INSTITUTE of CARTOGRAPHY

Dreidimensionales Höhleninformationssystem

Entwicklung eines webbasierten Prototyps
unter Verwendung von HTML, Javascript, SVG und VRML

ETH Zürich
Vertiefungsblock Kartographie WS 04/05

Zürich, Januar 2005

Autor:

Severin Hohl
Untere Schmitten
3036 Detligen
shohl@student.ethz.ch

Betreuung:

Andreas Neumann (IKA)

Inhaltsverzeichnis

1 Einleitung ... 3

2 Formate für die dreidimensionale Darstellung ... 4

2.1 VRML ... 4

2.2 X3D ... 5

2.3 Das Szenegraph- und Knotenkonzept ... 7

3 Testen von 3D-Visualisierungssoftware ... 8

3.1 Verwendete Software ... 8

3.2 Handling ... 9

3.2.1 BS contact ... 9

3.2.2 Venues ... 9

3.2.3 Flux ... 10

3.2.4 Octaga ... 10

3.3 Performance ... 10

3.3.1 Grundlagen ... 10

3.3.2 Test-Dateien ... 11

3.3.3 Test ... 12

3.4 Bewertung der Resultate ... 13

4 Höhlenvermessung ... 15

4.1 Datenerfassung ... 15

4.1.1 Vermessungsinstrumente ... 15

4.1.2 Realisierung der Vermessung ... 15

4.2 Die Höhlenvermessungssoftware Toporobot ... 16

4.2.1 Die grundlegendsten Funktionen von Toporobot ... 16

4.2.2 Struktur der Höhlendaten ... 16

4.2.3 Exportmöglichkeiten ... 17

5 Höhleninformationssystem ... 18

5.1 Ziel und Inhalt des Höhleninformationssystems ... 18

5.2 Probleme bei der Implementierung mit X3D ... 19

5.2.1 Erfolgreiche Befehle zur Interaktion mit Javascript / X3D ... 19

5.3 Dateistruktur des Informationssystems ... 20

5.4 Aufbau der Webapplikation ... 21

5.5 Die wichtigsten Befehle zur Interaktion mit VRML ... 22

5.6 Die einzelnen Funktionen *23*
5.6.1 Position des Beobachters 24
5.6.2 Auf Höhle zoomen 24
5.6.3 Information über Gänge 25
5.6.4 Ebene neigen 25
5.6.5 Digitales Geländemodell 26
5.7 Änderungen im automatisch generierten VRML *27*

6 Schlussbemerkungen **29**
6.1 Erreichtes *29*
6.2 Kleinere Mängel *29*
6.3 Weiterführende Möglichkeiten *30*
6.4 Dank *30*

7 Quellenverzeichnis **31**
7.1 Quellen *31*
7.2 Nützliche Links *31*
7.2.1 Viewer 31
7.2.2 Allgemeine Informationen 31

1 Einleitung

Die Motivation zur Bearbeitung eines Themas im Bereich Höhlenvisualisierung kommt unter anderem aus persönlichen Erfahrungen mit der Höhlenvermessung. Im Gymnasium hatte ich die Möglichkeit, zusammen mit erfahrenen Höhlenforschern einen Teil der Neuenburgerhöhle in der Schrattenfluh LU zu vermessen. Dies weckte meine Faszination für die unterirdischen Welten und deren Erforschung, verschaffte mir aber auch Respekt vor der Arbeit der Höhlenvermesser. Die Resultate dieser Vermessung sichtbar zu machen und sie einem breiteren Publikum näher zu bringen, hat für mich deshalb einen besonderen Reiz.

Um eine Höhle virtuell erfahren zu können, ist es am einfachsten, sie dreidimensional zu sehen. Die Visualisierung von komplexen dreidimensionalen Objekten trägt viel zu deren Verständnis bei. Zudem verschafft die Verbindung von 3D, 2D und alphanumerischer Information, welche ein Höhleninformationssystem beinhaltet, dem Betrachter einen umfassenden Überblick über eine Höhle.

Die Entwicklung eines Höhleninformationssystems als Webapplikation gibt die Möglichkeit, einen Einblick in verschiedene Webtechnologien zu erhalten. Zum Einsatz kommen HTML, Javascript, SVG und VRML / X3D. In einem ersten Teil werden verschiedene frei erhältliche Viewer-Plugins für die 3D-Formate VRML und X3D getestet. Dies im Hinblick auf eine allfällige Verwendung einer dieser Viewer im Höhleninformationssystem und zum Kennenlernen der Möglichkeiten des zukunftsversprechenden 3D-Standards X3D. Das Ziel, X3D als 3D-Format einzusetzen, konnte leider nicht erreicht werden. Die Gründe dafür werden in diesem Bericht erläutert. Im zweiten Teil wird ein dreidimensionales Höhleninformationssystem auf der Basis von VRML als Webapplikation aufgebaut.

2 Formate für die dreidimensionale Darstellung

Zum besseren Verständnis dieser Arbeit folgt an dieser Stelle eine Beschreibung des Formates VRML und dessen Nachfolger X3D. Da das Höhleninformationssystem in der Hauptsache auf dreidimensionaler Information aufbauen soll, kommt den Formaten zur Darstellung solcher Inhalte grosses Gewicht zu.

2.1 VRML

VRML bedeutet Virtual Reality Modeling Language und ist eine Beschreibungssprache für 3D-Szenen, deren Geometrien, Ausleuchtungen, Animationen und Interaktionsmöglichkeiten. VRML wurde ursprünglich als 3D-Standard für das Internet entwickelt. Die meisten 3D-Modellierungswerkzeuge ermöglichen den Import und Export von VRML-Dateien, weshalb sich das Dateiformat auch als ein Austauschformat von 3D-Modellen etabliert hat.

Eine VRML-Darstellung zum Beispiel innerhalb eines Web-Browsers wird vom Computer des Betrachters in Echtzeit generiert. Das bedeutet, dass der Computer jedes einzelne Bild aus den vorhandenen Geometriedaten sowie dem Verhalten und den Bewegungen des "Besuchers" ständig neu berechnet. Komplexe VRML-Szenen stellen daher unter Umständen hohe Anforderungen an die Hardware. Wie schnell, beziehungsweise wie flüssig die Bewegungen erfolgen, hängt vom Prozessor(takt) und vor allem von der Grafikkarte des wiedergebenden Computers ab.

VRML-Dateien erkennt man an der Dateierweiterung ".wrl" (world), sie sind im Klartext (ASCII) geschrieben und können auch in einem einfachen Texteditor erstellt werden. [vgl.Wikipedia]

VRML basiert ursprünglich auf dem von Silicon Graphics entwickelten 3D-Standard "Open Inventor". Dieser wurde 1995 als VRML 1.0 übernommen und um die Funktion der Verlinkung bereichert. VRML 1.0 mangelte es jedoch an einigen elementaren Möglichkeiten, wie beispielsweise an der Beschreibung physischer Eigenschaften von Objekten. Ferner fehlten Funktionen der Animation, der Interaktion, Multimedia sowie der prozeduralen Programmierung.

Als Antwort auf diese Mängel entwickelten Sony und Silicon Graphics das Konzept der "Moving Worlds", das sich 1996 in einer Internet-Abstimmung unter insgesamt sechs konkurrierenden Industrievorschlägen als neuer Standard VRML 2.0 durchsetzte. Im September 1997 wurde VRML 2.0 nach einigen kleineren Änderungen als VRML97 durch die International Standards Organization (ISO) als Industriestandard ISO/IEC 14772 festgesetzt. [Eibl, 2001]

2.2 X3D

Extensible 3D, kurz X3D, ist eine 3D-Modellierungssprache, die in der Syntax auf XML[1] (XML-Encoding) oder auf VRML (Classic-Encoding) aufbauen kann. X3D bildet den offiziellen Nachfolger des VRML-Standards und ist seit August 2004 als ISO-Standard spezifiziert.
Ähnlich wie bei VRML lassen sich auch in X3D dreidimensionale virtuelle Welten, Spiele und interaktive Lernanwendungen in Echtzeit realisieren. Gegenüber VRML stehen jedoch bei X3D wesentlich mehr standardisierte Möglichkeiten und Schnittstellen bereit. Der Funktionsumfang von X3D kann durch die Angabe von Profilen (siehe Abbildung 1) eingeschränkt werden. [vgl. Wikipedia]

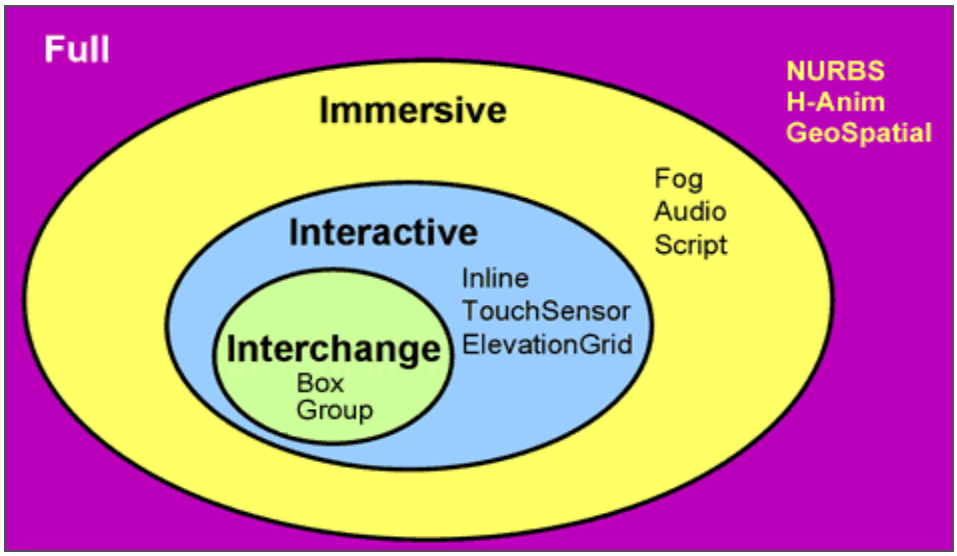


Abbildung 1 Profile von X3D [Web3D]

Das Konzept von XML bietet drei Vorteile: Erstens die strikte Trennung von Inhalt und Aussehen. Zweitens lassen sich mit XML die verschiedensten Elemente beschreiben. Es be-

[1] Die „Extensible Markup Language“ ist eine Konvention zur semantischen Auszeichnung von Texten.

darf lediglich einer entsprechenden Formulierung der DTD[2]. Drittens kann man den Sprachumfang durch eine entsprechende Erweiterung der DTD jederzeit ausbauen. Es lag nahe, diese Vorteile auch für ein weiterentwickeltes VRML zu nutzen.

VRML gilt mittlerweile als überholt. Um möglichst alle Aspekte der dreidimensionalen Darstellung im Internet umfassend zu berücksichtigen, erweiterte sich das VRML-Konsortium 1998 zum Web3D-Konsortium. Dieses verkündete im Februar 1999 die Einführung von XML zur Formulierung einer Beschreibungssprache. Ergebnis dieser Bemühungen war die DTD extensible 3D (X3D).

X3D ist modular konzipiert und besteht aus verschiedenen Komponenten:

- Core Kernel (X3D-1): Diese Komponente definiert die für gängige 3D-Darstellungen unabdingbaren Elemente wie Geometrie, Beleuchtung und Animation.
- VRML97 Profile (X3D-2): Dieser Teil der Beschreibung garantiert eine Rückwärtskompatibilität von X3D zu VRML97. Mit entsprechenden XSL[3]-Stylesheets lassen sich X3D-Dateien problemlos zu VRML97-Dateien konvertieren und so bereits mit aktuellen Browser-Plugins betrachten.
- Erweiterungen: Zahlreiche VRML-Erweiterungen können in das X3D-Format übernommen werden. Zusätzliche Erweiterungen lassen sich ebenfalls problemlos einbinden.

X3D lässt sich nahezu unbegrenzt erweitern. Um den Kern X3D-1, der nur die notwendigsten Spezifikationen enthält, werden weitere Profile angelegt, die auf die speziellen Bedürfnisse der einzelnen Anwender zugeschnitten sind. So wird etwa ein Biologe ganz andere Anforderungen an eine dreidimensionale Darstellung haben als ein Geograph.

Die Erweiterbarkeit von X3D hat in der Praxis Auswirkungen in zwei Richtungen: Zum einen muss für die Codierung eines X3D-Browsers nicht die komplette Spezifikation berücksichtigt werden. Das heißt: Für spezielle X3D-Anwendungen kann man auch spezielle Browser konzipieren, die in Größe und Laufzeitverhalten für den entsprechenden Anwendungsfall optimiert sind. Zum anderen sind die Hauptkomponenten jederzeit durch beliebige Module zu ergänzen. Erweiterungen liegen a priori in der Zielsetzung der Spezifikation und sind nicht

[2] Die „Document Type Definition“ beschreibt die für einen bestimmten Dokumenttyp zulässigen Elemente.
[3] Die „Extensible Style Language“ ist ein Stylesheet, das die Erscheinung der Elemente festlegt.

wie bei VRML97 proprietär. Neue Anwendungen lassen sich damit in die X3D-Spezifikation aufnehmen, ohne mit den bereits existierenden Modulen in Konflikt zu geraten. [Eibl, 2001]

2.3 Das Szenegraph- und Knotenkonzept

Die Struktur einer VRML- oder X3D-Szene ist grundsätzlich dieselbe. Die Darstellung dreidimensionaler Szenen erfolgt dabei mittels einer Baumstruktur, die Szenegraph genannt wird. Die Blätter des Hierarchiebaumes sind die Gestaltknoten der Szene und damit deren sichtbare Objekte. Es wird zwischen Knoten (Nodes) und Feldern oder Attributen (Fields) unterschieden. Ein Gestaltknoten verbindet eine geometrische Form (geometry) mit einer Erscheinung dieser Form (appearance). Es existieren aber auch Gruppierungsknoten, die dazu verwendet werden, geometrische Elemente zu verknüpfen.

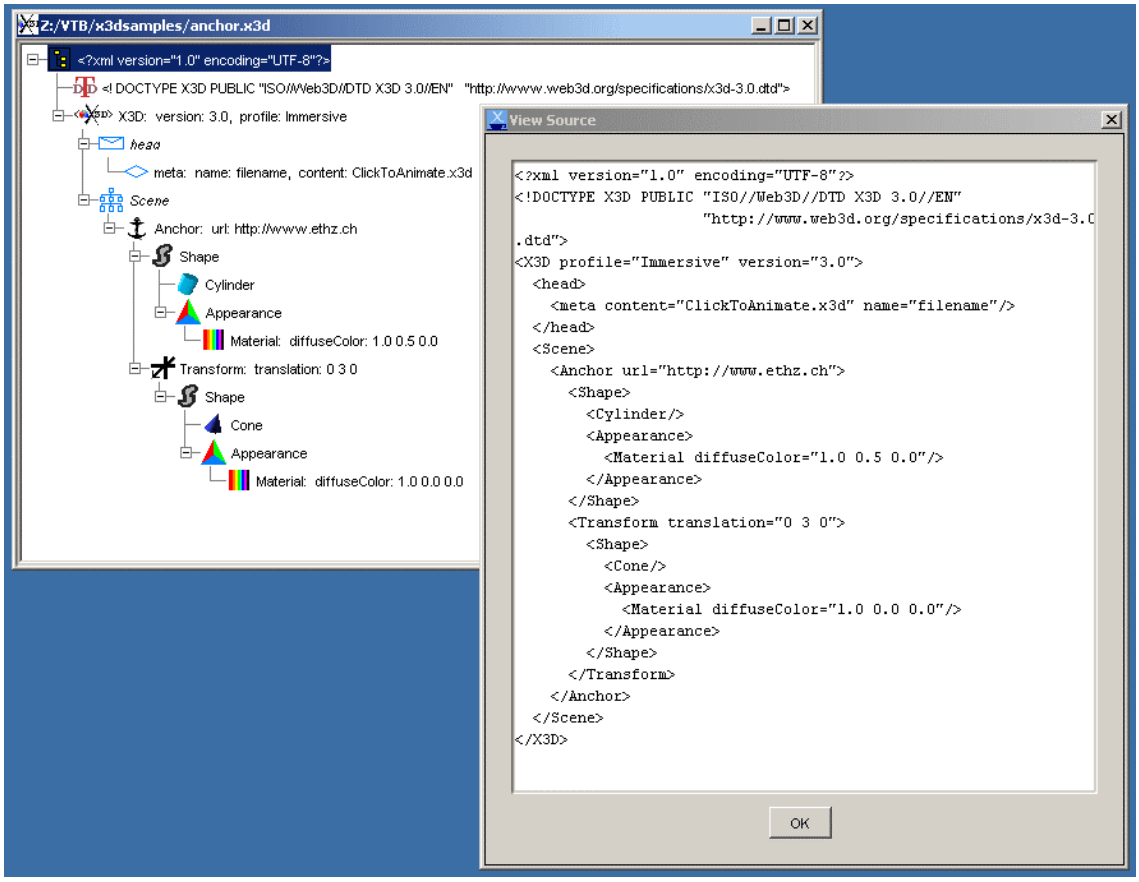


Abbildung 2 Aufbau und Syntax einer X3D-Szene

Abbildung 2 zeigt ein Beispiel mit X3D (in XML-Encoding) als Baumstruktur und als Code. Dabei stellt jedes Tag (z.B. Translation) einen Knoten dar, der wiederum Felder oder Attribute (z.B. translation) enthalten kann.

3 Testen von 3D-Visualisierungssoftware

Um dreidimensionale Inhalte am Bildschirm zu visualisieren, wird eine Software benötigt, die ein 3D-Format wie VRML oder X3D interpretieren und umsetzen kann. In einem ersten Schritt soll mit einer Internet-Recherche nach geeigneten Viewern gesucht werden, die als Freeware oder low-cost-Produkte verfügbar sind. Das Ziel ist einerseits, eine geeignete Software für den weiteren Verlauf der Arbeit zu finden und andererseits eine Übersicht über die verschiedenen Viewer und die Möglichkeiten bei der Verwendung von 3D-Formaten zu gewinnen und sich dabei in das Thema einzuarbeiten.

Die Software wird auf ihre Eignung bezüglich Handling und Leistungsfähigkeit im Umgang mit den zwei oben erwähnten 3D-Formaten getestet. Die Leistungsfähigkeit wird mit einer dreidimensionalen computergenerierten Szene getestet.

3.1 Verwendete Software

Die Internetseite des "web3d" (siehe Quellenverzeichnis) stellt im Zusammenhang mit der Förderung des 3D-Formates X3D eine Link-Liste mit Programmen zur Visualisierung von X3D zur Verfügung. Die verwendete Software stammt aus dieser Link-Liste. Alle Testprogramme sind rückwärtskompatibel mit VRML.

- **Octaga 1.6** (Octaga) : Vollversion, Standalone (auch als Plugin verwendbar)
- **Flux 1.2** beta test release (Media Machines): Plugin
- **Venues** (Vcom3D): Plugin
- **BS contact VRML/X3D 6.2** (Bitmanagement) : voll funktionsfähige Testversion (mit störendem Logo), Plugin

Es gibt natürlich noch weitere Freeware-Programme zur Darstellung von 3D-Formaten. Diese unterstützen aber entweder X3D, Windows oder den Internet Explorer (bei Plugins) nicht. Einen guten Überblick über 3D-Formate und Anwendungsprogramme verschafft die Hompage des NIST (siehe Quellenverzeichnis).

Die Tests mit den Plugins werden nur mit dem Browser Internet Explorer durchgeführt. Dies, um die Testphase nicht zu lange zu gestalten und weil der Internet Explorer der am weitesten verbreitete Browser ist.

3.2 Handling

Zum Testen des Handlings werden alle möglichen Menupunkte und Navigationsmöglichkeiten der Programme angewählt und ausprobiert. Die folgende Auflistung enthält die wichtigsten Eigenschaften für jedes Programm mit Bemerkungen zu ihrer Nützlichkeit und Zusatzinformationen über Spezialitäten, Lizenzgebühren u.ä.. Auf eine tabellarische Darstellung wird verzichtet, da grosse Unterschiede in den Bedienelementen vorhanden sind.

3.2.1 BS contact

- Menus mit rechter Maustaste abrufbar
- Navigation mit Tastatur kombiniert
- bildschirmfüllende Anzeige möglich
- Funktion zur Einstellung als default-Plugin für VRML und X3D
- Navigationsmodi: Gehen, Fliegen, Verschieben, Betrachten, Umschauen, Game (Bewegung auf Tastatur, Richtung und Geschwindigkeit mit Maus: praktisch), Springen, Aufrichten
- Darstellung der Szene als Punktwolke oder Drahtgitter
- Renderer: OpenGL, DirectX 7, DirectX 9
- Geschwindigkeitsregelung
- Sprachen: Deutsch, Englisch, Französisch u.a.
- über den Bildschirm wanderndes Logo bei nicht lizenzierter Testversion (störend)
- Lizenzgebühren für eine Lizenz: 300 EURO minus 30% Academic Discount

3.2.2 Venues

- kein GUI[4], Menus über rechte Maustaste
- Navigation mit Maus (linke, mittlere und rechte Taste)
- Navigationsmodi: Examine, Fly, Walk (Moduswechsel mit N), Anzeige des aktuellen Navigationsmodus in der Statusleiste
- viele Einstellungen mit Tastatur (z.B. Lage des Rotationszentrums, Anzeige der Raum- und Rotationskoordinaten, Wireframe-Darstellung, Lichtquellenposition usw.), nicht sofort ersichtliche aber gute Funktionalitäten

[4] Graphical User Interface: Grafische Benutzeroberfläche

- Möglichkeit der Einstellung von Venues als default-Plugin für VRML und/oder X3D im "Control Settings"-Menu
- Eigenheit: Bewegung stoppt nur bei einem zweiten Mausklick

3.2.3 Flux

- Menuleiste in Viewerfenster oder über rechte Maustaste
- Navigation mit Maus (linke, mittlere und rechte Taste), Shift: schnellere Bewegung
- Navigationsmodi: Examine, Explore (im Rechte-Maustaste-Menu ersetzt durch Fly und Walk), Seek
- Default-Viewpoint nicht der Szene angepasst
- Kein Zugriff auf Preferences und auf die Funktion "straighten up"

3.2.4 Octaga

- Ausgearbeitetes GUI aber ohne Shortcuts -> sehr intuitiv, schnell lernbar aber unpraktisch bei häufigem Gebrauch.
- Navigationsmodi: Walk, Examine, Fly, Slide, Pan, Lookat (Shift: schnellere Bewegung)
- Navigation mit Maus (linke Maustaste und Rad) und mit Pfeiltasten
- Stürzt bei grossen VRML-Szenen ab
- Möglichkeit der Einstellung von Octaga als default-Plugin für VRML oder X3D

3.3 Performance

3.3.1 Grundlagen

Alle Tests werden auf demselben PC durchgeführt. Der Rechner läuft unter dem Betriebssystem Windows 2000 Professional. Eingebaut sind ein AMD Athlon(TM) XP 2200+ Prozessor und 1GB Arbeitsspeicher. Als Grafikkarte wird eine Gforce FX 5200 mit 128MB Speicher verwendet.

3.3.2 Test-Dateien

Zum Testen der Leistungsfähigkeit der verschiedenen Visualisierungsprogramme wird eine 3D-Testszene generiert. Es wird in VRML und in X3D jeweils dieselbe Szene erstellt. Ziel ist es, das Format X3D mit seinem Vorgänger zu vergleichen. Dabei wird in beiden Formaten die DEF-USE-Syntax verwendet, um möglichst vergleichbare Dateien zu erhalten.
Nachfolgend jeweils der Anfang des Codes der Testszenen in VRML und in X3D:

```
#VRML V2.0 utf8

DEF box Shape {
  appearance Appearance {
    material Material {
    diffuseColor 0.6 0.4 0.2
    }
  }
  geometry Box { size 0.8 0.8 0.8
  }
}

Transform {translation 0 0 0 children[USE box]}
Transform {translation 0 1 0 children[USE box]}
Transform {translation 0 2 0 children[USE box]}
...
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE X3D PUBLIC "ISO//Web3D//DTD X3D 3.0//EN"
"http://www.web3d.org/specifications/x3d-3.0.dtd">

<X3D profile='Immersive'>
  xmlns:xsd="http://www.w3.org/2001/XMLSchema-instance"
  xsd:noNamespaceSchemaLocation="http://www.web3d.org/specifications/
  x3d-3.0.xsd">

    <Scene>
      <Group DEF="ChildElement">
        <Shape>
          <Box size="0.8 0.8 0.8"/>
          <Appearance>
            <Material diffuseColor='0.6 0.4 0.2'/>
          </Appearance>
        </Shape>
      </Group>

      <Transform translation="0 0 0"><Group USE='ChildElement'/></Transform>
      <Transform translation="0 1 0"><Group USE='ChildElement'/></Transform>
      <Transform translation="0 2 0"><Group USE='ChildElement'/></Transform>
    ...
```

Abbildung 3 Testdateien im Format VRML und X3D

Damit in kurzer Zeit Szenen mit vielen Knoten geschaffen werden können, werden diese mit der Skriptsprache Python erzeugt. Dabei kann nach dem Programmaufruf gewählt werden, wie viele Würfel in x-, y-, und z-Richtung erzeugt werden sollen. Die Würfel sind in den verschiedenen Test-Dateien folgendermassen angeordnet:

# Würfel total	# Würfel in x	# Würfel in y	# Würfel in z	Dateigrösse [kB]	
				VRML	X3D
100	10	10	1	5	8
1000	10	10	10	49	75
5000	100	10	5	244	377
10000	100	10	10	488	752
50000	100	10	50	2476	3795

Tabelle 1 Aufbau und Grösse der Testszenen

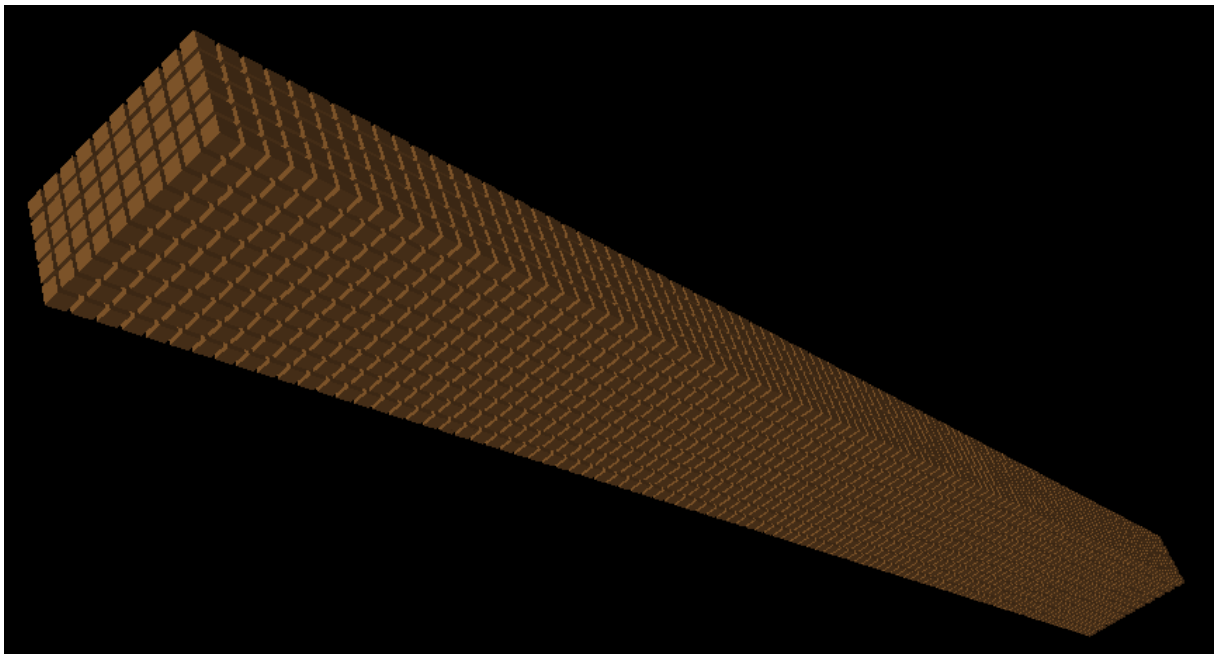


Abbildung 4 Test-Szene: Würfelquader 100x10x5

3.3.3 Test

3.3.3.1 Ladevorgang von Dateien

Der Befehl zum Öffnen der jeweiligen Datei erfolgt aus dem Programm (d.h. Octaga oder Internet Explorer für die Plugins). Die Dateien werden jeweils von einem Netzlaufwerk geladen. Alle Programme können die Dateien direkt öffnen bis auf Flux, welches die VRML-Dateien nicht direkt öffnen kann. Dennoch kann eine VRML-Datei in Flux angezeigt werden, indem man sie mit einem object-Tag in eine einfache HTML-Datei einbindet und diese dann im Explorer lädt.

# Würfel	BS contact		Flux		Octaga		Venues	
	X3D	VRML	X3D	VRML	X3D	VRML	X3D	VRML
100	1.8	0.5	0.6	0.6	0.2	0.2	1.2	1.4
1’000	2.0	0.6	0.6	0.7	0.5	0.2	1.8	1.8
5’000	4.9	0.8	18.2	15.8	1.6	1.6	11.3	10.5
10’000	16.4	1.2	57.2	68.5	9.5	9.5	43.0	41.8
50’000	-	3.4	>300.0	>300	-	-	>300.0	>300

Tabelle 2 Performance der Programme: Zeit für den Ladevorgang einer Datei in Sekunden

Bei 50’000 Würfeln reagieren die Programme BS contact und Octaga nicht mehr, bei Flux und Venues wird der Versuch nach 5 min abgebrochen.

Es wird auch getestet inwiefern Zeilenschaltungen und Leerzeilen (in X3D) einen Einfluss auf die Ladegeschwindigkeit haben (neue Zeile für jedes Transform-Tag und Einrücken des Group-Tags). Es kann kein Unterschied in der Ladezeit festgestellt werden.

3.3.3.2 Bewegungsschnelligkeit

Für die Bewegungsschnelligkeit einer Szene im "examine"-Modus wird eine Bewertung auf einer Skala von 1 bis 4 vorgenommen:

1. sehr weiche Bewegung
2. weiche Bewegung
3. schwaches Ruckeln, Bewegung hinkt Navigation hinterher
4. ruckelt

# Würfel	BS contact		Flux		Octaga		Venues	
	X3D	VRML	X3D	VRML	X3D	VRML	X3D	VRML
100	1	1	1	1	1	1	1	1
1’000	1	1	1	1	1	1	2	1
5’000	1	1	3	2	1	2	3	2
10’000	2	2	4	4	2	3	3	4
50’000		4						

Tabelle 3 Bewegungsschnelligkeit der Szene

Die Bewertung in dieser Form ist natürlich sehr subjektiv, und es ist für den Tester sehr schwierig, die Bewegungsschnelligkeiten zu vergleichen, ohne zwei Programme gleichzeitig laufen lassen zu können. Trotzdem kann aus dem Resultat die grobe Tendenz abgelesen werden.

3.4 Bewertung der Resultate

Aus den drei Tests (Handling, Ladevorgang und Bewegungsschnelligkeit) mit den 4 Visualisierungsprogrammen hebt sich vor allem das Plugin BS contact hervor. Es schneidet nicht nur im Vergleich mit den anderen Plugins, sondern auch im Verhältnis zu dem als Standalone-Version verwendeten Octaga sehr gut ab. BS contact erzielt in fast allen Tests die besten Resultate.

Enttäuschend an Octaga als Standalone-Produkt ist die Performance. Die mangelnde Stabilität bei grossen Dateien und der nicht wahrnehmbare Unterschied in der Bewegungsschnelligkeit zu BS contact erstaunen. Venues fällt punkto Performance deutlich ab, bietet aber einiges an - leider versteckter – Funktionalität. Die Navigationsmöglichkeiten sind nicht so gut aus-

gebaut wie in BS contact. Der Flux-Player hat zwar ein schönes GUI, kann aber in Bezug auf Performance und Handling nicht mit den andern Viewern mithalten. Neben den spärlichen Navigationsmöglichkeiten, können zudem die Einstellungen zur Darstellung der Szene nicht geändert werden.

Der Vergleich von VRML und seinem Nachfolger X3D in der Performance fällt eindeutig zugunsten von VRML aus. Sicher liegt dies zu einem grossen Teil an der schlankeren Syntax von VRML (siehe Dateigrössen in Tabelle 1). Ob es noch andere Gründe dafür gibt, kann an dieser Stelle nicht beurteilt werden.

4 Höhlenvermessung

Da das vorliegende Projekt auf Höhlendaten basiert, soll an dieser Stell ein kurzer Abriss über deren Akquisition und Verarbeitung gegeben werden. Das Kapitel basiert vorwiegend auf Informationen aus der Diplomarbeit von Andreas Neumann (siehe [Neumann, 2000]).

4.1 Datenerfassung

Die Vermessung von Höhlen findet unter z.T. widrigen Bedingungen statt. Es ist dunkel, kalt und meistens auch feucht. Dies stellt spezielle Anforderungen an das Vermessungsmaterial. Es muss klein, leicht, handlich und robust sein, damit es höhlentauglich ist. Es leuchtet ein, dass dabei nicht mit derselben Genauigkeit gerechnet werden kann, wie bei der oberirdischen Vermessung.

4.1.1 Vermessungsinstrumente

Die Distanzmessung erfolgt in der Höhle mit dem Messband oder mit einem elektronischen Handheld-Distanzmesser. Die Richtungsmessung wird mit einer Bussole, die mit einer Lichtquelle versehen ist, realisiert. Die Genauigkeit liegt ungefähr bei einem Grad, je nach Verhältnissen auch schlechter. Für die Neigungsmessung kommen spezielle Klinometer zum Einsatz, die mit einem Lot und einer Kreisscheibe mit Gradeinteilung versehen sind. Es gibt auch bereits elektronische Geräte, die einen Kompass und einen Neigungsmesser in sich vereinen. Die Höhenmessung erfolgt meist indirekt über die Neigungsmessung. Höhen von Eingängen können auch mit herkömmlichen Vermessungsmethoden wie Nivellement oder GPS bestimmt werden. Wenn die Grösse und die Zugänglichkeit der Höhle es zulassen, ist es durchaus auch möglich, Tachymeter oder sogar Laserscanner einzusetzen.

4.1.2 Realisierung der Vermessung

In aller Regel werden Höhlen mit einem Polygonzug vermessen. Dabei werden die Distanz, die absolute Richtung und die Neigung von Messpunkt zu Messpunkt bestimmt. Die Messpunkte werden mit Nagellack, Ölkreide oder fluoreszierende Streifen markiert. Bei jedem Messpunkt werden Daten über Abstände zu Wänden, Decke und Boden und ein Gangquer-

schnitt aufgenommen. Gangquerschnitte dienen später u.a. auch zur genäherten 3D-Modellierung des Höhlensystems. Zudem werden Grundrisse und meistens auch Längsschnitte erstellt.

4.2 Die Höhlenvermessungssoftware Toporobot

Toporobot ist ein in der Schweiz entwickeltes Computerprogramm zur Auswertung von Höhlenvermessungsdaten. Es erleichtert die Arbeit bei der Planerstellung wesentlich und bietet Darstellungsmöglichkeiten, die ohne EDV-Unterstützung undenkbar wären.

Daneben ist Toporobot auch eine vereinheitlichte Vermessungsnotation, welche sowohl die automatisierte Auswertung ermöglicht, als auch das praktische Vorgehen in der Höhle vereinfacht. Die Toporobot-Methode bewährt sich seit Jahren auf zahlreichen Expeditionen und ist inzwischen weitverbreitet. [vgl. Toporobot]

4.2.1 Die grundlegendsten Funktionen von Toporobot

- Komfortable Eingabe der Messdaten
- Validierung der Eingabedaten und Unterstützung bei der Fehlersuche
- Verwalten von wichtigen Attributen wie Vermesser, Datum, Bemerkungen, etc.
- Umgang mit verschiedenen Masseinheiten (Altgrad, Neugrad, Meter, Inch, etc.)
- Reduktion von Polarkoordinaten auf Rechtwinkelkoordinaten
- Ausgleich von Messstrecken bei Rundzügen (möglichst robust und schnell)
- Unterstützung wichtiger Import- und Export-Formate
- Druckmöglichkeiten (auch Grossformat- und Posterdruck)

4.2.2 Struktur der Höhlendaten

Um die Struktur der benutzten Höhlendaten zu verstehen, muss man wissen, wie die Höhlendaten in Toporobot verwaltet werden. Das grundlegende Konzept von Toporobot besteht in der Einteilung der Höhlen in Serien. Dabei werden zusammenhängende Messstrecken zu Polygonzügen zusammengefasst, die dann jeweils einen Gang oder Gangabschnitt bilden. Dabei ist es sinnvoll, die Serien den morphologischen Verhältnissen anzupassen. Messpunkte werden mit zwei Nummern gekennzeichnet: zuerst die Seriennummer, dann die Stationsnummer. Die Seriennummer wird mit eins beginnend und aufsteigend numeriert, am besten in der Rei-

henfolge ihrer Vermessung. Jede Serie muss mit dem Anfangs- oder Endpunkt an einer Serie hängen, die mit einem Eingang/Fixpunkt verbunden ist. Somit darf eine Serie auch rückwärts (in Richtung des Einganges) vermessen werden. Ist der Anfangspunkt einer Serie nicht identisch mit einem bereits bekannten Punkt einer vermessenen Serie, so erhält er die Stationsnummer 0.

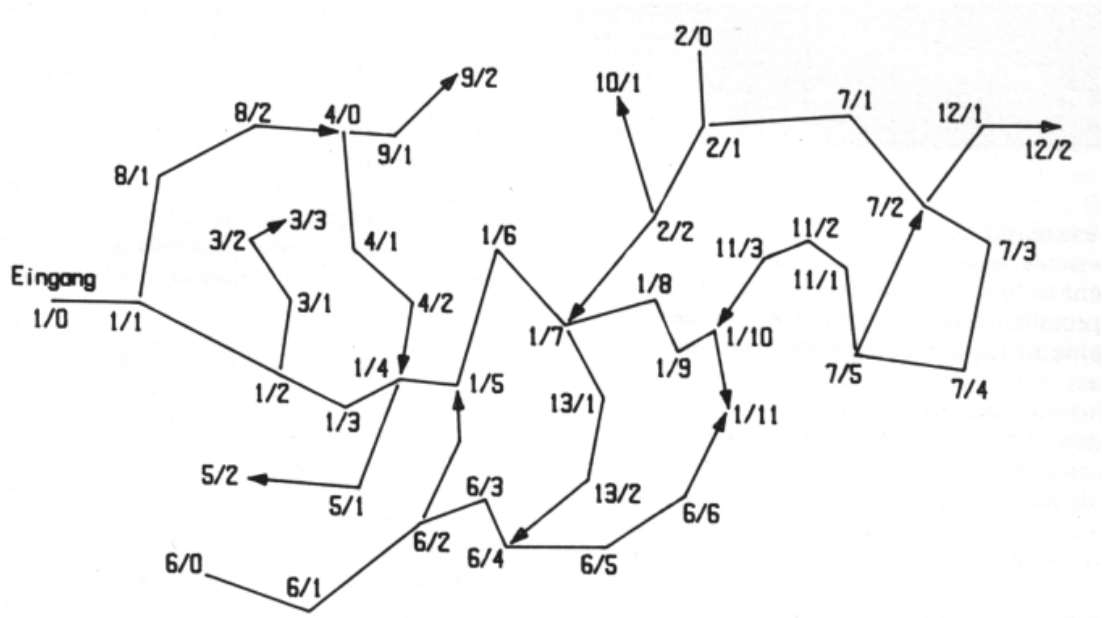


Abbildung 5 Struktur einer Höhle in Toporobot

Mit Toporobot hat man die Möglichkeit, die Höhle in hierarchischen Foldern zu gliedern, etwa nach Höhlenteilen. So kann auch eine ganze Region mit mehreren Höhlensystemen in der gleichen Datei verwaltet werden.

4.2.3 Exportmöglichkeiten

Was die graphischen Exportfilter anbelangt, ist Toporobot sicherlich eines der komplettesten Programme überhaupt. Alle relevanten und gut dokumentierten 2D- und 3D-Formate werden unterstützt und auch laufend an die Bedürfnisse angepasst. So besteht z.B. die Möglichkeit, SVG für den 2D-Graphikaustausch und die Präsentation im WWW zu exportieren. Ausserdem werden die Möglichkeiten der Verknüpfung von Attributdaten mit 3D-Objekten bei den 3D-Exportfiltern VRML und QuickDraw3D laufend ausgebaut. Für die Publikation der Messdaten fürs WWW kann ein graphisch ansprechender, tabellarischer Output mit Verknüpfungen zu anderen Attributtabellen erstellt werden. Im vorliegenden Projekt bilden diese grafischen und numerischen Exporte die Grundlage für den Aufbau des Höhleninformationssystems.

5 Höhleninformationssystem

5.1 Ziel und Inhalt des Höhleninformationssystems

Das Höhleninformationssystem (in der Folge HIS) soll dem Anwender auf anschauliche Weise ein Höhlensystem näher bringen. Dabei soll vor allem die visuelle, dreidimensionale Wahrnehmungsfähigkeit des Anwenders angesprochen werden. Ein komplexes, dreidimensionales Objekt wie eine Höhle, ist in einem virtuellen Raum für einen Menschen zumindest qualitativ einfacher zu erfassen als in zwei Dimensionen (z.B. Schnitte). Hier ist aber auch anzumerken, dass die 3D-Modellierung einen bedeutenden Informationsverlust mit sich bringt: Da das Modell auf den Profilen bei jedem Messpunkt basiert und dazwischen interpoliert wird, gehen viele Details der Höhlengeometrie verloren. Aber auch die grobe dreidimensionale Struktur der Höhle reicht für die Bedürfnisse des Informationssystems aus, da es in der Natur des Informationssystems liegt, zusätzliche, detailliertere zweidimensionale Informationen zur Verfügung zu stellen.

Durch die Interaktion mit dem 3D-Modell können im HIS numerische Daten und Detailpläne einzelner Höhlen oder Gänge aufgerufen werden. Diese Daten könnten auch direkt einzeln und ohne Informationssystem angeschaut werden. Der Vorteil des HIS liegt nun darin, dass diese Daten in semantisch geeigneter Weise geordnet und verlinkt werden. Zudem wird die Suche nach bestimmten Objekten durch visuelle Unterstützung und Navigationshilfen erleichtert. So können zum Beispiel alle Eingänge der Höhlen sichtbar gemacht oder einzelne Höhlen im 3D-Modell angefahren werden.

Das HIS stellt aber noch zusätzliche Analyse-Möglichkeiten zur Verfügung. Die Position des Betrachters im Modell kann zu jedem Zeitpunkt abgefragt werden. Durch Angabe von Azimut und Neigungswinkel kann eine in der Vertikalen verschiebbare Ebene in das Modell gelegt werden. Dies ermöglicht Gangneigungen oder Neigungen ganzer Höhlensysteme zu ermitteln. Durch Zuschalten eines digitalen Geländemodells kann das Höhlensystem bezüglich der Mächtigkeit der Überdeckung analysiert werden.

Das vorliegende HIS ist als Webapplikation konzipiert. Es verwendet ausschliesslich die offenen Standards HTML, Javascript, SVG und VRML. Der Grund, weshalb auf VRML zu-

rückgegriffen und nicht mit X3D gearbeitet wird, wird im nächsten Kapitel erläutert. Ein moderner Browser mit einem Plugin für SVG und dem VRML-Plugin Cortona sollte das HIS darstellen können. Da es aber erst ein Prototyp ist und dieser erst lokal auf Windows2000 mit dem Browser Internet Explorer 6.0 getestet wurde, ist die Webfähigkeit natürlich noch nicht garantiert. Probleme werden bereits bei der Darstellung von SVG in anderen Browsern auftreten, da nur der Internet Explorer das SVG-Plugin unterstützt.

5.2 Probleme bei der Implementierung mit X3D

Der ursprüngliche Gedanke war es, den 3D-Teil des HIS mit dem Format X3D (XML Encoding) und einem Viewer-Plugin aus Kapitel 3.1 darzustellen. Der Zugriff auf die Szene sollte mit Javascript erfolgen. Es musste aber festgestellt werden, dass die Interaktion mit X3D noch nicht einwandfrei funktioniert. Es bleibt die Frage, ob die Probleme auf das Format X3D oder auf den Viewer zurückzuführen sind. Im Falle von Bitmanagement's Viewer "Contact BS" ist anzunehmen, dass das SDK[5] Informationen dieser Art enthält, diese leider aber nicht gratis erhältlich sind.

Die Entscheidung, bei der Implementierung des HIS auf VRML und den Viewer Cortona zurückzugreifen, basiert auf zwei Punkten. Zum einen konnte nicht alle Funktionalität (dynamisches Einfügen von Knoten, Ändern von Mehrfachvariablen u.a.), bzw. die dafür benötigte Syntax gefunden werden. Zum andern ist X3D relativ neu. Dies hat zur Folge, dass die Hersteller von X3D-fähigen Viewern immer noch in deren Entwicklung stehen. So hat die Firma Media Machines zum Beispiel auf den Jahreswechsel 04/05 hin eine neue Version ihres Flux Players versprochen. Zudem ist im Gegensatz zu VRML die Sammlung hilfreicher Forumsbeiträge auf dem Internet noch klein.

5.2.1 Erfolgreiche Befehle zur Interaktion mit Javascript / X3D

Über die X3D Spezifikationen "framework & SAI[6]" und "language bindings (ECMAScript)" (siehe [Web3D]) können einige Befehle gefunden werden, die eine beschränkte Interaktion einer HTML-Seite über Javascript mit der X3D-Szene zulassen. Nachfolgend sollen einige Beispiele dafür aufgeführt werden:

---

[5] Ein Software Development Kit ist eine Sammlung von Programmen und Dokumentation zu einer bestimmten Software, die es Software-Entwicklern erleichtern bzw. erst ermöglichen soll, eigene darauf basierende Anwendungen zu erstellen. [Wikipedia]

[6] Scene Access Interface

Referenz auf die Szene:

```
x3dBrowser = document.x3dplugin.getBrowser();
scene = x3dBrowser.getExecutionContext();
```

Referenz auf den Knoten “sphere1”:

```
sphere = scene.getNode("sphere1");
```

Name und Version des X3D-Plugins abfragen:

```
x3dBrowser.getName();
x3dBrowser.getVersion();
```

Feldwert ändern:

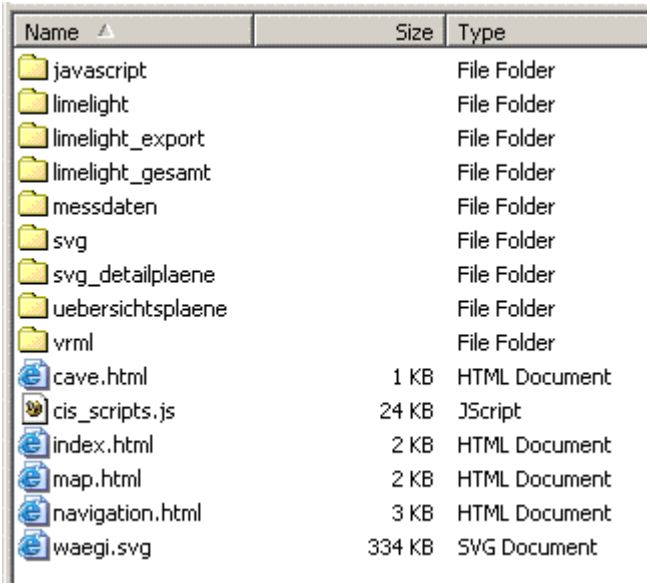
```
sphere.getField("radius").setValue(0.8);
```

Feldwert abfragen:

```
sphere.getField("radius").getValue()
```

5.3 Dateistruktur des Informationssystems

In Abbildung 6 ist die Datei- und Verzeichnisstruktur des HIS dargestellt. Alle Verzeichnisse und deren Inhalte sind bis auf zwei Ausnahmen direkte Exporte aus der Höhlenvermessungssoftware Toporobot. Die erste Ausnahme betrifft das DTM, welches im Nachhinein erstellt wurde. Das DTM, das ebenfalls im VRML-Format vorliegt, muss sich im gleichen Verzeichnis (vrml) befinden wie die Höhlendatei, da diese über ein Externproto darauf zugreift. Die zweite Ausnahme betrifft die beiden Javascript-Dateien im Verzeichnis "javascript". Diese Dateien, die die Definition der Höhlenobjekte bzw. die Höhlenbjekte selbst enthalten, werden aus einer Textdatei (Direkt-Export Toporobot) mit einem Perlscript erzeugt.



Name	Size	Type
javascript		File Folder
limelight		File Folder
limelight_export		File Folder
limelight_gesamt		File Folder
messdaten		File Folder
svg		File Folder
svg_detailplaene		File Folder
uebersichtsplaene		File Folder
vrml		File Folder
cave.html	1 KB	HTML Document
cis_scripts.js	24 KB	JScript
index.html	2 KB	HTML Document
map.html	2 KB	HTML Document
navigation.html	3 KB	HTML Document
waegi.svg	334 KB	SVG Document

Abbildung 6 Dateistruktur des HIS

Die HTML-, Javascript- und SVG-Dateien bilden das eigentliche Informationssystem. Die Datei index.html enthält das Frameset der Webapplikation. Von da aus werden cave.html, map.html und navigation.html aufgerufen. Die Scriptdatei cis_scripts.js ermöglicht die ganze Interaktion des Informationssystems. Der SVG-Plan waegi.svg wird in map.html eingebettet und dient als Übersichtskarte.

5.4 Aufbau der Webapplikation

Das HIS ist so konzipiert, dass die Visualisierungen in 2D und 3D sowie Navigation und Abfrage in einem einzigen GUI geschehen kann. Das GUI wird mit Hilfe von HTML im Internet Explorer realisiert.

Die Einteilung des Browser-Fensters wird mit Frames realisiert. Sie erfolgt nach dem Kriterium der Wichtigkeit der Inhalte für ein dreidimensionales HIS. Dabei wird dem Teil mit der 3D-Visualisierung der grösste Raum zugesprochen. Im Gegensatz zu den Bereichen Karte und Navigation passt sich der 3D-Bereich der Browserfenstergrösse an. Der Karten-Bereich oben rechts ist auf 400 mal 300 Pixel fixiert, wohingegen der Navigations-Bereich unten rechts nur in der Breite (300 Pixel) fixiert ist. Somit ist auch die Darstellung auf kleineren Bildschirmen (Anzeigehöhe des Browser-Fensters kleiner als ca. 800 Pixel) möglich, indem die Scrollbar in einem solchen Fall im Navigations-Bereich aktiviert würde.

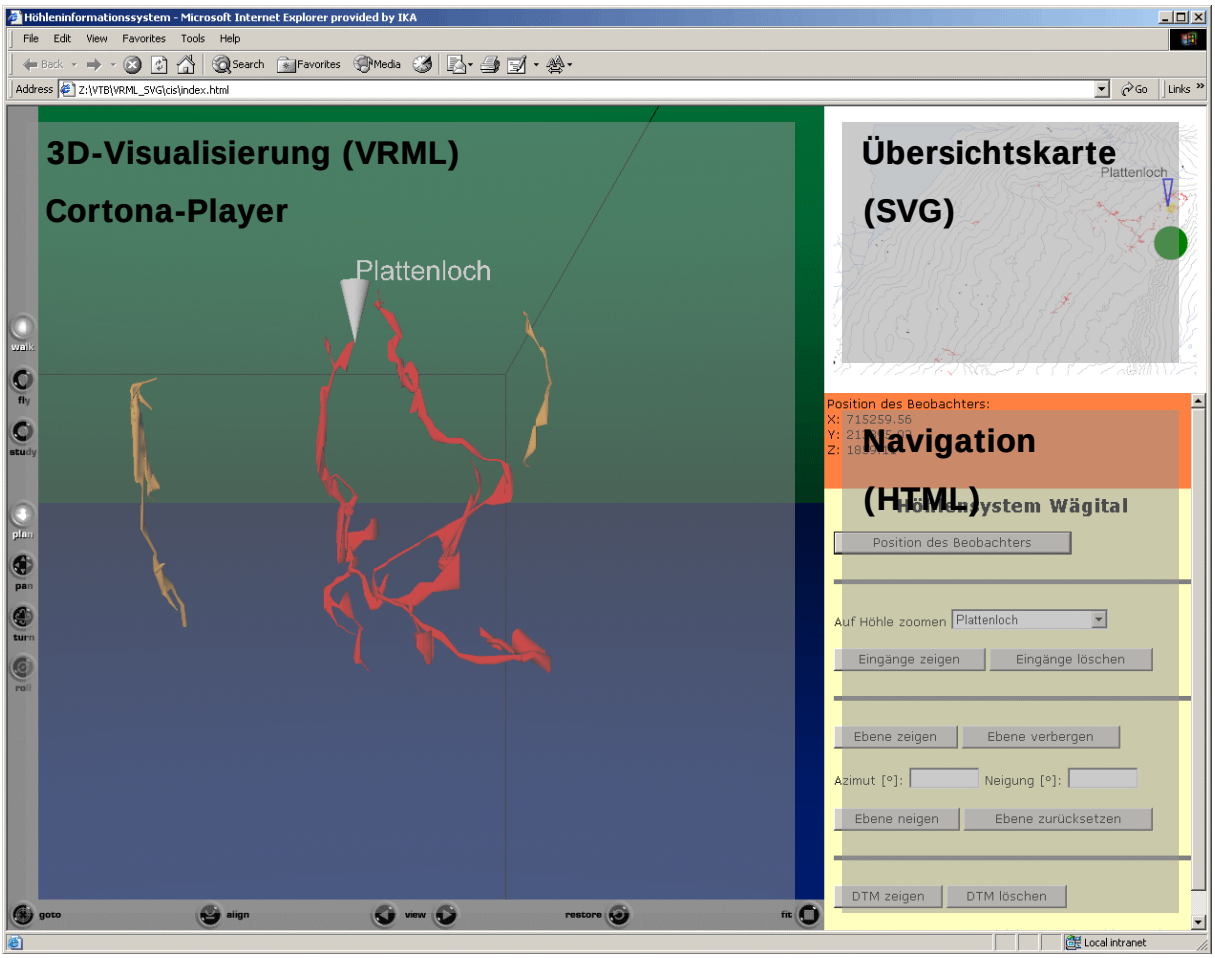


Abbildung 7 Aufbau der Webapplikation

Der Inhalt der Frames ist für die drei Bereiche je wieder ein HTML-Dokument. Dabei sind die VRML-Szene und SVG-Karte mit einem "embed"-Tag in diese HTML-Dokumente eingebettet.

Der Navigations-Bereich gliedert sich in zwei "div"-Container. Im orange hinterlegten Container unter der Karte werden je nach Abfrage dynamisch Informationen hineingeschrieben. Im darunter liegenden Container findet die Interaktion mit dem Höhlenmodell statt.

5.5 Die wichtigsten Befehle zur Interaktion mit VRML

Leider wurde nirgends ein Manual für den Zugriff auf VRML im Cortona Viewer mittels Javascript gefunden. Dies verursachte einen erheblichen Suchaufwand in Foren auf dem Internet. Das Forum von Parallel Graphics, dem Hersteller von Cortona, sei hier speziell erwähnt (siehe Quellenverzeichnis). Nachfolgend sollen daher die wichtigsten Javascript-Befehle zur Interaktion VRML / Javascript aufgelistet werden.

- **Zugriff** auf den Wurzelknoten der VRML-Szene (im 0. Frame, das eine HTML-Datei beinhaltet, in welcher ein embed-Tag mit der id="vrml" enthalten ist). Nachfolgend wird die Variable plugin weiterverwendet.

```
plugin = parent.frames[0].document.getElementById("vrml").Engine
```

- Erstellen von **Javascript-Variablen**, die VRML-Feldwerte übernehmen können und Setzen dieser Variablen:

```
ori_vrml = plugin.CreateField("SFRotation");
ori_vrml = plugin.Nodes("Dummy").Fields("orientation");
ori_vrml.x = 0;
ori_vrml.y = 1;
ori_vrml.z = 0;
ori_vrml.angle = 1.57;
```

- **VRML-Knoten erstellen und zuordnen**:

```
node_red = plugin.CreateNodeFromString("Appearance { material Material { diffuseColor 1 0 0 }}")
plugin.Nodes(shape).Fields("appearance") = node_red
```

- VRML-Knoten erstellen und **im Root Node** einfügen:

```
var node1 = plugin.CreateNodeFromString ("Transform {translation "+co.entrancex+" "+y_cone+" "+-co.entrancey+" rotation 1 0 0  3.14 children [Shape {geometry Cone { bottomRadius "+resize/2+" height "+resize*2+"  } appearance Appearance {material Material { diffuseColor 1 1 1 }}}]}");
plugin.RootNodes.Add(node1);
```

- Zuordnen von Werten auf **Multiple Fields** (MF):

```
plane_vrml = plugin.CreateField("MFVec3f");
plane_vrml.x(0) = 23;
plane_vrml.y(0) = 3;
plane_vrml.z(0) = 10;
plane_vrml.x(1) = 11;
plane_vrml.y(1) = 45;
plane_vrml.z(1) = 22;
```

5.6 Die einzelnen Funktionen

Wie erwähnt, wird die Ganze Funktionalität des Informationssystems von der Datei "cis_scripts.js" gesteuert. Der Javascript-Code inklusive einiger Kommentare kann also dort eingesehen werden. An dieser Stelle soll auf die Prinzipien der wichtigsten Funktionen eingegangen werden.

5.6.1 Position des Beobachters

Die Position des Beobachters in der 3D-Szene wird laufend in der 2D-Karte nachgeführt. Dazu muss die Position des Proximity-Sensors aus der VRML-Datei ausgelesen werden. Der Proximity-Sensor speichert zu jedem Zeitpunkt die Position und die Orientierung des Beobachters in einem vorgegebenen Bereich der 3D-Szene. Da dieser Bereich auf die Höhle beschränkt ist, muss er zuerst vergrössert werden, damit keine Fehlermeldungen bei einer Positionierung ausserhalb dieses Bereiches erscheinen. Das Positionsupdate erfolgt dann zweimal pro Sekunde, indem die Position des Proximity-Sensors auf die Position eines jeweils neu erzeugten SVG-Elementes übertragen wird.

Über den Butten "Position des Beobachters" werden die 3D-Koordinaten der Position in den div-Container im Navigations-Frame geschrieben. Die dynamische Erstellung von HTML über das DOM ist relativ umständlich, da jedes Element inklusive Zeilenschaltung (
) einzeln generiert werden muss.

5.6.2 Auf Höhle zoomen

Bereits beim Laden der Webapplikation werden die einzelnen Höhlen ("caveObjects") dynamisch eingelesen. Die Höhlendaten befinden sich im Verzeichnis "javascript" in der Datei "caveData.js". Die Höhlendaten sind in der Form eines Javascript-Objektes (als Array) abgelegt. Die Höhlennamen werden von da in das Pulldown-Menu im Navigations-Frame geschrieben.

Die Wahl einer Höhle über das Pulldown-Menu löst einige Operationen aus. Über die Eckpunkte der Box, in der sich eine Höhle befindet, kann die Richtung der grössten Ausdehnung einer Höhle berechnet werden. Der Viewpoint des Beobachters wird nun bezüglich der gewählten Höhle so ausgerichtet, dass er senkrecht auf die Richtung der grössten Ausdehnung in einem Abstand von 2.5 mal der grössten Ausdehnung zur Mitte der Box zu liegen kommt. Dabei bleibt die Blickrichtung immer in der X,Z-Ebene[7].

Nachdem allfällige vorherige Höhlen-Highlightings in der Szene gelöscht worden sind, wird die Eingangsmarkierung und die Beschriftung der Höhle im VRML gesetzt. Die Grösse der Eingangsbeschriftung passt sich der Distanz des Viewpoints für diese Höhle an. Anschliessend werden alle Gänge (Serien), die zu der Höhle gehören, rot eingefärbt. Dazu dienen die

[7] Das Koordinatensystem in VRML ist rechtshändig und die positiven Achsen zeigen in X nach rechts, in Y nach oben und in Z gegen vorne.

"seriesObjects", die sich in der gleichen Datei wie die "caveObjects" befinden. Jedes Serien-Objekt kann über ein Attribut mit der Höhlenzugehörigkeit einer Höhle zugeordnet werden.

Als letztes werden die wichtigsten Daten der Höhle in den div-Container im Navigations-Frame geschrieben. Unterhalb dieser Daten werden dynamisch Links auf zusätzliche Daten (Pläne und Messdaten), die in eigenen Dateien gespeichert sind, platziert. Die Übersichtspläne befinden sich im Verzeichnis "SVG" und die Messdaten im Verzeichnis "messdaten". Das Verzeichnis "svg_detailplaene" enthält Grundrisse, Längssschnitte und kombinierte Pläne. Die Information, ob und wieviele dieser Detailpläne vorhanden sind, ist wiederum in den "caveObjects" gespeichert.

5.6.3 Information über Gänge

Der Aufruf von Informationen zu einem einzelnen Gang ist die einzige Funktion, deren auslösendes Event durch Anklicken eines Elementes in der VRML-Szene erzeugt wird. Bei allen anderen Funktionen wird die Befehlskette vom Navigations-Frame (HTML) ausgelöst. Die Übertragung des Events aus dem VRML in die externe Javascript-Datei wird im Kapitel 5.7 beschrieben.

Funktion	Auslösendes Frame	Funktionsdatei	Beeinflusste Frames
Ganginformationen	VRML →	cis_scripts.js →	Navigation, VRML
Auf Höhle zoomen	Navigation →	cis_scripts.js →	Navigation, VRML, SVG

Tabelle 4 Unterschiedliche Befehlsketten

Ähnlich wie bei der Höhlenfunktion werden hier Informationen über einen Gang in den div-Container im Navigations-Frame geschrieben. Auch für die Gänge werden dynamisch Links erzeugt. Die Messdaten der Gänge befinden sich im Verzeichnis "messdaten/gesamt.htm".

5.6.4 Ebene neigen

Durch Eingabe von Azimut und Neigung kann die verschiebbare Ebene in eine gewünschte Richtung geneigt werden. Die Berechnung der Neigung erfolgt über die Vektorrechnung. Ein Punkt (x, y, z) einer Ebene kann dabei folgendermassen dargestellt werden:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} + \lambda \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} + \mu \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

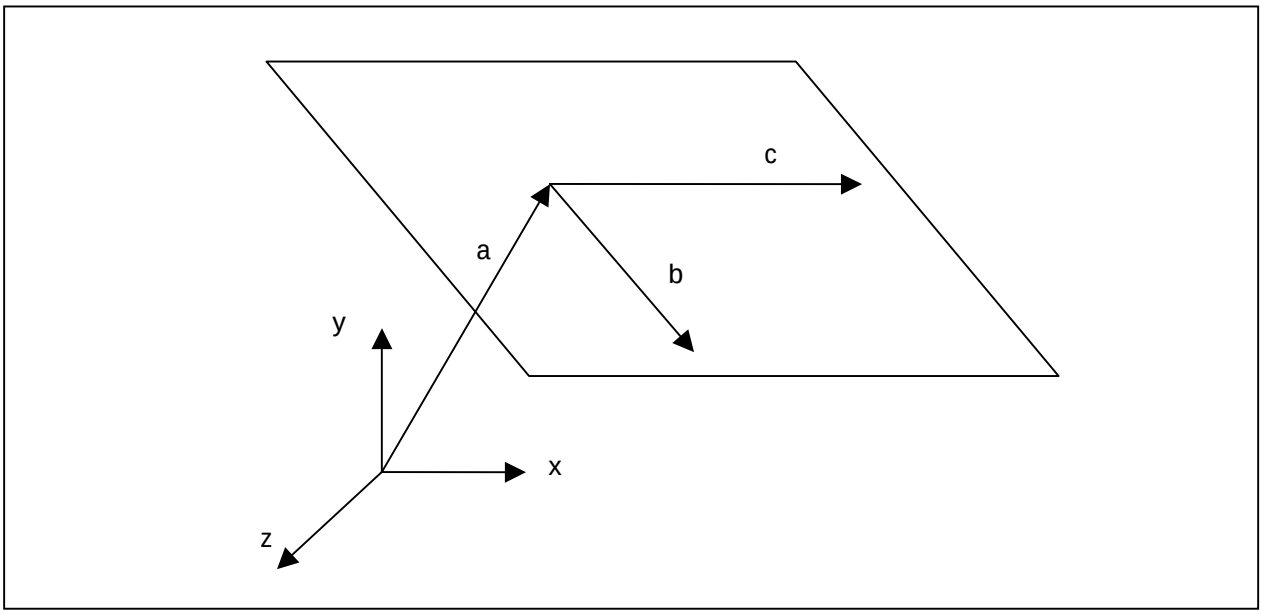


Abbildung 8 Ebenendefinition mit 3 Vektoren

Als Aufpunktvektor a kann eine Ecke der bestehenden Ebene verwendet werden. Die in der Ebene liegenden Vektoren, deren Linearkombination jeden Punkt der Ebene darstellen kann, können nun ausgerichtet werden. Vektor b erhält die Richtung des Azimutes in der X,Z-Ebene und die Neigung in der senkrecht darauf stehenden Ebene. Damit Vektor c auch in der Ebene liegt, wird dieser senkrecht auf b in die X,Z-Ebene gelegt. Mit dieser Ebenengleichung können nun die drei anderen Eckpunkte der Ebene berechnet werden, indem das Gleichungssystem jeweils nach den drei Unbekannten Höhe (y), λ und μ aufgelöst wird.

Die Verschiebung in Y-Richtung der Ebene im horizontalen Zustand ist durch die das Höhlensystem begrenzende Box limitiert. Die Bewegungsfreiheit der Ebene muss in geneigtem Zustand erweitert werden, indem die y-Werte der minimalen und maximale Position des "PlaneSensor" im VRML entsprechend angepasst werden.

5.6.5 Digitales Geländemodell

Das DTM kann über das Setzen des Feldes "whichChoice" auf 0 oder 1 ein- oder ausgeschaltet werden. Es wird als externer Prototyp in die Szene geladen.

5.7 Änderungen im automatisch generierten VRML

Das Ziel des Informationssystems ist es unter anderem, den automatisch generierten Höhlendaten-Output (siehe Kapitel 4.2) direkt und ohne Änderungen nutzen zu können. Dabei sollen nur die Dateien (HTML- und Javascript-Dateien), die für das Informationssystem benötigt werden, im Verzeichnissystem der Höhlendaten platziert werden, und das Höhlensystem sollte funktionsfähig sein.

Beim vorliegenden Prototyp wurde erst mit einem Höhlengebiet gearbeitet und getestet. Dabei mussten am automatisch generierten Datenoutput von Toporobot einige Änderungen in der VRML-Datei vorgenommen werden.

Eine Änderung betrifft die Schnittstelle der Touch-Sensoren nach aussen. Die Funktion "alertGalleryData" ermöglicht es nun, bei der Berührung eines Ganges im 3D-Modell, die ID des Ganges an die externe Script-Datei zu senden und diese dort für die Datensuche und das Gang-Highlighting weiterzuverwenden. Dabei wird die url Zeile in der Definition des Touch-Sensors folgendermassen geändert:

```
url ["javascript: function touched(active) {segtouched=segID; if (active == true) { var url=\"javascript:alertGalleryData('\"+segID+\"')\"; Browser.loadURL(url,'')}}"]
```

Zwei Änderung betreffen die verschiebbare Ebene. Um die Eckkoordinaten mit Javascript ansprechen zu können, wird dem Node "Coordinate" die Definition "plane1" zugeordnet. Zudem wird zum Ein- und Ausschalten der Ebene ein Switch-Knoten um den Group-Knoten gelegt, der die Ebene enthält:

```
DEF slicer1Switch Switch {
 whichChoice 0
 choice  [
  Group {
   children [
    DEF slicer1 Transform {
     children Shape{
      appearance Appearance {
       material Material { transparency 0.4 diffuseColor 0.7 0.7 0.7 }
      }
     geometry IndexedFaceSet {
      coord DEF plane1 Coordinate {
       point [
        715312.29 1488.82 -213798.32,
        ...
```

Der Versuch, den ganzen Shape-Knoten mit "CreateNodeFromString" neu zu schreiben und einzusetzen, scheitert. Wahrscheinlich liegt es daran, dass das Feld "children" des Transform-Knotens "slicer1" ein spezieller Feldtyp ist, der nicht angesprochen werden kann.

Am Ende der VRML-Datei wird ein EXTERNPROTO-Knoten eingefügt. Dieser stellt die Verbindung zum DTM her, welches in einer eigenen Datei "tin.wrl" als PROTO definiert ist.

```
EXTERNPROTO dtm [
 exposedField SFFloat dtmTransp
]
"tin.wrl#dtm"

 DEF surfaceSwitch Switch {
  whichChoice 1
   choice  [
    Group {
     children [
      DEF dtmInst dtm {
       dtmTransp 0
      }
]}]}
```

Die DEFs der Serien (Gänge) in der VRML-Datei haben unterschiedliche Bezeichnungen. Es kommen die Bezeichnungen "s21_0", "s21_1" und "s21_2" vor. Der Javascript-Code ist aber nur in der Lage die Serien mit der Endung "_0" zu finden. Daher werden alle DEFs der Serien auf "_0" geändert.

6 Schlussbemerkungen

6.1 Erreichtes

Der nun vorliegende Prototyp eines dreidimensionalen HIS erfüllt einige grundlegende Bedürfnisse des an Höhlen interessierten Betrachters. In Interaktion mit dem 3D-Modell können Informationen über Höhlen und Gänge mit visueller Unterstützung abgefragt werden. Die zusätzlichen Funktionalitäten der neigbaren Ebene, des einblendbaren DTMs und der Anzeige aller Eingänge bieten weniger einen quantitativen als einen qualitativen visuellen Nutzen. Zudem ist das Informationssystem mit wenigen Anpassungen am Output der Höhlenvermessungssoftware Toporobot auf andere Höhlensysteme portierbar.

6.2 Kleinere Mängel

Die eingebauten Funktionalitäten erfüllen ihr Ziel, haben aber zum Teil noch Schönheitsfehler oder könnten zur verständlicheren Nutzung noch ausgebaut werden. Diese kleinen Mängel konnten vor allem aus Zeitgründen nicht mehr behoben werden.

- **Highlighting der Höhlen**: Werden nacheinander mehrer Höhlen über das Pulldown-Menu angewählt, wird nur jeweils die rote Einfärbung der letzten Höhle gelöscht. Die vorletzte Höhle bleibt rot eingefärbt.
- **Zoom auf Höhle**: Je nach Grösse des Frames, in dem die 3D-Szene angezeigt wird, passt bei einem Zoom auf eine Höhle nicht die ganze Höhle in den Anzeigebereich.
- **Positionsanzeige**: Eine nützliche Information wäre - zusätzlich zur Anzeige der Position in der SVG-Übersichtskarte - die Anzeige der Blickrichtung des Beobachters.
- **Blockierung der Navigation**: Nach einem Zoom auf eine Höhle wird im FLY-Modus aus unbekanntem Grund die Vorwärts-Rückwärts-Bewegung blockiert.

Zudem müsste die Webfähigkeit des HIS getestet werden. Dies bedeutet Tests in verschiedenen Browsern und auf verschiedenen Servern. Einige Probleme dürften z.B. mit der Anzeige von Umlauten auftreten oder - wie bereits erwähnt - mit der Darstellung von SVG in anderen Browsern.

6.3 Weiterführende Möglichkeiten

Das HIS ist fast beliebig erweiterbar. Die folgende Liste von weiteren Möglichkeiten für ein HIS wurde mit wenigen Änderungen aus [Neumann, 2000, Kap. 10.4] übernommen.

- Definition der Position und Blickrichtung des Beobachters in der SVG-Karte
- Anzeige anderer Themen: z.B. geologische Schichten, Hydrologie
- Ableiten von Azimut und Neigung aus einer platzierten Ebene
- Tools zur Distanzmessung
- Einfärben der Gänge nach Attributen (z.B. nach Höhe unter Geländemodell)
- Einfache Animationen, z.B. chronologisches Erscheinen der Gänge nach der Erforschungsgeschichte
- Mehr Interaktion SVG-Karte -> VRML-Szene
- Anzeigen von Foto-Standorten in SVG Übersichtskarte
- Verbessertes User-Interface
- Portierung des Systems auf X3D
- Visualisierung von Fundorten und Mess-Stationen (beispielsweise von Temperaturmessungen)
- usw.

6.4 Dank

Speziell bedanken möchte ich mich bei Andreas Neumann vom Institut für Kartografie der ETH Zürich. Er hat mich jederzeit mit viel Engagement unterstützt und mir wo möglich über die Untiefen der Programmierung hinweggeholfen. Seine Begeisterung für das Thema hat auch mich angesteckt.

Einige gute Impulse erhielt ich auch von Martin Heller, dem Entwickler der Höhlenvermessungssoftware Toporobot und Kapazität im Bereich Höhlenvisualisierung. Auch ihm vielen Dank für die Einblicke in seine Arbeit.

Die OGH (ostschweizerische Gesellschaft für Höhlenforschung) und insbesondere Andreas Dickert zeichnet sich verantwortlich für umfangreiche Dokumentationsarbeiten meiner Testdaten der Lachenstockhöhle im Wägital. Auch bei ihnen bedanke ich mich ganz herzlich.

7 Quellenverzeichnis

7.1 Quellen

Bei Quellen aus dem Internet werden so viele Angaben über das jeweilige Dokument gemacht, wie gefunden werden können. Wo kein Autor vermerkt ist, wird ein passendes Pseudonym für das Dokument verwendet. Das Datum in Klammern bezeichnet den Zeitpunkt des Zugriffs auf die Seite.

- *Eibl*, Maximilian, 2001, http://www.tecchannel.de/internet/760/0.html (19.01.2005)
- *Neumann*, Andreas: Beiträge zur Kartographie und Visualisierung von Karst- und Höhlensystemen (Diplomarbeit Universität Wien), Zürich, 2000, http://www.carto.net/neumann/diplomarbeit/ (19.01.2005)
- *Toporobot*, http://www.geo.unizh.ch/~heller/toporobot/Deutsch/index.html (24.01.2005)
- *Web3D*: http://www.web3d.org/x3d/overview.html (19.01.2005)
- *Wikipedia*: http://de.wikipedia.org/wiki/ (19.01.2005)

7.2 Nützliche Links

7.2.1 Viewer

- Bitmanagement Software GmbH (BS Contact): http://www.bitmanagement.com/
- Media Machines (Flux): http://www.mediamachines.com/index.html
- Octaga: http://www.octaga.com
- Vcom3D (Venues) : http://www.vcom3d.com/Viewer.htm
- ParallelGraphics (Cortona) : http://www.parallelgraphics.com/

7.2.2 Allgemeine Informationen

- VRML Plugin und Browser Detector (NIST: National Institute of Standards and Technology), gute Linkseite: http://cic.nist.gov/vrml/vbdetect.html
- Erklärungen zu VRML und X3D: http://xml.coverpages.org/vrml-X3D.html

- Tooltips (X3D Node-Beschreibungen): http://www.web3d.org/x3d/content/X3dTooltipsGerman.html
- VRML Forum : http://www.neeneenee.de/vrml/forum/php/index.php