Inhaltsverzeichnis

Vorwort

Die vorliegende Ausarbeitung entstand im Rahmen meiner Diplomarbeit an der Fachhochschule Konstanz. Das Thema der Diplomarbeit umfasst die Entwicklung eines onlinefähigen Systems zur interaktiven 3D Visualisierung von Landschaftsausschnitten.

3D Visualisierung ist ein sehr interessanter, jedoch auch ein, wie ich während meiner Diplomarbeit festgestellt habe, sehr komplexer Teil der Computerwissenschaften.
Ich habe durch die Bearbeitung dieses Themas festgestellt, dass ich ein starkes Interesse im 3D Bereich habe und deshalb einen Nebenjob in dieser Sparte begonnen, in welchem ich nach der Beendigung dieser Diplomarbeit in Vollzeit arbeiten werde.

Besonderer Dank gilt meinem Betreuer Herrn Prof. Dr. Hedstück, Fachhochschule Konstanz, für die hervorragende Betreuung meiner Diplomarbeit.

Weiterhin möchte ich auch Herrn Christian W. Richter vom Verein Europäisch-Karibische Gesellschaft e.V. in Konstanz danken. Ursprünglich sollte in dieser Diplomarbeit eigentlich ein Ausschnitt eines Nebelwald Schutzgebietes in der Karibik visualisiert werden, jedoch waren die entsprechenden 3D Daten noch nicht bearbeitungsfähig. Daraufhin vermittelte mich Herr Richter an das Institut für Landschaftsökologie und Naturschutz in Singen.

Ich möchte auch ganz herzlich Herrn Alfons Krismann vom Institut für Landschaftsökologie und Naturschutz in Singen für die Bereitstellung der Daten danken. Auch möchte ich mich bei allen Kollegen der Firma IFAD A/S hier in Dänemark bedanken, besonders bei Nikolai V. Christensen, der immer ein offenes Ohr für meine Fragen hatte. Mein Dank gilt auch Eva Rommelfanger, welche sich bereit erklärte, meine Ausarbeitung korrekturzulesen. Schließlich gebührt auch meiner Freundin Belinda Klemmensen für Ihre moralische Unterstützung während meiner Diplomarbeit herzlicher Dank.

Odense, den 23 März 2002 Andreas Mößner

Abbildungs- und Tabellenverzeichnis

Abbidlungsverzeichnis:

Tabellenverzeichnis:

Abkürzungsverzeichnis

API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
CD-ROM	Compact Disk Read Only Memory
DARPA	Defence Advanced Research Projects Agency
DHM	Digitalles Höhenmodell
DTED	Digital Terrain Elevation Data
EAI	External Authoring Interface
EG	Elevation Grid
FPS	Frames per Second
GIF	Graphics Interchange Format
HTML	Hypertext Markup Language
HTTP	Hyper Text Transfer Protokol
ISO	International Standard Organisation
JPEG	Joint Photographics Expert Group
LOD	Level of Detail
MPEG	Motion Pictures Expert Group
MS	Microsoft
OPENGL	Open Graphics Library
PIXEL	Picture Element
ROAM	Real-time Optimally Adapting Meshes
SGI	Silicon Graphics Inc.
SIGGRAPH	Special Interest Group On Computer Graphics
SRI	Stanford Research Institute
URL	Unified Resource Locator
VET	Viewing Experience Technology
VRML	Virtual Reality ModelleingLanguage
X3D	Extensible 3D
XML	Extended Markup Language

1. Einführung

Im Rahmen dieser hier vorliegenden Diplomarbeit sollen die Techniken und Möglichkeiten zur Erstellung eines virtuellen 3D Modells eines natürlichen Landschaftsausschnittes erörtert werden. Zu entwickeln ist ein 3D Modell des Naturschutzgebietes Hohentwiel, welches die Möglichkeit bietet, die integrierten Inhalte interaktiv im Internet zu betrachten.

Der Hohentwiel, Wahrzeichen der Stadt Singen im Bundesland Baden-Württemberg, ist ein durch vulkanische Aktivitäten entstandener Berg, welcher auf Grund seiner einzigartigen Pflanzen- und Tierwelt seit 1941 unter Naturschutz steht. Das geschützte Gebiet erstreckt sich über eine Fläche von 108 ha, und mit einer Höhe von fast 270 m „thront“ der Hohentwiel über der Stadt Singen.

Die staatliche Naturschutzverwaltung Baden-Württemberg, genauer das Institut für Landschaftsökologie und Naturschutz in Singen, unterhält einen so genannten „Vulkanpfad“, welcher dem interessierten Besucher und Naturfreund auf rund 3 km Länge die Entstehung und Entwicklung des Berges näher bringt. Der Pfad führt rund um den Hohentwiel und beschreibt mit insgesamt 13 Informationstafeln am Rande des Weges alles Wissenswerte zum Naturschutzgebiet.

Ziel dieser Arbeit ist eine Untersuchung und Bewertung der momentan verfügbaren Tools und Standards, um Landschaften in der Größenordnung des Hohentwiel interaktiv im Internet zu visualisieren. Daraufhin soll eine Entscheidung getroffen werden, welche Methode sich am besten dazu eignet, die Landschaft des Hohentwiel zu implementieren. In diese Entscheidung fließen unter anderem der Preis, die Handhabung und die Flexibilität der untersuchten Methoden mit ein.
Des weiteren soll detailliert aufgezeigt werden, wie der Landschaftsausschnitt des Hohentwiel mit Hilfe der gefundenen Methode implementiert wird. Das so entstehende System sollte auch auf einem älteren Rechner mit einer langsamen Modemverbindung lauffähig sein und dem Benutzer ein Höchstmaß an Bedienungskomfort bieten.

2. Aktuelle Entwicklungen

Im folgenden Kapitel soll beschrieben werden, welche Techniken zur 3D Visualisierung im Internet zum Zeitpunkt der Erstellung dieser Ausarbeitung zur Verfügung stehen und inwiefern diese speziell im Hinblick auf die Modellierung virtueller Landschaften geeignet sind.

2.1 VRML97

Im Internetbereich ist 3D seit einigen Jahren eines der wichtigsten Schlagwörter geworden.

Diese Entwicklung begann Mitte der neunziger Jahre, genauer im Jahre 1994, mit der Einführung des VRML 1.0 Standards. VRML steht für Virtual Reality Modeling Language und ist eine Sprache, die zur Darstellung dreidimensionaler Objekte im Internet dient. VRML Dateien sind ähnlich wie HTML Dateien als ASCII Textdatei abgespeichert und werden von einem entsprechenden VRML Browser interpretiert und dargestellt. Im Jahre 1997 wurde die Version 2.0 von VRML als ISO Standard verabschiedet, welche auch unter dem Namen VRML97 bekannt ist. VRML97 bietet, aufbauend auf VRML 1.0, zusätzlich die Möglichkeit animierte Szenen zu entwickeln und die Einbindung von Multimediadateien wie Audio und Video. Trotz der enormen Möglichkeiten, die VRML bietet, konnte es sich nie richtig durchsetzen und geriet im Laufe der Jahre etwas in Vergessenheit, ja wurde sogar von verschiedenen Institutionen als unbrauchbar abgestempelt. Dies sind jedoch stark subjektive Aussagen, da sich in der VRML Welt doch noch einiges tut und momentan unter der Schirmherrschaft des Web3D Konsortiums an der Entwicklung eines neuen Standards gearbeitet wird, welcher auf VRML97 aufbaut und bald unter dem Namen X3D verabschiedet werden soll.

VRML ist durch seine offene Architektur und seine Erweiterbarkeit durch verschiedene Scriptsprachen ein Datenformat, welches sehr gut dazu geeignet ist, große Mengen an 3D Daten - wie sie z.B. speziell bei der Modellierung von Landschaften auftreten - performant zu verarbeiten und darzustellen. Da VRML für das Internet konzipiert wurde ist es natürlich auch möglich, VRML Inhalte über das Internet anzuschauen. Hierzu wird ein von verschiedenen Firmen kostenlos zur Verfügung gestelltes VRML-Plugin benötigt, welches sich in einen Internet-Browser, wie z.B. Internet Explorer oder Netcape integriert. Ist einmal ein Plugin in einem Browser installiert, können mit diesem Browser VRML Inhalte dargestellt werden.

Jedoch treten in diesem Zusammenhang häufig Probleme auf. Die verschiedenen Plugin Hersteller halten sich unterschiedlich genau an die VRML97 Spezifikation und haben oft ihre eigenen Erweiterungen in ihre Plugins eingebaut. Dies führt dazu, dass Inhalte in verschiedenen Plugins unterschiedlich aussehen können und dass manche Details gar nicht funktionieren. Wenn man also in seiner Anwendung ein breites Angebot von VRML Plugins unterstützen will ist es wichtig, nicht die proprietären Erweiterungen der Hersteller zu benutzen, sondern entsprechend sehr eng an der VRML97 Spezifikation, welche von allen Browsern unterstützt werden sollte, zu programmieren.

2.2 X3D

Das Web3D-Konsortium arbeitet derzeit an der Weiterentwicklung des VRML-Standards und dabei speziell an der Kombination von Web3D-Techniken mit dem XML-Format, dem Nachfolger von HTML. Genannt wird diese neue Technik X3D, was für Extensible 3D steht. X3D ist es eine Implementierung des VRML Standards in XML. Das heißt, VRML Inhalte werden im XML Format beschrieben. Dies hat unter anderem den Vorteil, dass sich ein Programmierer, der mit XML vertraut ist, viel schneller mit VRML zurechtfindet. Des Weiteren ist es ein Vorteil, dass sich XML zur Zeit ziemlich stark entwickelt und momentan das Datenbeschreibungsformat der Zukunft darstellt. Somit ist auch die Kompatibilität mit anderen Produkten gewährleistet. Jedoch beschränkt sich X3D nicht nur auf die Implementierung der VRML Spezifikation in XML, es sollen so zusätzlich alle Multimedia-Inhalte des Internets vereint werden, seien es Text, Sound, Video, 2D-Grafiken oder Echtzeit-3D-Grafiken. X3D soll kleine leistungsfähige Komponenten, wie z.B. eine schnelle 3D-Runtime Engine enthalten. Außerdem soll es ein plattformunabhängiges Dateiformat besitzen, das abwärtskompatibel mit VRML97 sein soll. Schließlich soll es, wie schon angedeutet, fortschrittliche XML-Integration bieten. Das Ziel des Web3D-Konsortiums ist die Vereinigung von X3D und VRML97 zu einem neuen offiziellen ISO-Standard. Die X3D Arbeitsgruppe steht in engem Kontakt mit den führenden Herstellern von 3D Browsern für das Internet. Die Browserfirmen versprechen sich viel von dem neuen Standard, und es ist geplant, ihn sukzessive in neue Produkte zu integrieren.

2.3 GeoVRML

Eine weitere Arbeitsgruppe des Web3D-Konsortiums ist die GeoVRML Arbeitsgruppe. Dieses Projekt hat zum Ziel, VRML 97 so zu erweitern, dass mit einem gewöhnlichen VRML Plugin auch geospezifische Daten wie z.B. verschiedene Koordinatensysteme dargestellt werden können. Diese Erweiterung resultiert in gegenüber VRML97 leicht abgeänderten Knoten, welche erweiterte Informationen speichern. Um GeoVRML Daten zu visualisieren, muss eine zusätzliche Komponente auf dem jeweiligen Computer installiert sein, welche ein gewöhnliches VRML Plugin GeoVRML fähig macht.

2.4 Java3D

Im Rahmen der Anstrengungen der Firma Sun Microsystems, ihre Programmiersprache Java zur Allzweck-Sprache des Internets zu machen, wurde die Java3D API entwickelt. 1997 wurde auf der Messe SIGGRAPH97 eine erste Spezifikation dieser neuen Technik vorgestellt. Java3D ist eine Sammlung von Klassen, welche als Erweiterungspaket für Java angeboten werden. Java3D ist keine eigenständige Software, es kann vielmehr als eine Klassenbibliothek für Java gesehen werden, die dessen Fähigkeiten im 3D Bereich beträchtlich erweitert. Java3D bietet ähnlich wie VRML, die Möglichkeit zur Erstellung umfassender 3D Welten. Jedoch ist Java3D um einiges komplexer als VRML und bietet dem Programmierer somit auch weit mehr Möglichkeiten zur Gestaltung seiner Objekte, da in Java3D Programmen natürlich auch alle Sprachkonstrukte der Standard Java API verwendet werden können. Java3D bietet die Möglichkeit, dreidimensionale Inhalte, welche in anderen Programmen, wie zum Beispiel 3D Studio Max®, erstellt wurden zu importieren. Dies ist möglich über so genannte Java3D-Loader. Diese Loader lesen die angegebene Datei und konvertieren die enthaltenen Daten in die interne Java3D Datenstruktur. Die meisten dieser Loader unterstützen jedoch nicht die volle Spezifikation des Datenformats, dessen Daten sie importieren sollen. So unterstützt zum Beispiel der Java3D VRML-Loader noch nicht alle Knoten, die für VRML97 spezifiziert sind. Jedoch ist besonders der Java3D VRML-Loader unter ständiger Bearbeitung und wird im Hinblick auf die VRML97 Spezifikation laufend vervollständigt, da er mitunter der wichtigste Loader ist, um 3D Inhalte in Java3D zu importieren. Zu erwähnen wäre hier auch, dass der VRML Loader vom gleichen Programmierteam des Web3D Konsortiums stammt, welches auch am oben erwähnten

neuen VRML Standard X3D arbeitet. Somit kann dieser Loader auch die schon implementierten Knoten des neuen Standards lesen und wird auch in dieser Richtung momentan stark erweitert. Zum Zeitpunkt der Drucklegung dieser Ausarbeitung steht als neueste Version Java3D 1.3 Beta zur Verfügung.

Java3D ist eine umfangreiche API zur Darstellung dreidimensionaler Inhalte. Es lassen sich sehr komplexe Systeme implementieren, welche durch eine ausgefeilte Technik zum Zugriff auf 3D Beschleunigungshardware und eine hochperformante 3D Engine auch zeitgemäß dargestellt werden können. Da Java durch seine Applet Technologie bekanntlich internetfähig ist, ist es auch möglich, Java3D Inhalte online zur Verfügung zu stellen. Jedoch sind hiermit verschiedene Komplikationen verbunden: um ein Java3D Modell im Browser darstellen zu können, muss auf dem jeweiligen Computer die Java- und zusätzlich die Java3D Runtime-Engine installiert sein. Von beiden gibt es verschiedene Versionen, welche unterschiedlich gut zusammenarbeiten. Des Weiteren unterstützen viele Internet-Browser noch nicht die Java 2 Spezifikation, und somit ist nicht gewährleistet, dass ein Java3D Programm in jedem Internet Browser läuft.

2.5 TerraVision

Unter dem Namen TerraVision stellt SRI International ein verteiltes, interaktives System zur Landschaftsvisualisierung zur Verfügung. Entwickelt wird dieses System im Rahmen des DARPA gesponserten Digital Earth Projekts, welches sich zum Ziel gesetzt hat, eine offene Infrastruktur zur Verfügung zu stellen, welche es ermöglicht abhängig vom Standort weltweit nach Landschaftsdaten zu suchen.

TerraVision ist ein System, das ermöglicht, interaktiv große geographische Gebiete zu visualisieren. Es ist in der Lage, sehr umfangreiche Mengen lokal oder entfernt gespeicherter Daten, wie Luft- und Satellitenbilder, topographische Daten und Wetterdaten, Gebäude und andere Landschaftsmerkmale, abzurufen und zu einem Modell zu vereinigen. Die Daten, die in der Größenordnung von Terrabytes und möglicherweise über verschiedene Server im Web verteilt vorliegen, können mit der SRI GeoWeb Technologie automatisch abgefragt werden. Das System implementiert ausgeklügelte Algorithmen zur Performance Optimierung wie z.B. multiresolution Tiles und Level of Detail (LOD) Management.

Die Vollversion des Systems besteht aus verschiedenen Komponenten und Tools, mit welchen es möglich ist, eigene Landschaftsdaten so aufzubereiten, dass sie im Internet präsentiert werden können, oder aber vorhandene Daten im Netz abzurufen und in eine eigene Anwendung einzubauen. TerraVision benutzt eine erweiterte Version des Apache Webservers als Webschnittstelle. Der Endbenutzer des Systems benötigt ein von SRI zur Verfügung gestelltes Plugin, welches sich im Internet Explorer oder Netscape installieren lässt und die Benutzung des Systems ermöglicht.

2.6 G-Vista

Die deutsche Firma G-Graphics vertreibt eine Produktsuite zur Visualisierung großer Landschaften. Kern dieser Suite ist die G-Render Library, eine auf Direct3D, bzw. OpenGL aufbauender 3D Engine. Das Produkt G-Vista ist eine ActiveX
Komponente, welche in alle ActiveX fähigen Anwendungen wie z.B. den Microsoft Internet Explorer oder aber Word und Powerpoint eingebettet werden kann und zur Darstellung von 3D Terraindaten dient. Des Weiteren bietet die Firma ein Tool namens G-Scene, welches zur einfachen Erstellung von virtuellen Landschaften dient. Das System unterstützt verschiedene Detaillierungsgrade der Landschaft, welche sich auch kontinuierlich anpassen, wenn man sich in der Landschaft bewegt. Dies bedeutet, dass sukzessive detaillierte Daten nachgeladen werden, je kleiner der Abstand zu einem Objekt (z.B. einem Berg oder einem Haus) wird. Die Implementierung dieser Technik ist sehr gut gelungen. Auf der Demoseite der Firma kann man einen virtuellen Flug durch die Schweiz starten. Das System kann mit der immensen Datenmenge der kompletten Schweiz vorzüglich umgehen, sogar ein Zoom aus einer Höhe von mehreren Kilometern bis auf einzelne Gebäude und Strassen in Zürich geht bei einer schnellen ADSL Verbindung reibungslos vonstatten. Außer der äußerst performanten 3D Engine glänzt das System mit leicht erlernbarer Navigation und mit Hyperlinks, welche Objekten in der 3D Welt zugewiesen werden können und welche direkt auf gewählte Webseiten leiten.

2.7 VET

Es gibt noch eine Reihe anderer Web3D Entwicklungen, die sich in letzter Zeit einen Namen gemacht haben. Unter anderem hat z.B. die Firma Viewpoint ein Produkt namens VET hervorgebracht. VET ermöglicht die Darstellung von 3D Daten in einem Browser und besticht durch Features wie Licht- und Schatteneffekte. Darüber hinaus weist VET ein ausgeklügeltes Verfahren zum inkrementellen Laden von 3D-Objekten vor. Selbst bei einer schlechten Modemverbindung baut der Browser schon nach

kürzester Zeit eine bildliche Darstellung auf. Insgesamt sind VET Dateien relativ kompakt, was durch entsprechende 3D Komprimierung der Geometriedaten und Texturen erreicht wird.

2.8 Cult3D

Ein ähnliches System bietet auch die schwedische Firma Cycore mit ihrem Produkt Cult3D. Auch dieses Produkt besticht durch seine Darstellungsqualität und kurze Ladezeiten.

Beide Firmen verfolgen ein spezielles Geschäftsmodell zur Finanzierung ihrer Produkte. Der Kunde bekommt die Software zur Erstellung von Web3D Inhalten kostenlos, jedoch muss für die Veröffentlichung der Inhalte im Web eine Lizenzgebühr bezahlt werden. Mit anderen Worten, der Betreiber einer 3D Webseite bezahlt, wenn er seine 3D Modelle als VET oder Cult3D Datei ablegt.

VET und Cult3D benötigen ein Plugin, welches im Webbrowser installiert wird. Dieses Plugin ermöglicht die interaktive Darstellung von 3D Inhalten innerhalb von Webseiten. Jedoch ist die Leistungsfähigkeit dieser zwei Produkte im Bezug auf Landschaftsdarstellung sehr eingeschränkt. Normalerweise enthält ein Landschaftsmodell viele tausend, gar hunderttausend Polygone, welche von der jeweiligen 3D Engine gerendert und dargestellt werden müssen. VET und Cult3D sind jedoch nicht für diese Art von 3D Modellen vorgesehen. Die Stärken liegen mehr in Bereichen wie Produktpräsentation, z.B. von einem Auto oder einem Computer. Solche 3D Modelle sind bei weitem nicht so umfangreich wie 3D Terrain Modelle und enthalten auch entsprechend weniger Polygone.

3. Untersuchung und Bewertung von Tools und Standards

In diesem Kapitel werden die im vorigen Kapitel aufgezeigten Tools und Standards genauer durchleuchtet und auf ihre Verwendbarkeit für die Landschaftsmodellierung hin bewertet.

3.1 VRML97

VRML97 ISO/IEC 14772-1:1997	
Name	VRML97 Standard

Homepage	http://www.web3d.org/technicalinfo/specifications/vrml97/index.htm
Hersteller	VRML baut auf der OpenInventor Technologie von SGI auf, wurde dann aber als offener Standard verabschiedet und wird seither von einem unabhängigen Konsortium verwaltet.
Preis	Die Benutzung des Standards ist kostenlos. Es existieren eine Reihe kostenloser Tools zur Erstellung und Darstellung von VRML Inhalten, es sind aber auch kostenpflichtige, kommerzielle Produkte erhältlich.
Plattformen	Windows, Unix, Linux, Mac, SGI, Windows CE
Dateiformat	VRML Dateien werden im unverschlüsselten ASCII Format abgelegt. Sie können zusätzlich mit einem Zip-Programm komprimiert werden.
Webtauglichkeit	VRML wurde für das Internet entwickelt und lässt sich daher sehr einfach ins Web integrieren. Jedoch bestehen einige Probleme: VRML Dateien sind nicht streaming fähig, das heißt eine Datei muss immer zuerst vollständig herunter geladen werden bevor sie angezeigt werden kann. Des Weiteren können die Dateien ziemlich groß werden, da im Gegensatz zu anderen Formaten keine speziellen 3D Komprimierungen vorgesehen sind.
Hauptsächliche Anwendungsgebiete	VRML wird praktisch in allen Gebieten der 3D Visualisierung eingesetzt, konnte sich aber im Internet, für welches es eigentlich entwickelt wurde, nie so richtig durchsetzen. Heute wird es hauptsächlich von wissenschaftlichen Instituten und Forschungseinrichtungen für die Visualisierung spezifischer Anwendungen eingesetzt.
Tauglichkeit für Landschaftsmodelle	VRML97 bietet einen Knoten namens „ElevationGrid“ mit welchem sich 3D Landschaftsdaten kompakt in VRML darstellen lassen. Bei sehr großen Modellen mit einigen hunderttausend Polygonen ist die 3D Engine jedoch sehr schnell überfordert und kann die Daten nicht mehr flüssig darstellen. VRML bietet jedoch durch seine offene Struktur die Möglichkeit, diese Einschränkungen zu umgehen und ist somit gut geeignet um virtuelle Landschaften zu implementieren.

Table 1 VRML 97

3.2 X3D

Name	X3D
Homepage	http://www.web3d.org/fs_x3d.htm
Hersteller	X3D wird von der X3D Arbeitsgruppe unter der Schirmherrschaft des Web3D Konsortiums entwickelt. Die X3D Arbeitsgruppe steht in engem Kontakt mit den führenden 3D Browser Firmen und unter anderem auch mit der MPEG Arbeitsgruppe.
Preis	Die Verwendung von X3D ist kostenlos
Plattformen	Theoretisch läuft X3D auf allen Plattformen, auf denen ein X3D kompatibler Browser zur Verfügung steht.

Dateiformat	X3D Dateien werden momentan als Textdateien im XML Format abgelegt. In der weiteren Entwicklung von X3D soll aber auch ein kompaktes binäres Dateiformat eingeführt werden, welches dann streamingfähig ist.
Webtauglichkeit	Sehr gute Webtauglichkeit, da es eine ganz neue Technologie ist und speziell auf die heutigen Anforderungen im Internet zugeschnitten ist.
Hauptsächliche Anwendungsgebiete	X3D steckt momentan noch in den Kinderschuhen und ist somit noch nicht sehr weit verbreitet. Es soll aber in näherer Zukunft als Standard verabschiedet werden und VRML97 ablösen. Die X3D Arbeitsgruppe hat einen X3D fähigen Browser entwickelt, der auf Java3D aufbaut und auch VRML97 Inhalte darstellen kann. Diese Implementierung ist jedoch noch sehr stark fehlerhaft und kann somit als nicht mehr als ein Prototyp angesehen werden.
Tauglichkeit für Landschaftsmodelle	Da X3D noch im Entwicklungsstadium ist, sind mehrere Knoten, welche zur Implementierung größerer Landschaftsausschnitte sehr wichtig sind, noch nicht oder nur unzureichend vorhanden. Falls aber X3D bald als Standard verabschiedet werden wird und all die versprochenen Features beinhaltet, wird es voraussichtlich sehr gut dazu geeignet sein, Landschaften zu visualisieren.

Table 2 X3D

3.3 GeoVRML

	
Name	GeoVRML
Homepage	http://www.geovrml.org
Hersteller	GeoVRML wird von der GeoVRML Arbeitsgruppe unter der Schirmherrschaft des Web3D Konsortiums entwickelt. Es stellt eine Erweiterung des VRML97 Standards bezüglich der Verwendung von Geographischen Daten wie z.B. verschiedener Koordinatensysteme dar.
Preis	GeoVRML ist kostenlos
Plattformen	Windows, Linux, Mac, Unix, SGI

Dateiformat	GeoVRML Dateien werden wie VRML Dateien im unverschlüsselten ASCII Format abgelegt. Sie können zusätzlich mit einem Zip-Programm komprimiert werden. Der Unterschied zu VRML Dateien, ergibt sich aus der teilweise vorgenommenen Erweiterung der normalen VRML Knoten um geospezifische Aspekte. Diese Erweiterung wird durch VRML Externprotos realisiert.
Webtauglichkeit	Hier gelten im Allgemeinen die gleichen Aussagen wie für VRML. Zusätzlich ist noch zu erwähnen, dass die Erweiterungen in GeoVRML für die Darstellung großer Landschaften insgesamt auch die Performance im Web steigern können.
Hauptsächliche Anwendungsgebiete	GeoVRML wird, wie der Name schon sagt, hauptsächlich in der Modellierung großer Landschaftsmodelle, welche auch Unterstützung für geographische Angaben enthalten müssen, verwendet.
Tauglichkeit für Landschaftsmodelle	GeoVRML eignet sich sehr gut zum Visualisieren großer Geländeausschnitte. Jedoch ist die Implementierung des Erweiterungsplugins noch ziemlich fehlerhaft und stürzt deshalb auch oft ab. Es ist jedoch teilweise möglich, Ideen und Ansätze wie sie in GeoVRML verfolgt werden auch in normalem VRML97 zu implementieren. Solche Inhalte können dann von jedem gewöhnlichen VRML Browser dargestellt werden, ohne dass ein Erweiterungsplugin erforderlich ist.

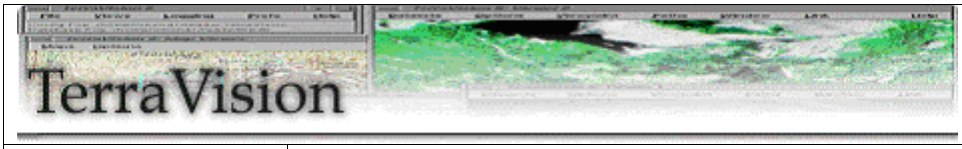
Table 3 GeoVRML

3.4 Java3D

	
Name	Java3D API
Homepage	http://java.sun.com/products/java-media/3D
Hersteller	Java3D wird von der Firma Sun, welche auch die Programmiersprache Java vertreibt, entwickelt
Preis	Java3D ist kostenlos
Plattformen	Java3D läuft momentan unter Windows, Linux, IBM AIX, HP Unix und SGI IRIX. Eine Portierung für Apple Mac ist vorgesehen.
Dateiformat	Java3D besitzt kein eigenes Dateiformat zum Laden und Speichern von Geometriedaten. Es existieren jedoch eine Vielzahl so genannter Java3D-Loader welche es ermöglichen, Geometriedaten anderer 3D Formate einzulesen. Loader gibt es unter anderem für folgende Formate: Milkshape3D, AC3D, Maya OBJ, DXF, VRML97, X3D, OpenFlight, LWO, NFF, 3DStudio 3DS, VTK
Webtauglichkeit	Java3D ist, da es eine Erweiterung zur Standard Java API darstellt, grundsätzlich webfähig. Das heißt, Java3D Programme können als Applet implementiert werden. In der Realität gibt es hierbei jedoch häufig Probleme, da Webbrowser oft nicht die aktuellste Version von Java unterstützen, welche aber meist von Java3D benötigt wird.
Hauptsächliche Anwendungsgebiete	Im kommerziellen Bereich konnte sich Java3D bis jetzt noch nicht sehr stark behaupten. Es wird hauptsächlich in der Forschung und im Open-Source Bereich eingesetzt. Vereinzelt gibt es auch Projekte, die Java3D basierte Spiele entwickeln. So wird z.B. momentan an einer Java3D basierten Rennsimulation gearbeitet, welche auf der Messe JAVA ONE 2002 vorgestellt werden soll.
Tauglichkeit für Landschaftsmodelle	Java3D scheint durchaus geeignet zu sein um 3D Landschaftsmodelle umzusetzen. Es kann eine Vielzahl verschiedener 3D Datenformate lesen und darstellen. Des Weiteren ist Java3D mit der Programmiersprache Java im Hintergrund natürlich sehr mächtig.

Table 4 Java3D

3.5 Terra Vision



Name	TerraVision
Homepage	http://www.tvgeo.com/
Hersteller	SRI International (früher: Stanford Research Institute)
Preis	Kostenloser Download
Plattformen	Momentan ist TerraVision für Windows, Linux und SGI IRIX erhältlich. Eine Version für Apple Mac ist in Planung.
Dateiformat	TerraVision unterstützt multiple Dateiformate. Unter anderem können VRML und GeoVRML Dateien eingebunden werden. Außerdem bietet TerraVision die Möglichkeit, Daten von öffentlichen Servern abzufragen und einzubinden. So ist es z.B. möglich, Terraindaten und die zugehörigen Bilddaten direkt von einem USGS-Server (United States Geological Survey) abzufragen und in TerraVision einzubinden, ohne dass die Daten vorher heruntergeladen und lokal installiert werden müssen.
Webtauglichkeit	TerraVision wurde speziell im Hinblick auf Webtauglichkeit entwickelt. Es unterstützt die Verteilung von Ressourcen über beliebig viele Computer in einem Netzwerk. Es kann sich hierbei um einen einzelnen PC, ein lokales privates Netzwerk oder aber z.B. auch um ein globales Netzwerk wie das Internet handeln. TerraVision setzt verschiedene, sehr effiziente Techniken ein, welche die Darstellung großer Mengen an Terraindaten über das Netz erst ermöglichen.
Hauptsächliche Anwendungsgebiete	TerraVision wird momentan hauptsächlich im Forschungsbereich eingesetzt. Anwendungsgebiete sind unter anderem militärische Missionsplanungen, Organisation von Hilfseinsätzen in Katastrophengebieten und die Planung von Maßnahmen im Umweltschutzbereich.
Tauglichkeit für Landschaftsmodelle	TerraVision ist sehr gut dazu geeignet, große Landschaftsmodelle zu visualisieren. Es wurde speziell daraufhin entwickelt, sehr umfangreiche Datenmengen effizient zu verarbeiten und darzustellen.

Table 5 TerraVision

3.6 G-Vista

G-Vista:	
Name	G-Vista, G-Scene, G-Render Library
Homepage	http://www.g-graphix.de/
Hersteller	G-Graphix, Freiburg
Preis	G-Vista Viewer Plugin kostenlos erhältlich. Für G-Scene, das zum Erstellen von 3D Inhalten im G-Vista Format benötigt wird, konnten leider keine Preisinformationen ermittelt werden.
Plattformen	Windows Plattform
Dateiformat	Eigenes proprietäres Dateiformat. Import von verschiedenen gängigen Terrainbeschreibungsformaten und Bildmaterialien ist möglich.
Webtauglichkeit	Das G-Vista Dateiformat ist speziell auf sehr große Datenmengen ausgelegt und integriert eine Reihe effektiver Methoden, um 3D Landschaftsvisualisierungen über das Internet zu bieten. Die Bedienung des Viewers ist sehr intuitiv und äußert leicht erlernbar. Die Darstellung der Landschaft erreicht eine bisher nicht da gewesene Qualität. So werden z.B. auf der Demohomepage des Anbieters Bildauflösungen von bis zu einem halben Meter erreicht und das bei ziemlich flüssigem Streaming über eine ADLS Leitung. Obiges bedeutet, dass Objekte mit einer Größe von 50 cm in der Landschaft unterschieden werden können.
Hauptsächliche Anwendungsgebiete	G-Vista ist ein ziemlich neues Produkt und somit noch nicht sehr weit verbreitet. Zum Kundenkreis von G-Graphix zählen unter anderem die Schweizer Firma Geonova, welche die komplette Schweiz in einer Auflösung von 25 m, und unter anderem die Kantone Zürich, Ob- und Niedwalden in einer Auflösung von 50 cm visualisiert hat. Weitere Anwendungsgebiete wären z.B. 3D-Atlanten, 3D-Routenplanung, Wetterinformationen(3D Wetterflug)
Tauglichkeit für Landschaftsmodelle	G-Vista bietet sehr hohe Detailgenauigkeit und Realitätsnähe für Landschaftsmodelle und ist somit sehr gut geeignet,l virtuelle Landschaften darzustellen.

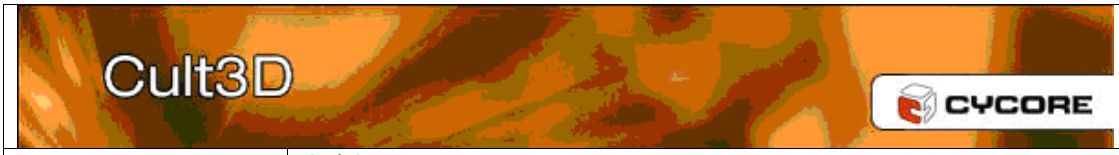
Table 6 G-Vista

3.7 VET

	
Name	VET
Homepage	http://www.viewpoint.com/
Hersteller	Viewpoint
Preis	Der Viewpoint Media Player, ein Plugin, welches zur Darstellung von Viewpoint Dateien benötigt wird, kann gratis aus dem Internet herunter geladen werden. Außerdem bietet die Firma Viewpoint ein Set kostenloser Softwarekomponenten zur Erstellung von 3D Inhalten, welche mit dem Viewpoint Media Player dargestellt werden können. Des Weiteren existieren auch eine Reihe von Programmen von Drittherstellern, welche 3D Daten im VET Viewpoint Format speichern können. Die Firma Viewpoint bietet zwar alle benötigten Softwarekomponenten zur Erstellung und Darstellung von VET Dateien kostenlos an, jedoch muss eine Lizenz erworben werden, um diese Dateien online zu veröffentlichen. Diese Lizenz ist für private Anwender kostenlos und gilt für sechs Monate, kann aber laut Aussage von Viewpoint problemlos verlängert werden.
Plattformen	VET wird momentan für MS Windows Plattformen und für Apple Macintosh angeboten.
Dateiformat	Die Firma Viewpoint hat sein eigenes proprietäres Dateiformat entwickelt. Herz dieses Formats ist die binäre .mts Datei. Diese Datei enthält alle Geometrie-, Material- und Texturdaten in stark komprimierter Form.
Webtauglichkeit	VET von der Firma Viewpoint besticht durch einige herausragende Features. So baut sich, wenn man Viewpoint Inhalte online betrachtet, auch mit einer langsamen Modemverbindung schon nach kürzester Zeit eine Darstellung auf, welche sich dann sukzessiv detailliert. VET Dateien sind mit einem speziellen 3D Kompressionsverfahren gepackt und somit insgesamt äußerst kompakt.
Hauptsächliche Anwendungsgebiete	Viewpoint VET wird momentan hauptsächlich in der Produktvisualisierung eingesetzt. So werben zum Beispiel Autohersteller und Computerhersteller mit digitalen Modellen im Viewpoint Format. Aber auch Gebäudemakler und z.B. Sony mit ihrem Roboterhund AIBO gehören zum Kundenkreis von Viewpoint.
Tauglichkeit für Landschaftsmodelle	Leider eignet sich Viewpoint nicht besonders gut dazu, große Landschaftsmodelle darzustellen. Auch bietet der Viewpoint Media Player keine „Walk Through“ Funktionen, sondern eher solche Funktionen wie drehen und zoom, wie man es eben für Produktvisualisierungen braucht. In Viewpoint VET sind keinerlei Funktionen implementiert, um virtuelle Landschaften effizient darzustellen.

Table 7 ViewPoint VET

3.8 Cult3D



Name	Cult3D
Homepage	http://www.cult3d.com/
Hersteller	Cycore, Sweden
Preis	Wie die Firma Viewpoint bitet auch die Firma Cycore Tools zur Erstellung von 3D Inhalten, Werkzeuge zur Veröffentlichung im Netz und ein Plugin für den Webbrowser, um Cult3D Dateien darstellen zu können, kostenlos zum download an. Des Weiteren bietet Cycore ein kostenloses Kovertierungstool an, das Formate von Programmen wie 3D Studio Max und Maya in das Cult3D Format umwandeln kann. Um Cult3D Anwendungen online zu stellen bedarf es einer Lizenz. Die Lizenz ist für private Anwender und für nicht kommerzielle Anwendungen kostenlos.
Plattformen	Der Cult3D Viewer läuft unter Windows, Mac, Linux und Solaris. Der Cult3D Designer zum Kreieren von 3D Inhalten ist momentan nur für Windows Plattformen verfügbar.
Dateiformat	Das Cult3D Datei Format ist ein binäres Fileformat welches Modelle, Animation, Sound, Interaktionselemente und eingebetteten Javacode beinhaltet. Dieses Fileformat wird vom Cult3D Designer und dem Konvertierungstool verwendet und ist schreib- und lesefähig. Für Daten, welche ins Netz gestellt werden sollen, gibt es ein nur Lese Fileformat, das außerdem auch noch mit speziellen 3D Algorithmen komprimiert wird.
Webtauglichkeit	Wie auch VET von Viewpoint besticht Cult3D mit streamingfähigen 3D Inhalten. Das heißt, der Cult3D Viewer baut schon nach kürzester Zeit eine Darstellung auf, auch wenn noch nicht das ganze File vom Netz heruntergeladen ist. Auch spezielle Beleuchtungseffekte und Features wie Partikelsysteme (z.B. zur Simulation eines Springbrunnens) lassen beim Benutzer fast keine Wünsche offen.
Hauptsächliche Anwendungsgebiete	Cult3D's Hauptanwendungsgebiet liegt in der Produktvisualisierung. Kunden reichen von Autoherstellern über Handyfirmen, bis hin zur NASA, die mit Cult3D ihr Spaceshuttle präsentiert und zu einem virtuellen Probeflug einlädt.
Tauglichkeit für Landschaftsmodelle	Cult3D bietet in ihrem Viewer zwar Möglichkeiten, sich auch innerhalb einer 3D Welt zu bewegen, jedoch ist die Größe dieser Welt stark eingegrenzt und der Viewer stößt schnell an seine Performancegrenze. Dies liegt vermutlich hauptsächlich daran, dass die Runtime Engine nicht speziell für solche Anwendungsgebiete optimiert worden ist, da der Anwendungsbereich nicht im Gebiet der Landschaftsvisualisierung gesehen wird.

Table 8 Cult3D

3.9 Fazit:

Nach eingängiger Untersuchung der in diesem Kapitel vorgestellten Produkte bzw. Standards muss nun eine Auswahl getroffen werden, welche Technik für die Implementierung des Virtual Reality Modells verwendet wird. In die Entscheidung fließen unter anderem der Preis der verwendeten Technik, die Leistungsfähigkeit für Landschaftsmodelle und die gebotene Flexibilität ein.

Die Produkte Viewpoint VET und Cult3D sind der Untersuchung nach nicht für die Implementierung von großen virtuellen Landschaften geeignet und scheitern somit an diesem Kriterium.

Die Produktsuite von G-Graphix wäre voraussichtlich optimal geeignet dieses Projekt zu implementieren, jedoch ist die Software nicht frei verfügbar, und auf Anfrage konnte auch keine Preisinformation ermittelt werden. Die Firma G-Graphix wurde zwei Mal per Mail kontaktiert und hat auf diese Anfragen leider nicht reagiert.

GeoVrml und TerraVision sind zwei Kandidaten, die auf die Visualisierung von Terrain spezialisiert sind. Jedoch bringen sie auch einige Probleme mit sich. So braucht man z.B. für die Darstellung von GeoVRML Inhalten ein spezielles Plugin, welches ein gewöhnliches VRML Plugin erweitert und befähigt, GeoVRML zu interpretieren. Zum einen läuft dieses Plugin nicht sehr stabil, das heißt, es kommt des Öfteren zu Systemabstürzen, andererseits ist es eben eine Komponente mehr, welche sich der Endbenutzer herunterladen und installieren muss. Des Weiteren würden die meisten Features, welche GeoVRML bietet, in diesem Projekt nicht benutzt werden. TerraVision scheidet als Kandidat aus, da es ziemlich unstabil läuft. Es bietet leider auch nicht die benötigte Flexibilität wie z.B. Filmunterstützung, welche für die Anwendung des Hohentwiel sehr wichtig ist.

Java3D in Verbindung mit VRML wäre wahrscheinlich gut geeignet für Terrainvisualisierungen. Jedoch wäre hier der Zeitaufwand, um ein lauffähiges System zu implementieren, zu hoch. Es müsste so z.B. in Java3D eine komplette View- und Steuerkomponente erstellt werden, welche in etwa einem VRML-Browser Plugin entspricht. Des Weiteren hat Java3D momentan auch noch Probleme damit, als Applet in einem Webbrowser zu laufen, da es oft zu Versionskonflikten zwischen der unterstützen Java RuntimeVersion und der Java3D Version kommt. Auch ist es dem Endbenutzer kaum zuzumuten, alle benötigten Komponenten herunterzuladen und so zu konfigurieren, dass sie problemlos zusammenarbeiten.

X3D ist durch seine noch nicht ganz vollständige Implementierung ungeeignet, da einige Konzepte, die für die Realisierung von virtuellem Terrain, wie z.B. der

Elevation Grid Knoten, noch nicht bzw. unvollständig implementiert sind. Auch wird es noch nicht von den gängigen VRML Browser Plugins unterstützt.

Die Entscheidung fällt somit auf VRML97, da diese Beschreibungssprache alle benötigten Features für die Landschaftsvisualisierung zur Verfügung stellt und auch die benötigte Flexibilität mit sich bringt. Die Implementierung des Vorhabens in VRML97 bringt unter anderem folgenden Vorteile mit sich:

- Das System läuft auf unterschiedlichen Hardwareplattformen (es muss nur ein VRML97 kompatibles Browser Plugin installiert sein.

- Die Benutzung von VRML ist kostenlos, und auch viele Browser Plugins sind kostenlos erhältlich.

- Der Download und die Installation von VRML Plugins gehen in der Regel schnell und einfach vonstatten.

- VRML ist durch seine flexible Struktur in der Lage, die großen Datenmengen für dieses Projekt flüssig darzustellen.

- VRML lässt sich optimal in HTML Seiten integrieren und sogar durch HTML Seiten steuern.

- VRML ist ein etablierter Standard, der von vielen Produkten unterstützt wird.

4. Untersuchung und Bewertung von VRML Browser Plugins

Wie schon in den vorigen Kapiteln erwähnt wurde gibt es viele verschiedene Browser Plugins, welche es ermöglichen, VRML Inhalte darzustellen. In diesem Kapitel sollen die gebräuchlichsten Plugins näher untersucht und nach verschiedenen Kriterien bewertet werden.

4.1 Cosmo Player

Cosmo Player	
Name	Cosmo Player 2.1
Hersteller	Cosmo Software (Computer Associated International)
Homepage	http://ca.com/cosmo/
Preis	kostenlos
Plattformen	Windows, Macintosh, SGI IRIX
Browser	Netscape Navigator 3.x und höher Internet Explorer 4.x und höher
Größe	3,2 MB
Installation	Manueller Download und Installation. Neuere Browser Versionen von Netscape und Microsoft werden nicht automatisch erkannt. Man kann das Plugin jedoch umständlich über die Funktion „Nicht unterstütze Browser“ installieren. Dazu muss für den Netscape Navigator manuell das Plugin Verzeichnis angegeben werden. Insgesamt kein zufrieden stellender Installationsablauf.
Handhabung	Auf den ersten Blick fällt die Navigationsleiste am unteren Rand des Bildschirms positiv auf. Hier können die gerade gewünschten Navigationsmodi einfach ausgewählt werden. Jedoch ist die Steuerung insgesamt sehr gewöhnungsbedürftig. Auch gibt es keine Möglichkeit, die Steuerleiste auszublenden, um z.B. den ganzen Bildschirm für die 3D Darstellung zu verwenden.
Leistungsfähigkeit	Cosmo Player unterstützt in der Version 2.1 die komplette VRML97 Spezifikation. Es werden die Sprachen deutsch, englisch und französisch unterstützt. Cosmo Player benutzt vorhandene 3D Beschleunigungskarten, entweder durch OpenGL oder Direct3D.

Table 9 CosmoPlayer

4.2 Blaxxun Contact

	
Name	Blaxxun Contact 5.1
Hersteller	Blaxxun Interactive AG, München
Homepage	http://www.blaxxun.de
Preis	kostenlos
Plattformen	Windows
Browser	Netscape Navigator ab Version 3.0 Internet Explorer ab Version 4.0 Opera Webbrowser ab Version 5.0 (nicht offiziell unterstützt)
Größe	1,5 MB (Automatische Installation) 5,6 MB (Manuelle Installation)
Installation	Automatischer Download und Installation, kein Reboot, bzw. Neustart des Browsers nötig. Automatische Installation wirkt jedoch nur für den Browser, aus dem die Installation gestartet wurde und nicht für die anderem im System installierten Browser. Bei manueller Installation, wird für alle im System vorhandenen und unterstützen Browser ein Plugin installiert.
Handhabung	Anfangs gewöhnungsbedürftig, da kein dauerhaft sichtbares Funktionsmenü vorhanden ist. Ein temporäres Menü kann jedoch über die rechte Maustaste, bzw. Doppelklick mit der linken Maustaste geöffnet werden. Von Vorteil ist, dass der ganze Bildschirm für die 3D-Darstellung verwendet wird. Die Navigation ist intuitiv und sehr schnell erlernbar. Durch Tasten wie ALT und CTRL und Shift kann temporär in einen anderen Bewegungsmodus geschaltet werden.
Leistungsfähigkeit	Blaxxun Contact unterstützt die komplette VRML97 Spezifikation und darüber hinaus auch eine Reihe weiterer interessanter Features wie 3D Sound, Multitextures, Environment Mapping (Spiegeln der Umgebung auf einer Oberfläche) und dynamisches Nachladen von Dateien. Außerdem gibt es den Blaxxun Contact in insgesamt 16 verschiedenen Sprachen. Das Plugin unterstützt die Libraries OpenGL oder Direkt3D, zwischen welchen ohne Neustart der Anwendung gewechselt werden kann. Außerdem sind viele Feineinstellungen zur Graphik- und Soundperformance per Konfigurationsmenü möglich. Blaxxun Contact ist der einzige VRML Browser im Test, welcher verzögertes Nachladen von Dateien bietet, welches als eine Art Streaming benutzt werden kann.

Table 10 Blaxxun Contact

4.3 Cortona

Name	Cortona 3.1
Hersteller	Parallel Graphics
Homepage	http://www.parallelgraphics.com/products/cortona
Preis	kostenlos
Plattformen	Windows, Macintosh, Windows CE 3.x
Browser	Netscape Navigator ab Version 4.0 Internet Explorer ab Version 4.0
Größe	0,96 MB (automatische Installation) 1,31 MB (manuelle Installation)
Installation	Auswahl zwischen automatischer und manueller Installation. Automatische Installation installiert Cortona mit den gebräuchlichsten Features für den Browser, aus dem die Installation gestartet wurde. Kein Neustart des Browsers nötig. Manuelle Installation installiert Cortona komplett für Internet Explorer und Navigator. Die Installations-Datei muss manuell heruntergeladen und gestartet werden. Manuelle Installation enthält eine Deinstallationsroutine und erweiterte Funktionen wie z.B. Konfiguration der Render Engine.
Handhabung	Der Cortona VRML Client hat die üblichen Browser Modi wie z.B. „gehen“, „fliegen“, „untersuchen“ und „verschieben“. Was sehr gut gefällt ist, dass diese Funktionen entweder über die rechte Maustaste oder aber über eine abschaltbare Funktionsleiste am unteren Rand des Bildschirms aktiviert werden können. Mit Zusatztasten wie ALT oder Space kann vorübergehend in einen anderen Bewegungsmodus geschalten werden. Insgesamt ist die Benutzeroberfläche übersichtlich gestaltet und einfach zu erlernen und zu bedienen.
Leistungsfähigkeit	In der Vollversion (nach manueller Installation) bietet Cortona eine Vielzahl von Einstellungsmöglichkeiten zur Performance Optimierung. Besitzer eines PIII Prozessors können einstellen, dass dessen erweiterter Chipsatz benutzt wird. Außerdem bietet Cortona als einziger Browser einen Vollbildmodus. Bei installiertem RealPlayer bietet Cortona die Möglichkeit, Movietextures im Real Format abzuspielen.

Table 11 Cortona

4.4 Horizon Browser

	
Name	Horizon Browser 2.5 Beta
Hersteller	OpenWorlds
Homepage	http://www.openworlds.com
Preis	kostenlos
Plattformen	Windows, SGI Irix
Browser	Internet Explorer oder Standalone Version (Kein Webbrowser benötigt)
Größe	2,74 MB (Standalone Version)
Installation	Automatische Installation für Internet Explorer; Standalone Version muss manuell heruntergeladen und installiert werden.
Handhabung	Bedienungsmenü über rechte Maustaste. Keine Funktionsleiste vorhanden. Im Test funktionierte die Auswahl verschiedener Bewegungsmodi nicht, das Programm läuft insgesamt sehr instabil; häufig Abstürze des gesamten Programms bzw. beim Internet Explorer Plugin des kompletten Browsers. Die Einstellungsmöglichkeiten sind für 3D Laien sehr verwirrend, jedoch können Profis den Browser genau auf ihre Bedürfnisse abstimmen.
Leistungsfähigkeit	Laut Aussagen des Herstellers wird die komplette VRML97 Spezifikation und teilweise auch schon die des VRML Nachfolgers X3D unterstützt. Spezielle Features von Horizon sind Schatten und Spiegeleffekte, Bump Mapping (realistischer wirkende Texturen) und Feuer- und Raucheffekte (Partikelsysteme). Der Browser wirkt sehr viel versprechend, jedoch ist er derzeit aufgrund der häufigen Abstürze und komplizierten Bedienung nicht zu empfehlen.

Table 12 Horizon Browser

4.5 Pivoron Player

Name	Pivoron Player 1.0
Hersteller	Nexternet Inc
Homepage	http://www.nexternet.com/download/index.html
Preis	Kostenlos
Plattformen	Windows
Browser	Netscape Navigator ab Version 4.0 Internet Explorer ab Version 4.0
Größe	1,4 MB
Installation	Manueller Download
Handhabung	Pivoron Player kommt mit einer etwas verwirrenden Funktionsleiste zur Auswahl des Navigationsmodus daher, welche nicht ausgeblendet werden kann. Die Navigation in einer Scene ist vergleichsweise schwierig und anfangs gewöhnungsbedürftig.
Leistungsfähigkeit	Der Pivoron Player basiert auf Cosmo Player 2.0. Die Firma Nexternet hat die Rechte an Cosmo Player erworben und diesen teilweise erweitert. Piveron Player unterstützt die komplette VRML97 Spezifikation und wurde von Nexternet speziell hinsichtlich des EAI (External Authoring Interface) erweitert. EIA wird für die externe Steuerung des Plugins z.B. über eine Java Klasse verwendet. Pivoron läuft in Microsofts Internet Explorer 6.0, hat aber mit einem Testfile, welches MovieTextures enthält, Probleme und kann die Filme nicht abspielen. Unter Netscape 6.2 produziert das Plugin beim Start eine Endlosschleife, welche, wenn nicht sofort gestoppt, endlos viele leere Netscape Fenster öffnet.

Table 13 Pivoron Player

4.6 Fazit

Die in diesem Kapitel vorgestellten Browser Plugins stellen natürlich nur eine Auswahl der vorhandenen Plugins dar. Es wurden nur die momentan am häufigsten benutzten bzw. die gebräuchlichsten VRML Plugins in den Test mit aufgenommen. Im Rahmen dieses Plugin Tests, wurden noch eine Reihe älterer Anwendungen, wie z.B. Microsoft's VRML 2.0 Viewer, Viscape's VRML Viewer, WorldProbe von UpperCut Software und WorldView von Computer Associated getestet. Jedoch arbeiten diese nicht oder nur unzureichend mit modernen Browsern wie z.B. Internet Explorer ab Version 5.0 und Netscape ab Version 6.0 zusammen, sie wurden deshalb auch nicht in den Test einbezogen. Dies liegt wahrscheinlich daran, dass diese Plugins

nicht für die Typen von modernen Browsern implementiert wurden, und da diese Plugins nicht mehr weiterentwickelt werden, werden moderne Browser auch in Zukunft nicht unterstützt.

Prinzipiell ist zu sagen, dass die Referenzanwendung für das 3D Modell des Hohentwiel in allen Browsern welche in die Untersuchung mit aufgenommen wurden, mehr oder weniger fehlerfrei läuft.
Jedoch wird die für das Modell wichtige MovieTexture von den Plugins Cosmo Player, Pivoron Player und Horizon Browser nicht oder nur unvollständig unterstützt. Deshalb, und auch weil die Navigation in diesen Browsern sehr gewöhnungsbedürftig ist, werden sie im Rahmen dieser Untersuchung als unzureichend eingestuft.

Die Browser Plugins Blaxxun Contact und Cortona stellen beide die Referenzanwendung fehlerfrei dar. Auch in Bezug auf die Leistung und Handhabung sind sie in etwa beide gleichauf. Jedoch hat Blaxxun Contact einen entscheidenden Vorteil. Das Plugin beherrscht das inkrementelle Nachladen von VRML-Dateien, was wiederum sehr wichtig für die Performance des Modells ist. An dieser Stelle soll ein wenig vorgegriffen werden auf den Abschnitt „Verbesserung der Performance“ (5.3.). Das Modell wurde in mehrere hundert kleine Dateien aufgeteilt, um erstens die Menge an Dreiecken, welche die 3D Engine auf einmal berechnen muss, zu reduzieren und zweitens, um die Zeit, welche zum Download des Modells gebraucht wird, zu minimieren. Blaxxun ist das einzige Plugin welches nicht beim Start komplett alle Dateien der Anwendung in den Speicher lädt. Es lädt nur die Dateien, die momentan im Betrachtungswinkel des Benutzers sind. Weitere Dateien werden hintereinander nachgeladen, wenn der Benutzer sich in der Welt bewegt und die entsprechenden Landschaftsstücke in seinen Blickwinkel geraten. Durch diese Funktion kann die Zeit für den Start der Anwendung stark minimiert werden.

Ausserdem ist Blaxxun Contact auch das einzige Plugin, welches auf Anhieb alle im System installierten Web Browser erkennt und die entsprechenden VRML Plugins korrekt installiert. Ein weiteres Plus für Blaxxun ist, dass ein HTML Tag in die Web Seite, von welcher aus das 3D Modell gestartet wird, eingebaut werden kann, welches es ermöglicht, das Plugin halbautomatisch zu installieren. Wenn z.B. ein Benutzer

noch kein VRML Plugin installiert hat und die Web Seite mit dem 3D Modell besucht, wird er automatisch dazu aufgefordert, das Blaxxun Plugin zu installieren. Wenn der User dies bestätigt, wird das Plugin automatisch heruntergeladen und installiert, darüber hinaus hat der User danach die Möglichkeit, die VRML Seite zu betrachten ohne seinen Webbrowser neu zu starten müssen.

Ein weiterer wichtiger Aspekt ist, dass Blaxxun3D ständig weiterentwickelt wird und so auch immer die neueste Version von Webbrowsern unterstützt wird.

Aufgrund der oben genannten Argumente wird für die Hohentwiel 3D Anwendung das VRML Plugin Blaxxun Contact verwendet.

5. Implementierung des 3D Modells des Hohentwiel

5.1 Einführung

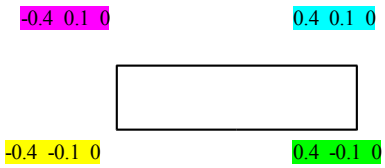
In diesem Kapitel soll der vollständige Arbeitsablauf der Erstellung des virtuellen 3D Modells des Hohentwiel beschrieben werden. Dies umfasst die Beschaffung der Rohdaten, deren Umwandlung in eine passende Struktur, die benutzten Methoden und Techniken und die benötigten Tools und Werkzeuge.

5.2 Rohdaten

Digitale Landschaftsdaten sind in Deutschland normalerweise über die Landesvermessungsämter erhältlich. Die Rohdaten werden meist als DHM (Digitales Höhenmodell) ausgeliefert. Oft sind auch entsprechende Luftaufnahmen des Landschaftsausschnitts erhältlich. Digitale Höhendaten sind in verschiedenen Auflösungen verfügbar. Als Auflösung wird die Rasterweite der Daten bezeichnet. So kann ein digitales Höhenmodell z.B. alle 20 m gesampelt sein, was bedeutet, dass in einem XY Raster von z.B. 1000x1000 m auf beiden Achsen alle 20 m eine Höhenangabe vorliegt. Je höher die Auflösung der Daten ist desto detailgetreuer ist das spätere 3D Modell, soll heißen desto mehr Einzelheiten können in dem Modell erkannt werden. Jedoch bedeutet eine höhere Auflösung auch steigenden Speicherbedarf und natürlich ferner eine erhöhte Rechenzeit, da der Computer bzw. die Graphikkarte mehr Informationen berechnen muss.

Die erste Version der Rohdaten wurde vom Institut für Landschaftsökologie und Naturschutz in Singen auf CD-ROM geliefert. Es handelte sich um ein DHM im 3D Studio Max Format. Diese Datei wurde mit dem Programm 3D Studio Max geöffnet und als VRML97 Datei exportiert. Jedoch wies die entstandene VRML Datei eine Größe von mehreren Megabyte auf und war daher nicht gut geeignet für Onlineanwendungen, bei welchen es bei langsamen Verbindungen auf jedes Kilobyte ankommt. Die enorme Größe der Datei resultierte aus der Art und Weise, wie der Exporter von 3D Studio Max VRML-Dateien schreibt. 3D Studio Max exportiert VRML als ***Indexed Face Set***. Dies ist ein Konstrukt in VRML in welchem 3D Daten zu einem Objekt gespeichert sind. Anhand des folgenden Code-Beispiels soll dieser Sachverhalt genauer veranschaulicht werden:

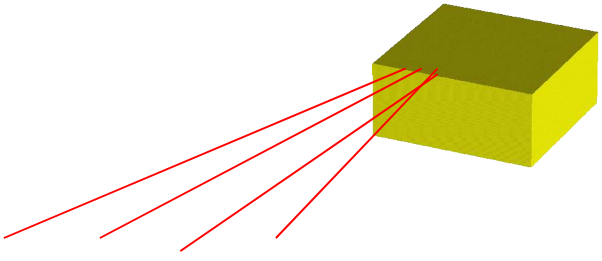
```
geometry IndexedFaceSet {
  coord Coordinate {
  point [ -0.4 -0.1 0,
   0.4 -0.1 0,
   0.4 0.1 0,
   -0.4 0.1 0 ]
  }
  coordIndex [0 1 2 3 ]
}
```



Dieser Code zeichnet ein einfaches Rechteck, 80 cm lang, 20 cm breit und ohne Tiefe. Nebenstehende Zeichnung zeigt dieses Rechteck und die Koordinaten der vier Ecken. Das Feld ***coordIndex*** enthält Informationen darüber, wie die vier gezeichneten Punkte verbunden werden. Die Werte in ***coordIndex*** stehen für den Punkt an der entsprechenden Stelle im Feld ***point.*** So werden in diesem Beispiel die Punkte, angefangen mit dem Index 0 bis Index 3, einfach der Reihe nach verbunden, und so entsteht ein Rechteck.

Wie man in obigem Beispiel sehen kann, speichert das VRML Konstrukt ***Indexed Face Set*** also immer zu jedem Punkt eines dreidimensionalen Objekts die XYZ-Koordinaten ab; zusätzlich wird noch Speicherplatz für den Koordinatenindex benötigt. Dies ist für die meisten geometrischen Objekte auch unbedingt nötig und unerlässlich. Digitale Landschaftsdaten stellen hier jedoch eine Ausnahme dar. Wie weiter oben im Text schon einmal erwähnt, werden Landschaftsdaten meist als DHM ausgeliefert. Diese Daten kommen heutzutage oft von Satelliten, welche das Gelände nach einem bestimmten Muster abtasten.

Folgendes Beispiel zeigt, wie eine solche Vermessung vonstatten geht:



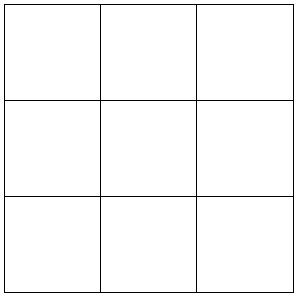


Figure 1 Vermessung

Der Satellit überfliegt das Land und sendet z.B. alle 20 m einen Radarstrahl zur Erde. Der Strahl wird von der Erdoberfläche reflektiert und geht zurück zum Satelliten. Anhand der Zeit, die der Strahl für den Weg zur Erde und zurück gebraucht hat, kann die Höhe des Punktes auf der Erde bestimmt werden, den der Radarstrahl angepeilt hat. Diese Höheninformation wird zusammen mit dem genauen Längen- und Breitengrad abgespeichert. Auf diese Art und Weise entsteht dann z.B. eine derartige Matrix, wie wir sie in obiger Abbildung sehen können. An jedem Punkt, an dem zwei oder drei Linien zusammentreffen, liegen dann Werte für den genauen Standort auf der Erde (XY) und Höhenangaben für diesen Punkt (Z) vor.

Die Tatsache, dass die Daten in immer gleich bleibenden Abständen vorliegen, können wir uns bei der Umwandlung in eine VRML Struktur zunutze machen. Wenn wir die Daten in VRML als ein ***Indexed Face Set*** abspeichern, müssen wir zu jedem Punkt alle drei Koordinaten, also XYZ angeben. Es gibt jedoch in VRML ein Konstrukt Namens ***Elevation Grid***, welches speziell für die Implementierung von Landschaftsdaten vorgesehen ist. Im Unterschied zum ***Indexed Face Set*** speichert ***Elevation Grid*** nur die Höhenangaben (Z) ab, nicht aber die Werte für X und Y. Somit ist die entstehende VRML Datei viel kleiner als eine Datei mit ***Indexed Face Set***. Nachstehendes Code-Beispiel zeigt ein ***Elevation Grid*** Konstrukt in VRML, welches unserer Matrix vom vorhergehenden Beispiel entspricht:

```
geometry ElevationGrid{
```

```
xDimension 4
                                zDimension 4
                                xSpacing 20
                                zSpacing 20
          height [1 2 0 1,  2 2 0 1,  2 0 0 1,  1 0 2 0]

}
```

Die Felder ***xDimension*** und ***zDimension*** geben an, wie viele Höhenwerte in X-Richtung und in Y-Richtung vorliegen. Die Felder ***xSpacing*** und ***zSpacing*** geben den Abstand der Höhenangaben in x und y Richtung an. (In unserem Beispiel 20m, da der Satellit alle 20 m eine Höhenangabe berechnet hat.) Das Feld ***height*** gibt dann die tatsächlichen Höhenwerte an. Diese sind von links nach rechts und von oben nach unten angeordnet. Folgendes Beispiel demonstriert diesen Aspekt:

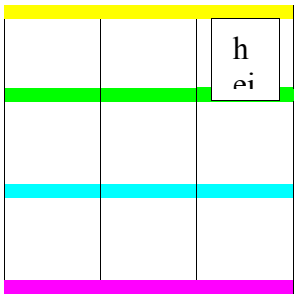


Figure 2 Matrix

Sehr deutlich zu sehen ist in diesem Beispiel, dass die vier Werte vor dem ersten Komma die Höhenwerte für die erste Reihe in der Matrix, geordnet von links nach rechts, angeben und dass das letzte Quadrupel von Werten für die letzte Reihe in der Matrix steht.

Der ***Elevation Grid*** Knoten in VRML ist eine elegante Möglichkeit, Landschaftsdaten Speicherplatz sparend darzustellen, da nur die Höhenangaben eines digitalen Höhenmodells in die Struktur mit einfließen.

Folgende Abbildung zeigt unser Beispiel, in einem VRML-Browser dargestellt:

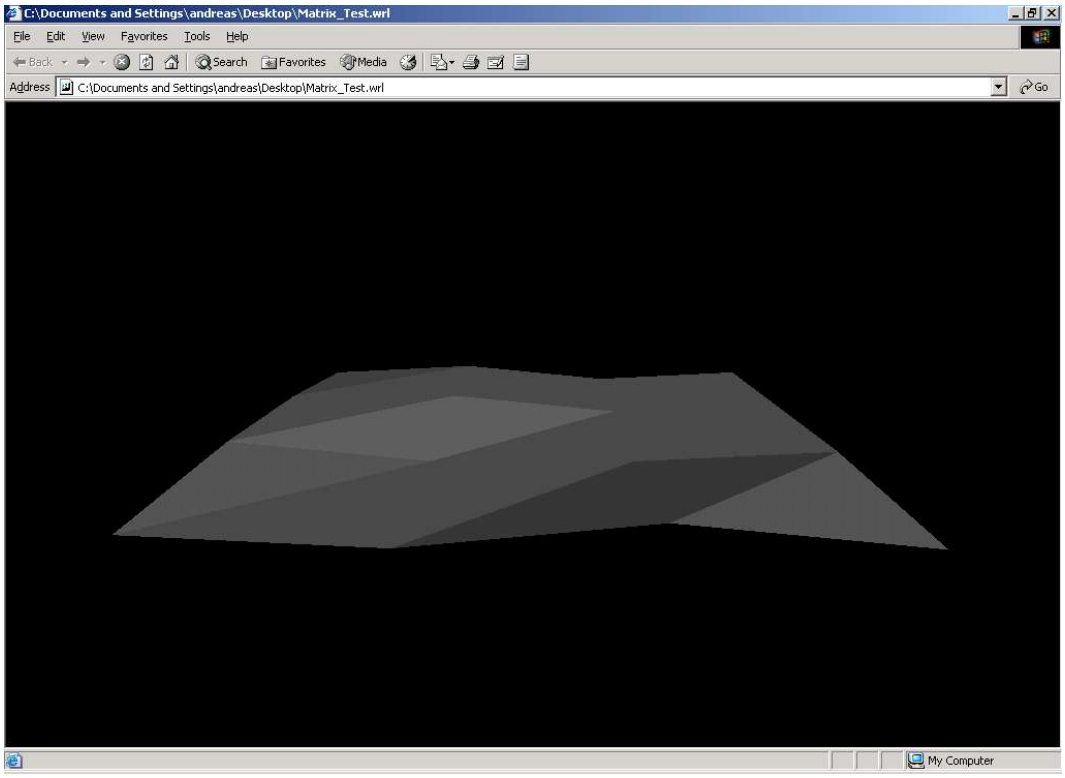


Figure 3 EG in Browser

Nach diesen weiterführenden Erklärungen zur Darstellung von Landschaftsdaten in VRML wende wir uns wieder den Rohdaten, welche in 3D Studio Max Format vorliegen, zu.

Wie weiter oben im Text schon erwähnt kann der 3D Studio Max VRML Exporter nur Daten als ***Indexed Face Set*** exportieren. Das ist jedoch, wie die oben aufgeführten Beispiele belegen, für Landschaftsdaten nur bedingt zweckdienlich. Bedingt zweckdienlich soll in diesem Zusammenhang heißen, es ist geeignet, wenn es bei der Anwendung nicht auf die Dateigröße ankommt. Prinzipiell gesehen macht es keinen Unterschied, ob die Daten in VRML als ***Elevation Grid*** oder als ***Indexed Face Set*** abgespeichert werden; das Resultat, welches man im VRML Browser sieht, ist vollkommen identisch. Der kritische Faktor ist nur die Dateigröße. Da es bei diesem Projekt aber sehr wohl auf die Größe der VRML-Datei ankam, war die 3D Studio Max Datei als Datenausgangsbasis nicht zu gebrauchen.

Es wurde daher das Institut für Landschaftsökologie und Naturschutz in Singen kontaktiert und nachgefragt, ob die Möglichkeit besteht, die Rohdaten in einem

anderen Format zu liefern, bei dem es möglich ist, nur die Höhenangaben zu extrahieren, welche ja für das VRML ***Elevation Grid*** Konstrukt benötigt werden.

Das ILN in Singen verwies direkt an das Landesamt für Flurneuordnung und Landentwicklung Baden-Württemberg, welches die Rohdaten für den Hohentwiel geliefert hatte. Dieses stellte daraufhin die Daten im XYZ-Format zur Verfügung. Die Daten liegen bei diesem Format als Textdaten in einer ASCII Datei vor. Folgendes Beispiel zeigt einen Ausschnitt der erhaltenen Daten:

```
3484767.000   5291140.000   475.400
3484767.000   5291150.000   476.000
3484767.000   5291160.000   476.700
3484767.000   5291170.000   476.700
3484767.000   5291180.000   476.200
3484767.000   5291190.000   475.400
          .
          .
          .
3484777.000   5291140.000   475.800
          .
          .
          .
3484787.000   5291140.000   475.900
```

Die erste Spalte der Daten stellen die X-Werte dar, die zweite Spalte die Y-Werte und die dritte Spalte die Höhenangaben(Z). Wie zweifellos zu erkennen ist, sind die Daten in Y-Richtung mit einer Genauigkeit von 10 m gesampelt. Zu diesem Ergebnis kommt man, wenn man von einer Zahl in der Y-Spalte die darüber stehende Zahl abzieht, so z.B. 5291150.000 - 5291140.000 = 10. Auch in der X-Richtung liegt alle 10 m ein Wert vor. 3484777.000 - 3484767.000 = 10. Daraus resultiert, dass die Daten in einem Matrixformat mit einer Schrittweite von 10 m in X und Y Richtung vorliegen. Nun muss nur noch bestimmt werden, wie viele Werte jeweils in X und Y Richtung vorliegen. Für die X-Richtung ist dies relativ einfach, es muss nur die Anzahl der der Werte gezählt werden, bis sich eine Zahl ändert, so also die Anzahl der Werte von 3484767.000 bis 3484777.000. Dies sind in unserem Beispiel 166. Wenn man jetzt die gesamte Anzahl der Zeilen in der Datei durch 166 teilt, kommt man auf die Anzahl der Werte in Y-Richtung, welche 242 ist. Die Daten bilden also eine Matrix von

166x242, welche alle 10 m gesampelt ist. Aus diesen Angaben kann jetzt auch einfach die Größe der Landschaft berechnet werden. Hierzu multipliziert man nur die X und Y Werte mit der Schrittweite von 10 m und erhält so 1.660x 2.420 m. Dies ist also die Größe des zu visualisierenden Terrains und entspricht genau 4.017.200m², also ca. 4,02 km².

Es sind nun alle Angaben vorhanden, welche benötigt werden um die Daten in einem VRML ***Elevation Grid*** Knoten zu speichern.

5.2.1 Konvertierung der Rohdaten

Die Rohdaten liegen, wie wir gesehen haben, in einer Textdatei vor, und zwar in nach XYZ-Werten geordneten Spalten. Da die Daten jedoch in ein VRML ***Elevation Grid*** Konstrukt exportiert werden sollen, ist als Datenbasis nur die Spalte mit den Höhenangaben interessant. Diese Höhenangaben müssen im Textfile mit den Rohdaten kopiert werden, und in eine VRML Detei eingefügt werden. Für diese Arbeit wurde eine Trial-Version des professionellen Texteditors UltraEdit 32 [ULTRAEDIT] verwendet. Dieser unterstützt unter anderem das Markieren nach Spalten, was zum Beispiel mit Notepad unter Windows nicht möglich ist. Die nächste Abbildung zeigt UltraEdit mit einem Ausschnitt der markierten Höhendaten.

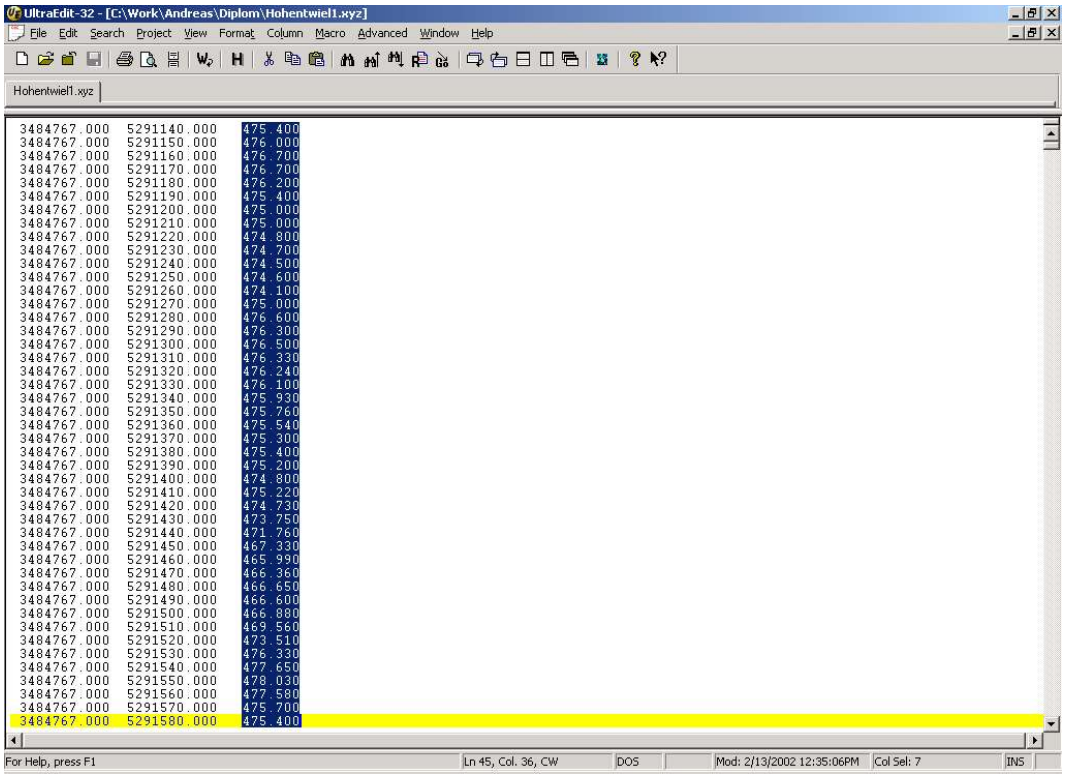


Figure 4 Höhendaten in UltraEdit

Die Höhendaten wurden kopiert und in unten stehendes VRML ***Elevation Grid*** Konstrukt in das Feld ***height*** eingefügt.

#VRML V2.0 utf8

```
Transform {
children
  Shape {
    appearance Appearance {
      material Material { }
    }
    geometry ElevationGrid{
      xDimension 166
      zDimension  242
      xSpacing 10
                              zSpacing 10
      height [
      475.400
      476.000
          .
          .
          .]
    }
  }
}
```

Die im vorigen Abschnitt errechneten Werte für ***xDimension***, ***zDimension*** und ***xSpacing***, ***zSpacing*** wurden in die entsprechenden Felder eingetragen. Das Resultat eines ersten VRML ***Elevation Grid*** Files sieht in einem VRML Browser wie folgt aus:

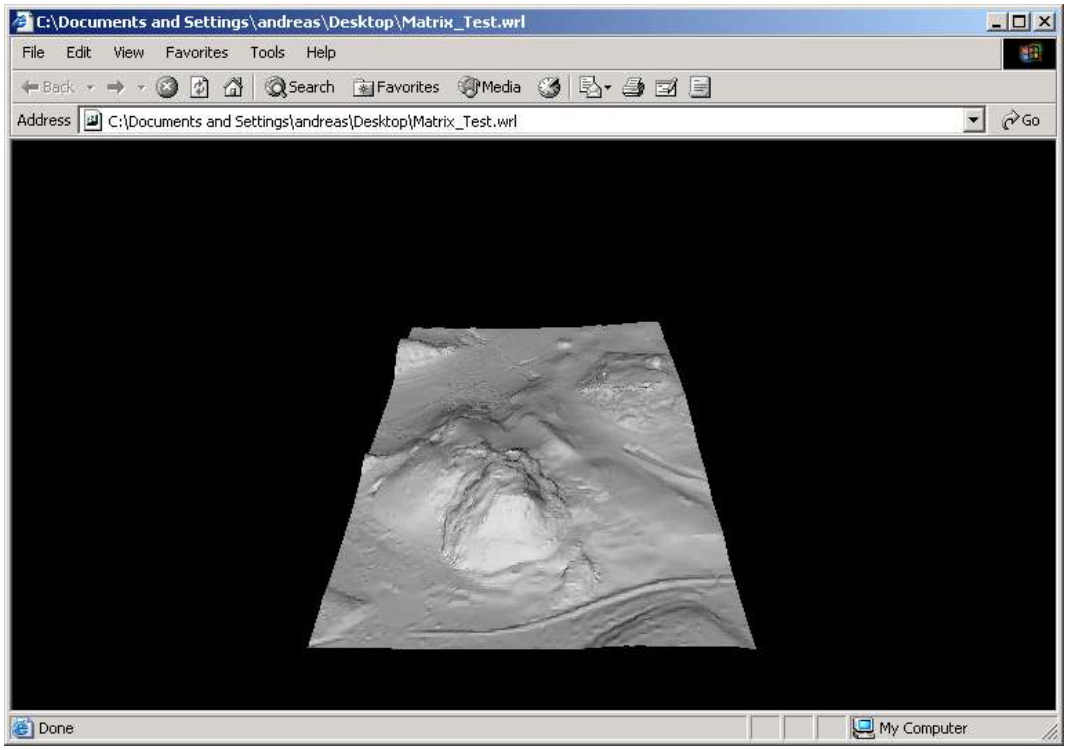


Figure 5 Beispiel EG im Browser

5.2.2 Textur

Das ***Elevation Grid*** VRML File, welches im letzen Abschnitt erzeugt wurde, wirkt in seiner Rohform natürlich noch ziemlich fremd und erinnert mehr an eine Marslandschaft als an das Naturschutzgebiet Hohentwiel.

Um Landschaften in 3D möglichst naturgetreu darzustellen, ist es möglich, ein Luftbild der Landschaft über das 3D Modell zu legen. Diese Technik wird im allgemeinen 3D-Sprachgebrauch Texture Mapping genannt.

Das Landesamt für Flurneuordnung und Landentwicklung Baden-Württemberg lieferte zusammen mit den Rohdaten für den Hohentwiel ein sehr hochauflösendes Luftbild der Region (siehe folgende Abbildung).

Figure 6 Luftbild

Dieses Luftbild hat in etwa eine Auflösung von 3 Metern. Das heißt, wenn man oben stehendes Bild vergrößert darstellt bzw. eine Zoom-Funktion darauf anwendet, können Objekte mit einer Größe von ca. 3 Metern (z.B. ein Auto) noch als individuelles Objekt erkannt werden.

Ziel unserer Bemühungen ist es, dieses Luftbild so über das 3D Modell zu legen, dass entsprechende auf dem Bild zu erkennenden Objekte an der richtigen Stelle im 3D Modell auftauchen. Nach mehreren Versuchen, das Bild über das 3D Modell zu legen, wurde herausgefunden, dass das Bild dazu an der X-Achse gespiegelt und um 90 Grad nach links gedreht werden muss. Außerdem wurde das Bild in eine Größe von 2048x2048 Pixel konvertiert, da alle VRML-Browser Texturen auf eine Größe, die ein Vielfaches von 2 ist, filtern. Liegt die Textur in einer anderen Größe vor, muss der VMRL Browser extra Rechenzeit dafür aufwenden, das Bild zu skalieren. Die Konvertierung des Bildes wurde mit der Trial Version von PaintShop Pro [PAINTSHOP] umgesetzt. Folgende Abbildung zeigt das modifizierte Luftbild der Region Hohentwiel in Paintshop Pro.

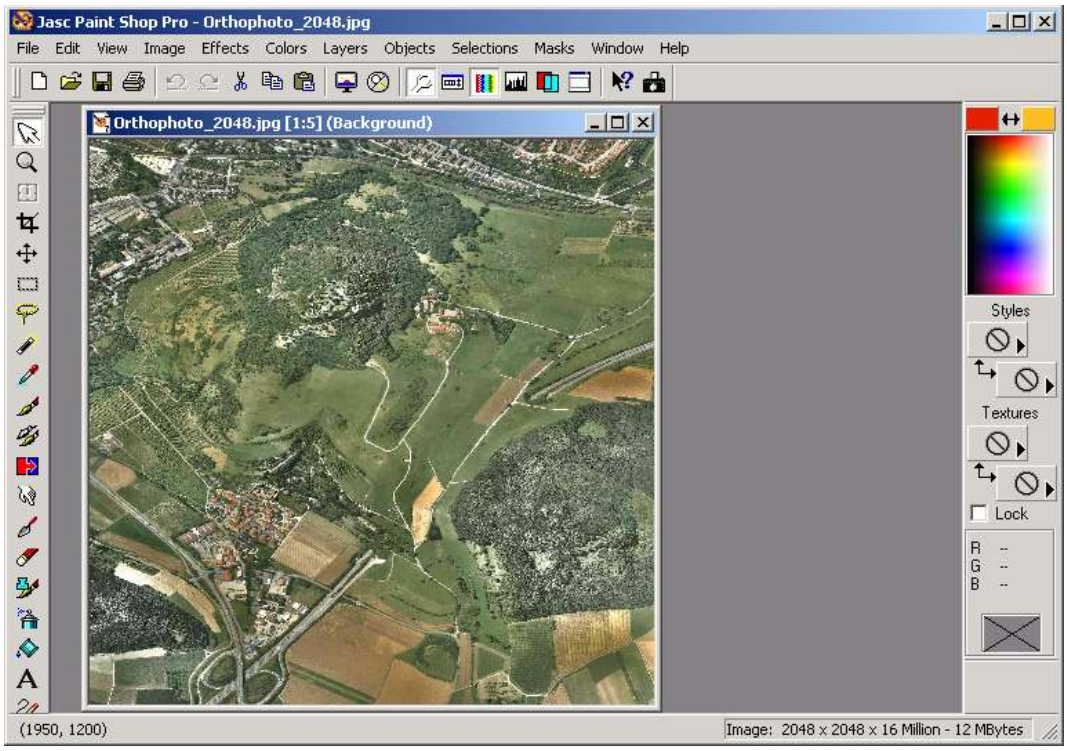


Figure 7 Luftbild in PaintShop

Dieses modifizierte Luftbild kann nun über das 3D Modell gelegt werden, also als Textur in dem VRML-File referenziert werden.

5.2.3 Image Texture Node

Das Konstrukt ***Image Texture*** ist in VRML dafür zuständig, eine Textur einem bestimmten geometrischen Objekt zuzuordnen. Der ***Image Texture*** Knoten sieht folgendermaßen aus:

```
ImageTexture {
      url [ ]
      repeatS TRUE
      repeatT TRUE
}
```

Das Feld ***url []*** gibt den Namen und den Ort der zu verwendenden Bilddatei an (z.B. textures/image.jpg). Bei diesem Beispiel würde die Bilddatei im Ordner images liegen, immer relativ gesehen zum Speicherort der VRML-Datei.

Die Felder ***repeatS*** und ***repeatT*** geben an, ob das Bild vertikal und horizontal wiederholt werden soll, wenn es nicht groß genug für das 3D Objekt ist.

Eine Textur muss in einer VRML Struktur immer innerhalb eines ***Appearance*** Knotens angegeben werden. Dieser Knoten beschreibt das Aussehen einer geometrischen Figur. Als Unterknoten von ***Appearance*** kann der ***Material*** Knoten stehen, welcher z.B. die Farben für ein 3D Objekt oder dessen Reaktion auf Licht angibt. Auch der Knoten ***Image Texture*** ist als ein Subknoten von ***Appearance*** definiert. Folgender Code zeigt die VRML Struktur des Beispiels im vorigen Abschnitt mit dem markierten ***Image Texture*** Knoten an der richtigen Stelle.

```
#VRML V2.0 utf8

Transform {
children
  Shape {
    appearance Appearance {
      material Material { }

      texture                    ImageTexture {
        repeatS TRUE
        repeatT TRUE
        url                   "Orthophoto_2048.jpg"
      }
    }
    geometry ElevationGrid{
      xDimension 166
      zDimension  242
      xSpacing 10
                              zSpacing 10
      height [
      475.400
      476.000
          .
          .
          .]
    }
  }
}
```

Folgendes Bild zeigt nun das Hohentwiel Modell mit dem entsprechenden Luftbild als ***Image Texture*** in einem VRML Browser

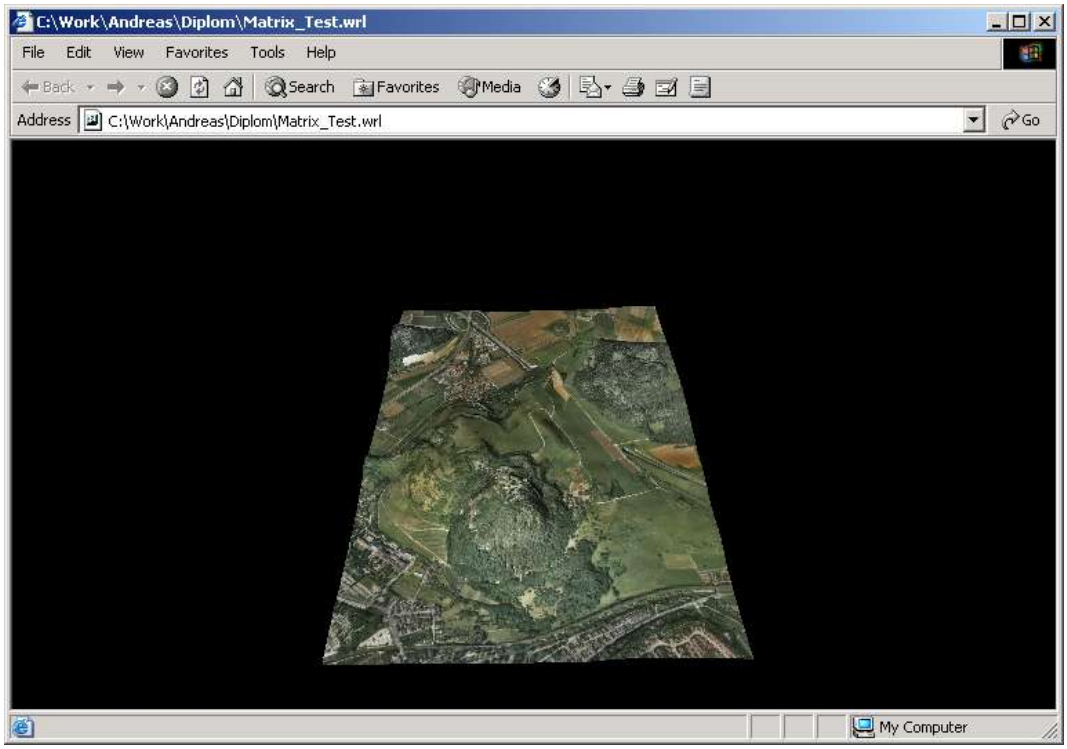


Figure 8 EG mit Texture im Browser

Wenn man die Landschaft um den Hohentwiel kennt, kann man auf obigem Bild erkennen, dass das Luftbild korrekt auf die 3D Landschaft abgebildet wurde. Wenn man sich jedoch im 3D Modell bewegt, fällt einem auf, dass die Landschaft insgesamt sehr dunkel wirkt. Das liegt daran, dass bis jetzt noch keine Lichtquelle angegeben wurde, die die Landschaft beleuchtet. Der Grund dafür, dass wir ohne Lichtquelle überhaupt etwas sehen, ist, dass jeder VRML Browser die Option eines so genannten Headlights bietet, welches standardmäßig eingeschaltet ist. Ein Headlight ist am besten vergleichbar mit dem Licht, welches z.B. Minenarbeiter auf ihrem Helm tragen. Es leuchtet immer in die Richtung, in die man gerade sieht. Jedoch ist ein Headlight nur eine sehr schwache Lichtquelle und dementsprechend ziemlich ungeeignet dafür, eine Landschaft in der Größenordnung des Hohentwiel Gebiets auszuleuchten. Folgender Abschnitt zeigt, wie in VRML eine Lichtquelle hinzugefügt wird.

5.2.4 Point Light Node

In VRML gibt es verschiedene Arten von Lichtquellen, welche in einer Scene definiert werden können. Für die Beleuchtung des Hohentwiel Modells soll der Point ***Light Knoten*** verwendet werden, welcher im folgenden Abschnitt detailiert erklärt wird.

Ein ***Point Light*** hat in einer VRML-Welt eine bestimmte Position und scheint aus dieser Position mit einer gewissen Lichtfarbe und Intensität in alle Richtungen. Folgendes Bild verdeutlicht diesen Sachverhalt.

Wie man gut erkennen kann, strahlt das ***Point Light*** Licht in alle Richtungen aus. Es ist daher besonders angebracht, das Sonnenlicht zu simulieren, welches ja auch von einer weit entfernten Lichtquelle in alle Richtungen ausgesandt wird.

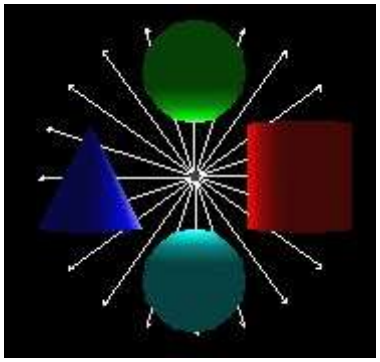


Figure 9 Point Light

Der ***Point Light*** Knoten hat folgende Struktur:

```
PointLight {
  on TRUE
  intensity 1
  ambientIntensity 0
  color 1 1 1
  location 0 0 0
  attenuation 1 0 0
  radius 100
}
```

Mit dem Feld ***on*** kann das ***Point Light*** an- oder ausgeschaltet werden. Das Feld ***intensity*** gibt die Leuchtkraft an. Werte in diesem Feld können zwischen 0.0 und 1.0 liegen. ***AmbientIntesity*** beschreibt den Faktor, inwiefern dieses Licht an der Umgebungsbeleuchtung beteiligt ist. Mögliche Werte sind wiederum zwischen 0.0 und 1.0. Das Feld ***color*** gibt die Farbe des Lichts an (1 1 1 entspricht z.B. einer Mischung der drei Farbkanäle RGB mit gleichen Teilen, somit einem weißen Licht).

Location beschreibt, wo die Lichtquelle in der Scene platziert wird (XYZ Koordinaten). ***Attenuation*** ist ein 3D Vektor, welcher angibt, wie das Licht an Intensität verliert, wenn die Distanz zur Lichtquelle größer wird. Das Feld ***radius*** gibt an, wie weit die Lichtstrahlen gehen sollen. In obigem Beispiel sind das 100 Meter. Alle Objekte, die weiter als 100 Meter entfernt liegen, werden nicht beleuchtet.

Ein ***Point Light*** Knoten ist ein eigenständiges VRML Konstrukt und kann daher an jeder Stelle in der VRML Struktur platziert werden. Es ist allerdings normalerweise üblich, Lichter am Anfang der VRML Datei zu platzieren. Der Beispiel Code mit ***Point Light*** Implementierung sieht demnach folgendermaßen aus:

```
#VRML V2.0 utf8

PointLight {
  intensity                    1
  color                      1 1 1
  location                 0 10000 0
  radius                     11000
}

Transform {
children
  Shape {
    appearance Appearance {
      material Material { }
              .
              .
              .
```

Wie unschwer zu erkennen ist, sind nicht alle Felder gesetzt, welche üblicherweise für ein ***Point Light*** definiert sind. Wenn Felder in einem VRML Knoten nicht gesetzt sind, benutzt der VRML-Browser automatisch Standardwerte. Obenstehendes ***Point Light*** erzeugt eine weiß strahlende Lichtquelle mit maximaler Intensität und einer Reichweite der Strahlen von 11.000 Metern. Die Lichtquelle steht direkt über dem XY Punkt 0,0 des Modells auf einer Höhe von 10.000 Metern. Um die Sonne zu simulieren könnten wir die Lichtquelle auch 250 Millionen Kilometer über dem Terrain platzieren und den Radius der Strahlen entsprechend erhöhen, allerdings macht dies für VRML keinen Unterschied verglichen mit jetziger Konfiguration.

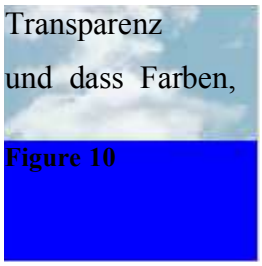
5.2.5 Background Node

Nachdem jetzt das 3D Modell richtig ausgeleuchtet ist, fehlt nur noch ein realistisch wirkender Hintergrund. Das Konstrukt ***Background*** erlaubt es uns, in VRML Farben und Texturen für den Boden und den Himmel der virtuellen Welt zu definieren. Der Background Knoten hat folgende Struktur:

```
Background {
  skyColor [ 0 0 0 ]
  skyAngle [ ]
  groundColor [ ]
  groundAngle [ ]
  backUrl [ ]
  bottomUrl [ ]
  leftUrl [ ]
  rightUrl [ ]
  frontUrl [ ]
  topUrl [ ]
}
```

Das Feld ***skyColor*** gibt die Farbe des Himmels als RGB Wert an. Das Feld ***skyAngle*** kann benutzt werden, um Farbverläufe zu erzeugen, so z.B. einen Himmel, dessen Farben im Zenit blau sind und dann Richtung Horizont in eine hellblaue Farbe übergehen. Um diesen Effekt zu erhalten müssen im Feld ***skyColor*** die RGB Werte für blau und hellblau angegeben werden und im Feld skyAngle ein Winkel im Bogenmaß, welcher angibt, bis zu welchem Grad am Horizont (ausgehend vom Zenit) der Farbverlauf reichen soll. Wenn man einen unifarbenen Himmel wünscht, muss im Feld ***skyColor*** nur eine Farbe angegeben werden, und das Feld ***skyAngle*** sollte weggelassen werden. Das Feld **groundColor** gibt die Farbe des Untergrundes der Welt an. Für **groudAngle** gilt das gleiche wie für ***skyAngle***. Die verschiedenen Felder mit der Endung ***Url*** können benutzt werden, um den Horizont und den Boden durch Texturen noch realistischer wirken zu lassen. Enthalten die Texturdateien Transparenzen, so scheint an diesen Stellen die gewählte Himmel- oder Bodenfarbe durch.

Die Arbeit mit Transparenzen in Bilddateien soll hier an einem kleinen Beispiel näher erläutert werden. Mit einem Bildbearbeitungswerkzeug wie z.B. Paint Shop Pro kann man in einer Bilddatei eine Farbe als die transparente Farbe definieren. In nebenstehender Abbildung ist dies die blaue Farbe im unteren Teil des Bildes.



bedeutet, dass der ganze blaue Bereich faktisch durchsichtig ist, welche hinter diesem Bereich liegen, durchscheinen.

Figure 10 **Background mit Transparenz**

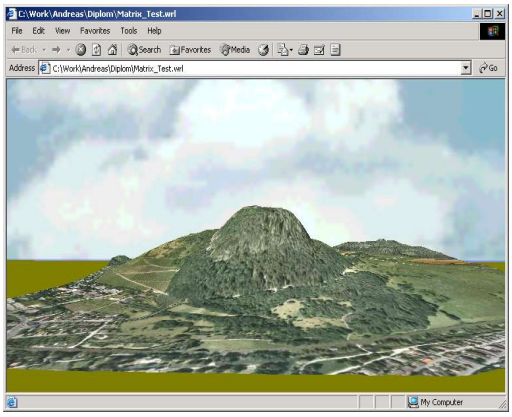
Dieses Bild wird jetzt den Feldern ***backUrl***, ***leftUrl***, ***rightUrl***, **frontUrl** zugewiesen. Für das Feld ***topUrl*** wird die gleiche Bilddatei verwendet, jedoch ohne Transparenz.

Der ***Background*** Knoten sieht somit folgendermaßen aus:

```
Background {
                        groundColor [
                           0.5 0.5 0
                           0.5 0.5 0
                              ]
                     groundAngle [1.57]

                     backUrl "sky41.gif"
                     leftUrl "sky41.gif"
                    rightUrl "sky41.gif"
                    frontUrl "sky41.gif"
                     topUrl  "sky4.gif"
}
```

Er liefert im Browser folgendes Ergebnis:



Wie man gut erkennen kann scheint im unteren Teil die grüne Farbe, welche als ***groundColor*** definiert wurde, durch. Wenn wir keine Transparenz in der Bilddatei verwendet hätten, sähe das Modell aus als schwebe es in einem Wolkenmeer. Mit Transparenz scheint es nun auf den grünen Grund fixiert zu sein. .

Figure 11 EG mit Background im Browser

Nachdem der ***Background*** Knoten eingefügt wurde, wirkt das VRML Modell des Hohentwiel entscheidend realistischer. Wenn man sich in der Welt bewegt, sieht man am Horizont nahezu echt wirkende Wolken, und falls man sich dem Rand des Modells nähert, kommt auch die schon erwähnte Bodenfarbe zum Vorschein.

Im nächsten Abschnitt soll nun die Performance des Modells verbessert werden.

5.3 Verbesserung der Performance

Das VRML Modell in seinem momentanen Status weist auf einem PIII mit 700 Mhz eine annehmbare Performance vor, man kann sich also flüssig im Modell bewegen. Bei einem Test mit einem leistungsschwächeren Computer (PII 300) geht die Performance jedoch so stark zurück, dass eine flüssige Bewegung kaum mehr möglich ist. Da jedoch das Modell einer möglichst großen Anzahl von Internetbenutzern zugänglich gemacht werden soll, muss es so optimiert werden, dass es auch auf Computern der älteren Generation lauffähig ist. Des Weiteren soll es auch auf die Benutzung über das Internet hin optimiert werden. Das heißt, auch User mit einer langsamen Internet-Verbindung sollten in der Lage sein, das Modell ohne größere Schwierigkeiten zu erforschen.

5.3.1 LOD

Es gibt nun verschiedene Ansätze, die Performance eines VRML Modells zu verbessern. Der wichtigste hierbei ist Level of Detail (LOD) Management. Um zu verstehen, wie LOD funktioniert, soll hier nun kurz erklärt werden, wie ein VRML Modell im Allgemeinen aufgebaut ist. In den vorherigen Abschnitten wurde das VRML ***Elevation Grid*** Konstrukt erklärt. ***Elevation Grid*** ist eine Möglichkeit, Geometriedaten in VRML darzustellen. Man konnte auch sehen, dass die einzigen Daten, welche der ***Elevation Grid*** Knoten benötigt, eine Matrix mit einer bestimmten Ausdehnung in X und Y Richtung und eine große Menge von Höhenangaben sind. Der VRML Browser errechnet aus diesen Daten, wie er die Landschaft darstellen soll. Er geht die Matrix von oben links nach unten rechts durch, bildet entsprechend der Höhenangaben Dreiecke und hängt diese aneinander. Dies bedeutet, dass das Modell des Hohentwiel aus vielen kleinen Dreiecken zusammengesetzt ist.

Folgendes Bild zeigt diesen Sachverhalt:

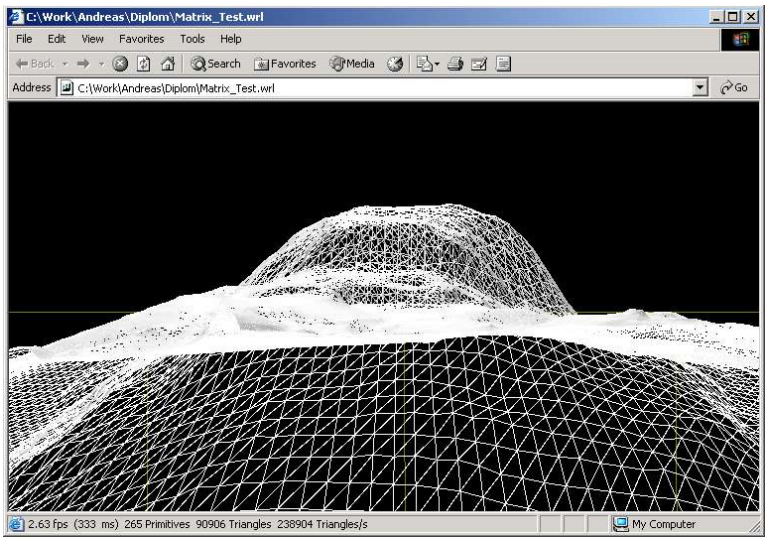


Figure 12 EG Gitternetzlinien

Wenn wir uns jetzt im 3D Modell bewegen, muss der Computer (bzw. die Graphikkarte) für jede Bewegung, die wir machen, alle diese Dreiecke neu berechnen, und das kann einen älteren Computer schnell an die Grenzen seiner Leistungsfähigkeit bringen. Ein kleines Rechenbeispiel soll diesen Sachverhalt etwas näher beleuchten. Unser Modell vom Hohentwiel besteht aus etwa 90.000 Dreiecken. Um eine realistische, flüssige Bewegung im Modell zu gewährleisten, muss eine Framerate von mindestens 10 fps (frames per second) gewährleistet sein. Dies bedeutet, dass das Bild, welches auf dem Bildschirm zu sehen ist, zehn mal in der Sekunde neu gezeichnet wird. Um 10 fps zu erhalten, muss die Grafikkarte 10 x 90.000 = 900.000 Dreiecke berechnen. Dies ist für moderne Graphikkarten kein Problem, jedoch kommen ältere Graphikkarten mit den Berechnungen nicht mehr schnell genug nach und deshalb sinkt die Framerate. Wenn die Framerate sinkt, werden nicht mehr genügend Bilder am Bildschirm aufgebaut und die Bewegung wirkt nicht mehr flüssig. Das kann im Extremfall soweit kommen, dass der Bildschirm bei der Bewegung durch die Landschaft z.B. nur jede zweite Sekunde aufgefrischt wird und die Bewegungen somit extrem stockend wirken.

Nach diesen weiterführenden Erklärungen kehren wir zurück zum Level of Detail Management. LOD setzt genau an dem Punkt an, der in oben stehendem Beispiel besprochen wurde. Es versucht, die Zahl der Dreiecke, welche berechnet werden müssen, zu reduzieren. Der Ansatz, den LOD benutzt, kommt wie bei so vielen Dingen in der Computerwelt aus dem richtigen Leben. Stellen wir uns vor, wir

befinden uns im Naturschutzgebiet Hohentwiel. Wir stehen 500 Meter vom Berg entfernt und schauen auf den Berg. Aufgrund der großen Entfernung können wir natürlich nicht viel erkennen. Wir sehen zwar die Konturen des Berges, vielleicht auch Konturen von großen Objekten, aber wir erkennen keine Details.
Allerdings berechnet der Computer in einem 3D Modell stumpfsinnig alle Dreiecke, egal wie weit entfernt vom Berg sich der Benutzer des Modells befindet. Genau hier setzt Level of Detail Management an. Mit LOD kann man einem Objekt verschiedene Detailstufen zuweisen. Wenn wir z.B. sehr weit von einem Objekt entfernt sind, wird das Objekt in seiner niedrigsten Detailstufe angezeigt. Niedrigste Detailstufe heißt, dass das Objekt nur sehr wenige Dreiecke umfasst, welche der Computer berechnen muss. Wenn wir uns nun dem Objekt nähern, wird automatisch das Objekt in einer höheren Detailstufe nachgeladen und ersetzt das vorherige. Es ist dem Programmierer selbst überlassen, wie viele verschiedene Detaillevel er für ein Objekt verwenden will, und es kommt natürlich auch darauf an, wie umfangreich das Modell ist. Es zeigt sich jedoch, dass diese Technik, wenn sie richtig angewendet wird, die Graphikkarte eines Computers stark entlasten kann.

Der LOD Knoten in VRML ist folgendermaßen aufgebaut:

```
LOD {
  level []
  center 0 0 0
  range []

}
```

Das Feld ***level*** beinhaltet eine Liste geometrischer Figuren in normalerweise verschiedenen Detaillierungsgraden. Dies könnte in unserem Beispiel das Gebiet des Hohentwiel in verschiedenen Auflösungen sein. Mit ***center*** wird ein Punkt festgelegt, von welchem aus der VRML Browser Entfernungen misst. Im Feld ***range*** werden die Entfernungsangaben der verschieden Detaillevel angegeben. Wenn wir z.B. im Feld ***level*** drei verschiedene Detaillierungsgrade für unser Modell angeführt haben, geben wir im Feld ***range*** an, wie weit diese verschiedenen Repräsentationen unseres Modells vom Punkt ***center*** entfernt liegen. Ein kleines Beispiel macht diesen Sachverhalt etwas deutlicher:

```
LOD {
  level [
      Inline {url "HighRes.wrl"}
      Inline {url "MedRes.wrl"}
      Inline {url "LowRes.wrl"}]
  center 0 0 0
  range [200, 400]

}
```

Wie man in obigem Beispiel sieht, sind im ***level*** Feld drei Modelle mit jeweils unterschiedlicher Auflösung angegeben. (Mit Hilfe des ***Inline*** Knotens können Referenzen zu anderen VRML Dateien hergestellt werden.) Im Feld ***range*** stehen die Entfernungsangaben von 200 bzw. 400 Metern. Obiges wird vom VRML Browser folgendermaßen interpretiert: Er stellt bis zu einer Entfernung von 400 Metern das in ***LowRes.wrl*** hinterlegte Modell dar. Nähert man sich mehr als 400 Meter, wird das in ***MedRes.wrl*** hinterlegte Modell nachgeladen und ersetzt das andere. Wenn man sich auf mehr als 200 Meter dem Punkt ***center*** nähert, wird das in ***HighRes.wrl*** hinterlegte Modell geladen, welches die höchste Detaillierungsstufe darstellt.

5.3.2 Tiling

Eine zweite Möglichkeit, die Performance eines 3D Terrainmodells zu verbessern, wird Tiling genannt. Bei dieser Technik wird das Terrain in viele kleine Stücke (Tiles) aufgeteilt, das heißt, ein 3D Landschaftsmodell wird nach einem vordefinierten Muster so aufgeteilt, dass mehrere kleinere Teildateien entstehen, welche logisch miteinander verknüpft sind. Dies hat zum einen den Vorteil, dass nicht das komplette Modell im Hauptspeicher des Computers gehalten werden muss, sondern immer nur die Teildateien, welche gerade im Blickwinkel des Betrachters sind, was vor allem für sehr große Landschaftsgebiete, welche sehr große Datenmengen umfassen, die nicht komplett in den Speicher geladen werden können, Sinn macht. Zweitens bringt die Aufteilung in kleinere Landschaftsstücke den Vorteil einer kürzeren Ladezeit über das Internet. Wenn z.B. ein Landschaftsmodell in einer großen VRML Datei gespeichert ist, muss der Benutzer lange warten (natürlich abhängig von der Geschwindigkeit seiner Internetverbindung), bis die komplette VRML Datei herunter geladen und somit startklar ist. Wird jedoch der Ansatz des Tilings benutzt, lädt z.B. der VRML

Browser Blaxxun Contact nur diejenigen Teildateien herunter, welche im Ausgangszustand des Modells im Blickfeld des Anwenders sind. Alle anderen Dateien werden hintereinander nachgeladen, wenn sich das Blickfeld des Benutzers ändert, d.h. wenn sich der Benutzer im Modell bewegt.

In dieser Diplomarbeit soll eine Kombination dieser beiden Ansätze verwendet werden, was einerseits die Möglichkeit mit sich bringt, das Modell auch auf Computern älterer Generation zu betrachten und andrerseits die Downloadzeit für das Modell so optimiert, dass es auch über langsamere Internetverbindungen funktionsfähig ist.

5.3.3 Rez-Tools

Um ein 3D Modell wie das des Hohentwiel in viele kleinere Teildateien zu teilen, bedarf es unbedingt eines Automationsmechanismus, da solche Arbeiten von Hand erstens sehr zeitaufwendig und zweitens auch stark fehleranfällig wären.

Es existiert eine Sammlung von Werkzeugen, genannt Rez-Tools [REZTOOLS], welche unter anderem automatisiertes Tiling und LOD-Management beherrschen. Es handelt sich hierbei um eine Sammlung von Java Klassen, welche zum kostenlosen Download angeboten werden.

In diesem Abschnitt soll nun erklärt werden, wie mit Hilfe der Rez-Tools die ursprüngliche Terraindatei entsprechend unserer Zielsetzungen optimiert wird.
Hierbei soll auch anhand von Beispielen die Funktionsweise der Tools näher beschrieben werden.

Rez bietet die Möglichkeit, eine große ***Elevation Grid*** VRML Datei einzulesen und in eine vom Anwender vorgegebene Anzahl von Teildateien aufzuteilen. Das Tiling der Dateien kann automatisch mit dem LOD Ansatz (5.3.1) kombiniert werden. Der Anwender legt fest, wie viele verschiedene Levels of Detail, d.h. wie viele unterschiedliche Detailstufen das endgültige Modell enthalten soll. Anhand dieser Angaben teilt Rez das ursprüngliche Modell auf.

Wenn der Anwender z.B. drei Detaillierungsstufen auswählt, wird das Modell folgendermaßen aufgeteilt:

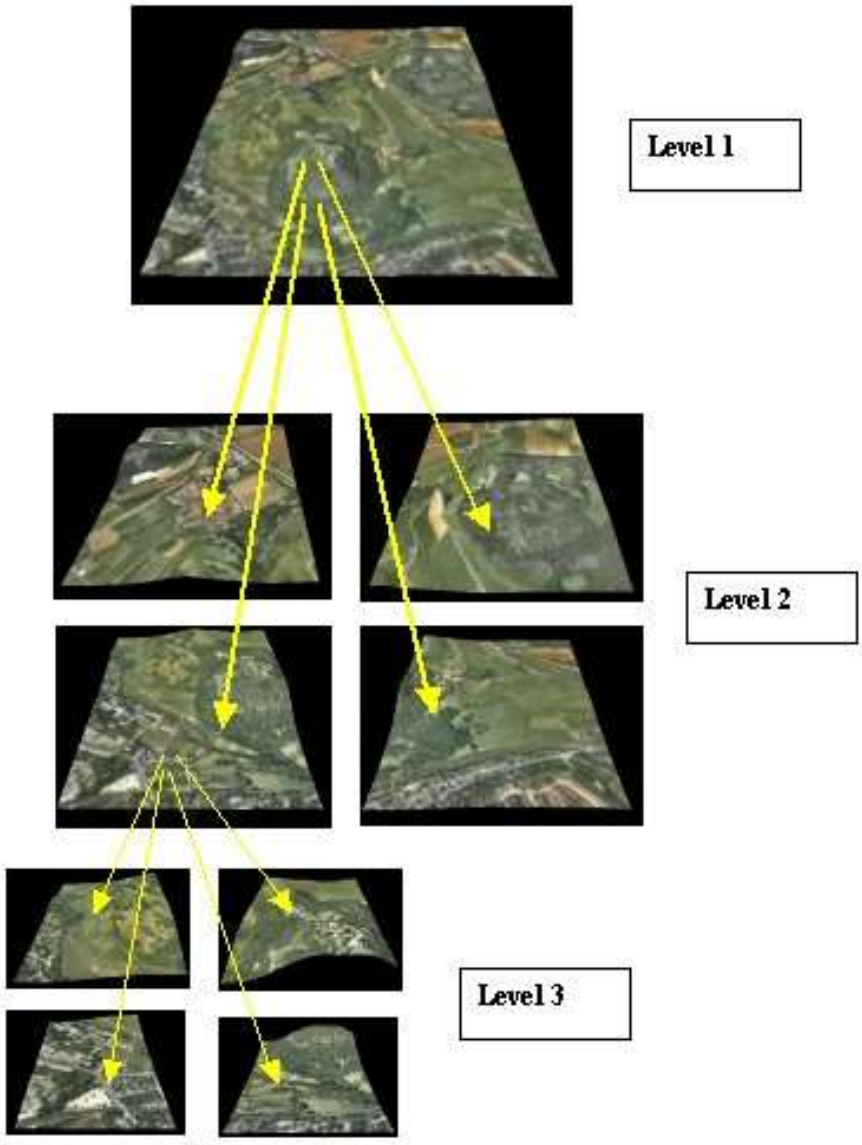


Figure 13 LOD Rez Tools

Hierbei wird ersichtlich, dass das Terrain in eine Baumstruktur aufgeteilt wird. Im ersten Level der Struktur befindet sich eine VRML Datei mit sehr niedriger Auflösung, d.h. sehr wenigen Details. Im zweiten Level wird diese Datei in vier Teildateien aufgeteilt, welche eine höhere Auflösung haben. Jede dieser vier Teildateien wird wiederum in vier Dateien aufgeteilt. Im letzen Level des Baumes (hier Level 3) befinden sich die VRML Dateien mit der höchsten Auflösung. Die Technik, immer ein Tile mit vier neuen Tiles zu ersetzen, wird QuadTree Technik genannt und findet in verschiedenen Variationen auch in neueren Computerspielen ihre Anwendung. Wie man am obigen Beispiel sehen kann, nimmt die Zahl der

Teildateien mit jedem Detaillevel um das vierfache zu. Bei z.B. fünf Detaillevels teilt das Rez Tool unser VRML File in 341 Teildateien auf bei sechs Detaillevels wären es schon 1365 Teildateien. Das Problem hierbei ist, dass jede dieser Teildateien einzeln über das HTPP Protokoll angefordert werden muss. Wenn vom VRML Browser eine Teildatei gebraucht wird, sendet er eine Anforderung an den Webserver, auf dem das Modell gespeichert ist. Der Webserver sendet daraufhin eine Mitteilung zurück zum VRML Browser, dass er die Anforderung erhalten hat und bereit zum Senden ist. Der VRML Browser sendet nun wieder eine Mitteilung, dass er bereit zum Empfangen ist, und erst dann geht die eigentliche Datenübertragung vonstatten. Diese Art der Sicherung von Datenübertragungen wird 3-way-handshake genannt und ist typisch für das HTTP Protokoll. Wenn jedoch viele kleine Dateien übertragen werde, führt dieses Protokoll sehr schnell zu einem großen Overhead. Es ist daher genau abzuwägen, wie viele Detaillevel wirklich gebraucht werden, um das System insgesamt performant zu halten.

Nach mehreren Tests hat sich herausgestellt, dass eine Struktur mit fünf Stufen, d.h. fünf Detaillevels für die Hohentwiel Anwendung die beste Performance bietet.
Die folgenden Zeilen zeigen nun detailliert, wie das 3D Modell mit Hilfe der Rez-Tools optimiert wird.

Die Rez Tools sind eine Sammlung von Java Klassen, die auf Kommandoebene agieren. Rez ist ein Framework, mit dem es möglich ist, verschiedene gebräuchliche Terrain Formate (z.B. DTED, Gtopo30, ArcView, VRML Elevation Grid) einzulesen und in verschiedenen optimierten Formaten abzuspeichern. In dieser Diplomarbeit benutzen wir den Import Filter von Rez für ***Elevation Grid*** Daten und schreiben unser optimiertes Terrain als VRML ***Elevation Grid*** in QuadTree Format. Rez benutzt zur Konfiguration des Tools eine Textdatei mit einem frei wählbaren Namen, so z.B. ***config.txt***. In dieser Datei stehen z.B. Pfadangaben wie das Quellverzeichnis und das Zielverzeichnis. Es sind unter anderem auch Angaben dazu, in welcher Art und Weise welches Plugin für das Lesen und Schreiben von Daten verwendet werden soll und mit welcher Genauigkeit Gleitkommazahlen dargestellt werden sollen, in dieser Datei gespeichert. Des weiteren benutzt Rez eine Batchdatei zum Starten der Java Klasse und zur Übergabe von Parametern. Die Parameter von Rez sind die folgenden:

ConfigFile: Pfad und Name der Konfigurationsdatei

FirstTreeLevel: Start Level des Baumes (z.B. 0)

FinalTreeLevel: Letzter Level des Baumes (z.B. 4)

DetailScale: Skalierungsfaktor für LOD Range Werte

GzipFlag: kann gesetzt werden, um Dateien zu komprimieren

SamplingFlag: Wenn dieses Flag gesetzt ist, entfernt Rez Polygone aus Dateien mit niedrigem Detaillevel (hat entscheidenden Einfluss auf die Performance).

SamplingIncrement: Faktor für die Sample Größe

HorizontalScale: Faktor, um den das Terrain (x,z) vergrößert wird

HeightScale: Faktor, mit welchem die Höhenangaben (y) multipliziert werden

MinOutputTileDimension: minimale Matrixgröße eines einzelnen Samples

MaxOutputTileDimension: maximale Matrixgröße eines einzelnen Samples

Translation x: Verschiebung der Tiles auf der x-Achse

Translation z: Verschiebung der Tiles auf der z-Achse

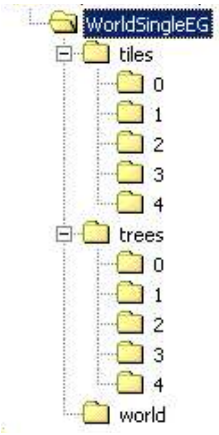
TreeType: für Binary Tree(y) oder Quadtreee (n)

Für die Optimierung der Hohentwiel Daten wird Rez mit folgenden Parametern aufgerufen:

```
java -DdebugOn=false -classpath .;.\Rez.jar rez.Rez C
onfig.txt 0 4 1.35 n y 0 0 1 40 50 0 0 n
```

Obiges bedeutet, dass unsere Einstellungen in der Datei Config.txt liegen, dass wir eine Baumtiefe von fünf Levels haben wollen (0-4), einen LOD-Range Scale von 1.3, keine Komprimierung der Dateien, Optimierung der Daten niedriger Detaillevel, keine Vergrößerung der Sample Größe, keine Vergrößerung des Terrains (0) , keine Vergrößerung der Höhen (1), minimale Größe von Samples soll 40 sein, maximale Größe von Samples soll 50 sein, keine Verschiebungen und dass das Ausgabeformat Quadtree sein soll.

Rez legt im Zielpfad (hier WorldSingleEG) folgende Dateistruktur an:



Rez legt ein Verzeichnis ***tiles*** an, in dessen Unterverzeichnissen sich die aufgeteilten ***Evelation Grid*** Dateien befinden, so z.B. in Verzeichnis 0 eine Datei, in Verzeichnis 1 vier Dateien, usw. Im Verzeichnis ***trees*** sind keine geometrischen Informationen hinterlegt. Die Dateien in den Unterverzeichnissen von ***trees*** steuern den Aufruf der wirklichen Terraindateien (in tiles) und haben zusätzlich die LOD Logik implementiert. Im Verzeichnis ***world*** liegt die Startdatei, mit welcher das Modell geladen wird.

Figure 14 Dateistruktur

Im Folgenden soll beschrieben werden, wie die einzelnen Dateien zusammenhängen und nacheinander aufgerufen werden. In der Startdatei im Verzeichnis ***world*** können Angaben zum Aussehen der VRML-Welt, wie z.B. Licht, Hintergrund, Navigationseinstellungen und Aussichtspunkte definiert werden. Die Startdatei ruft dann den ***Root*** Knoten des Baumes, sprich die VRML Datei, welche im Verzeichnis ***trees/0*** liegt, auf. Die Struktur dieser Datei wird nachstehend vereinfacht dargestellt:

```
LOD {
  level [ Group { children [
    Inline {url "../../trees/1/world0-0.wrl"}
    Inline {url "../../trees/1/world0-1.wrl"}
    Inline {url "../../trees/1/world1-0.wrl"}
    Inline {url "../../trees/1/world1-1.wrl"}
    ]}
    Inline {url "../../tiles/0/world0-0.wrl"}]
  center 0 0 0
  range  2500
}
```

Obenstehender Code bewirkt folgendes:
Bis zu einer Entfernung von 2500 Metern wird das ***Elevation Grid*** File ***"../../tiles/0/world0-0.wrl"*** dargestellt, kommt man dem Terrain näher als 2500 Meter, so werden die vier Dateien in ***trees/1*** nachgeladen. Diese vier Dateien haben grundsätzlich die gleiche Struktur wie obige Datei. Das heißt, jede der vier Dateien definiert wieder vier Folgedateien aus dem ***tree*** Verzeichnis und eine Datei mit ***Elevation Grid*** aus dem ***tiles*** Verzeichnis. Somit entsteht eine verlinkte Struktur, welche bewirkt, dass der Terrain, je näher man ihm kommt, umso detaillierter dargestellt wird.

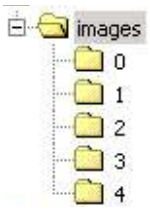
Um dieses optimierte Modell zu vervollständigen, muss nun noch das Luftbild, welches über das Terrain gelegt werden soll, dieser Struktur angepasst werden. Auch hierzu gibt es ein Werkzeug welches mit den Rez Tools ausgeliefert wird. Dieses Werkzeug heißt SmoothImageSlicer und teilt ein Bild im JPEG oder GIF Format so auf, dass es mit der optimierten Struktur verwendet werden kann.

SmoothImageSlicer hat folgende Parameter:

FilePath	absoluter Pfad zur Bilddatei
FromLevel	Start Level (z.B. 1)
ToLevel	Stop Level (z.B. 4)
VerboseFlag	zeigt zusätzliche Informationen auf der Konsole
ImageWidth	x-Dimension der Zieldateien in Pixel (z.B. 128)
ImageHeight	y-Dimension der Zieldateien in Pixel (z.B. 128)
BinaryTreeFlag	Flag bei Verwendung von BinaryTrees (bei QuadTree n)
GifOutputFlag	Flag für GIF Ausgabe (bei JPEG Ausgabe n)

SmoothImageSlicer wird nun für das Luftbild des Hohentwiel mit folgenden Parametern aufgerufen:

```
Java -Xmx250M SmoothImageSlicer C:\Work\Andreas\Graph-
ic\Rezneu\tools\slicer\world.jpg 1 4 y 256 256 n n
```



SmoothImageSlicer lieferert als Ergebnis folgende Verzeichnisstruktur: Das Luftbild des Hohentwiel wurde durch das ImageSlicer Tool an die Strukturen unseres optimierten Terrains angepasst, d.h. in den verschiedenen Unterverzeichnissen liegen nun die Texturen, die zu den ***Elevation Grids*** in ***tiles*** passen mit der entsprechenden Auflösung.

Figure 15 Images Verzeichnis

Der Ordner ***images*** muss nun nur noch in das Verzeichnis ***WorldSingleEG*** kopiert werden. Der Verzeichnisbaum sieht danach wie folgt aus:

Figure 16 Root Vezeichnis

Unbedingt zu beachten ist, dass die Ausgangsdatei für die Bildaufteilung (hier world.jpg) genau gleich benannt werden muss wie die Ausgangsdatei für die ***Elevation Grid*** Aufteilung (hier world.wrl) Dies ist sehr wichtig, da die Rez-Tools anhand dieser Angaben die Namensvergabe für alle Subdateien festlegen. Sind die Namen unterschiedlich, kann der VRML-Browser die in den ***Elevation Grid*** Dateien angegebenen Texturen nicht finden, da diese dann den falschen Namen tragen.

Um die vorgenommene Optimierung abzuschließen, muss nun noch die Startdatei im Verzeichnis ***world*** mit Informationen wie Licht und Hintergrund ergänzt werden, wie ich bereits in den Abschnitten PointLightNode und BackGroundNode beschrieben habe.

5.3.4 Fazit der Optimierung

Durch die Implementierung des Level of Detail Managements und die Aufteilung des Terrains in kleinere, verkettete Teildateien, wurde die Performance des 3D Modells entscheidend gesteigert. Das Modell ist nach Abschluss der Optimierungen auch auf einem älteren Rechner mit langsamer Internetverbindung sinnvoll einzusetzen. Jedoch treten durch die Optimierungen auch Probleme auf. Durch die Aufteilung des Terrains in viele kleine Teildateien entstehen teilweise sichtbare, feine Risse in der Landschaft. Diese Risse treten dort auf, wo zwei Teildateien zusammengefügt sind. Das Problem ist, dass partiell das LOD Management ein Tile schon durch ein höher auflösendes Tile ersetzt hat und das Nachbartile noch mit einer niedrigeren Auflösung dargestellt wird. Da die benachbarten Tiles dann verschiedene Auflösungen haben, stimmen die Höhenangaben an den Kanten der Tiles manchmal nicht exakt überein, und deshalb werden Risse sichtbar. In der ersten Version des optimierten 3D Modells waren diese Risse oft ziemlich auffallend und störend. Jedoch habe ich durch verschiedene Versuche herausgefunden, dass die Risse bei Vergrößerung der Tilegröße abnehmen. Bei einer Tilegröße zwischen 40 und 50 treten Risse nur noch ziemlich selten auf und stören somit das Gesamtbild nur geringfügig. Die völlige Beseitigung der Risse wäre nur durch die Erweiterung der Rez Tools möglich. Das heißt, die Java Klasse, welche die Terrain Tiles erstellt, müsste so erweitert werden, dass die Höhenangaben benachbarter Tiles auch in den verschiedenen LOD Stufen übereinstimmen. Jedoch hatte ich für diese Änderung der Rez Tools, was eine intensive Einarbeitung in die internen Programmabläufe erfordert hätte, während meiner Diplomarbeit leider keine Zeit mehr zur Verfügung.

5.4 Vulkanpfad

In diesem Abschnitt soll gezeigt werden, wie der Vulkanpfad in das 3D Modell implementiert wird. Der Vulkanpfad ist, wie schon kurz im Kapitel Einführung erklärt wurde, ein Wanderweg, welcher auf rund 3 km Länge um den Hohentwiel führt. Am Rande dieses Weges sind 13 Schautafeln aufgestellt, welche dem Besucher Informationen zum Berg und zum Naturschutzgebiet bieten.
Diese Tafeln sollen auch in das virtuelle 3D Modell des Hohentwiel integriert werden. Das Institut für Landschaftsökologie und Naturschutz in Singen lieferte hierzu hochauflösende Farbbilder, welche genau dem Inhalt der realen Tafeln entsprechen. Im 3D Modell sollen nun an den entsprechenden Stellen Tafeln mit den entsprechenden Farbbildern als Textur integriert werden. Des Weiteren soll bei manchen Tafeln im 3D Modell die Möglichkeit bestehen, Informationsvideos darauf abzuspielen. Das heißt, die Tafeln sollen teilweise auch als eine Art Leinwand dienen.

Die Tafeln wurden mit einer Trial Version der Software Cosmo Worlds [COSMOWORLD] erstellt. Dies ist ein professioneller VRML Builder, welcher nahezu alle Funktionen der VRML97 Spezifikation unterstützt. Die Tafeln wurden aus zwei so genannten Basis Geometrien zusammengesetzt, und zwar aus einem Würfel (VRML Bezeichnung ***Box***) und einem Zylinder (VRML Bezeichnung ***Cylinder***). Basis Geometrien in VRML existieren für einfache Objekte wie die obigen und zusätzlich für Kugel und Kegel. Der Vorteil von Basis Geometrien ist, dass man nicht extra alle einzelnen 3D Koordinaten für ein Objekt, wie z.B. einen Würfel angeben muss, sondern nur die Bezeichnung und Werte für dessen Größe. Basis Geometrien können dann wie alle Geometrien in VRML skaliert, gedreht und verschoben werden. Die Werte des Würfels wurden so gesetzt, dass er ein flaches Quadrat darstellt anstatt eines Würfels. Auf dieses Quadrat wird dann als Textur ein Bild der Vulkanpfadtafel abgebildet. Folgende Abbildung zeigt eines der Schilder in dem Programm Cosmo Worlds:

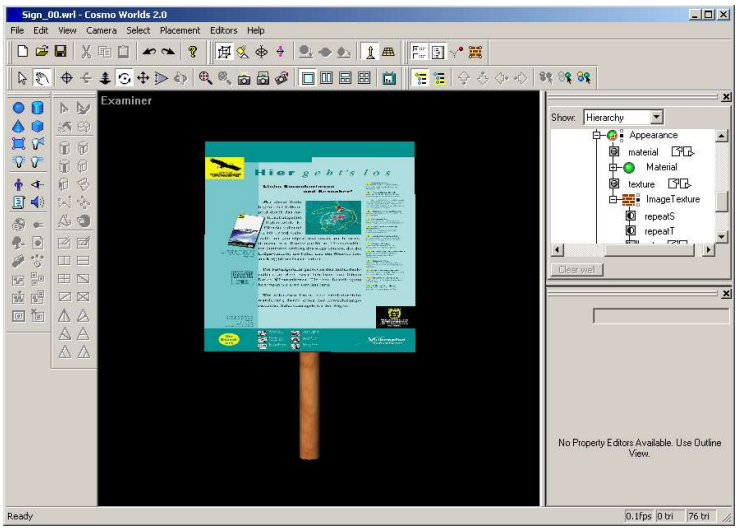


Figure 17 Schild in Cosmo Worlds

Für jedes der Schilder, wurde außerdem der ***Billboard*** Knoten integriert. Das ***Billboard*** Konstrukt in VRML bewirkt, dass Objekte wie z.B. ein Schild immer automatisch so gedreht werden, dass der Betrachter sie am besten sieht. Dies bedeutet, dass das Schild sich um seine eigene Achse (Z-Achse) drehen kann und sich automatisch immer dem Betrachter zuwendet. Dies hat den Vorteil, dass die Navigation für den Benutzer erleichtert wird. Folgender Code, zeigt beispielhaft, wie ein Schild aufgebaut ist:

```
Billboard {
      Transform {
                                                    ..
        Shape {
                                        texture  ImageTexture {
                                                    ..
                                          }
                                  geometry Cylinder {
                                                    ..
                                          }
                                       }
                                    Shape {
                                                    ..
                                        texture  ImageTexture {
                                                    ..
                                          }
           geometry                    Box {
               ..
                                          }
                                       }
                                     }
 }
```

Wie ersichtlich ist, wurden die zwei Basis Geometrien (***Cylinder*** und ***Box***) unter einem ***Transform*** Knoten angelegt, welcher wiederum unter dem ***Billboard*** Knoten liegt. Der ***Transform*** Knoten bewirkt, dass sich Transformationen wie Verschiebungen, Rotationen und Skalierungen auf das komplette Schild, das heißt auf dessen Pfosten (***Cylinder***) und dessen Oberteil (***Box***) auswirken. Alle Schilder wurden als eigenständige VRML Datei abgespeichert und wurden in das Modell des Hohentwiel mittels ***Inline*** Anweisung integriert.

Wie schon erwähnt sollen manche Schilder auch die Möglichkeit haben, Filme welche zum Text des Schildes passen, abzuspielen. Dies soll folgendermaßen funktionieren. Der Benutzer kann mittels Mausklick auf das Schild dessen Funktion wechseln. Das heißt standardmäßig wird das Bild des Schildes dargestellt, wenn aber der Benutzer darauf klickt, wird anstatt des Bildes ein Film eingeblendet und auf dem Schild abgespielt.

5.4.1 Movie Texture Node

Filme können in VRML mit dem ***Movie Texture*** Konstrukt auf eine Geometrie abgebildet werden. Dieses Konstrukt ist dem ***Image Texture*** Knoten, welchen wir schon kennen gelernt haben, sehr ähnlich. Es hat jedoch noch einige zusätzliche Felder zur Steuerung des Films eingegliedert. Dies sind Felder, die bestimmen, wann der Film gestartet und beendet wird (***startTime, stopTime***), wie schnell der Film abläuft (***speed***) und ob sich der Film wiederholt (***repeat***). Wie auch bei ***Image Textures*** ist zu beachten, dass der Film in einem Format, das ein vielfaches von 2 ist, vorliegt (z.B. 256x256), da sonst der VRML Browser eine zeitaufwendige Konvertierung in eine solche Größe vornehmen muss und da der Film sonst in manchen Browsern überhaupt nicht läuft.

Um dem Benutzer die Möglichkeit zu geben, zwischen dem normalen Bild und einem Film auf einem Schild mittels Mausklick auszuwählen, werden ein ***TouchSensor*** und ein ***Switch*** Knoten in Verbindung mit einem ***Script*** Knoten verwendet.

5.4.2 TouchSensor Node

Der ***TouchSensor*** Knoten wird in VRML verwendet, um interaktiv auf Benutzereingaben mit der Maus zu reagieren. Er wird normalerweise als ein Gruppenknoten verwendet. Das heißt, es können andere VRML Knoten wie z.B. Geometrie Konstrukte oder aber ein Schild in der Hierarchie unter ihm stehen, und somit für Mauseingaben „empfänglich“ gemacht werden. Der ***TouchSensor*** Knoten hat nur ein einziges Feld, mit welchem er an- bzw. ausgeschaltet werden kann. Wenn sich der Mauszeiger über einem Objekt, welches den ***TouchSensor*** Knoten integriert hat, befindet, wandelt sich der Zeiger in den meisten Browsern in eine Hand um, was dem Benutzer symbolisieren soll, dass dieses Objekt interaktiv ist. Der Knoten funktioniert folgendermaßen: wenn der Benutzer den Mauszeiger über ein Objekt, welches den ***TouchSensor*** Knoten integriert hat, bewegt, sendet der Knoten einen so genannten Event aus, in diesem Fall den Event ***isOver***. Klickt der Benutzer auf das Objekt, wird der Event ***isActive*** ausgelöst, und wenn der Benutzer die Maustaste wieder loslässt, wird der Event ***touchTime*** ausgelöst. Auf diese Events kann nun in einem ***Script*** Knoten reagiert werden, und es können entsprechende Aktionen ausgeführt werden. In unserem Fall soll der Event ***touchTime*** an den ***Script*** Knoten weitergeleitet und dort verarbeitet werden.

5.4.3 Script Node

Der ***Script*** Knoten in VRML wird in der Regel dazu verwendet, um auf Events zu reagieren und entsprechende Aktionen auszuführen, wie z.B. Berechnungen anzustellen oder aber andere Events zu senden. ***Script*** Knoten werden in der Sprache VRML Script verfasst, welche eine Untermenge der Java Script Sprache darstellt. Ein ***Script*** Knoten implementiert normalweise ein ***EventIn*** Feld, eines oder mehrere ***EventOut*** Felder und eine Funktion, welche aufgerufen wird, wenn ein ***EventIn*** an den ***Script*** Knoten gesendet wird. Zusätzlich können auch noch Datentypen, welche für den internen Programmablauf benötigt werden, definiert werden. Der ***Script*** Knoten für die Steuerung unseres Schildes sieht folgendermaßen aus:

```
DEF MyScript Script {
  eventIn      SFTime         touchTime
  eventOut     SFInt32 output
  eventOut     SFTime          startT
  eventOut     SFTime           stopT
```

```
    field                          SFBool       choice TRUE
    url                          "javascript:

        function touchTime(value, time) {
            if (choice) {

                                 choice = FALSE;

                                 startT = time;

                                   output = 0;



                                                                }
            else {

                                 choice = TRUE;

                                 stopT  = time;

                                   output = 1;









                                                                }
        }
    "
}
```

Dieses Script hat ein ***EventIn*** Feld namens ***touchTime*** und drei verschiedene ***EvenOut*** Felder, sowie ein boolsches Feld Namens ***choice***. Der ***Script*** Knoten funktioniert folgendermaßen: Der Knoten empfängt den EventIn ***touchTime***, welcher den Zeitstempel enthält, an dem der Benutzer die Maustaste losgelassen hat. Daraufhin wird die Funktion ***touchTime*** aufgerufen. Diese überprüft, ob das Feld ***choice*** wahr ist. Trifft dies zu, wird das Feld ***choice*** unwahr gesetzt, dem EventOut ***startT*** wird der Zeitstempel zugewiesen, an dem der Benutzer die Maustaste losgelassen hat, und dem EventOut ***output*** wird der Wert 0 zugewiesen. Falls das Feld ***choice*** unwahr ist, werden im gleichen Stil die Befehle in der **else** Anweisung ausgeführt.

Hier nun ein kleines Beispiel, das die Funktion des ***Script*** Knotens näher erklären soll: Der Benutzer drückt die Maustaste auf einem Schild, welches momentan das Bild

einer Vulkanpfadtafel darstellt. Daraufhin sendet der ***touchSensor*** Knoten des Schildes einen EventOut ***touchTime***. Dieser Event wird vom ***Script*** Knoten empfangen und sorgt dafür, dass die Funktion ***touchTime*** ausgeführt wird. In dieser Funktion wird sodann ein ***EventOut*** produziert, welcher an den ***Switch*** Knoten (dieser wird im nächsten Abschnitt näher erklärt) gesendet wird und dafür sorgt, dass die Textur des Schildes in eine ***Movie Texture*** geändert wird. Außerdem wird noch ein ***EventOut*** produziert, welcher die Startzeit des Filmes setzt und somit dafür sorgt, dass der Film startet. Wenn nun der Film auf dem Schild läuft und der Benutzer drückt wiederum die Maustaste, wird die ganze Prozedur rückgängig gemacht, das heißt, es wird wieder die normale ***Image Texture*** des Schildes dargestellt und der Film wieder abgeschaltet. Dies geschieht in der ***else*** Klausel der ***touchTime*** Funktion.
Die verschiedenen Event Felder der Knoten sind untereinander mit so genannten ***route*** Anweisungen verbunden. Eine solche Anweisung sieht z.B. so aus:

```
ROUTE MyTouchSensor.touchTime TO MyScript.touchTime
```

Obige Anweisung bewirkt, dass der EventOut ***touchTime*** des ***TouchSensor*** Knotens an den EventIn ***touchTime*** des ***Script*** Knotens gesendet wird.

5.4.4 Switch Node

Der ***Switch*** Knoten in VRML ist als Gruppenknoten definiert. Dies bedeutet, er kann mehrere Unterknoten, auch Kinder genannt implementieren. Die Besonderheit des ***Switch*** Knotens besteht in seiner Eigenschaft, immer nur eines der ihm untergeordneten Kinder darzustellen. Die Kinder im ***Switch*** Knoten sind der Reihe nach von 0 bis n durchnummeriert. In unserem Fall haben wir zwei Kinder, und zwar erstens ein Schild, welches eine ***Movie Texture*** implementiert und zweitens ein Schild, welches eine ***Image Texture*** enthält. Zwischen diesen beiden Schildern kann jetzt gewechselt werden, indem man das Feld ***whichChoice*** des ***Switch*** Knotens auf den entsprechenden Wert setzt, also entweder 0 für das ***Movie Texture*** Schild oder auf 1 für das ***Image Texture*** Schild. Die Möglichkeit, dieses Feld zu setzen, schaffen wir durch eine Route Anweisung, welche folgendermaßen aussieht:

```
ROUTE MyScript.output TO SWITCH.whichChoice
```

Hier wird also der EventOut ***output*** des ***Script*** Knotens an das Feld ***whichChoice*** des ***Switch*** Knotens gebunden. Wenn jetzt also im ***Script*** Knoten dem EventOut ***output*** ein Wert zugewiesen wird, wird dieser automatisch dem Feld ***whichChoice*** des ***Switch*** Knotens zugewiesen, und somit kann zwischen den zwei verschiedenen Ausführungen der Schilder gewechselt werden. Zusätzlich zu den oben erwähnten Route Anweisungen sind noch zwei weitere notwendig, welche steuern, wann der Film gestartet bzw. gestoppt werden soll.

5.4.5 Filmformat

Standardmäßig kann VRML nur Filme im MPEG1 Format als ***Movie Texture*** darstellen. MPEG1 ist ein verhältnismäßig altes Komprimierungsformat, und deshalb sind Filme, welche in diesem Format gespeichert sind, auch entsprechend groß. Die Filme, welche vom Institut für Landschaftsökologie und Naturschutz in Singen geliefert wurden, waren im Schnitt etwa 12MB groß und daher für Online-Anwendungen nur sehr schlecht zu gebrauchen. Es wurde deshalb nun nach einer Möglichkeit gesucht, die Filme trotzdem in das Modell zu integrieren. Zuerst wurde versucht, die Auflösung der Filme zu reduzieren und gleichzeitig auch deren Framerate herunterzusetzen, um die Dateigröße zu verringern. Jedoch setzt der MPEG1 Standard hier Grenzen. So kann z.B. die Framerate nicht beliebig reduziert werden, sie muss mindestens 10 Frames pro Sekunde betragen. Auch die Auflösung des Videos muss bestimmten Kriterien entsprechen, und so war es leider nicht möglich, die Datenmenge der Filmdateien auf ein vernünftiges Maß für Online-Anwendungen zu reduzieren.

Schließlich wurde die die Möglichkeit gefunden, die Filme in das Real Media Format zu konvertieren, welches eine sehr kompaktes Komprimierungsformat darstellt. Diese Umwandlung der Filmdaten wurde in dem von Real Media kostenlos zur Verfügung gestellten Tool Real Producer Basic [REALPRODUCER] vorgenommen. Das Programm erlaubt es, verschiedene Einstellungen zur Qualität des Videos, der Qualität der integrierten Audiospur und auch der Bildwiederholungsrate (Framerate). Mit diesem Programm war es möglich, die Videodateien auf durchschnittlich ca. 400 KB zu verkleinern. Natürlich mussten dafür starke Einbußen in der Darstellungsqualität

hingenommen werden, und die Videos laufen aufgrund der verringerten Bildwiederholungsrate nicht mehr ganz flüssig ab. Jedoch können diese Filme nun aufgrund ihrer relativ geringen Größe auch über das Internet dargestellt werden und somit in das Modell des Hohentwiel integriert werden.

Filme in Real Media Format abzuspielen gehört nicht zur Standard VRML97 Spezifikation. Jedoch haben verschiedene Hersteller wie z.B. Blaxxun und Parallel Graphics diese Möglichkeit in ihre VRML Browser integriert. Ältere Browser jedoch, welche nicht mehr weiterentwickelt werden, haben diese Erweiterungen nicht und können deshalb solche Videodaten auch nicht abspielen.

Insgesamt wurden sechs Filme ins Real Media Format übertragen, und wie in den vorhergehenden Abschnitten beschrieben in die entsprechenden Vulkanpfadschilder integriert. Im Ganzen wurden 13 Schilder in das Hohentwiel Modell eingesetzt, sechs Schilder mit Videofunktionalität und weitere sieben Schilder, welche nur den Tafelinhalt darstellen.

5.4.6 Positionierung der Schilder

Die Schilder wurden nun mit Hilfe einer Karte, auf welcher der genaue Verlauf des Vulkanpfads und dessen Schilder aufgezeichnet sind, in das 3D Modell des Hohentwiel integriert. Hierzu ist unbedingt ein Tool wie z.B. das schon vorher erwähnte Cosmo Worlds erforderlich. Es ist zwar möglich, die genauen Koordinaten, an denen die Schilder stehen sollen, per „Try and Error“ Methode festzustellen. Dies erfordert aber unendliche Zeitreserven, da die Position immer in einem Texteditor gesetzt werden und danach in einem VMRL Browser überprüft werden muss. Sehr viel einfacher geht dieser Schritt mit einem Programm wie Cosmo Worlds vonstatten, welches das komplette 3D Modell darstellt und dann die Möglichkeit bietet, Objekte wie z.B. unsere Schilder einfach und sehr genau zu platzieren. Wie schon einmal weiter oben im Text erwähnt wurden die Schilder mittels ***Inline*** Knoten in das Modell integriert. Das heißt, jedes der Schilder ist eine eigenständige VRML Datei, welche in das 3D Modell des Hohentwiel geladen wird. Dies hat den Vorteil, dass die Schilder einfach geändert werden können ohne den Inhalt des 3D Modells ändern zu müssen.

5.4.7 Viewpoints

Viewpoints werden in VRML verwendet, um dem Benutzer des Modells vordefinierte Sichten auf die Welt, welche aus einer Liste im VRML Browser ausgewählt werden können, zur Verfügung zu stellen. Viewpoints werden auch oft mit Kameras verglichen, welche an bestimmten Koordinaten in der virtuellen Welt aufgestellt werden und die Sicht auf ein bestimmtes Objekt, bzw. in eine bestimmte Richtung bieten. In dem Modell des Hohentwiel sollen solche Viewpoints für jedes der integrierten Vulkanpfadschilder angeboten werden, so dass der Benutzer auf einfache Art und Weise zu den Schildern navigieren kann. Der Viewpoint Knoten in VRML hat folgenden Aufbau:

```
ViewPoint {
      fieldOfView 0.785398
      position 0 0 10
      orientation 0 0 1 0.785398
      description "MyViewPoint"
      jump TRUE
}
```

Das Feld ***fieldOfView*** beschreibt den Sichtwinkel des Benutzers und wird als Winkel im Bogenmaß angegeben. Kleine Winkel entsprechen in etwa einem Teleobjektiv bei einer Kamera, größere Winkel entsprechen einem Weitwinkelobjektiv. ***Position*** ist ein 3D Vektor und beschreibt die Koordinaten in der virtuellen Welt, an denen sich der Viewpoint befindet. Das Feld ***orientation*** gibt die Blickrichtung des Viewpoints an. Die ersten drei Werte stehen für die xyz Achsen und der vierte Wert ist ein Winkel im Bogenmaß, der bestimmt, um wie viel Grad der Viewpoint rotiert werden soll. Im obigen Beispiel würde der Viewpoint um 90 Grad relativ zur z-Achse rotiert. ***Description*** enthält die Bezeichnung des Viewpoint, welche in der Auswahlliste des VRML Browsers angezeigt wird. Das Feld ***jump*** ist ein boolsches Feld und legt fest, ob der Benutzer beim Wechseln zwischen verschiedenen Viewpoints auch gleichzeitig seine Position im Modell wechselt. Das heißt, wenn ***jump*** wahr ist, wird beim Ändern des Viewpoints die Position des Benutzers auch auf die Position des Viewpoints gewechselt. Ist ***jump*** unwahr, bleibt der Benutzer an seiner alten Position.

Für das Einfügen der Viewpoints in das 3D Modell wurde wiederum das Programm Cosmo Worlds verwendet, da es wie schon weiter oben im Text erwähnt die Möglichkeit bietet, Objekte (und so auch Viewpoints) auf einfache Art und Weise sehr genau im Modell zu platzieren.

5.5 Animation

Das Modell soll die Möglichkeit bieten, dem Benutzer ohne dessen Zutun alle interessanten Stellen in der Visualisierung des Hohentwiel vorzustellen. Der Benutzer soll zwischen einem Rundgang, welcher ihn auf dem Vulkanpfad um den Berg führt, oder aber einem Rundflug um den Berg auswählen können.

Eine Möglichkeit, dies zu realisieren, ist durch animierte Viewpoints. Man kann sich das so vorstellen, dass man mit einer Kamera den kompletten Vulkanpfad abläuft, und danach die Route, welche die Kamera beschritten hat, abgespeichert. Diese Route kann daraufhin animiert werden, d.h. die Kamera bewegt sich dann automatisch auf der vorgegebenen Route und zeigt somit dem Benutzer alles Sehenswerte. Auch für den Rundflug wird eine Route festgelegt, auf welcher sich die Kamera selbständig bewegen kann.

Die Animationen wurden wiederum mit dem Werkzeug Cosmo Worlds erstellt. Dieses Programm beinhaltet einen so genannten Keyframe Animator, mit dem Objekte auf einfache Weise animiert werden können. Der Keyframe Animator arbeitet folgendermaßen: Zu Beginn muss festgelegt werden, wie lange die Animation dauern soll. Danach muss das Objekt, welches animiert werden soll, ausgewählt werden– in unserem Fall ist das ein Viewpoint (eine Kamera). Die Kamera muss dann in eine Startposition gebracht werden. Danach kann man die Kamera an eine beliebige Position bewegen und auf der Zeitskala einen Keyframe abspeichern. Man bewegt so die Kamera entlang einer fiktiven Route oder aber im Falle der Vulkanpfad Route entlang des Vulkanpfads und speichert an mehreren Stellen einen Keyframe ab. Cosmo Worlds berechnet dann aus diesen Keyframes die Animation, das heißt, wie sich die Kamera später automatisch bewegt. Je mehr Keyframes gesetzt werden, umso genauer wird die Animation. Folgendes Beispiel soll zeigen, wie die ersten paar Sekunden der Vulkanpfad Tour in dem Programm Cosmo Worlds realisiert werden.

Zuerst muss das komplette Modell des Hohentwiel in Cosmo Worlds geöffnet werden. Dann wird eine neuer Viewpoint (Kamera) zu dem Modell hinzugefügt, welcher ***Vulkanpfad Tour*** genannt wird. Die Kamera wird dann im Modell so platziert, dass das erste Schild des Vulkanpfads im Blickfeld der Kamera ist. Nun wird auf der Zeitskala des Keyframe Animators am Zeitpunkt 0.0 ein erster Keyframe gesetzt und somit der Anfang der Animation festgelegt. Siehe Abbildung:

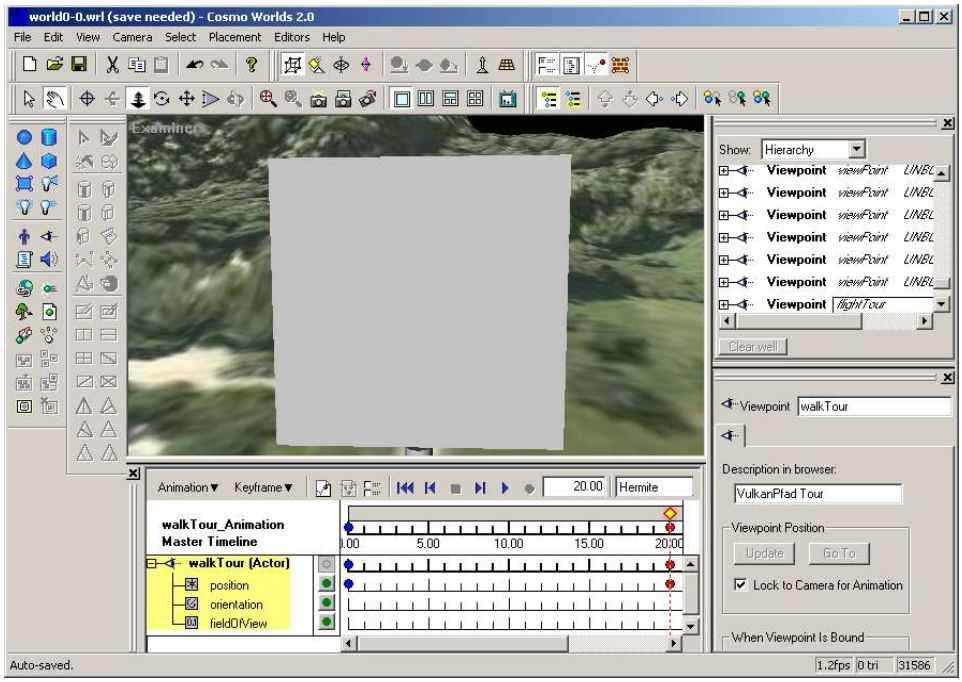


Figure 18 Animation in Cosmo Worlds

Danach soll die Kamera zwanzig Sekunden in dieser Position verbleiben, um dem Benutzer das Lesen des Schildes zu ermöglichen. Es wird deshalb ein zweiter Keyframe am Zeitpunkt 20 gesetzt und dem Programm mitgeteilt, dass zwischen den zwei Keyframes keine Bewegung stattfinden soll. Danach wird die Kamera über mehrere Haltestationen zum nächsten Schild bewegt. An jeder Haltestation wird ein Keyframe abgespeichert. Cosmo Worlds kann aus diesen Informationen eine Animation erstellen, bei der die Kamera auf direktem Wege entlang der Route, welche die Keyframes beschreiben, bewegt wird. Auf direktem Wege bedeutet, dass die Kamera den kürzesten Weg zwischen zwei Keyframes wählt. Dies heißt wiederum, dass an Stellen, an denen der Vulkanpfad eine Kurve macht, auf jeden Fall ein Keyframe gesetzt werden muss, da sich die Kamera sonst nicht entlang des Weges bewegt.

Auf diese Art und Weise wurden zwei Animationen in das Modell des Hohentwiel integriert, welche in einem VRML Browser durch einfaches Anwählen im Viewpoint Menü gestartet werden können.

6. Schlußbetrachtung

In dieser Ausarbeitung wurden Techniken und Verfahren zur interaktiven Visualisierung von Landschaften im Internet untersucht und bewertet. Es hat sich herausgestellt, dass sich der VRML97 Standard, obwohl er schon einige Jahre alt ist, momentan am besten dazu eignet, virtuelle Landschaften zu implementieren. Es wurde weiter detailliert beschrieben, wie das 3D Modell des Hohentwiel mit Hilfe von VRML97 implementiert wurde.

6.1 Erreichen der Zielsetzungen

Ziel dieser Diplomarbeit war es, ein Verfahren zu finden, welches unter den Kriterien von Preis, Handhabung und Flexibilität am besten dazu geeignet ist, die Landschaft des Hohentwiel im Internet zu visualisieren. Dieses Ziel wurde durch die Wahl des kostenlosen VRML97 Standards erreicht.

Ein weiteres Ziel war es, die Implementierung der virtuellen Landschaft anhand der gefundenen Methode detailliert aufzuzeigen. Auch dieses Ziel wurde durch die konsequente Beschreibung einzelner Implementierungsphasen erreicht.

Die Anforderung, dass das System auch auf einem älteren Rechner mit langsamer Internetverbindung lauffähig sein sollte, konnte nicht vollständig umgesetzt werden. Trotz aller vorgenommenen Optimierungen braucht die Übertragung der Daten über eine Modemverbindung länger als einem Benutzer zuzumuten wäre. Auch ist das System auf einem Rechner der ersten Pentium Generation (PI 200) zwar lauffähig, jedoch nur sehr langsam. Somit wurden die Zielsetzungen im Bezug auf die Performance leider nicht erreicht.

Das Ziel, ein konsequent benutzerfreundliches System zu implementieren, wurde fast vollständig umgesetzt. Durch die Wahl von Blaxxun Contact als VRML Browser ist es möglich, das benötigte VRML Plugin halbautomatisch zu installieren. Des weiteren verkürzt sich bei der Benutzung dieses Browsers die Startzeit der Anwendung verglichen mit anderen Browsern erheblich. Jedoch hat dieser Browser den Nachteil,

dass sich kein permanentes Menü einblenden lässt, welches speziell für unerfahrene Benutzer von Vorteil gewesen wäre.

6.2 Ausblick

In den letzen Wochen vor der Fertigstellung dieser Ausarbeitung wurde eine Java3D Terrain Diskussionsliste [J3DTERRAIN] ins Leben gerufen. Dies ist eine Mailingliste, welche sich speziell mit der Implementierung von großen Landschaftsausschnitten mit Java3D befasst. Hier werden unter anderem Algorithmen zur Optimierung von Terrain diskutiert, und es wurde auch schon teilweise eine Codebasis mit Beispielimplementierungen aufgebaut.

Unter der Zusammenarbeit verschiedener Programmierer wurde im Rahmen dieser Mailingliste eine Basisimplemntierung des ROAM Algorithmus in Java3D zur Verfügung gestellt. ROAM steht für „Real-time Optimally Adapting Meshes“ und ist ein Algorithmus, welcher in Echtzeit Terrain Informationen so optimiert, dass die Relation zwischen Darstellungsqualität und Anzahl der zur Darstellung benötigten Dreiecke optimal wird. ROAM verfolgt einen ähnlichen Ansatz wie das in dieser Diplomarbeit verwendete Level of Detail Management (LOD). Es versucht die Anzahl der zu berechnenden Dreiecke zu reduzieren und somit den Rechenaufwand für ein Modell zu minimieren.

Der hauptsächliche Unterschied zwischen ROAM und dem in dieser Diplomarbeit verwendeten LOD Ansatz ist, dass ROAM die Terraininformationen zur Laufzeit des Programms optimiert, während LOD eine statische Optimierung darstellt. Somit hat ROAM den Vorteil, dass die Detailgenauigkeit des Terrains dynamisch, dass heißt zur Laufzeit des Programms eingestellt werden kann. Dies bedeutet wiederum, dass z.B. der Benutzer einer 3D Welt die Auflösung des Terrains in der Anwendung einstellen kann, und der Terrain somit optimal an die jeweilige Hardware Umgebung angepasst werden kann.

In Ihrem jetzigen Stadium kann die Java3D ROAM Implementierung noch nicht in Verbindung mit dem VRML97/X3D Loader für Java3D verwendet werden. Jedoch

soll dieser Loader in den nächsten Monaten so angepasst werden, dass er die ROAM Implementierung unterstützt.

Somit wäre die Möglichkeit gegeben, VRML Elevation Grid Daten in Java 3D zu laden und durch den ROAM Algorithmus zu optimieren.

Deshalb, und auch wegen der in der neuen JDK Version 1.4 eingeführten Java WebStart Technologie, welche die Benutzung von Java-Anwendungnen über das Internet entscheidend verbessert, sehe ich in Java3D in Verbindung mit VRML/X3D eine interessante Alternative zu „purem“ VRML.

7. Quellenverzeichnis

7.1 Internetquellen

Floppy's VRML97 Tutorial	http://web3d.vapourtech.com/
Java3D Mailing Liste	http://java.sun.com/products/java-media/3D/forDevelopers/interest_group.html
VRML Knoten Referenz	http://xarch.tu-graz.ac.at/publ/tutorial/vrml/lighthouse/toc.html
VRML97 Standard	http://www.web3d.org/Specifications/VRML97/

7.2 Benutzte Tools

[COSMOWORLD]	http://www.sgi.com/software/cosmo/worlds.html
[J3DTERRAIN]	http://www.j3d.org/mailing_lists.html
[PAINTSHOP]	http://www.jasc.com/
[REALPRODUCER]	http://www.realnetworks.com/products/producer/basic.html
[REZTOOLS]	http://www.surak.com.au/~chris/vrml/RezIndex.html
[ULTRAEDIT]	http://www.ultraedit.com/