490
Zeughaus	2077	1661	416

Tabelle 9 – Polygonzahl vor und nach Bearbeitung mit Polygon Reduction Editor

Ein weiteres Tool zur Optimierung von VRML-Welten ist Chisel (Abb. 32), das auch in einer Version für Windows vorliegt. Dieses Programm erlaubt es, zahlreiche Optimierungen vollautomatisch durchführen zu lassen. Dabei überprüft es die Syntax, reduziert die Dateigröße und komprimiert das Ergebnis ebenfalls mit GZIP. Die Analyse der Einzelmodelle mit Chisel ergab, dass relativ selten von DEF/USE-Konstrukten Gebrauch gemacht wurde. Für diesen Fall wird eine Option angeboten, solche Konstrukte automatisch zu erzeugen. Dabei werden mehrfach verwendete Objekte voneinander abgeleitet. Das hat den Vorteil, dass die Dateigröße noch weiter schrumpft, was in Bezug auf gute Performance mehr als wünschenswert ist. Weitere Optionen gestatten das automatische Entfernen nicht verwendeter Objekte oder die Reduzierung der Auflösung bei Fließkommawerten. Ähnlich dem Polygon Reduction Editor können auch mit Chisel Dreiecke oder Kanten entfernt werden, um die Polygonzahl zu reduzieren. Chisel verlangt dem Anwender ebenso mehrere Versuche ab, bis brauchbare Ergebnisse erzielt werden. Oftmals werden durch die Reduktion Fehler in der Syntax erzeugt, die das Modell unbrauchbar machen. Um solche zu

vermeiden, empfiehlt es sich, die Reihenfolge der Operationen zu variieren, da diese teilweise voneinander abhängig sind. Insgesamt können auch mit Chisel sehr gute Reduktionswerte erreicht werden (Tabelle 10).

	ohne Chisel		mit Chisel	
	unkomprimiert	**komprimiert**	**unkomprimiert**	**komprimiert**
Brandenburger Tor	142 KB	13 KB	92 KB	11 KB
Domkirche	59 KB	5 KB	51 KB	6 KB
Kronprinzenpalais	331 KB	23,4 KB	178 KB	15,5 KB
Opernhaus	194 KB	20,9 KB	104 KB	15 KB
Zeughaus	86,8 KB	6 KB	48,7 KB	6 KB

Tabelle 10 – Ergebnisse mit Chisel

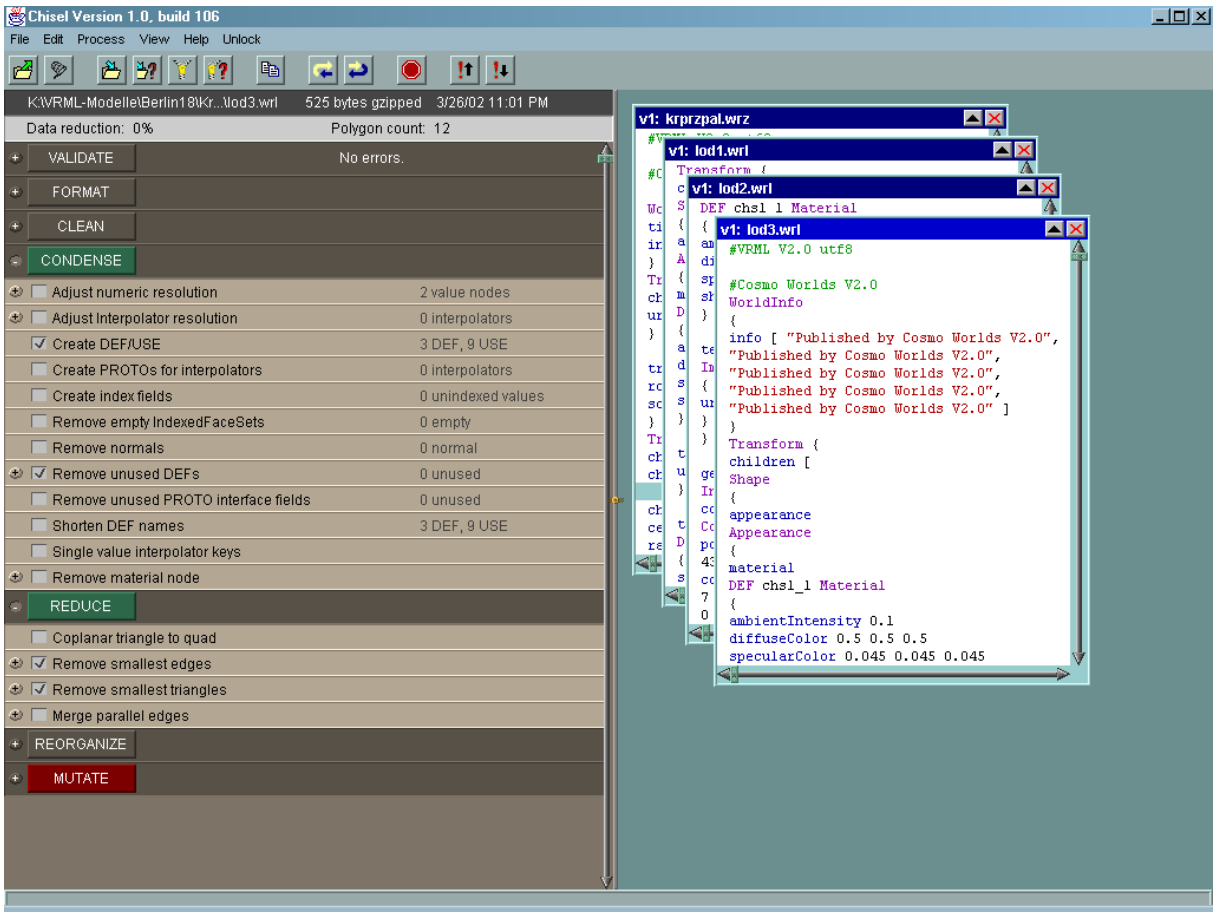

Abbildung 32 – Chisel für Windows

Die Ergebnisse aus Tabelle 10 zeigen, dass eine Bearbeitung der VRML-Modelle mit dem Optimierungsprogramm Chisel eine deutliche Reduzierung der Dateigröße bewirken kann.

Die Platzersparnis wirkt sich insbesondere bei der Kombination mehrerer Dateien aus, wie sie im 3D-Informationssystem praktiziert wird.

Der vierte Punkt der Optimierung betrifft den Einsatz von Inlines in dem VRML-Modell. Über Inlines lassen sich Teile eines Modells in separaten Dateien auslagern. Diese müssen dann nicht sofort geladen werden, sondern werden erst übertragen, wenn der Anwender in den Sichtbereich des Objekts gelangt. Dieser Umstand lässt sich insbesondere bei Verwendung von Levels-of-Detail (LOD) ausnutzen. In Abbildung 33 ist die Definition von LODs veranschaulicht.

```
children LOD {
center 50.2 0 -30.2
range [ 500, 2000 ]
level [     Inline { url "lod1.wrl" }
            Inline { url "lod2.wrl" }
            Inline { url "lod3.wrl" }
      ]
}
```

Abbildung 33 – LOD unter Verwendung von Inlines

In der Ausgangsposition des 3D-Informationssystems findet sich der Anwender in der Vogelperspektive wieder. Der Abstand zu allen Einzelmodellen ist so groß, dass zu Beginn nur das am wenigsten detaillierte Modell geladen wird. Nähert er sich einem Gebäude, werden die detaillierteren Stufen sukzessive nachgeladen. Der VRML-Modellierer Cosmoworlds erlaubt es, gruppierte Elemente, in diesem Fall die verschiedenen Detailstufen eines Modells, in Inlines umzuwandeln, sie in separaten Dateien abzulegen und den Quellcode entsprechend anzupassen. Diese Vorgehensweise hat den Vorteil, dass der Anwender nicht warten muss, bis alle Bestandteile des Systems geladen sind, sondern seine Tour bereits kurz nach dem Start beginnen kann.

Der Mechanismus der Bounding Box dient der Optimierung des Rendering-Prozesses einer VRML-Szene. Bounding Boxes werden gemeinsam mit Inlines definiert. Eine Bounding Box ist genaugenommen ein quaderförmiges Gebilde, dessen Ausmaße im optimalen Fall genau den Ausmaßen des per Inline zu ladenden Objektes entsprechen.

```
Inline {
   url "Kronprinzenpalais/krprzpal.wrz"
   bboxCenter    -624 20 -784
   bboxSize 55 40 55
}
```

Abbildung 34 – Definition eines INLINE-Knotens mit Bounding Box

Im Browser wird die Bounding Box eines Inlines normalerweise nicht dargestellt. Einige Browser verwenden sie dennoch als eine Art Platzhalter für die noch zu ladenden Objekte.

Da sie aufgrund ihrer simplen Geometrie (Quader) einfach zu berechnen ist, kann schnell festgestellt werden, ob das umschlossene Objekt im Sichtbereich des Anwenders liegt oder nicht. Falls das nicht der Fall ist, kann es vom Rendering vorerst ausgeschlossen werden, was den Ladevorgang beschleunigt. Im Fall des 3D-Informationssystems, wurden zu allen Inline-Knoten in der Hauptdatei entsprechende Bounding Boxes definiert (Abb. 34).

3.4 Benutzerschnittstelle

Der Entwicklung der Benutzerschnittstelle muss beim Entwurf eines 3D-Informationssystems besondere Achtung gelten. Sie stellt die Verbindung zwischen dem Benutzer des Systems und den darin gebotenen Funktionen dar. Die Erfahrung zeigt, dass eine umständliche Benutzerführung den Endanwender immer wieder abschreckt Webseiten zu besuchen oder ein System zu benutzen. Daher sollte der Bedienerfreundlichkeit besonderes Augenmerk geschenkt werden.

In der Praxis wird des Weiteren zwischen zwei verschiedene Arten von Benutzerschnittstellen unterschieden:

- textbasierte Benutzerschnittstelle
- grafische Benutzerschnittstelle

Die textbasierte Benutzerschnittstelle wird benutzt, wenn beispielsweise eine Anwendung unter Angabe von Parametern beim Start gesteuert werden soll. Wie der Name bereits verrät, gibt es hier keinerlei grafische Steuerelemente. Alle zur Bedienung nötigen Kommandos, werden über eine Konsole eingegeben. Als Beispiel sind hier zahlreiche Anwendungen unter dem Betriebssystem Unix zu nennen. Abbildung 35 zeigt das Komprimierungsprogramm gzip bei Einsatz unter Unix.

Abbildung 35 – Beispiel für textbasierte Benutzerschnittstelle (gzip unter Unix)

Abbildung 36 – Beispiel für grafische Benutzerschnittstelle (Windows Explorer)

Bei der grafischen Benutzerschnittstelle kommen, wie der Name bereits andeutet, zahlreiche grafische Steuerelemente zum Einsatz. Die Bedienung der Funktionen einer

Anwendung erfolgt in der Regel über Knöpfe, Schieberegler, Listboxen oder ähnliches und mittels einer Maus. Bestes Beispiel für die Verwendung einer grafischen Benutzerschnittstelle ist Microsoft Windows (Abb. 36).

Das hier entwickelte 3D-Informationssystem ist eine grafische Anwendung. Daher kommt zu dessen Steuerung auch nur eine grafische Benutzerschnittstelle in Frage. Eine textbasierte Benutzerschnittstelle ist nicht vorgesehen. Bei der Bedienung des 3D-Informationssystems muss zudem zwischen Navigation und Funktionalität unterschieden werden. Die Steuerelemente zur Navigation in der 3D-Umgebung sind in der Regel Bestandteil des verwendeten VRML-Browsers und unterliegen damit nicht dem Einfluss des Entwicklers. Anders verhält es sich bei der Schnittstelle zur Funktionalität des Systems. Hier hat der Entwickler alle Freiheiten ein benutzerfreundliches System zu entwerfen.

3.4.1 Integration in VRML und Einbettung in HTML

Das 3D-Informationssystem wurde in der 3D-Bescheibungssprache VRML realisiert. Für die grafische Benutzerschnittstelle kamen wiederum zwei Möglichkeiten in Frage.

VRML ist in seiner Konzipierung stark auf einen Einsatz im Internet ausgerichtet. Für alle Webbrowser existieren Plugins, die es dem Browser möglich machen, VRML-Welten darzustellen. Angesichts dessen bietet sich die Möglichkeit an, VRML mit HTML zu verknüpfen. Im Falle des 3D-Informationssystem wäre eine Einbettung der Steuerelemente in HTML möglich. Über das Embed-Tag lassen sich Multimedia-Inhalte in HTML einbinden. So kann über Embed die VRML-Welt des Informationssystems in eine HTML-Datei integriert werden. Die Steuerelemente zur Navigation sind, abhängig vom verwendeten Browser, Bestandteil des VRML-Plugins. Um die Funktionen des Informationssystems abzurufen, müssen in dem HTML-Sourcecode entsprechende Bedienelemente integriert werden. Dies kann entweder über die HTML-Standardfunktionen geschehen (Form-Tag) oder über ein Java-Applet. Mittels des EAI (External Authoring Interface) kommuniziert das Applet mit dem VRML-Code und ermöglicht somit die Weitergabe von Benutzereingaben, um die entsprechenden Funktionen auszulösen. Der Nachteil dieser Methode ist, dass man durch Verwendung mehrerer Technologien (VRML, HTML, Java) potenzielle Fehlerquellen eröffnet. Die Realisierung dieser Lösung erfordert umfangreiche Kenntnisse in den drei Technologien. Außerdem ist das EAI erst seit der Verabschiedung von X3D fester Bestandteil des Standards.

Eine zweite Methode ist die Implementierung der Benutzerschnittstelle in VRML. VRML bringt zur Interaktion zwischen Benutzer und 3D-Welt von vornherein alle Möglichkeiten mit. Die Steuerelemente müssen als normale 3D-Objekte modelliert werden. Unter Verwendung von Sensoren und Scripten können Knöpfe erzeugt werden, welche die gewünschten Funktionen auslösen. Der große Vorteil dieser Methode liegt in der Unabhängigkeit der verwendeten Plattform. Eine Einbettung in HTML ist weiterhin möglich, aber nicht zwingend

erforderlich. Das System lässt sich auf diese Weise als Anwendung im Internet genauso verwenden wie als Standalone-Version mit einem reinen VRML-Browser.

Für das 3D-Informationssystem kommt deshalb eine VRML-Benutzerschnittstelle zum Einsatz. Da als Anwendung sowohl Netzwerk als auch lokale Datenträger vorgesehen sind, ist die Unabhängigkeit vom Webbrowser als Vorteil anzusehen. Im Folgenden werden die einzelnen Funktionen und deren Technik näher erläutert.

3.4.2 Funktionen

Die für das 3D-Informationssystem entwickelte Benutzerschnittstelle hat einige grundlegende Richtlinien zu erfüllen:

- Die Oberfläche muss übersichtlich sein
- Alle Funktionen des 3D-Informationssystems müssen steuerbar sein
- Intuitive Bedienung
- Sie darf nicht störend sein

Die Oberfläche ist grundsätzlich in zwei Teile untergliedert. Der erste Teil ist die immer im Sichtfeld des Anwenders befindliche Menüleiste. Die Informationsterminals, die sich vor den Gebäuden befinden, gehören zu Teil zwei. Bei beiden ist die Übersichtlichkeit durch eine überschaubare Anzahl von Schaltern gewährleistet. Zu einigen Funktionen existieren zusätzliche aktivierbare Steuerelemente. Ist die Funktion deaktiviert werden entsprechend auch alle Zusatzschalter ausgeblendet. Das erleichtert dem Anwender die Navigation. In der Mitte der Menüleiste befindet sich ein Display, welches ihn über den aktuellen Zustand des Systems informiert. Auf diese Weise weiß er jederzeit, welche Epoche der Zeitleiste gerade dargestellt wird. Die Bedienelemente der Informationsterminals sind ebenfalls auf das nötigste beschränkt. Eine intuitive Bedienung wird durch eine Bebilderung der Knöpfe ermöglicht. Da sich die Menüleiste ständig im Sichtfeld des Anwenders befindet, sollte eine Möglichkeit existieren, sie auszublenden, damit sie beim Erforschen der 3D-Welt nicht störend wirkt.

Im Folgenden werden alle Funktionen und Bedienelemente des 3D-Informationssystems näher erläutert. Dabei werden insbesondere die verwendeten Techniken beschrieben

Menüleiste

Zweck der Menüleiste ist es in erster Linie, die Bedienelemente der Funktionen der Zeitleiste und Messlatte in einer einheitlichen Oberfläche zu vereinen. Ein integriertes Display gibt zusätzliche Informationen aus (Abb. 37).

Abbildung 37 – Menüleiste mit Bedienelementen

Um eine Sichtbehinderung zu vermeiden, ist die Menüleiste so gestaltet, dass sie auf Knopfdruck ausgeblendet werden kann. Dafür mussten zwei Animationen erstellt werden. Die erste sorgt dafür, dass sie nach hinten geklappt wird. Die zweite bringt sie wieder zum Vorschein. Für beide Animationen kam ein OrientationInterpolator zum Einsatz (Abb. 38).

```
DEF Interpolator_klapp_hoch OrientationInterpolator {
  key  [ 0, 1 ]
  keyValue  [ 0 0 1  0,
              1 0 0  2.21 ]
}
DEF Interpolator_klapp_runter OrientationInterpolator {
  key  [ 0, 1 ]
  keyValue  [ 1 0 0  2.21,
              0 0 1  0 ]
}
```

Abbildung 38 – Definition eines OrientationInterpolator

Dieser ist nötig um eine Rotation zu ermöglichen. Über das Feld keyValue werden die Koordinaten der Ausgangs- und Endposition festgelegt. Die Zwischenwerte werden automatisch durch den Interpolator errechnet. Dadurch wird eine weiche Animation ermöglicht. Über einen TimeSensor kann die Geschwindigkeit gesteuert werden, mit der die Animationen abläuft. Dafür wird im Feld cycleInterval ein Wert größer als 0 angegeben. Je größer der Wert ist, desto langsamer läuft die Animation ab. Um in VRML Animationen zu starten, ist ein sogenannter Trigger nötig. In diesem Fall fungiert ein TouchSensor als Trigger. Beim Drücken wird das Ereignis touchTime ausgelöst und an ein Script weitergeleitet. Dort übernimmt eine Funktion die Auswertung des aktuellen Zustands der Menüleiste (hoch- oder runtergeklappt). Entsprechend wird ein eventOut an den TimeSensor übermittelt, der seinerseits ein weiteres eventOut an den OrientationInterpolator sendet und damit die Animation auslöst. Ein drittes in dem Script definiertes eventOut sorgt dafür, dass gleichzeitig mit der Menüleiste die Messlatte ausgeblendet wird. Wird die Menüleiste nach oben geklappt, verbleibt nur noch der TouchSensor zum Auslösen der Animation im Sichtfeld des Anwenders.

In Abbildung 39 ist die Animation der Menüleiste schematisch dargestellt.

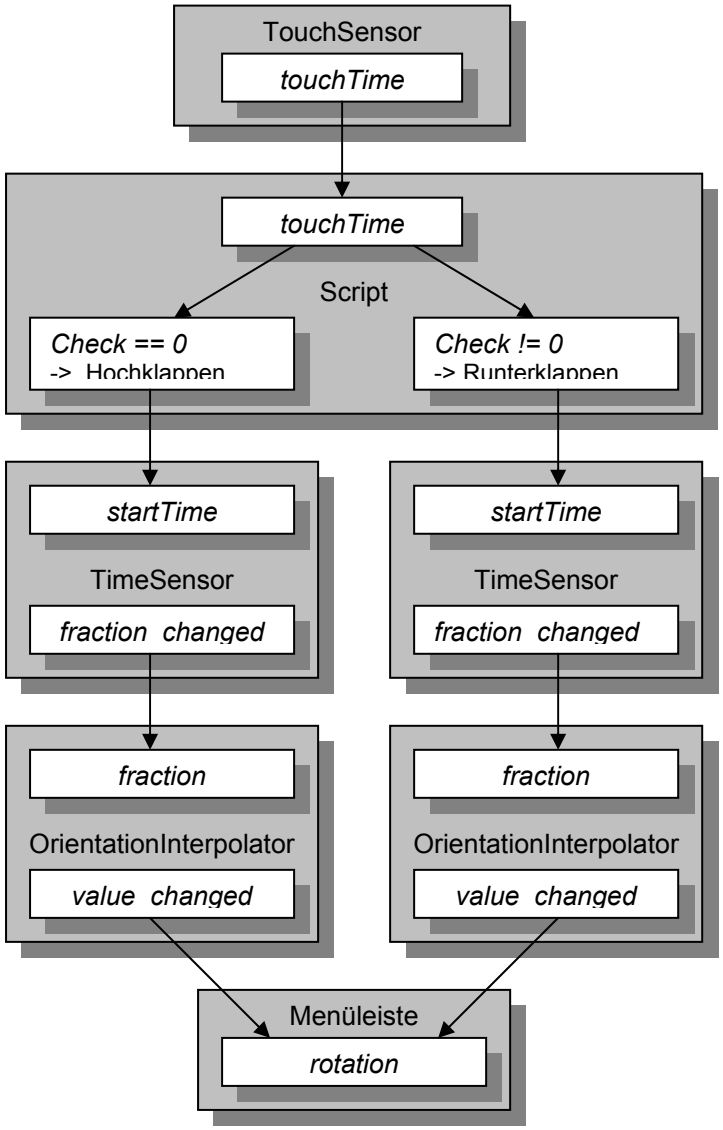

Abbildung 39 – Animation der Menüleiste

Zeitleiste

Unter der Funktion Zeitleiste ist ein Mechanismus zu verstehen, der es dem Anwender erlaubt, zwischen verschiedenen Zeitepochen zu wechseln. Dadurch wird ihm ein direkter Vergleich zwischen Vergangenheit und Gegenwart gestattet. In der hier entwickelten Ausbaustufe stehen als Epochen das 18.Jahrhundert und das 21.Jahrhundert zur Verfügung. Der Realität entsprechend sind beispielsweise im Modell des 21.Jahrhunderts das Stadtschloss entfernt und der Fernsehturm hinzugefügt worden.

Um diese Funktion zu realisieren, wird zwischen den verschiedenen Epochen mittels eines Switch umgeschaltet. Ein Switch ermöglicht das wechselnde Anzeigen von ihm untergeordneten Knoten. Die Einzelmodelle werden per Inline in das VRML-Modell integriert.

Alle Inlines, die zu einer Epoche gehören müssen gruppiert und dem Switch zugeordnet werden. Dadurch entsteht die folgende Struktur (Abb. 40):

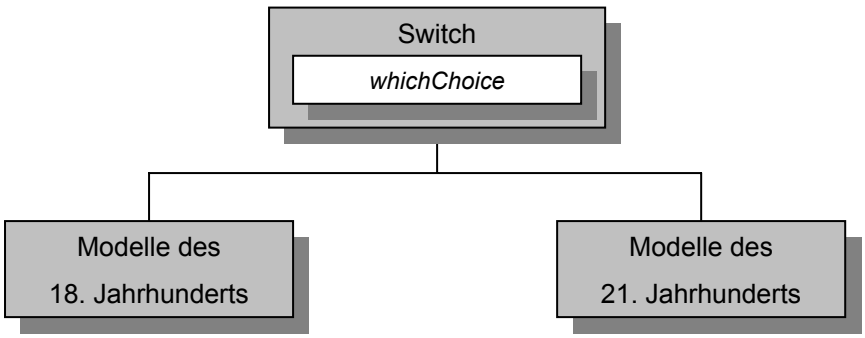

Abbildung 40 – Struktur des Zeitleisten-Switch

WhichChoice ist ein Feld des Switch-Knotens. Über dieses Feld wird gesteuert, welche der untergeordneten Gruppen jeweils dargestellt wird. Das Feld ist vom Datentyp SFInt32 (32-Bit Integer). Ist der Wert des Feldes 0, so wird die erste Option angezeigt, bei 1 die zweite und so weiter. In diesem konkreten Beispiel würde jedoch bei Werten größer 1 nichts dargestellt, da nur zwei Optionen definiert sind. Voreingestellt ist der Wert 0, damit das Modell nach dem Laden das 18.Jahrhundert darstellt. Um dem Benutzer die Möglichkeit einzuräumen selbst zwischen den Epochen hin- und herzuschalten, bedarf es einer entsprechenden Bedienungsmöglichkeit in der Benutzerschnittstelle. Die einfachste Möglichkeit ist die Definition eines Umschaltknopfes. Dafür werden eine Kugel und eine Touchsensor definiert. Beide werden wiederum gruppiert, damit nur die Kugel als Schalter fungiert. Die Verbindung zwischen dem Touchsensor und dem Switch wird mittels eines Scripts hergestellt. Das ist nötig, da durch Betätigen des Umschaltknopfes der Wert von whichChoice geändert werden muss und eine solche Änderung auf direktem Weg nicht möglich ist, weil die Datentypen von whichChoice und den Feldern des Touchsensors nicht kompatibel sind. Das Script besitzt ein eventIn (touchTime) und ein eventOut (whichChoice). Vom eventIn ist eine Route zum Feld touchTime des Touchsensors definiert. Das hat zur Folge, dass ein Ereignis immer dann ausgelöst wird, wenn der Touchsensor angeklickt wird. Dadurch wird die in dem Script definierte Funktion touchTime abgearbeitet. Die Funktion liest den Wert von whichChoice aus und inkrementiert ihn um 1. Danach wird geprüft, ob der neue Wert von whichChoice größer als 1 ist und, falls das zutrifft, wieder auf 0 zurückgesetzt. Das ist nötig, um zu verhindern, das bei Werten größer 1 keine der Epochen dargestellt wird. Um den neuen Wert für whichChoice an den Switch weiterzuleiten, ist wiederum eine Route vom eventOut des Scripts (whichChoice) zum eventIn des Switches (whichChoice) definiert.

Das folgende Beispiel (Abb. 41) zeigt das verwendete Script:

```
DEF script_zeitleiste Script {
  eventIn      SFTime  touchTime
  eventOut     SFInt32 whichChoice
  url  "javascript:
  function touchTime(value, time)
  {
    if (++whichChoice > 1) whichChoice = 0;
  }
" }
ROUTE zeitleiste_touchsensor.touchTime TO
script_zeitleiste.touchTime
ROUTE script_zeitleiste.whichChoice TO _3.set_whichChoice
```

Abbildung 41 – Zeitleisten-Script

Als Zusatzfunktionalität hat der Anwender die Möglichkeit, zwischen verschiedenen Stadtplänen zu wechseln (Abb. 42). Auch hier steht ihm ein Plan von 1778 (18.Jahrhundert) und 2002 (21.Jahrhundert) zur Verfügung. Das Umschalten erfolgt wiederum über einen Knopf in der Benutzerschnittstelle. Das Script ist mit dem für die Zeitleiste identisch.

Abbildung 42 – Zeitleiste: 18.Jh.-alter Stadtplan, 21.Jh.-neuer Stadtplan (v.l.)

Messlatte

Die Messlatte ist ein Werkzeug, welches speziell für den professionellen Anwender gedacht ist. Sie soll es ermöglichen, an den Modellen Längenmessungen vorzunehmen. Auf diese Weise soll es beispielsweise Historiken möglich sein, Breite und Höhe eines Modells zu bestimmen. Um die Messlatte praktikabel zu gestalten, ist eine Möglichkeit vorgesehen, sie in alle drei Richtungen (x, y und z) zu verschieben. Des Weiteren soll sie um 90° gekippt werden können, um Höhenmessungen zu erleichtern. Damit die Messlatte nicht störend wirkt, kann sie auch ausgeblendet werden. Als Messlatte dient ein, Quader auf dem ein

Streifenmuster als Textur aufgebracht ist. Die Textur ist in 20 Segmente unterteilt, wobei jedes einzelne einer Länge von 50 cm entspricht. Daraus ergibt sich eine Gesamtlänge von 10 m. Um sie zu verschieben, muss zuerst ein PlaneSensor definiert werden. Dieser erlaubt es, das an den PlaneSensor gekoppelte Objekt mit der Maus zu bewegen. Versuche zeigten jedoch, dass die Messlatte auf diese Weise nur schwer zu kontrollieren ist. Aus diesem Grund musste ein anderer Weg gefunden werden. Die Lösung für das Problem ergab sich durch die Verwendung von Schiebereglern (Slider). Um eine komfortable Steuerung zu ermöglichen, wurden für y- und z-Richtung eigene Slider erstellt. Sie sind kleine Quader, die - an einen PlaneSensor gekoppelt - hin und her geschoben werden können. Die Bewegung der Slider wird über ein Script auf die Positionswerte der Messlatte übertragen. Die folgende Abbildung (Abb. 43) zeigt schematisch das Routing bei Bewegung der Messlatte in y-Richtung.

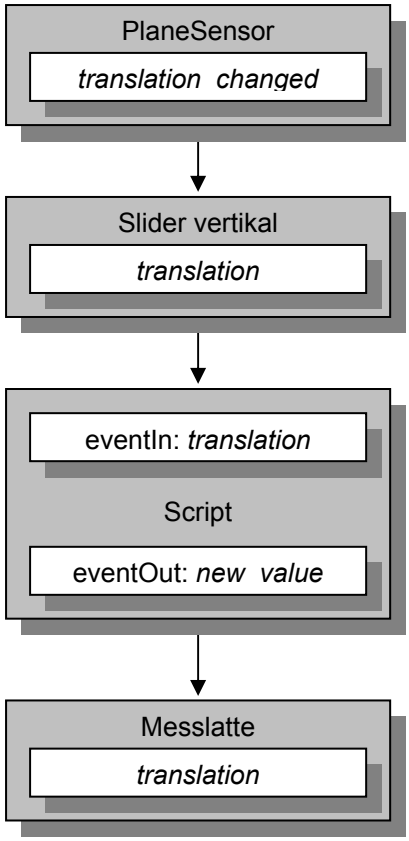

Abbildung 43 – Routing der Messlatte

Ein Planesensor erlaubt standardmäßig die Bewegung in zwei Richtungen. Um den vertikalen Slider auf y-Richtung zu beschränken, musste die Bewegungsspanne in x-Richtung auf 0 eingestellt werden. Damit der Slider auch in y-Richtung die „Führung“ nicht verlässt, wurde auch diese Spanne auf entsprechende Werte gesetzt. Diese Einstellungen

werden über die Felder min- und maxPosition vorgenommen. Analog wird der Slider für horizontale Bewegungen erstellt.

Wird die Messlatte direkt an den Slider gekoppelt, sind die Bewegungen zu gering, da der Größenunterschied zwischen Slider und Messlatte zu groß ist. Eine Bewegung des Sliders um 10 Einheiten nach rechts, bewirkt nur eine sehr geringe Verschiebung der Messlatte. Zu diesem Zweck ist zwischen Slider und Messlatte ein Script zwischengeschaltet. Das Script hat als eventIn translation und ist an das Feld translation des Sliders gekoppelt. Es arbeitet wie die Übersetzung bei einer Gangschaltung, indem es die Eingangswerte mit einem fest vorgegebenen Faktor multipliziert. Die Werte werden in new_value gespeichert und an das translation-Feld der Messlatte geleitet. Auf diese Weise werden die kleinen Bewegungen des Sliders in für die Messlatte ausreichend große umgewandelt. Abbildung 44 zeigt das dazugehörige Script.

```
DEF script_slider_v Script {
  eventIn      SFVec3f translation
  eventOut     SFVec3f new_value
  url  "javascript:
  function translation (value, timestamp)
  {
    value[0] = value [0] * 1;
    value[1] = value [1] * 100;
    value[2] = value [2] * 100;
    new_value[0] = value[0];
    new_value[1] = value[1];
    new_value[2] = value[2];
    return new_value;
  }
" }
ROUTE slider_v_links_planesensor.translation_changed TO
_25.set_translation
ROUTE _25.translation_changed TO script_slider_v.translation
ROUTE script_slider_v.new_value TO
messlatte_27.set_translation
```

Abbildung 44 – Slider-Script

Werden die Bewegungen der vertikalen und horizontalen Slider direkt auf die Messlatte übertragen, treten Überlagerungen der Bewegung auf. Um das zu umgehen, ist ein weiterer Trick nötig. Zusätzlich zur Messlatte wird ein übergeordneter Gruppenknoten mit einem eigenen translation-Feld erstellt. Auf diese Weise wird die Bewegung des horizontalen Sliders auf die Messlatte selbst und die des vertikalen Sliders auf den übergeordneten Gruppenknoten übertragen. Das Problem der Überlagerung tritt dadurch nicht mehr auf.

Im folgenden Schema (Abb. 45) ist die Schachtelung der PlaneSensoren dargestellt:

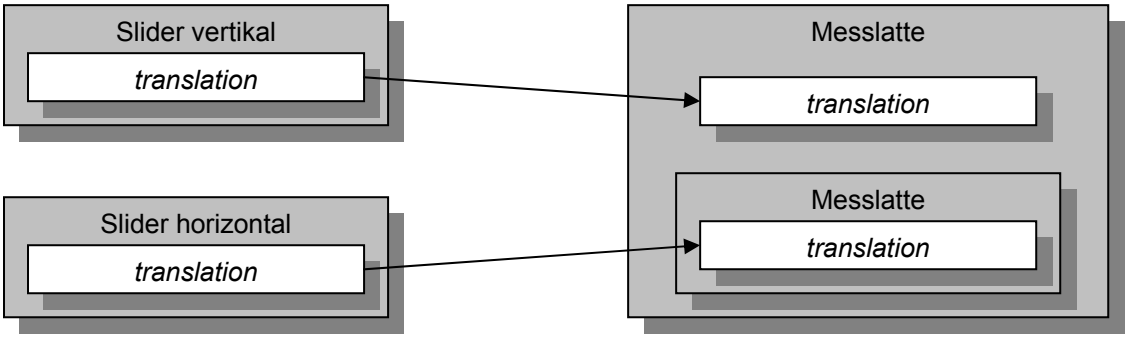

Abbildung 45 – Verschachtelung der PlaneSensoren

Damit ist die Verschiebung in y- und z-Richtung gewährleistet. Die Bewegung in x-Richtung kann ohne Umwege über einen weiteren PlaneSensor realisiert werden. Auch hier muss die Bewegung auf die gewünschte x-Richtung beschränkt werden. Um ein Kippen der Messlatte um 90° möglichen zu machen, wird wieder ein Switch verwendet. Als Unterobjekte des Switches sind zwei Messlatten definiert, eine horizontal und eine vertikal. Umgeschaltet wird über einen Touchsensor, wie er bereits bei der Zeitleiste verwendet wurde (Abb. 46). Um die Messlatte und die nötigen Steuerelemente komplett auszublenden, ist ebenfalls ein Switch nötig. Alle zur Messlatte gehörenden Objekte (Messlatte, Steuerelemente) werden dafür gruppiert. Das zum Kippen und Ausblenden der Messlatte nötige Script ist mit dem bei der Zeitleiste verwendeten identisch.

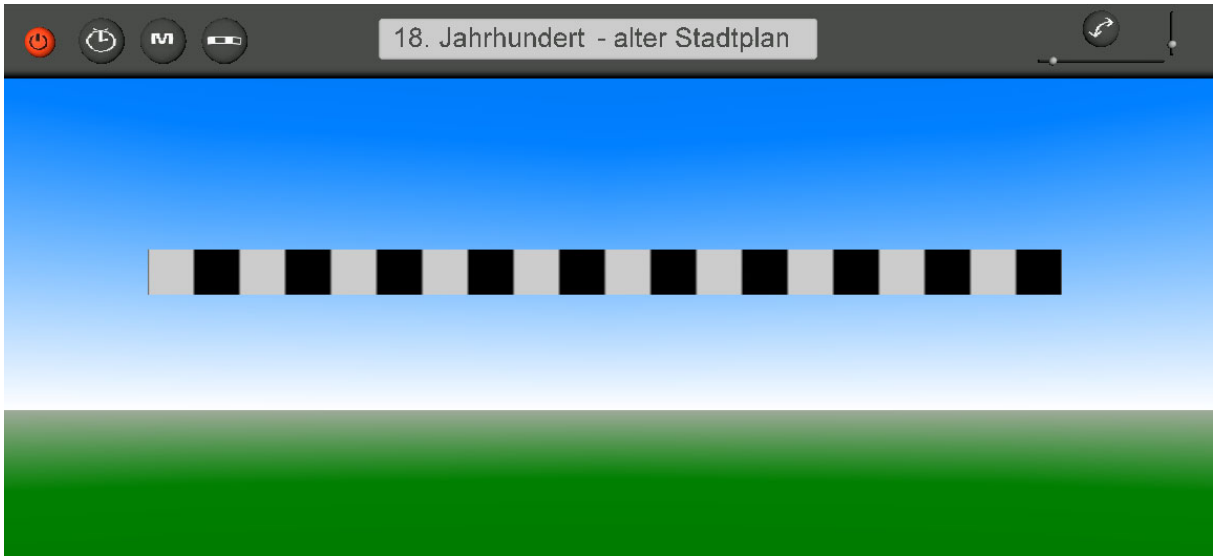

Abbildung 46 – Menüleiste mit aktivierter Messlatte

Infoterminals

Neben den 3D-Modellen werden von einem 3D-Informationssystem auch Informationen traditioneller Art erwartet. Zu diesem Zweck ist jeweils vor einem Gebäude ein Informationsterminal aufgebaut. Diese bieten die Möglichkeit, verschiedene Abbildungen des Gebäudes abzurufen. Der Anwender hat die Möglichkeit, über zwei Knöpfe vorwärts und rückwärts durch die gespeicherten Abbildungen zu blättern. Des Weiteren kann ein Hyperlink aktiviert werden, der auf eine HTML-Seite mit weiterführenden Details über das Gebäude verweist. Die Technik, die sich hinter den Informationsterminals verbirgt, ist ähnlich der bei der Zeitleiste verwendeten. Die Abbildungen werden einfach als Textur auf einen Quader aufgebracht, der auf dem Terminal als virtueller Bildschirm dient. Für jedes einzelne Bild existiert ein solcher Quader. Alle zusammen sind in einem Switch zusammengefasst. Um das Blättern durch alle Bilder zu ermöglichen, sind zwei Steuerelemente (Touchsensoren) nötig, die über ein Script den Wert des whichChoice-Felds des Switches manipulieren. Abhängig davon ob der Vorwärts- oder Rückwärtsknopf betätigt wird, muss der Wert inkrementiert oder dekrementiert werden. Zu diesem Zweck besitzt das Script zwei eventIns (touchTime_next und touchTime_prev) und ein eventOut (whichChoice). Die beiden eventIns sind jeweils per Route mit dem zugehörigen Knopf verbunden (Vorwärts und Rückwärts), das eventOut mit dem whichChoice-Feld des Switches. Wird einer der beiden Knöpfe betätigt, wird die zugehörige Funktion des Scripts (Abb. 47) ausgeführt und ein neuer whichChoice-Wert zurückgeliefert.

```
DEF toggle_script Script {
  eventIn      SFTime  touchTime_next
  eventIn      SFTime  touchTime_prev
  eventOut     SFInt32 whichChoice
  url  "javascript:
  function touchTime_next (value, timestamp)
  {
    if (whichChoice++ > 3) whichChoice = 0;
    return whichChoice;
  }
  function touchTime_prev (value, timestamp)
  {
    if (whichChoice-- < 1) whichChoice = 4;
    return whichChoice;
  }
" }
ROUTE ts_next.touchTime TO toggle_script.touchTime_next
ROUTE ts_prev.touchTime TO toggle_script.touchTime_prev
ROUTE toggle_script.whichChoice TO _6.set_whichChoice
```

Abbildung 47 – Infoterminal-Script

Abbildung 48 – Infoterminal

Zusätzlich zu den Abbildungen sollen Informationen in Textform angeboten werden. Diese werden über ein drittes Steuerelement abgerufen (Abb. 48). Betätigt man dieses, öffnet sich in einem neuen Fenster eine HTML-Seite mit entsprechenden Informationen. Dafür muss an den Knopf ein Anchor-Knoten gebunden werden. Dieser erlaubt unter Angabe einer URL, auf eine beliebige HTML-Seite zu verweisen. Dazu lassen sich weitere Parameter spezifizieren.

```
Anchor {
  children  Transform {
    children     Shape {
      appearance Appearance {    material   Material { } }
      geometry   Box { }
    }
    translation  0 1 0
  }
  url  "info.html"
  parameter "target=_blank"
}
```

Abbildung 49 – Anchor-Knoten

Bei Anklicken des Knopfes darf die Position des Anwenders im 3D-Modell nicht verändert werden. Würde der Verweis im selben Fenster geöffnet, könnte der Anwender nur unter Betätigung des Zurück-Buttons des Webbrowser in die 3D-Welt zurückgelangen. Als Folge stünde er wieder am Ausgangspunkt des Informationssystems. Der Parameter „target=_blank“ (Abb. 49) verhindert das, und erhöht dadurch die Benutzerfreundlichkeit des Systems. Der Anwender befindet sich nach Schließen des Informationsfensters an der selben Position im System wieder.

Neben der Möglichkeit die Abbildungen einfach nur zu betrachten, soll der Anwender ebenfalls einen direkten Vergleich zwischen Abbildung und 3D-Modell erhalten. Da das Informationsterminal direkt vor dem Gebäude platziert ist, ist dieser Vergleich schwer möglich. Zu diesem Zweck ist eine weitere Funktion integriert. Über einen Knopf (TouchSensor) kann eine Instanz des Infoterminals angerufen werden. Diese lässt sich mittels eine Sliders in z-Richtung vom Gebäude wegbewegen. Auf diese Weise ist gewährleistet, dass genügend Abstand zwischen dem Terminal und dem Gebäude ist, um einen Vergleich durchführen zu können. Da es sich bei dem verschiebbaren Teil um eine Instanz des Infoterminals handelt, können zur Bedienung die Steuerelemente beider Versionen benutzt werden. Eine Überschneidung der Funktionen gibt es dabei nicht. Zum Ein- und Ausblenden der Instanz kommt wie bei der Messlatte ein Switch zum Einsatz. Der Slider funktioniert ebenso wie bei der Messlatte. Ein Script übernimmt die Übersetzung von kleinen in große Bewegungen. Abbildung 50 zeigt den Aufbau des Infoterminals.

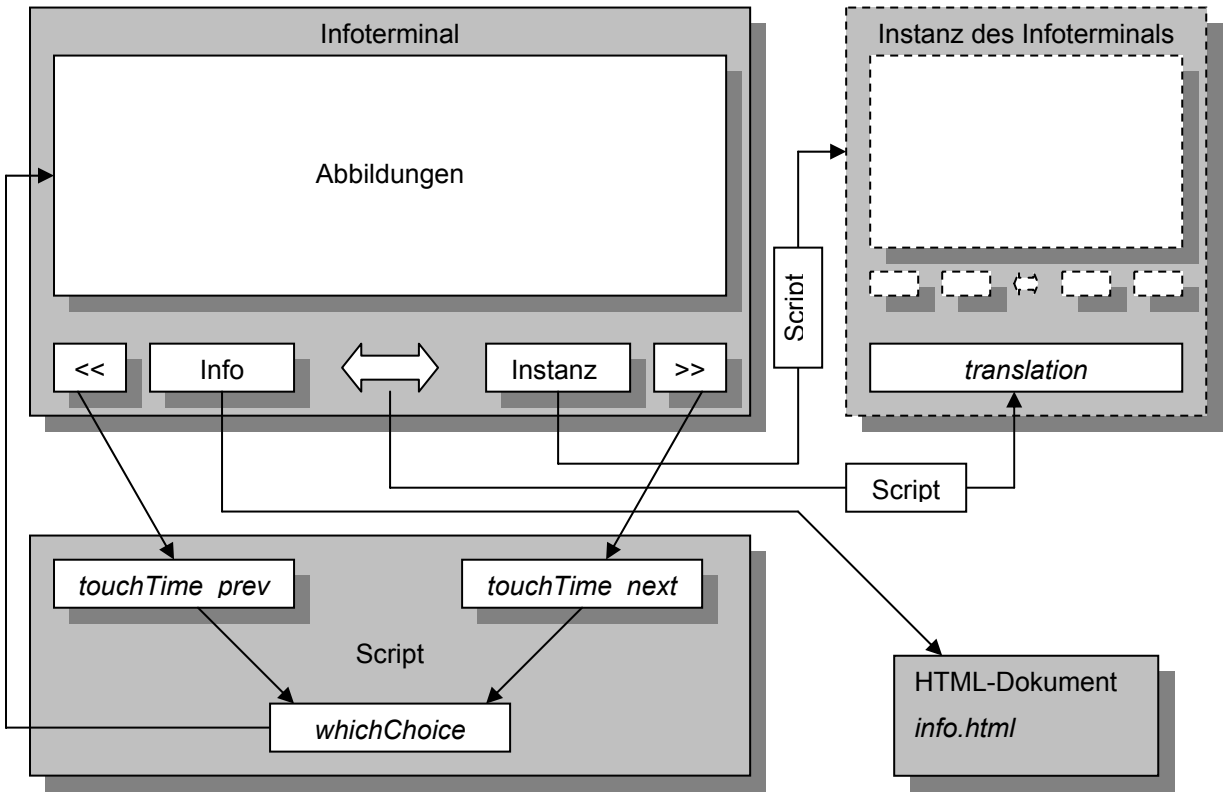

Abbildung 50 – Schematische Darstellung des Infoterminals

3.4.3 Head Up Display

Eine Benutzerschnittstelle sollte im Sinne eines ergonomischen Systems möglichst bedienerfreundlich sein. Dazu gehört auch, dass alle Bedienelemente immer erreichbar sind. In VRML erstellte Slider oder Knöpfe verhalten sich aber zunächst genauso wie jedes andere 3D-Objekt. Das heißt sie sind statisch im Raum und der Benutzer bewegt sich um sie herum. Für eine Benutzerschnittstelle, die für den Anwender stets erreichbar sein soll, ist dieses Verhalten denkbar ungünstig. Bewegt er sich von den Bedienelementen zu weit weg, ist eine Steuerung der Funktionen unmöglich. Im Idealfall sollte sich die Schnittstelle mit dem Anwender mitbewegen, also relativ zu seiner Position nicht bewegen. Die Lösung für dieses Problem ist ein Head Up Display (HUD), das die gesuchte Funktionalität zur Verfügung stellt. Alle Objekte, die Bestandteil des HUD sind, bleiben in ihrer Position relativ zum Anwender unverändert. Bewegt er sich in eine beliebige Richtung, so bewegen sich alle Objekte des HUD mit ihm mit. In VRML lässt sich ein HUD durch eine Kombination eines ProximitySensors mit HUD-Objekten realisieren. Ein ProximitySensor wird verwendet, um beim Betreten eines festgelegten Bereiches durch den Anwender Ereignisse auszulösen. Dieser Bereich wird über die Felder center und size des Sensors definiert. Um das HUD im 3D-Informationssystem zu integrieren, wurde ein Prototyp erzeugt. Dieser definiert einen ProximitySensor, dessen Zentrum bei Koordinate [0,0,0] liegt. Die Größe ist mit einer Kantenlänge von 50000 recht groß bemessen. Damit wird aber gewährleistet, dass sich der Anwender nach dem Laden des Systems bereits im Einflussbereich des Sensors befindet. Um Objekte in das HUD aufzunehmen, muss ein Feld vom Typ MFNode definiert werden. In dieses Feld können mehrere Knoten eingefügt werden. In diesem Fall sind das die Bestandteile der Benutzerschnittstelle. Ein ProximitySensor hat dazu noch mehrere eventOuts. Position_changed und orientation_changed werden ausgelöst, sobald sich der Anwender innerhalb des Sensorbereiches bewegt. Eine Route verbindet position_changed mit dem center-Feld des Sensors. Dadurch wird bei jeder Bewegung das Zentrum des Sensors der Position des Anwenders gleichgesetzt und er kann den Bereich des Sensors nicht verlassen. Weitere Routen koppeln die Positionsänderung an die Position des Feldes, welches die HUD-Objekte beinhaltet. Auf diese Weise behalten auch sie ihre Position relativ zum Anwender bei. Die Objekte der Benutzerschnittstelle müssen nun nur noch so ausgerichtet werden, dass sie an einer günstigen Position im Blickfeld das Anwenders liegen. Die Abbildungen 51 und 52 zeigen den schematischen Aufbau des HUD und den zugrundeliegenden Prototypen.

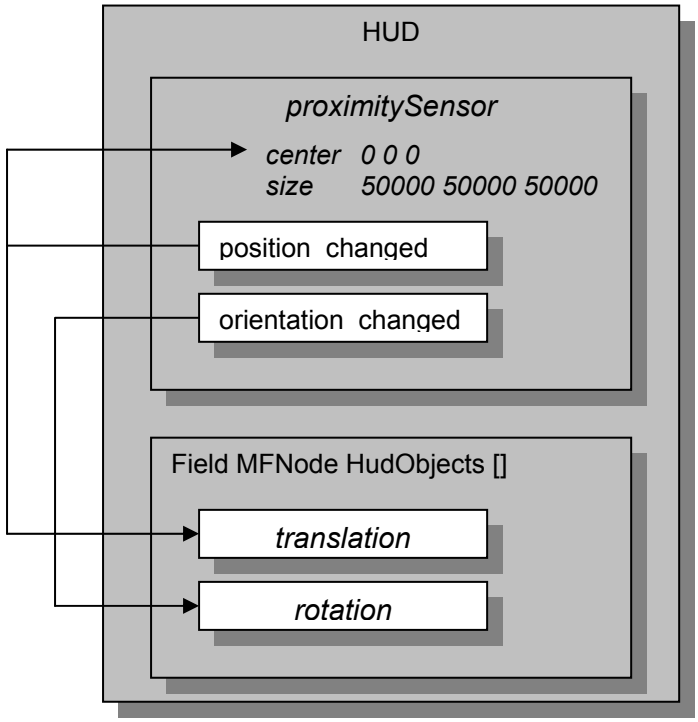

Abbildung 51 – Schema des Head Up Display

```
PROTO HUD [  exposedField SFVec3f position   0 0 0
             exposedField SFVec3f scale 1 1 1
             field MFNode  HudObjects  []
             field SFVec3f proximitySize 50000 50000 50000 ]
{
  Group {
    children [
      DEF prox_1 ProximitySensor { center   0 0 0
                                   size IS proximitySize }
      Collision {
       children DEF HUDshapes_1 Transform {
         children Transform {
           children IS HudObjects
           translation IS position
           scale IS scale
           }
         translation   0 0 0
        rotation 0 0 1  0
         }
      collide FALSE
      }
    ]
  }
ROUTE prox_1.position_changed TO proximity1_0.set_center
ROUTE prox_1.position_changed TO HUDshapes_1.set_translation
ROUTE prox_1.orientation_changed TO HUDshapes_1.set_rotation
}
```

Abbildung 52 – Prototyp des HUD

3.4.4 Erstellen der Testumgebung

Alle beschriebenen Funktionen wurden mit dem VRML-Modellierungsprogramm Cosmoworlds realisiert. Dieses gestattete ein einfaches Modellieren der Bedienelemente. Über den integrierten Script-Editor konnten problemlos Scripte geschrieben und integriert werden. Dabei hatte man die Auswahl zwischen VRMLScript, JavaScript und Java. Alle für die Benutzerschnittstelle benötigten Funktionen wurden mit JavaScript implementiert. Dafür wurde für jede einzelne Funktion zunächst ein Testmodell erstellt, um die richtige Funktionsweise zu überprüfen. Anschließend wurden alle in die Benutzerschnittstelle übertragen. Abbildung 53 zeigt einen weiteren Ausschnitt des Gesamtmodells.

Abbildung 53 – Ausschnitt aus dem Gesamtmodell (Nikolaikirche)

3.5 Anforderungskatalog für Erweiterungen

In zukünftigen Ausbaustufen des 3D-Informationssystems ist eine Erweiterung mit weiteren Einzelmodellen vorgesehen. Diese Einzelmodelle können entweder neue Gebäude oder auch Infrastrukturen darstellen. Um eine Integration zu erleichtern, ist jedoch ein verbindliches Format nötig. Die 3D-Modellierer müssen sich bei ihrer Arbeit an diese Konventionen halten. Damit soll gewährleistet werden, dass das Gesamtsystem performant und in seinem Aufbau übersichtlich bleibt. Gleichzeitig muss es möglich sein, alle

Einzelmodelle verteilt speichern zu können. Zu diesem Zweck müssen bereits in der Modellierungsphase einige Grundregeln beachtet werden. Wo es möglich ist, sollten Objekte referenziert werden, um DEF/USE-Konstrukte zu ermöglichen. Ebenfalls sollte weitestgehend auf IndexedFaceSets verzichtet werden, da diese für große Dateien und träge Darstellung mitverantwortlich sind. In diesem Zusammenhang ist ebenfalls darauf zu achten, dass die Grundprimitive durch die Texturierung nicht in IndexedFaceSets umgewandelt werden. Dieses Verhalten ist vor allem bei 3D-Studio MAX aufgefallen. Die Erfahrung mit den vorhandenen Modellen des 3D-Informationssystems hat gezeigt, dass ein detailliertes Modell nicht zwangsläufig eine Größe von 100 KB oder mehr haben muss. Dateigrößen von 50 bis 80 KB sind definitiv realisierbar. Des Weiteren ist die Integration von mindestens 3 Level of Detail erforderlich. Diese müssen zudem jeweils ein Inline bilden und in die Hauptdatei des Modells eingebunden sein. Als Dateinamen für die einzelnen Inlines sind lod1.wrl bis lodx.wrl zu wählen, wobei lod1.wrl für das am meisten detaillierte und lodx.wrl für das am wenigsten detaillierte Modell steht. Alle verwendeten Texturen müssen in einem der Formate GIF, JPEG oder PNG gespeichert sein und sollten eine maximale Auflösung von 72 dpi besitzen. Um den Integrationsaufwand zu minimieren muss das Modell bereits richtig positioniert und skaliert sein. Dafür wird dem Modellierer eine VRML-Datei mit dem passenden Stadtplan zur Verfügung gestellt. Das Infoterminal, welches dem Modellierer ebenfalls zur Verfügung gestellt wird, ist in das Modell einzufügen. Die als Abbildung verwendeten Texturen sollten den Dateinamen tex001 bis texxxx haben. Als Grafikformat ist GIF, JPEG oder PNG zu verwenden. Die entsprechenden Scripte und Texturen sind an die Anzahl der Abbildungen anzupassen. Die Verzeichnisstruktur hat folgenden Ansprüchen zu genügen (Abb. 54):

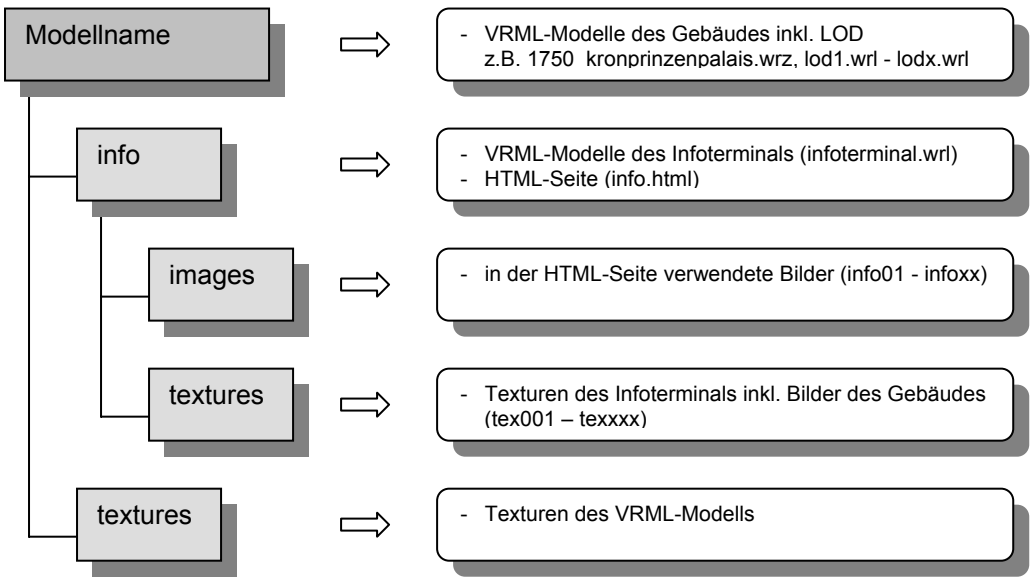

Abbildung 54 – vorgegebene Verzeichnisstruktur des Anforderungskatalogs

Des Weiteren ist eine HTML-Datei mit textuellen Informationen zu erstellen. Aus dem Dateinamen des VRML-Modells sollte der Name und die Epoche, in die das Modell einzuordnen ist, abzulesen sein. Dabei ist eine vierstellige Jahreszahl gefolgt von einem Unterstrich und dem Namen des Modells als Dateiname anzugeben, z.B. 1750_Kronprinzenpalais.wrz. Alle VRML-Dateien sind mit dem ZIP-Algorithmus zu komprimieren. Die Hauptdatei des Modells trägt die Endung .wrz, alle anderen sind mit .wrl zu bezeichnen.

Mit den hier beschriebenen Anforderungen an ein VRML-Modell des 3D-Informationssystems ist gewährleistet, dass eine Integration weiterer Modelle unproblematisch ist. Des Weiteren ist zu erwarten, dass die Performance des Gesamtmodells für die meisten Anwender ausreichend ist.

3.5.1 Vorgehensweise zum Einfügen neuer Modelle

Um eine neues 3D-Modell in das Informationssystem zu integrieren sind 3 Schritte nötig:

1. Ist das Modell für eine noch nicht im System existierende Zeitepoche gedacht, muss in der Hauptdatei in dem Zeitleisten-Switch ein neuer Knoten eingefügt werden. Innerhalb des Switches müssen die Knoten chronologisch aufsteigend angeordnet sein. Um die neue Zeitepoche anwählen zu können muss in dem Script für die Zeitleiste (script_zeitleiste) in der Funktion touchtime die Bedingung für whichchoice um die Anzahl der hinzuzufügenden Epochen erhöht werden.

   z.B: + 1 Epoche

   ```
   if (++whichChoice > 1) whichChoice = 0;
   ```

   ⇩

   ```
   if (++whichChoice > 2) whichChoice = 0;
   ```

2. Da die aktuell gewählte Epoche in dem Display innerhalb der Benutzerschnittstelle angezeigt werden soll, müssen bei Einfügen einer neuen Zeitepoche auch hier Modifikationen vorgenommen werden. Wie im ersten Schritt ist im entsprechenden Switch ein neuer Knoten der Chronologie folgend einzufügen. Als Vorlage sollte dabei einer der bereits vorhandenen Knoten dienen. Bei diesen handelt es sich um Text-Knoten. Im Feld string des Text-Knotens muss nun der Name der Epoche eingetragen werden.

   z.B.: `string "19. Jahrhundert"`

3. Das Modell muss mittels eines Inline-Knotens in den Zeitleisten-Switch eingebunden werden. Da ein Modell eventuell in mehreren der definierten Epochen angezeigt werden soll, muss es in allen in Frage kommenden Epochen integriert werden.

z.B.:

```
Inline {
  url "Modellname/dateiname.wrz"
  bboxCenter    100 0 100
  bboxSize 250 50 250
}
```

Die Werte für die Bounding Box müssen entsprechend den Abmessungen und der Position des Modells angepasst werden.

4 Test und Bewertung

4.1 Allgemeines

Das vorliegende Informationssystem ist für eine Auflösung von 800 x 600 Pixel optimiert. Damit ist die Auflösung des VRML-Fensters gemeint. Eine höhere bzw. niedrigere Auflösung stellt technisch kein Problem dar, jedoch müsste die Position der Benutzerschnittstelle angepasst werden. Tests haben ergeben, dass sie sich bei Veränderung der Auflösung nach vorn bzw. hinten verschiebt. Im ungünstigsten Fall sind dadurch einige der Bedienelemente nicht mehr zu erreichen, wodurch die Funktionalität des Systems stark eingeschränkt wird. Um die Auflösung von 800 x 600 Pixel bei der Anzeige im Webbrowser zu garantieren, wurde die VRML-Datei mittels des EMBED-Tag in HTML eingebunden. Das Tag erlaubt über verschiedene Parameter die Größe des eingebetteten Inhalts festzulegen.

```
<EMBED SRC="berlin18.wrl" BORDER="None" WIDTH="800"
HEIGHT="600">
```

In der hier gezeigten Version hat das 3D-Informationssystem eine Gesamtgröße von etwa 6,4 Megabyte. Der Umfang der Einzelmodelle schwankt zwischen 30 und bis zu 650 Kilobyte. Bei einer Übertragung über das Internet muss zwar nicht das komplette System auf einmal geladen werden, jedoch werden einige Ansprüche an die Verbindung gestellt. Eine DSL-Verbindung ist in jedem Fall empfehlenswert. Mit zunehmendem Umfang des Systems würde die Ladezeit jedoch ein vertretbares Maß überschreiten. Aus diesem Grund muss beim Modellieren der Einzelmodelle in jedem Falle auf Optimierung geachtet werden (siehe Kapitel 3.3.1).

4.2 Bedienung des Systems

Das 3D-Informationssystem kann ausschließlich mit der Maus bedient werden. Abhängig vom verwendeten VRML-Browser lässt sich jedoch auch die Tastatur alternativ oder ergänzend zur Navigation benutzen. Im VRML-Browser Cosmoplayer kann über die Tasten Ctrl, Alt und Shift die Navigation zwischen der normalen Bewegung, Drehung und Gleiten umgeschaltet werden. Über den Ziffernblock ist es möglich, die verschiedenen Navigationspunkte anzusteuern. Am oberen Rand des Bildschirm befindet sich die Menüleiste, die dem Anwender alle Funktionen des Systems zugänglich macht. Durch Verwendung eines Head Up Display ist sie stets in Reichweite. Um in bestimmten Situationen nicht im Wege zu sein, ist ein Wegklappen der Menüleiste möglich. Um den Anwender nicht durch zahlreiche Schaltflächen zu verwirren, werden auch innerhalb der Menüleiste nicht benötigte Bedienelemente ausgeblendet.

4.3 Performance-Tests

Bei den Performance-Tests kam es vor allem darauf an festzustellen, welche Systemressourcen bei der Benutzung des 3D-Informationssystems besonders gefordert sind. Dabei geht es im Speziellen um die Ladegeschwindigkeit und die Darstellung auf dem Client. Als Testplattform kam ein AMD Athlon mit 1000 MHz unter Windows 2000 Professional zum Einsatz. Dieser war mit 448 MB RAM und einer 3D-Grafikkarte Elsa Erazor III ausgestattet. Die Dauer der Datenübertragung wurde anhand einer ISDN- und einer DSL-Verbindung überprüft (64 KBit / 768 KBit).

Der Test zeigte, dass das Testsystem den Anforderungen des 3D-Informationssystems bei der Darstellung durchaus gewachsen war. Die Wiedergabe erfolgte weitgehend ruckelfrei. Einzig beim Wechsel von einem Navigationspunkt zum anderen traten aufgrund der interpolierten Bewegung deutliche Ruckbewegungen auf. Die Navigation mittels der im VRML-Browser integrierten Navigationselemente war problemlos möglich. Die gute Darstellung hängt vor allem von der schnellen CPU und der 3D-Grafikkarte ab, die geometrische Berechnungen in Hardware durchführen kann. Angesichts der Leistungsfähigkeit der heutzutage verkauften PCs kann man bei einem zwei Jahre alten Athlon 1000 sicherlich von einem Einsteigersystem sprechen. Es ist abzusehen, dass der Anteil entsprechender Rechner am Gesamtmarkt bald überwiegt, und damit das System von einem Großteil der Anwender genutzt werden kann.

Anders sieht die Situation bei der Ladegeschwindigkeit aus. Abhängig vom verwendeten Medium muss mit langen Ladezeiten gerechnet werden. Wird das System von einem lokalen Datenträger gestartet, sind die Ladezeiten kürzer, aber nicht unbeträchtlich. Zudem wird hierbei erst das Gesamtsystem geladen. In Tabelle 11 sind die Ladezeiten zusammengefasst.

	Lokal	DSL - 768 KBit	ISDN - 64 KBit
System benutzbar	46 s	15 s	k.A.
System vollständig	46 s	105 s	k.A.

Tabelle 11 – Ladezeiten des 3D-Informationssystems

Bei der Verwendung einer DSL-Verbindung ist bereits nach 15 Sekunden eine Navigation durch das System möglich. Allerdings sind nach dieser Zeit noch nicht alle Modelle geladen. Erst nach 105 Sekunden ist das Gesamtsystem komplett und damit uneingeschränkt nutzbar. ISDN scheint komplett ungeeignet, da die Übertragung entweder unvollständig war oder abgebrochen wurde. Im ersten Fall wurde lediglich die Benutzerschnittstelle und die Grundplatte geladen, die Einzelmodelle wurden nicht übertragen. Die Ursache hierfür ist wohl in der zu geringen Bandbreite zu suchen. Nach neuesten Marktforschungsergebnissen

sind jedoch bereits zum jetzigen Zeitpunkt 15 % aller privaten Internetnutzer mit einem Breitbandanschluss versorgt. Die Tendenz ist weiter steigend. Damit dürfte in naher Zukunft auch dieses Nadelöhr beseitigt sein.

4.4 Bewertung und Ausblick

Als Ergebnis der Arbeit steht ein 3D-Informationssystem, welches in dieser Form neuartig ist. Informationssysteme existieren zwar, einige sogar mit historischem Ansatz. Keines bietet dem Anwender aber die Möglichkeiten und Freiheiten des hier entwickelten 3D-Systems. Trotzdem bietet es dem Entwickler noch reichlich Potenzial für Verbesserungen. Das betrifft die dargebotenen Inhalte, die Performance oder einfach Ideen für weitere Ausbaustufen. Zum ersten Punkt wäre es natürlich wünschenswert, dass sich eine Vielzahl von 3D-Modellierern bereit erklärten, weitere Modelle für alle Zeitepochen zu entwickeln. Dabei sollte neben Gebäuden auch die Infrastruktur nicht unbeachtet bleiben. Mit zunehmender Komplexität gewinnt der zweite Punkt an Bedeutung. In der jetzigen Form verkraftet das System kaum mehr Modelle, da die Performance stark zu wünschen übrig lässt. Das ist nur durch konsequente Optimierung, die bereits in der Modellierungsphase einsetzt, zu beheben. Zu diesem Zweck existiert auch ein Anforderungskatalog, an dem sich alle weiteren Modelle strikt orientieren müssen. Ebenso wäre es sinnvoll, die bereits in das System integrierten Modelle, nach den geforderten Gesichtspunkten neu zu erstellen. Das Performance-Problem wird sich mit zunehmender Leistungsfähigkeit von Client-PCs und Netzwerken zwar in seinen Auswirkungen abschwächen, verlassen sollte man sich darauf jedoch nicht.

In weiteren Ausbaustufen des Systems lassen sich mit Sicherheit zahlreiche weitere Ideen verwirklichen. Zum Beispiel wäre eine Modellierung des Innenraumes der Gebäude denkbar. Diese muss nicht zwangsläufig mittels VRML erfolgen. Eine Integration von sphärischen Quicktime VR Panoramen böte dem Anwender eine ganz neue Dimension des Erlebens von Architektur. Die hier entwickelte Version des 3D-Informationssystems ist für eine Präsentation auf Monitoren oder eventuell auch größeren Leinwänden ausgelegt. In zukünftigen Versionen könnte eine Benutzerschnittstelle enthalten sein, die den Einsatz von Datenhandschuhen oder VR-Helmen unterstützt. Diese Technologie ist heutzutage kaum verbreitet, in Museen ist eine solche Verwendung jedoch durchaus vorstellbar. In diesem Zusammenhang muss auch die Frage gestellt werden, ob ein Monitor dem Anwender den richtigen Eindruck verschaffen kann. Das Gefühl von der Größe der Bauwerke kann auf einem Standard-17-Zoll-Bildschirm nicht wirklich vermittelt werden.

Auf der technischen Seite ist eine Umsetzung in den neuen 3D-Standard X3D denkbar. Die damit verbundene Unterstützung von XML bietet im Bereich der Informationssysteme vielfältige Möglichkeiten. Informationen ließen sich besser strukturieren und damit dem Anwender einfacher vermitteln. In Anlehnung an das 3D-Informationssystem könnten auch andere Plattformen präsentiert werden. Die Verwendung der 3D-Technologie für virtuelle

Marktplätze im Bereich des e-Commerce ist nicht undenkbar. Über dies wäre es beispielsweise möglich, kulturelle Angebote (Theater, Museen usw.) zu vermarkten oder modellierten Märkten Leben einzuhauchen. Durch Integration von Avataren könnte dem Anwender die Möglichkeit gegeben werden, mit anderen Besuchern zu interagieren (z.B. Chat).

Allgemein kann jedoch gesagt werden, dass 3D als Ergänzung zu den klassischen Medien wie Text und Bild in jedem Falle seine Berechtigung hat. 3D bietet neue und vielfältige, derzeit mit Sicherheit noch nicht vorstellbare Möglichkeiten der Informationsdarbietung. Da VRML97 bereits seit fünf Jahren verfügbar ist, bis jetzt jedoch nur wenige Anwendungen existieren, ist im Augenblick unklar, inwieweit 3D von den Nutzern in Zukunft angenommen wird. Die dreidimensionale Präsentation im Internet fristet zum heutigen Zeitpunkt leider noch ein Nischendasein, hat jedoch neben den etablierten Medien hinsichtlich der Fähigkeiten zur Darstellung virtueller Realitäten definitiv eine Zukunft.

5 Abkürzungsverzeichnis

CD-ROM	Compact Disc-Read Only Memory
DSL	Digital Subscriber Line
EAI	External Authoring Interface
GIF	Graphics Interchange Format
GIS	Geografische Informationssysteme
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
HUD	Head Up Display
IBR	Image Based Rendering
ISDN	Integrated Services Digital Network
ISO / IEC	International Standards Organization / International Electro-Technical Commission
JPEG	Joint Photographic Experts Group
LOD	Level of Detail
MPEG	Motion Pictures Expert Group
PNG	Portable Network Graphics
QTVR	Quicktime Virtual Reality
RAM	Random Access Memory
URL	Uniform Resource Locator
VR	Virtual Reality
VRML	Virtual Reality Modelling Language
W3C	Web 3D Consortium
WWW	World Wide Web
X3D	Extensible 3D
XML	Extensible Markup Language

6 Quellenverzeichnis

6.1 Literaturverzeichnis

[1] Jörg Kloss; Robert Rockwell u.a
VRML 97 – Der neue Standard für interaktive 3D-Welten im WWW
Addison Wesley, 1998

[2] Rikk Carey; Gavin Bell
The Annotated VRML 2.0 Reference Manual
Addison Wesley, 1997

[3] Mark Pesce
Cyberspace-Welten erkunden und erschaffen
Carl Hanser Verlag, 1997

[4] Oliver Schlüter
VRML Sprachmerkmale, Anwendungen, Perspektiven
O'Reilly Verlag, 1998

[5] Jörn Loviscach
VRML 2.0
magazin für computertechnik, c't 5/97, S.236

6.2 Informationsquellen im Internet

[6] http://www.4teamwork.ch/rueedi/
11/2001, Cyber Chicago

[7] http://www.doc.mmu.ac.uk/RESEARCH/virtual-museum/index.html
12/2001, Virtual Museum MMU

[8] http://www.pa-malta.org/npi/mdina/mainFrame.html
12/2001, Heritage Management System

[9] http://www.vrseattle.com/
03/2002, VR Seattle

[10] http://www.viewtec.ch/techdiv/vr_info_d.html
12/2001, Definition Virtual Reality

[11] http://www.ikb.mavt.ethz.ch/projects/fmea/chapter/virtualreality.html
12/2001, Definition Virtual Reality

[12] http://www.bildindex.de/
03/2002, Bildarchiv zur Kunst und Architektur in Deutschland

[13] http://www.vrml.org/technicalinfo/specifications/vrml97/index.htm
12/2001, VRML97-Referenz

[14] http://www.web3d.org/x3d/
12/2001, X3D

[15] http://selfaktuell.teamone.de/
12/2001, SelfHTML 8.0

[16] http://www.apple.com/quicktime/qtvr/authoringstudio/index.html
03/2002, Apple Quicktime VR Authoring Studio

[17] http://www.vrml-fokus.de/
04/2002, Vivatech's VRML-Fokus

[18] http://www.aliaswavefront.com/en/Home/homepage.shtml
04/2002, Alias|Wavefront Maya PLE

[19] http://www.spazz3d.com/
03/2002, Virtock Technology Spazz3D

[20] http://www.realviz.com/
03/2002, RealViz Stitcher 3

[21] http://www.trapezium.com/
02/2002, Trapezium Chisel

7 Eigenständigkeitserklärung

Hiermit versichere ich, dass ich die vorliegende Diplomarbeit selbständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel verfasst habe. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt.

Andreas Vogel
Berlin, 7.Mai 2002