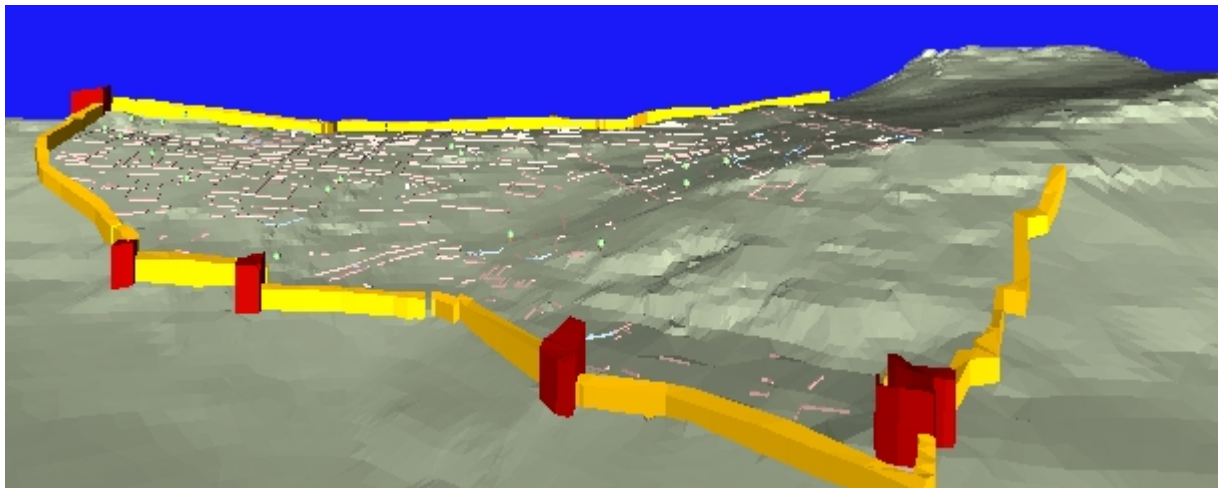


Vertiefungsblock K7 3D-Visualisierung einer archäologischen Ausgrabungsstätte

VTB K7 WS 2000/01 D-BAUG / Kult-Ing
Semesterarbeit - Technischer Bericht



Michael Boller
Martin Fopp

9. Februar 2001

Vorwort

Im Rahmen des Vertiefungsblockes K7 durften wir mit der Erstellung einer 3D-Visualisierung einer archäologischen Ausgrabungsstätte eine interessante und lehrreiche Aufgabe bearbeiten.

Der vorliegende Bericht richtet sich an ein Fachpublikum. Für das Verständnis sind Kenntnisse in JavaScript, VBScript und VRML hilfreich. Für allfällige Fragen zum bearbeiteten Thema stehen wir jederzeit gerne zur Verfügung. Des Weiteren sind wir gerne bereit interessierten Personen unser Resultat zu demonstrieren. Die Webseite kann auch unter <http://www.karto.ethz.ch/teaching/studentwork.html>, dann den entsprechenden VTB wählen, ausprobiert werden.

Im Verlauf unserer Arbeit konnten wir auf die Unterstützung von zahlreichen Personen zählen. Für die vielen guten Tipps und Ratschläge möchten wir ihnen ganz herzlich danken. Ein spezieller Dank gilt an dieser Stelle insbesondere folgenden Personen:

- **Prof. Dr. Lorenz Hurni** (IKA), dass er die Schirmherrschaft für den Vertiefungsblock am IKA übernahm
- **Christoph Brandenberger** (IKA), für die Organisation der Semesterarbeit und die gute Unterstützung bei der Bearbeitung.
- **Andreas Neumann** (IKA), für die ausgezeichnete und geduldige Betreuung, sowie für die Mitarbeit
- **Uta Samaga** (IKA / TU Dresden), für die Vorarbeit, Mithilfe und die vielen Informationen
- **unseren MitstudentInnen** , für die gute Zusammenarbeit

Die Verfasser

Michael Boller
bollermi@student.ethz.ch

Martin Fopp
mfopp@student.ethz.ch

Zusammenfassung

Die Aufgabe, die im Rahmen des Vertiefungsblockes K7 an der Professur für Kartographie gestellt wird, umfasst die Auslotung der Möglichkeiten zur Darstellung einer archäologischen Ausgrabungsstätte in 3D mittels VRML. Dazu stehen digitale Daten der antiken Akropolis von Palairos bei Stratos in Westgriechenland zur Verfügung. Diese Stätte wird vom griechischen Kultusministerium in Kooperation mit dem Deutschen Archäologischen Institut (DAI) ausgegraben und terrestrisch vermessen. Von ihnen stammen auch die Daten, das sind DXF-Dateien und eine ausgedruckte Karte. Zusätzlich steht auch noch eine Diplomarbeit von Niels Oberhauser (FH Oldenburg & DAI) zur Verfügung.

Die Daten werden in 3D Studio Max R3.1 importiert und weiterbearbeitet. So wird das importierte TIN je nach Wichtigkeit des Gebietes in der Auflösung angepasst. Solche Massnahmen werden getroffen, damit die Anforderungen an Browser und Hardware in einem sinnvollen Mass bleiben. Des Weiteren wird die Stadtmauer modelliert und Funde mit Symbolen versehen, sowie die Grundrisse der aufgenommenen Strukturen von antiken und mittelalterlichen Überresten extrudiert. Zusätzlich werden noch Kameras gesetzt. Nach dem Export aus 3D Studio Max in eine VRML-Datei, wird diese in den beiden Texteditoren VrmIpad und UltraEdit32 weiterbearbeitet und vor allem der Code optimiert. Das heisst es wird die VRML-Eigenschaft genutzt einmal definierte Objekte, darunter fallen auch definierte Farben eines Objektes, per USE-Befehl beliebig häufig wiederzuverwenden. Es werden auch noch Touchsensoren und Skripte, wobei hier ein neuer Prototype erstellt wird, eingebaut. Diese werden für die verschiedenen Interaktionen genutzt.

All dies wird in einer ersten Phase für den Internet Explorer 5+ und den Cortona Player designt. In einer nächsten Phase, die aus Zeitgründen nicht mehr realisiert werden konnte, stände die Portierung auf andere Browser und Plugins. In einem letzten Schritt ist die Webseite gestaltet worden. Diese erlaubt Interaktionen mit der VRML-Datei. Dazu werden die beiden Programmiersprachen JavaScript und VBScript genutzt.

Das Resultat ist eine Webseite, die die archäologische Ausgrabungsstätte Palairos einerseits in 3D präsentiert und andererseits diese mittels Interaktionen informativer gestaltet. So kann man per Knopfdruck antike bzw. mittelalterliche Überreste farblich hervorheben lassen. Eine weitere Funktion ist, dass beim Darüberfahren mit der Maus über die verschiedenen Symbole der Fundstücke dynamisch eine Legende in den Browser ausgegeben wird. Des Weiteren besteht die Möglichkeit per Mausclick sich weitere Informationen anzeigen zu lassen, dies konnte aus Materialgründen leider nur für die Stadtmauer realisiert werden. Zur besseren Orientierung in der VRML-Szene gibt es eine Karte, auf der die aktuelle Position als roter Punkt dargestellt wird. Zu guter Letzt besteht noch die Möglichkeit eines manuellen Rundgangs über vordefinierte Aussichtspunkte zu machen.

Inhaltsverzeichnis

	Vorwort.....	1
	Zusammenfassung.....	3
1	Situationsanalyse.....	7
1.1	Einleitung.....	7
1.1.1	Historische Stutte Palairos.....	8
1.1.2	Topographie.....	8
1.1.3	Palairos.....	9
1.2	Auftrag.....	10
1.3	Ausgangsdaten.....	11
1.4	Techniken und Werkzeuge.....	11
2	Datenverarbeitung.....	13
2.1	Vorarbeiten und Bereitstellung der Daten.....	13
2.2	Visualisierung und Bearbeitung in 3D Studio Max.....	15
2.2.1	Import.....	15
2.2.2	Ebenen-/Gruppenkonzept.....	16
2.2.3	Gelandemodellierung.....	17
2.2.4	Gebaude-/Mauermodellierung.....	17
2.2.5	Felsen.....	18
2.2.6	Treppen.....	19
2.2.7	Funde.....	21
2.2.8	Vegetationsdarstellung.....	21
2.2.9	Gewasser.....	21
2.3	Bearbeitung VRML.....	21
2.3.1	Export nach VRML.....	21
2.3.2	Bearbeitung in VrmIPad.....	22
2.3.3	Bearbeitung in UltraEdit32.....	26
2.4	Progammierung der interaktiven Elemente der Webseite.....	26
2.4.1	Antike uberreste hervorheben.....	27
2.4.2	Mittelalterliche uberreste hervorheben.....	27
2.4.3	Stadtmauer aus-/einblenden.....	28
2.4.4	Ansichten wechseln.....	29
2.4.5	Positionsanzeige.....	30
2.4.6	Legende zu den Fundstucken.....	31
2.4.7	Information zu einem Objekt.....	31
3	Technische Grundlagen.....	32
3.1	Einbinden von Cortona in HTML Dokumente.....	32
4	Prasentation der Daten.....	33
4.1	Bedienungsanleitung und Benutzerfuhrung.....	33
4.1.1	Bedienungselemente.....	33
4.1.2	Navigieren in Cortona.....	33
4.1.3	Optionen im Cortona Plugin.....	34
4.1.4	Interaktivitat mit der Welt.....	35
4.2	Benutzerfuhrung auf der Homepage.....	35
	Schlusswort.....	37
	Quellenverzeichnis.....	39
	Abbildungsverzeichnis.....	41
	Abkurzungen.....	43
	Anhang.....	45

1 Situationsanalyse

1.1 Einleitung

Akarnanien – eine Landschaft in Nordwestgriechenland, ein gebirgiges Gebiet mit buchtenreichen Küsten, kulturell gleichermaßen von Ägäis und Adria beeinflusst - hat der Altertumswissenschaft viel zu bieten¹.

Schon in den Jahren 1896 bis 1902 begann der Archäologe Ferdinand Noack, die Ruinenstädte Westgriechenlands topographisch zu erfassen. Er fertigte auch erstmals topographische Pläne der antiken Stätte Palairos.

Das Deutsche Archäologische Institut (DAI), mit seinen derzeit ca. 100 Fachwissenschaftlern und Hauptsitz in Berlin, arbeitet seit 1989 mit der 6. Ephorie für prähistorische und klassische Altertümer in Patras zusammen. Seit dieser Zeit dauern auch die archäologischen Regionalforschungen zur wissenschaftlichen Erschließung der weitgehend unbekannt westgriechischen Landschaft Akarnanien an.

Diese Forschungen basieren auf der Initiative des DAI, Athenische Abteilung, vom Ende des 19. Jahrhunderts. Mit der Wiederaufnahme dieser Initiative, unter anderem auch in Palairos, soll eine umfangreiche Bestandsaufnahme aller historischer Monumente und die Erforschung der Änderung der Siedlungsstruktur von der Vorgeschichte bis zur Moderne erreicht werden².



Abbildung 1.1. Übersichtskarte, Lage von Palairos

Durch die Umsiedlung der Bewohner in die Siegesstadt Nikopolis war Palairos nie zerstört oder geplündert worden. Aus diesem Grund erlangte Palairos für die Archäologen besondere Bedeutung. Die Wahrscheinlichkeit fundierter Forschungsergebnisse in Bezug auf Siedlungsstruktur, Architektur und Sozialstruktur ist im Vergleich zu anderen Ausgrabungsstätten höher.

In Palairos wird neben einer lückenlosen Oberflächenbegehung, Luftbildauswertungen, archäologischen Planaufnahmen, Sondagen und Flächenausgrabungen sowie die Auswertung überlieferter Reiseberichte und Dokumente, auch die geodätische Geländevermessung zur Analyse herangezogen. Die geodätischen Geländevermessungen haben im Frühjahr 1997 begonnen und wurden im Rahmen der Messkampagnen 1998 und 1999 weiter vervollständigt.

Ein wesentlicher Teil der Publikation von Forschungsergebnissen ist vor allem die Aufarbeitung und Dokumentation der geodätischen Arbeiten, die der Geodät im Rahmen seines Aufgabengebietes umfassend bereitstellen kann. Sie beinhaltet eine topographische Aufnahme der antiken Stätte Palairos unter Erfassung detaillierter Darstellungen der Geländeformen, Bauwerke, Quellen, markanter Bäume und Besonderheiten des Gebietes und dessen Dokumentation in einer 3D-Darstellung und Webseite.

Die 3D-Darstellung soll hauptsächlich die Geländeformen visualisieren. Sie wirkt unterstützend bei der Dokumentation der Inhalte der topographischen Karte und ermöglicht archäologische Befunde auf Grund der Oberflä-

1. Akarnanien, Oberhammer - Gesellschaft e. V. München, S.13, Bertold, Schmid, Wacker

2. Internet: <http://www.dainst.de/de/var/aufgaben.html>

chenformen besser zu erkennen. Des Weiteren bildet die 3D-Darstellung die Grundlage für vielseitige Weiterverarbeitungsmöglichkeiten.

Die wissenschaftliche Disziplin der Archäologie leidet unter ihrem schlechten Image, da die Wissenschaft und deren Publikationen fast überwiegend Fachkreisen vorbehalten bleiben. So wird mit dem Internet ein neues Medium genutzt, um die Ergebnisse einer breiteren Öffentlichkeit wissenschaftlich und medienwirksam zu vermitteln. Die Darstellung der Ergebnisse dieser Arbeit im Internet soll den Archäologen als Grundlage für spätere Publikationen über die antike Stätte Palairos im Internet dienen.

Aus geodätischer Sicht geht es hier primär um die Verwirklichung fundierter Inhalte der Kartenwerke und nur sekundär um die dafür notwendigen vermessungstechnischen Voraussetzungen. Die Kommunikation zwischen Archäologen und Geodäten sowie die Einarbeitung des Geodäten in archäologische Thematiken bilden die Basis zur Realisierung fundierter, aussagekräftiger Kartenwerke.

1.1.1 Historische Stätte Palairos

Schon im Verlauf des 19. Jahrhunderts beschrieben mehrere Reiseschriftsteller die gut erhaltenen Reste der antiken Polis Palairos etwas nordöstlich von Kechropoula und deren nähere Umgebung. Neben den schriftlichen Überlieferungen spielen Funde, Bodenuntersuchungen, Interpretationen der Topographie und Mauerstrukturen sowie alte Kartenwerke eine wichtige Rolle. Die daraus gewonnenen Erkenntnisse können einen umfassenden Einblick in die historische Entwicklung von Palairos geben.

1.1.2 Topographie

An der Westküste Griechenlands, östlich der Insel Lefkada, befindet sich auf einem markanten Ausläufer des Gebirges der Plagia-Halbinsel innerhalb der grossen Senke von Zaverda die antike Stätte Palairos.

Das Siedlungsplateau der Stätte liegt in einem flachen Sattel, dessen östlicher Hang in die 240 m hohe Akropolis überleitet. Der westliche Hang steigt kontinuierlich zum Berg Prophetis Ilias mit dem Stadtheiligtum an. Nach Norden erstreckt sich ein steil abfallender Hang bis zum Voulkariasee, der heute noch zur Bewässerung der mehrheitlich aus Tabakanpflanzungen bestehenden Felder dient. Südlich verläuft sich der Hang in die fruchtbare Ebene von Zaverda.

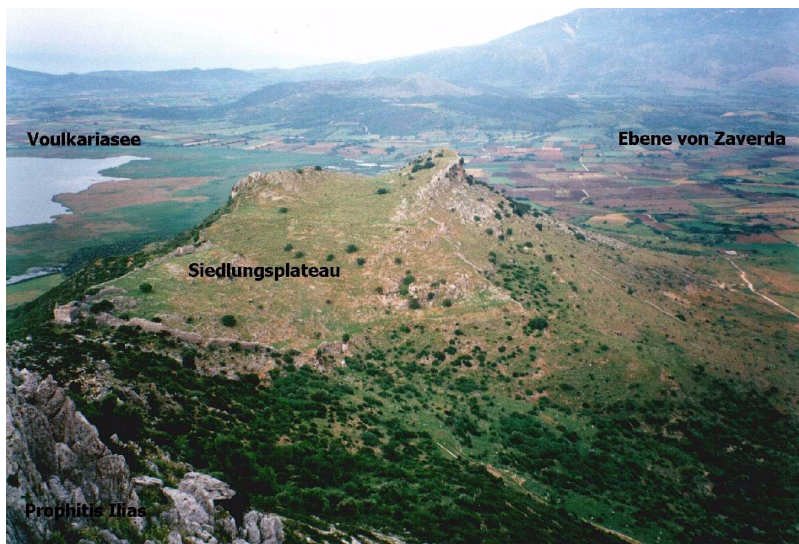


Abbildung 1.2. Siedlungsplateau und Südhang der antiken Stätte Palairos

Das Gelände, in steileren Regionen von Felsen zerschnitten, ist überwiegend mit fast undurchdringlichen Hecken, Bäumen und niedrigen Sträuchern bewachsen und dient zur Weidung der Schafe, Kühe und Ziegen der in der Umgebung angesiedelten Bauern. Durch diese Nutzung wird zum Leidwesen der Archäologen zahlreiches historisch, archäologisch wertvolles Material zerstört. Die Ebene von Zaverda ist von kleinen Feldwegen durchschnitten und erst in jüngster Zeit wurde ein Weg bis hin zur antiken Stadt angelegt. Der gesamte Stadtbereich einschliesslich Nekropolen³ ist von Terrassen durchzogen. Hierbei wurden nicht nur natürlich vorkommende Terrassen, sondern häufig auch künstlich geschaffene Terrassen zur Bebauung genutzt. Auch sind Strassenzüge parallel zu den Höhenlinienverläufen zu erkennen. Südlich des Proastion⁴ ist der Hang von zwei tiefen Gräben

3. Friedhof

durchzogen, die teilweise auch eingefasst waren. Sie dienten neben den Zisternen vermutlich der Wasserversorgung; auch der Graben im Tal zwischen Kechropula und Nekropole war vermutlich ein schon in der Antike genutzter Wasserlauf.

1.1.3 Palairos

Der Name der Stadt Palairos kann nicht aus dem griechischen abgeleitet werden und wird von E. Kirsten als illyrisch bezeichnet. Das Ethnikon ‚Palaireus‘ wird bei Stephanos von Byzanz genannt und durch eine Inschrift aus Ruga bestätigt. Die Existenz einer Stadt Palairos wird in Bezug auf die Lage ebenfalls schriftlich überliefert.



Abbildung 1.3. Historische Lage von Palairos

Strabon erwähnt die Polis an zwei Stellen, welche die Lokalisierung bei Kechropula sehr wahrscheinlich machen. In seinem zehnten Buch wird Palairos neben Alyzeia, Leukas und anderen Poleis als akarnanische Stadt aufgezählt, was einige Kapitel weiter noch präzisiert wird. Dort beschreibt Strabon, dass der Salzsee Myrtuntion, der mit dem heutigen Voulkariasee gleichzusetzen ist, zwischen Leukas und dem Ambrakischen Golf läge und man von Leukas aus als nächstes nach Palairos und Alyzeia gelangen könne. Eine weitere Bestätigung für die Identifizierung von Palairos mit Kechropula bieten zwei Inschriften, die in der Oberstadt an der Strasse als Basen für Ehrenmonumente aufgestellt waren.

Die antike Stätte Palairos erstreckt sich vom Siedlungsplateau bis weit hinter Kechropula, was neueste Untersuchungen bestätigen. Anhand umfangreicher Oberflächenfunde und sichtbaren Mauerbeständen aus mehreren Zeitepochen kann die Stadt Palairos in Kastro, Akropolis, Proastion und die Nekropolen aufgeteilt werden. Die Kastro⁵ ist von einer gut sichtbaren, hohen Stadtmauer umgeben. Sie umschliesst die noch erkennbaren Strassenverläufe, Grundmauern von Wohnhäusern, Zisternen, öffentliche administrative Gebäude und Plätze sowie die Agora⁶ mit ihren vielfältigen Ehrenmonumenten. Anschliessend an die Oberstadt, durch eine massive Mauer getrennt, liegt die Akropolis. Sie wurde vermutlich ausschliesslich als Fluchtburg in Gefahrensituationen genutzt und weist ebenfalls Spuren von Bebauung auf. Zahlreiche prunkvoll gearbeitete Tore in der Stadtmauer belegen die Erschliessung der umliegenden Gebiete. Im Norden am Fusse des Berges vermuten die Archäologen den Hafen von Palairos, welcher sich an den Ufern des Voulkariasees befunden haben muss, als dieser noch Zugang zum Meer besass. Auf dem Weg dorthin befindet sich die nördliche Nekropole. Ausserhalb der südlichen Stadtmauer beginnt die Proastion. Auffallend sind verschiedene aufwendig errichtete Stützterrassen, die den Hang durchschneiden und ebene Flächen für Gebäude und Grabterrassen schaffen. Diese sind immer wieder durch Aufgänge miteinander verbunden. Vermutlich wurden sie in früheren Analysen mit Resten der fehlenden Stadtmauer verwechselt, die sich nur im Osten zur Ebene von Zaverda befindet und mit Beginn der Südnekropole ohne erkennbaren Abschluss endet. Das südliche Tor der Oberstadt erschliesst den Zugang zur Proastion im Osten. Der Verlauf der Hauptstrasse, die sich vermutlich in Serpentinaen in die Ebene von Zaverda schlängelt, ist bis heute nicht genau rekonstruiert und basiert auf Theorien. Der westliche Teil der Unterstadt ist geprägt von Mauerfragmenten verschiedener Zeitepochen sowie erkennbaren Strassen und einigen Quellen, welche neben den Zisternen der Kastro die Wasserversorgung von Palairos sicherstellten. Genauere und neue Erkenntnisse über die Siedlungsstruktur der Unterstadt erhofft man sich durch die Neuvermessung dieses Gebietes.

4. Unterstadt

5. Oberstadt

6. öffentlicher Platz

Mit dem Ende der östlichen Stadtmauer beginnt die südliche Nekropole. Charakteristisch für jede antike Stadt erstreckt sie sich ausserhalb des Innenstadtbereiches entlang der Hauptausfallstrassen und um das südlich liegende Dorf Kechropula. In diesem Bereich finden sich neben aufwendigen, auf Stützterrassen errichteten Familiengräbern, zahlreiche Reste von Pithos- und Kammergräbern, die überwiegend von Grabräubern zerstört sind.

Die nördliche Nekropole säumt eine Strasse, die sanft abwärts in die Bucht des Voulkariasees führte und die im Gegensatz zum südlichen Friedhof erst ab dem 4. Jahrhundert vor Christus intensiver belegt wurde. Bis auf wenige Ausnahmen stammen die Grabinschriften aus klassischer und hellinistischer Zeit. An die südliche Nekropole anschliessend auf einem kleinen Hügel liegt das mittelalterliche Dorf Kechropula. Die überwiegend mittelalterliche und moderne Bebauung liessen bisher eine Nutzung in der Antike nicht vermuten, doch vereinzelt Funde und Mauern sowie Grabenkanalisierungen, welche eindeutig aus antiker Zeit stammen, lassen einen antiken Ursprung vermuten.

Die umfangreiche detaillierte topographische Erfassung dient als Grundlage weiterer wissenschaftlich archäologischer Forschungen, sowie der Sicherung von Befunden, die im Zuge der heutigen Gebietsnutzung der dort ansässigen Bevölkerung immer mehr zerstört werden.

1.2 Auftrag

Ausgangslage

Im Rahmen einer Semesterarbeit am Institut für Kartographie der ETH Zürich soll für das Gebiet Palairos in eine digitale kartenverwandte Darstellung umgesetzt werden. Für die Visualisierung und Präsentation im Internet wird die VRML verwendet. VRML gibt dem Programmierer die Möglichkeiten der dreidimensionalen Modellierung einer virtuellen Welt, mit zahlreichen Möglichkeiten der Visualisierung und Umsetzung digitaler Daten.

Produkt der Arbeit

Es soll ein Internetauftritt realisiert werden, der dem Besucher ermöglicht, sich frei im Gelände der Ausgrabungsstätte Palairos zu bewegen und diese zu erkunden.

Ziel und Zweck des Berichts

Dieser technische Bericht soll die Hintergrundinformationen für den interessierten Leser bieten. Die Entstehung sowie die technischen Aspekte unserer Arbeit werden im Detail erläutert:

- Einführung und historischer Überblick der Ausgrabungsstätte Palairos
- Technische Umsetzung in VRML und Gestaltung einer interaktiven Webseite. Beschreibung der Vorgehensweise, sowie der entstandenen Probleme.

Zeitplan

Der grobe Zeitplan wurde von der Assistenz IKA ab- und vorgegeben. Die Terminplanung ist den Studierenden überlassen worden und mündete in einem verfeinerten Zeitplan⁷.

Übungsbestimmungen und Betreuung

Bei der Erarbeitung des technischen Berichts ist systematisch vorzugehen. Die Betreuung am Institut erfolgt durch folgende Mitarbeiter.

- Christoph Brandenberger, 01/ 633 30 32, brandenberger@karto.baug.ethz.ch
- Andreas Neumann, 01/ 633 30 37, neumann@karto.baug.ethz.ch
- Andrea Terribilini, 01/ 633 30 29, terribilini@karto.baug.ethz.ch
- Uta Samaga, 01/ 633 30 31, nur bis Dezember 2000
- Herr Schwandner (DAI), architektur@dainst.de

Abzugebende Arbeiten

Die abzugebenden Arbeiten wurden von der Assistenz IKA vorgegeben.

- Technischer Bericht (inkl. Karten, Tabellen und Anhang)
- Präsentation im Internet
- Poster
- CD-ROM
- Webseite⁸ (fakultativ)

7. Anhang A

8. <http://www.karto.ethz.ch/>

1.3 Ausgangsdaten

Ausgangsdaten

Unter den Ausgangsdaten stand ein DHM in digitaler Form zur Verfügung. Die erste Schwierigkeit bestand darin, die im DXF-Format vorliegenden Daten für unsere Bedürfnisse anzupassen. Für die Bearbeitung derart umfangreicher Daten und den Export ins VRML-Format wird das bekannte Rendering- und Modellierungsprogramm 3D Studio Max R3.1 eingesetzt.

Karte

Für die Modellierung stand eine nicht überarbeitete Vektorkarte des Gebietes zur Verfügung, welche die Lage der Relikte vorgibt. Sie diente auch zur groben Orientierung im Gelände.

DTM

Aufgrund der Ausgangsdaten wurde ein DTM erarbeitet als Grundlage für die weitere Arbeit. In dieses Höhenmodell des Geländes wurde die Ausgrabungsstätte schematisch eingefügt. Die Mauern und sonstige Relikte wurden in 3D Studio Max modelliert, nachbearbeitet und zu einem Ganzen zusammen gefügt.

Tabelle 1.1. Ausgangsdaten

Ausgangsdaten	
DTM	DXF Format
Übersichtskarte	DXF Format

1.4 Techniken und Werkzeuge

Nachfolgend werden zuerst die eingesetzten Techniken beschrieben und wie sie für die Umsetzung des Projekts benutzt wurden. Anschliessend werden noch kurz die eingesetzten Programme vorgestellt.

HTML/CSS

Die Seitenbeschreibungssprache HTML/CSS für die Gestaltung der Webseite. HTML ist eine Beschreibungssprache, die normalen Text über sog. Tags auszeichnet, zum Beispiel `fett` würde in einem Browser als **fett** ausgegeben. Zusammen mit CSS kann man in modernen Browsern noch wesentlich genauer in das Layout eingreifen, zum Beispiel kann man die Position eines Objektes, wie in unserem Fall die Karte, auf der Webseite pixelgenau festlegen.

VRML

Die Beschreibungssprache VRML wird für die Umsetzung und zur Darstellung der virtuellen Welt der Ausgrabungsstätte Palairos. VRML ist eine Modellierungssprache für die Darstellung von dreidimensionalen Objekten und/oder Welten. Zur Darstellung reicht aber ein normaler Browser, wenn das entsprechende Plugin vorhanden ist. In dieser Sprache sind einerseits eine Reihe von primitiven Objekten vordefiniert (Kugel, Kegel, Quader, Zylinder), die auch kombiniert werden können. Andererseits besteht die Möglichkeit komplexe Objekte über sogenannte IndexedFaceSets zu konstruieren. Ein weiterer Vorteil, dass die Datei mit einer Beschreibungssprache plattformunabhängig gestaltet wird und diese von einem Viewer interpretiert werden kann.

JavaScript und VBScript

Die Interaktion von Benutzer und „Virtueller Welt“ wurde mittels JavaScript und VBScript umgesetzt. JavaScript, genauer gesagt ECMAScript⁹, ist die Programmiersprache um interaktive Elemente auf einer Webseite zu erstellen. Sie entstand aus Netscapes JavaScript und Microsofts JScript. VBScript ist ein Ausschnitt der Sprache Visual Basic von Microsoft, die speziell für Webanwendungen optimiert wurde. Im Zusammenspiel Cortona Player und Internet Explorer ist sie mächtiger als JavaScript.

3D Studio Max R3.1

Die gesamte Modellierung der virtuellen Welt erfolgte in 3D Studio Max Release 3.1 von KINETIX. Es stellt alle nötigen Funktionen zur Verfügung und überzeugt durch seine gute Bedienbarkeit sowie den gewaltigen Funktionsumfang. Darüber hinaus stellt die Software einen wirklich guten Export nach VRML zur Verfügung.

VrmlPad

Die Nachbearbeitung des VRML Codes erfolgte mit dem Editor von ParallelGraphics, der speziell für VRML optimiert ist. So bietet er unter anderem Syntax-Highlighting, eine Fehlerprüfung und die Möglichkeit das VRML-File komprimiert zu speichern.

9. siehe ECMA -262 unter <http://www.ecma.ch>

Cortona Player

Für die Darstellung der virtuellen Welt im Browser wurde der Cortona Player von ParallelGraphics gewählt. Da dieser durch seine Stabilität und die qualitativ gute und auch schnelle Darstellung überzeugt und als eines der wenigen VRML-Plugins momentan weiterentwickelt wird.

Internet Explorer 5+

Heutzutage der wohl meistverbreitete Webbrowser. Deshalb stellten wir uns der Herausforderung, alles in diesem Browser zu realisieren. Was anfangs den Nachteil hatte, dass es praktisch keine Vorlagen und sonstige Unterlagen zu finden gibt. Trotzdem konnten Interaktionen zwischen VRML-Plugin und Browser in beiden Richtungen realisiert werden.

2 Datenverarbeitung

Im Folgenden wird die Entstehung und die Weiterverarbeitung der Datensätze bis zum fertigen Produkt und die Probleme beschrieben.

2.1 Vorarbeiten und Bereitstellung der Daten

Als Ausgangsinformationen lagen die Diplomarbeit von Niels Oberhauser (FH Oldenburg & DAI) und die daraus entstandenen Datensätze vor. Für die Weiterbearbeitung zur topographischen 2D-Karte und zum 3D-Modell liefen folgende AutoCAD-Dateien die geometrischen und semantischen Grundlagen.

- palairos.dwg (Messdaten kartiert, ebenengetrennt, mit Höhenmodell in Form von Höhenlinien, Details siehe unten)
- gitter3d.dwg (Version, die als Rendervorlage erstellt wurde, wenige Objekte vorhanden, noch teilweise falsche Höhenkoordinaten der Objekte, Gebiet Prophitis Ilias fehlt, Höhenmodell in Form eines sehr engen Gitters, das zwischen den Höhenkurven interpoliert wurde und dadurch Kleinformen enthält, die im Höhenlinienbild gar nicht vorhanden sind)
- palairos3d.dwg (Version, die als Rendervorlage genutzt wurde, Gebiet Prophitis Ilias fehlt und es sind noch weniger Objekte im Modell, da wohl fehlerhafte Objekte aus palairos.dwg und gitter3d.dwg beseitigt statt bereinigt wurden, Vorteil: die Stadtmauer liegt komplett vor).

Aufgrund der Vollständigkeit sollte vor allem die Datei palairos.dwg genutzt werden.

Mit diesen Daten direkt weiterzuarbeiten war aus verschiedenen Gründen nicht möglich. Dabei traten folgende Probleme auf:

- ein schlechtes Ebenenkonzept, d.h. es gab zu viele Ebenen, die teilweise sogar leer waren, teilweise Objekte anderer Objektklassen enthielten, und es waren andererseits einige Objekte gleicher Klassen auf verschiedenen Ebenen verteilt
- viele Höhenangaben waren fehlerhaft, d.h. sie passten nicht ins Höhenmodell und waren auch offensichtlich falsch (teilweise auf 0m oder negative Höhenangaben usw.)

Probleme und Fehler

Diese Probleme und Fehler galt es zu beheben. Folgende Vorgehensweise schien günstig.

1. Erstellen einer neuen Datei mit einem grob bereinigtem Ebenensystem, zusammengesetzt aus palairos.dwg und palairos3d.dwg.
Grob bereinigt bedeutet, dass Ebenen mit gleicher Thematik zusammengeführt wurden und in wenigen Fällen, in denen in einer Ebene viele Elemente anderer Ebenen vorhanden waren, auch Objekte dieser Ebene getrennt und den entsprechenden anderen Ebenen zugeordnet wurden. Daraus ergab folgende neue Ebenenstruktur.

Tabelle 2.1. Ebenenstruktur - Ausgangsdaten

Ebenenname	Datei/Layer	Inhalt	Bemerkungen
ANTIKE	Palairos.dwg/ Kechantike Kechantikfundament Antike	Antike Bebauung (grundrisslich)	
BYZANTINISCH	Palairos.dwg/ Funde, Kechfunde	Byzantinische Bebauung (grundrisslich)	
DGM	Palairos.dwg/ dgm, \$hoehen, Elias, Kechdgm	Höhenlinien	Beschriftung entfernt
FELSEN	Palairos.dwg/ Kechmodernen, Felsbruch, \$böschung, Kechfels, Modtopographie, \$bruch	Felsregionen u. Kanten, Böschungen, Terrassen	Topographie aus Kechmodern, Schrift u. Einzelpunkte entfernt, Schriftfreistellung in Felsregionen entfernt
FUNDE	Palairos.dwg/ Funde, Kechfunde	Fundstellen, Gräber, Stelen, Nekropole	

Tabelle 2.1. Ebenenstruktur - Ausgangsdaten

Ebenenname	Datei/Layer	Inhalt	Bemerkungen
MITTELALTER	Palairos.dwg/ Mittelalter, Kechmittelalter, Solidmittelalter, Solidkechmittel	Mittelalterliche Bebauung (grundrisslich)	
STADTMAUERGER-IPPE	Palairos3d.dwg/ Mauerdeckel, Mauergitter	Gerüst für Stadtmauer mit richtigen Höhenkoten, inkl. Türme	Antike Bebauung im NO, die auch in diesem Layer war, entfernt
TOPOGRAPHIE-HEUTE	Palairos.dwg/ Modtopographie, Kechmodern, \$bruch	Wege, Zäune	Bruchkante in Kech, Felsen in Prophitis Ilias und Einzelpunkte entfernt
VEGETATION	Palairos.dwg/ Bäume-wald, Kechbaum	unbegehbare Buschfläche, Einzelbäume	
VERBINDUNG	Palairos.dwg/ Verbindung	Gedachte Verbindungslinien zwischen Mauerresten, sehr gemischter Inhalt, nicht eindeutig zuordnenbar (siehe Bsp. 1 u. 2)	
VERMSTRASSEN	Palairos.dwg/ Vermstrassen	Vermuteter Strassenverlauf	
WASSER	Palairos.dwg/ Wasser	Grabensole, Zisternen, Quellenbecken	Schriftzüge u. Einzelpunkte entfernt

Da diese Bearbeitung auch als Grundlage für die 3D-Visualisierung genutzt wird, musste die Datei entsprechend bearbeitet, insbesondere die Beschriftung entfernt werden. Für die Umsetzung als 3D-Modell und 2D-Darstellung sollte aber unbedingt die Originaldatei, bzw. der Plot der Originaldatei konsultiert werden, da durch die Beschriftungen noch teilweise Hinweise zu den Objekten, d.h. auch für die graphische Umsetzung, gegeben werden können. Auch wurden noch die verschieden grossen Signaturen für gleiche Objekte beibehalten, da noch nicht klar war, ob sie eine Bedeutung tragen oder zufällig entstanden sind.

1. Erstellung eines AutoCAD(-dwg)-Files für jede Ebene
Auf diese Art konnten die Elemente ebenenweise (als *.dxf-Files) an das Höhenmodell angepasst und vor allem anschliessend ebenenweise kontrolliert werden.
2. Berechnung eines TINs
Ein TIN ist notwendig, um die Höhen der Elemente aus den „Ebenen-dxf-Dateien“ mit einem Programm von A. Terribilini an das Höhenmodell anpassen zu können. Voraussetzung für die TIN-Generierung ist in diesem Fall eine ASCII-Datei mit den Stützpunktkoordinaten der Höhenkurven. Dabei gab es Probleme mit dem AutoCAD-Elementtyp B-Spline, in dem ein Grossteil der Höhenkurven vorlag und die sich weder ins dxf-Format konvertieren noch in Polylinien umwandeln liessen. Eine Lösung fand sich schliesslich über den Export aus AutoCAD als *.dxf-File der Version 14, den Import dieses Files in Microstation (PC-Version), einer *.dgn-Erzeugung in Microstation und dann das Auslesen als *.dxf-File (Version 10). Dadurch wurden die Splines in Linien zerlegt. Mit Programmen von A. Terribilini konnten aus diesem File dann wieder Koordinaten als ASCII-Text ausgelesen und daraus ein TIN generiert werden.
3. Anpassung von Objekthöhen an das Höhenmodell
Mit dem erzeugten TIN und den dxf-Files (Version 12) der einzelnen Ebenen, konnten die Objekthöhen an das Höhenmodell angepasst werden. Leider können in diesem Verfahren nur Polylinien und Linien verarbeitet werden. Deshalb mussten zuvor noch sämtliche komplexen Objekte in Polylinien oder Linien umgewandelt werden.
4. Erzeugung neuer *.dwg-Files pro Ebene
Diese *.dwg-Files, in denen die Objekthöhen an das Höhenmodell angepasst wurden, konnten nun als Kontrolle zum Vergleich mit den Ausgangsdateien und vor allem für eine Zusammenfassung der korrigierten Ebene

nen in einer neuen Datei („merged.dwg“) genutzt werden. Letztere ist jetzt eine korrigierte Version der palairos.dwg-Ausgangsdatei und wird für die Weiterbearbeitung verwendet.

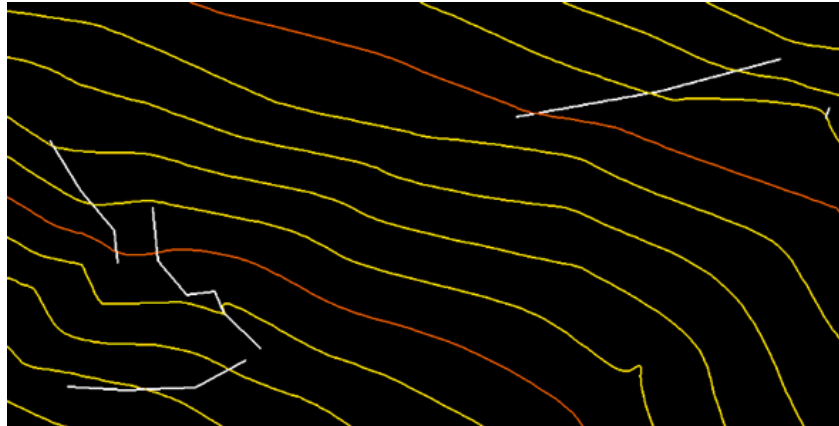


Abbildung 2.1. Bildschirmfoto aus der Ebene Verbindung



Abbildung 2.2. Screenshot aus AutoCAD

2.2 Visualisierung und Bearbeitung in 3D Studio Max

Zur 3D-Visualisierung der Szene wird mit dem Programm 3D Studio Max weitergearbeitet, da hier die Objekte modelliert und direkt als VRML-Datei ausgegeben werden können. Die interaktive Visualisierung mittels Software und anschließendem Export ins VRML-Format scheint in diesem speziellen Fall auch deshalb sinnvoll zu sein, da die Daten inhomogen vorliegen und manuelle Interpretation zur Bereinigung der Inhomogenität erfordern.

Aufgrund des grossen Umfangs der Datei, der zu Problemen beim Im- und Export der Daten führte, und der anfallenden umfangreichen manuellen Nachbearbeitung wurde vorläufig nur ein repräsentativer Ausschnitt als Muster und zum Austesten der Möglichkeiten bearbeitet.

Das von uns gewählte Gebiet umfasst das Gebiet innerhalb der Stadtmauern inklusive dieser. Bei der Modellierung des Ausschnitts führten vor allem folgende Gegebenheiten aus den Ausgangsdaten zu Problemen, diese werden nachfolgend an passender Stelle erläutert.

2.2.1 Import

- In 3D Studio Max können direkt AutoCAD-Dateien importiert werden.
- Beim Import von *.dwg-Dateien besteht die Möglichkeit, identische Punkte schon beim Datenimport mit der Schweissfunktion ("weld") zusammenzufügen. Diese Funktion ist in diesem Fall unbedingt zu aktivieren, da viele Mauer Grenzen nicht als ein Objekt, sondern durch mehrere Linienobjekte beschrieben sind (s.o.).
- Die "weld"-Funktion ist nur innerhalb eines Objektes verfügbar, was zur Folge hat, dass aus den alten Ebenen jeweils ein neues 3D-Studio-Objekt erzeugt werden muss ("derive objects by layer" + "convert to single objects"). Das hat den Nachteil, dass ehemalige Objekte aus dem neu entstandenen Ebenenob-

jekt manuell herausgelöst werden müssen. Es bringt aber auch den Vorteil, dass die Ebenenbezeichnungen als Objektnamen übernommen werden und somit die Zuordnung der Ebenen erhalten bleibt.

2.2.2 Ebenen-/Gruppenkonzept

- Es liegen noch viele der bereits erwähnten Fehler in der Ebenenzuordnung vor, die auch hier nach der groben Bereinigung¹⁰ nur zum Teil behoben werden konnten.
- Es können zwar noch Felskanten aus falschen Ebenen zur richtigen Ebene, der Ebene "Fels", übertragen werden. Der umgekehrte Fall ist aber schon nicht mehr korrigierbar, z.B. Mauerkanten, die in der Felsebene liegen, können nicht mehr zur richtigen Epoche und somit auch nicht der entsprechenden Ebene zugeordnet werden. Diesen beschriebenen Fall zeigt folgender Screenshot aus dem 3D Studio Max.



Abbildung 2.3. Bildschirmfoto (die violetten Elemente stammen aus der Ebene "Felsen")

- Weiterhin scheint es sinnvoll, das Ebenenkonzept aus dem AutoCAD im 3D Studio Max durch ein entsprechendes Gruppenkonzept zu erhalten. Der zusätzliche Vorteil der Gruppenstruktur ist, dass nicht nur die Ebenenstruktur erhalten bleibt, sondern sie durch Untergruppen noch weiter strukturiert werden kann. Ausserdem wird diese Struktur auch beim Import in das VRML-Format in Form von Groupnodes mit Children beibehalten, was sehr vorteilhaft für die weitere Editierung ist. Zusätzlich ist das auch praktisch bei der Programmierung der Interaktionen.
- Bei der elementweisen Bearbeitung tauchen immer wieder Unklarheiten über Objekte und ihre Ebenenzugehörigkeit auf. Diese sollten nicht gelöscht aber entsprechend bezeichnet und bei der Erstellung der endgültigen Szene eventuell weggelassen werden. Ist eine nachträgliche Klärung der Zugehörigkeit möglich, kann diese dann problemlos umgesetzt werden.
- Die Elemente der Ebene "Verbindung" wurden aufgrund der Unklarheiten über die Bedeutungen der Inhaltselemente vorläufig weggelassen.

Gruppen-Knoten-Konzept

Das neue Ebenen-, bzw. Gruppen-Knoten-Konzept sieht bisher folgendermassen aus.

Tabelle 2.2. Gruppen-Knoten-Konzept

Gruppe	Untergruppen	Inhalt und Beschreibung
TIN	tin-innen01, tin-mitte01, DTM_1.01	Höhenmodell in Form eines TIN in verschiedenen Auflösungen (siehe "GELÄNDEMOTELLIERUNG")
Antike		Antike Bebauung
Mittelalter		Mittelalterliche Bebauung
Funde		Geometrien aus AutoCAD
Funde-Signaturen	Fundorte, Grabstelen, Kapitelle, Schwellen, Säulen	Umsetzung aus "Funde" in Signaturen

¹⁰siehe Kapitel 2.1

Tabelle 2.2. Gruppen-Knoten-Konzept

Gruppe	Untergruppen	Inhalt und Beschreibung
Felsen		Bisher nur Treppen bearbeitet (siehe Gruppe "Treppen")
Treppen	Trepp-felsen, Trepp-antike	Modellierte Treppen, nach Ausgangsebene unterteilt
Vegetation		Geometrien aus AutoCAD
Bäume-Sign		Umsetzung aus "Vegetation" in Signaturen
Wasser		Zisternen
Verbindung		AutoCAD-Geometrien, noch nicht bearbeitet
Beleuchtung		Headlight
Cameras		Repräsentative Ansichten, Kamerafahrten
Stadtmauer		Stadtmauer

2.2.3 Geländemodellierung

Um das Geländemodell einzufügen, sollte möglichst das TIN verwendet werden, das zur Korrektur der z-Koordinaten verwendet worden ist, und nicht ein weiteres Modell aus den ursprünglichen Höhenlinien, um Unstimmigkeiten zu vermeiden. Dafür wurde das programminterne TIN-Format¹¹ in das vom 3D Studio Max lesbare dxf-Format umgewandelt.

So konnte es eingelesen werden. Ein kleines Problem war, dass Objekte und Geländemodell erst einmal scheinbar nicht zusammenpassten. Eine stichprobenweise Überprüfung ergab aber, dass beim Import der Objekte ein Fehler aufgetreten sein musste und dieser glücklicherweise nur eine Verschiebung ausmachte. Der genaue Fehlerbetrag der z-Koordinate betrug 200 m. Damit aber alle Elemente sicher auf, bzw. lieber unter als über der Geländeoberfläche zu liegen kamen, wurden die Objekte nur um 199,9 m verschoben. Mit dieser Ungenauigkeit muss gerechnet werden, da ja nur die Stützpunkte von den jeweiligen Objekten auf das Geländemodell gerechnet wurden und dieses nicht stetig verläuft, sondern durch die Dreiecksstrukturen mehr oder weniger stark gebrochen ist.

Aufgrund der grossen Punktdichte des vorliegenden TINs, kann es zu Problemen nach dem Export in eine VRML-Datei geben. Die Dateigrösse wird dann zu mächtig, so dass keine flüssige Navigation mehr möglich ist.

Da bisher sowieso nur ein Teil der Ausgrabungsstätte objektmässig modelliert wurde, bot es sich an folgendermassen vorzugehen: Im Bereich der modellierten Ausgrabungsstätte wurde das genaue Modell übernommen – aufgrund des optisch besseren Aussehens und damit Ungenauigkeiten und Überschneidungen zwischen Geländemodell und Objekten vermieden werden. Das übrige Modell wurde ausgedünnt. Damit es aber nicht zum optischen Bruch zwischen dem Modell innerhalb der Objektfläche und den umliegenden Gebieten kommt, erfolgte es in zwei Stufen.

Gearbeitet werden kann auch hier wieder mit dem "Optimize"-Modifizierer, allerdings mit anderen Einstellungen. Vor allem ist es günstig, auch wenn dadurch eine geringere Reduktion resultiert, den Wert zum Erhalt von typischen Dreiecksstrukturen ("Bias") relativ hoch zu setzen. Denn so bleiben die charakteristischen Formen der Geländeoberfläche noch gut erkennbar.

2.2.4 Gebäude-/Mauermodellierung

Bei der Modellierung des Ausschnitts führten vor allem folgende Gegebenheiten aus den Ausgangsdaten zu Problemen:

- Viele Umrisslinien von Mauerkanten lagen in Einzellinien zerstückelt vor, so dass sie keine geschlossenen Polygone bildeten.

¹¹ siehe Kapitel 2.2.2

- Andere Umrisse von Mauerresten waren durch nicht geschlossene Polygone und vor allem oft mit sich selbst kreuzenden Splines beschrieben.

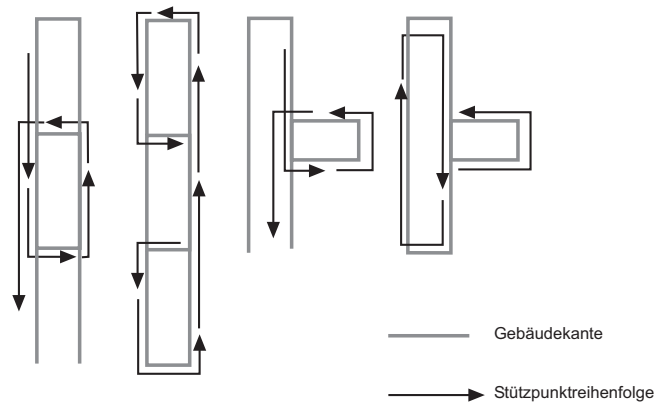


Abbildung 2.4. Stützpunktreihenfolge

Mit dem vorherigen Schritt sind die Probleme der zerstückelten Gebäudekanten zwar behoben, jedoch noch nicht die der Gebäudekanten mit falscher Stützpunktreihenfolge¹², da sie aufgrund der Überkreuzungen noch immer keine geschlossenen Polygone bilden.

Bei der manuellen Weiterverarbeitung ist deshalb besonders darauf zu achten, dass die Mauerelemente wirklich aus geschlossenen, sich nicht kreuzenden Linienzügen zusammengesetzt werden, damit sie beim Extrudieren auch zu geschlossenen Körpern werden. Des Weiteren ist zu beachten, dass die neuen einzelnen Elemente im Fall von Innenhöfen so zusammengesetzt sind, dass Innenhofrand und Aussenkante als zwei Splines in einem Objekt zusammengeführt sind. Um geschlossene Splines zu erzeugen, müssen teilweise Kanten ("edges") gelöscht werden, Stützpunkte ("vertices") wieder verbunden ("weld") und manchmal ganze Modellierungsstrukturen erneuert werden. Manchmal lassen sich Splines mit der Funktion "close" einfach schliessen.

Mit der "detach"-Funktion können dann die einzelnen Objekte von den ehemaligen Ebenen (die nach dem Import zu Objekten geworden sind) gelöst werden (Achtung! Nicht die "explode"-Funktion verwenden, da mit ihr wieder alle Polygone in Linienstücke zerlegt werden.). Dabei ist es ratsam die neuen Objekte nicht mit den "Default-Namen" (Shape + Nummer) sondern mit entsprechenden Bezeichnungen zu versehen. Z.B. wie bisher bei den Mauern verwendet: a - einfache Linie; b -eventuell nicht geschlossen; c - geschlossener Linienzug; d - geschlossener Linienzug und einfache Linien in einem Objekt. Dabei können auch mehrere Mauern in einem Objekt zusammengefasst werden.

Wenn die Objektstrukturen bereinigt sind, können die Mauerreste von der blossen Umrissdarstellung durch den "Extrude"-Modifizier zu Körpern mit einheitlicher Höhe generiert werden. Am besten geschieht dies mit der ganzen Gruppe (also Ebene) gleichzeitig. (bei Parametern: "Cap closed entities" wählen und "Smooth" ausschalten), damit der Vorgang bei allen Elementen einheitlich passiert und auch später noch Mauerumrisse hinzugefügt werden können, die dann automatisch extrudiert werden.

Zusätzlich sollte hinterher noch der "Optimize"-Modifizier - ebenfalls für die ganze Gruppe gleichzeitig - genutzt werden, um überflüssige Punkte zu eliminieren, die die spätere VRML-Datei nur unnötig vergrössern würden (günstige Parameter: Voreinstellung und "Bias = 0.0". Es ist ein Parameter, der hilft, bestimmte Dreiecksstrukturen zu erhalten und deshalb in diesem Fall deaktiviert (= 0.0) werden kann und sollte, da nur ebene Flächen vorhanden sind, die nicht durch Dreiecke zerlegt werden sollten). Die Resultate für einzelne Objekte der Gruppe lassen sich durch Öffnen der Gruppe und Anwählen der Objekte beurteilen - optisch und numerisch.

2.2.5 Felsen

Ein besonderes Problem liegt ausserdem noch in der Ebene "Felsen" vor, denn die als Bruchkanten angegebenen Linienelemente entsprechen häufig nicht der Form von Bruchkanten und stimmen zudem oft nicht mit dem Höhenlinienbild überein.

¹²siehe Abbildung 2.4

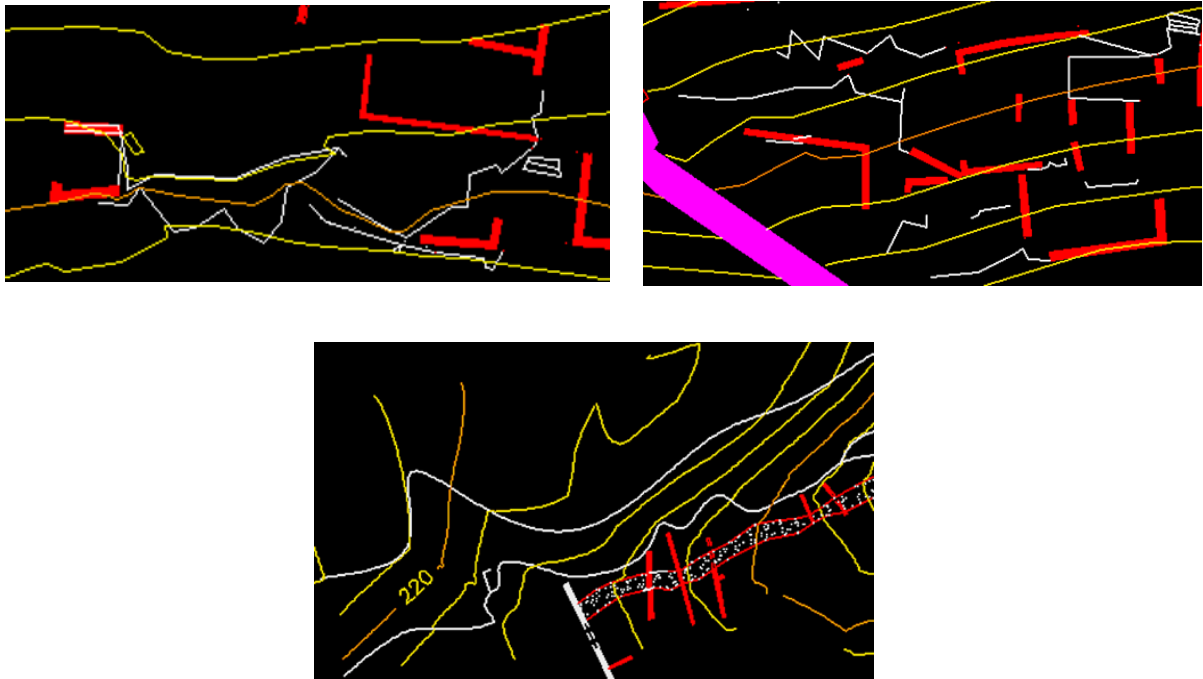


Abbildung 2.5. Bildschirmfotos Bruchkanten

Wenn auch die Bruchkanten zum Teil nicht mit dem Geländemodell übereinstimmen und nicht unbedingt die typische Form von Bruchkanten haben, so passt doch der Teil dieser Ebene, der felsiges Gelände beschreibt, recht gut zum Höhenmodell. Leider reichte die Zeit vorerst nicht, um dieses auszuführen, es ist aber gut vorstell- und machbar, dass man entweder Texturen über die entsprechenden Bereiche legt, oder den betreffenden Dreiecksstrukturen im TIN andere Oberflächeneigenschaften (Farbe, Lichtreflexionsverhalten, usw., was im VRML-Code dem "Appearance-Node" entspricht) zuweist.

Die Felskanten wurden in 3D Studio Max nachbearbeitet und den Gegebenheiten angepasst, sodass eine optisch bessere Darstellung erreicht wurde.

2.2.6 Treppen

Ein spezielles Problem beinhalten die Treppen (die hauptsächlich in der Ebene "Felsen" vorliegen), bei denen nicht zu erkennen war, ob sie stufenweise aufgenommen wurden. In der Ausgangsdatei liegen zwar alle Treppen linienweise vor, aber die Höhen sind scheinbar zufällig zugeordnet. Teilweise sind die potentiellen Stufen auf 0m gesetzt und teilweise haben sie Höhen, die allerdings keine Treppenform wiedergeben, sondern schräg verlaufen. Deshalb wurden sie, wie alle anderen Elemente in das Höhenmodell passend eingepasst¹³. Daher stellt sich die Frage, wie diese Strukturen in der 3D-Darstellung modelliert werden sollen.

Vor allem gibt es auch hier wieder Fälle, wie folgenden, wo unklar ist, wieso die Form auf diese Art modelliert wurde (rote Linie kennzeichnet eine Spline) und ob es sich dabei um eine Treppe handelt oder nicht.

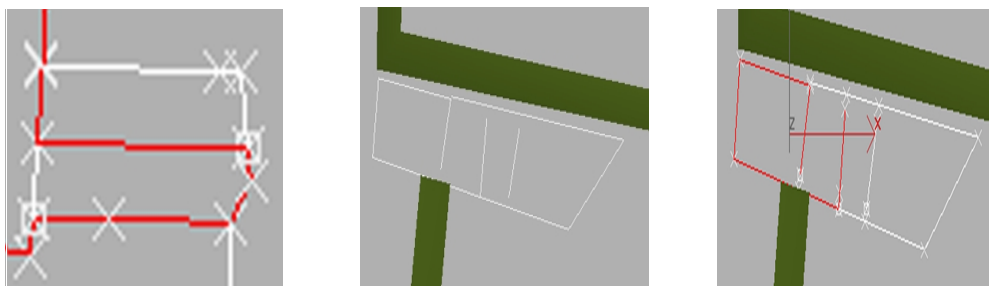


Abbildung 2.6. Bildschirmfotos Treppen

Zur Behandlung der Treppen scheint das folgende Vorgehen sinnvoll. Alle Treppen sollten in einer eigenen neuen Gruppe zusammengefasst und bearbeitet werden, da die meisten Treppen der Ebene "Fels" zugeordnet sind und

¹³.siehe Kapitel 2.2.2

dadurch eine epochale Gliederung nicht mehr möglich ist. Für eine bessere Übersicht und zum leichteren Nachvollzug der Modellentstehung, sollte die Herkunft der Treppen durch eine Untergruppenstruktur beibehalten werden (d.h. Gruppe "Treppen" mit Untergruppen "Treppen aus Antike" und "Treppen aus Felsen").

Um die Treppen als solche in diesem Modell zu veranschaulichen, wirkt es besser, wenn künstlich Stufen in die flach im Gelände liegenden Treppen eingefügt werden, da sowieso nicht klar ist, welche Form die Linien in der Ausgangsdatei beschreiben sollen und so aber die Treppenstruktur für den Betrachter gleich klar wird:

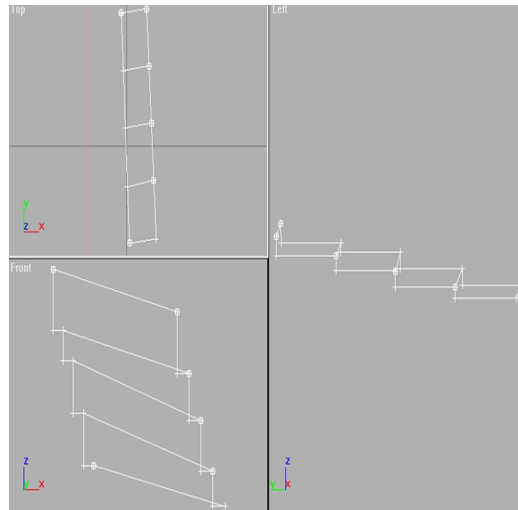


Abbildung 2.7. Bildschirmfoto Ausgangssituation

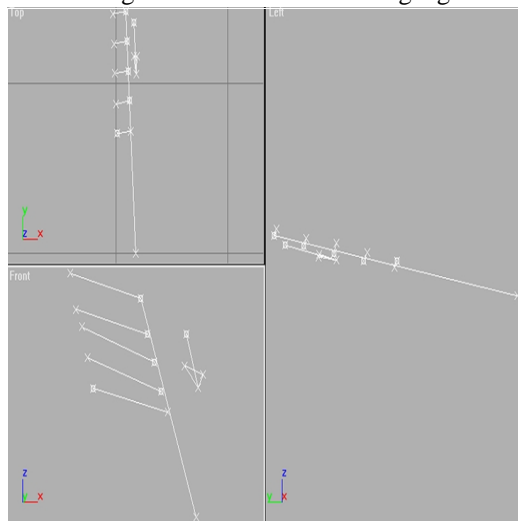


Abbildung 2.8. Bildschirmfoto modellierte Treppenform

Später müssen dann noch die Treppenstufen in das Geländemodell integriert, bzw. die entsprechenden, die Treppen verdeckenden Flächen aus dem Geländemodell herausgeschnitten werden.

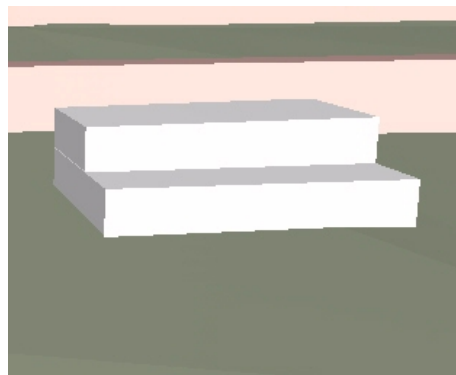


Abbildung 2.9. Treppe im Gelände

2.2.7 Funde

Für die Darstellung von Elementen aus der Ebene "Funde" ist es naheliegend, für die verschiedenen Kategorien dreidimensionale Signaturen zu entwerfen. Diese lassen sich gut in einer neuen Datei entwerfen und dann einfügen.

Nachdem das erste Symbol eingefügt ist, ist es wichtig, dass alle weiteren Symbole dieser Art als Instanzen und nicht als Kopien daraus entwickelt werden, damit nachträgliche Änderungen an Aussehen und Struktur auf alle Symbole automatisch übertragen werden und beim Export nach VRML "USE and DEF" genutzt wird.

Weiter müssen sie über die Signaturen, die auch ins Geländemodell eingepasst sind, positioniert werden. Dabei ist darauf zu achten, dass die Elemente passend im Gelände orientiert sind, z.B. die Säulen auf dem Untergrund liegend und nicht schwebend oder die Grabsteine mit dem Fuss im Gelände senkrecht stehend. Es ist nicht immer klar, welcher Punkt der Signatur als Referenzpunkt für das entsprechende Objekt dient. Deshalb muss dieser manchmal abgeschätzt werden.

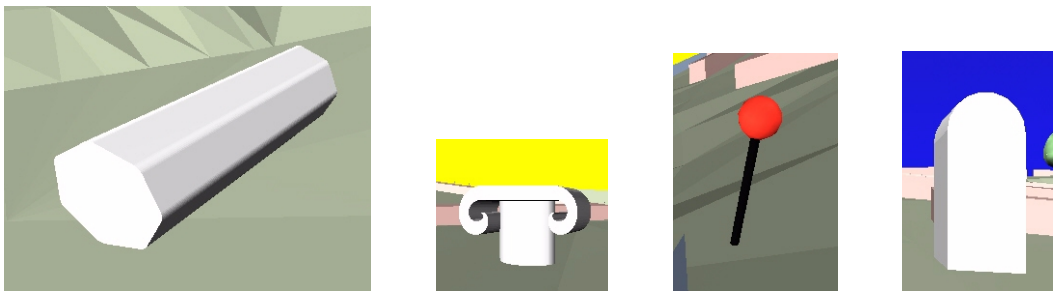


Abbildung 2.10. Fundsymbole

2.2.8 Vegetationsdarstellung

Das gleiche Vorgehen, wie für die Elemente der Ebene "Funde", bietet sich auch für die Baumsignaturen an.



Abbildung 2.11. Baumsymbol

2.2.9 Gewässer

Zu beachten bei der Bearbeitung der Ebene "Wasser" ist, dass die Wasseroberflächen, in diesem Fall betrifft es die Zisternen, horizontal verlaufen.

2.3 Bearbeitung VRML

Die mit 3D Studio Max modellierten virtuellen Daten wurden mit VmlPad und UltraEdit32 weiter bearbeitet. Aus der beschriebenen Problemsituation ergibt sich folgende weitere Bearbeitungsweise.

2.3.1 Export nach VRML

Es ist beim Export, bzw. schon bei der Modellierung der Szene, immer auf die resultierende Dateigröße zu beachten. Dies bedeutet ein stetes Abwägen zwischen der graphischen Feinheit und der möglichst flüssigen Navigation. Es sollte daher, wann immer möglich, die Punktzahl gering gehalten werden, das heisst besonders der „Optimize“-Modifizier genutzt werden, mit dem gezielt Strukturen herausgearbeitet können und doch die Punktzahl reduziert werden kann (siehe Modellierung der Mauerreste und des Geländes).

Beim Export ist dann zu überlegen, ob Normalenvektoren generiert werden sollen. Sie vergrößern die Datei zwar, allerdings ist der optische Effekt recht gut. Es sollte auf jeden Fall die Möglichkeit genutzt werden, dass in VRML vordefinierte Körper für die Geometrien genutzt werden („Primitives“ aktivieren).

Bisher fehlen im Probeausschnitt aus Zeitgründen noch die Darstellung der Reste der Stadtmauer und die Bearbeitung der Ebene Fels.

2.3.2 Bearbeitung in VrpmlPad

Dieser Editor bietet einigen Komfort bei der Programmierung und Anpassung einer VRML-Datei. So bietet er Syntax-Highlighting, sowie eine Fehlerüberprüfung.

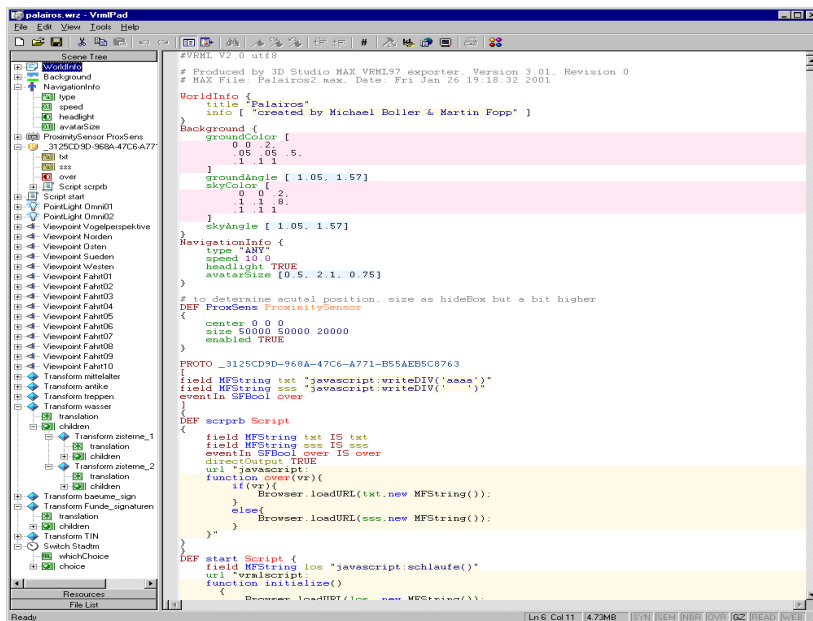


Abbildung 2.12. VrpmlPad-Screenshot

Mit ihm wird der Sourcecode angepasst, während im weiter unten besprochenen UltraEdit32 die Massarbeit, sprich die Replacefunktion, genutzt wird.

So wird ein ganzer Baum-Node mit *DEF* genau bezeichnet, damit nachher im UltraEdit32 der *USE*-Befehl gesetzt werden kann. Codebeispiel:

```
DEF b_ume_sign Transform {
  translation -12518 200.96 -9474.9
  children [
    DEF baum_symbol01 Transform {
      translation -122.03 20.098 -65.686
      rotation 0 1 0 -1.4661
      scale 0.011722 0.011722 0.011722
      children [
        DEF Sphere02 Transform {
          ...
        }
      ]
    }
  ]
}
DEF Cylinder02 Transform {
  DEF baum_symbol02 Transform {
    translation -120.46 19.153 -61.338
    rotation 0 1 0 -1.4661
    scale 0.011722 0.011722 0.011722
    children [
      USE Sphere02
      USE Cylinder02
    ]
  }
}
```

Dieses Ersetzen ist nun bei allen Bäumen gemacht worden, natürlich nur nicht beim definierten Baum.

Ähnlich wird auch bei den antiken Überresten verfahren. Denen wurde in 3D Studio Max wohl allen die gleiche Farbe zugeteilt, aber beim Export nach VRML wird nicht, wie angenommen, einmal der material-Node definiert, so dass er in den Folgenden mit USE wiederverwendet werden könnte, das wird hier nun nachgeholt. Das bietet auch einen weiteren Vorteil, man kann für alle antiken Überresten an einem einzigen Ort die Farbe ändern. Nachfolgend wird das an einem Codebeispiel gezeigt:

```

DEF a_010 Transform {
  translation 79.74 1.89 4.2773
  children [
    Shape {
      appearance Appearance {
        material DEF Antikcolor Material {
          diffuseColor 0.80784 0.65098 0.61961
          ambientIntensity 0.47582
          specularColor 0.045 0.045 0.045
          shininess 0.2875
          transparency 0
        }
      }
    }
  ]
}
...
DEF a_011 Transform {
  translation 79.74 1.89 4.2773
  children [
    Shape {
      appearance Appearance {
        material USE Antikcolor
      }
    }
  ]
}

```

Genau gleich wird auch bei den mittelalterlichen Überresten verfahren. Codebeispiel:

```

DEF c Transform {
  translation 17.019 -29.536 99.594
  children [
    Shape {
      appearance Appearance {
        material DEF MialCol Material {
          diffuseColor 0.59216 0.67843 0.78824
          ambientIntensity 0.76732
          specularColor 0.045 0.045 0.045
          shininess 0.2875
          transparency 0
        }
      }
    }
  ]
}
...
DEF a Transform {
  translation 17.019 -29.536 99.594
  children [
    Shape {
      appearance Appearance {
        material USE MialCol
      }
    }
  ]
}

```

Im VrmPad werden auch alle JavaScript-Funktionen programmiert. So wird einerseits ein PROTOTYPE definiert, das heisst es wird ein neuer Nodetyp erstellt. Dieser hat folgende Aufgabe, wenn die Funktion *over* aufgerufen wird, einen String dem Browser zu übergeben. Codebeispiel:

```

PROTO _3125CD9D-968A-47C6-A771-B55AEB5C8763
[
  field MFString txt "javascript:writeDIV('aaaa')"
]

```

```

    field MFString sss "javascript:writeDIV('  ')"
    eventIn SFBool over
  ]
  {
  DEF scrprb Script
  {
      field MFString txt IS txt
      field MFString sss IS sss
      eventIn SFBool over IS over
      directOutput TRUE
      url "javascript:
      function over(vr){
          if(vr){
              Browser.loadURL(txt,new MFString());
          }
          else{
              Browser.loadURL(sss,new MFString());
          }
      }"
  }
  }
}

```

Auf diesen neuen Node wird nun in den verschiedenen Fundobjekten zurückgegriffen. Damit nun die Funktion *over* ausgelöst wird, bedarf es aber noch eines Touchsensors für jedes betreffende Objekt. Ein Touchsensor gibt eine Rückmeldung, wenn der User mit der Maus z. B. über das Objekt darüberfährt oder es anklickt. Codebeispiel:

```

DEF Grabstele Transform {
    translation -52.015 -2.2928 -3.5596
    children [
        DEF PUSH TouchSensor { enabled TRUE }
        DEF grabstele Transform {
    ...

```

Zudem muss nun auch der Bezug zum Prototypen hergestellt werden, das geschieht folgendermassen, Codebeispiel:

```

DEF      sc _3125CD9D-968A-47C6-A771-B55AEB5C8763 {
          txt "javascript:writeDIV('<h6>Grabstele</
h6>') "
      }

```

Sprich es wird der Funktion „writeDIV“ eine Ausgabe übergeben. Diese Funktion existiert in der HTML-Datei „palairos.html“ und gibt dann den ihr übergebenen Text dynamisch in die HTML-Datei aus, quasi eine interaktive Legende. Dazu muss aber noch das Skript mit dem Touchsensor verbunden werden, was über die Definition einer *Route* geschieht. Codebeispiel:

```

ROUTE PUSH.isOver TO sc.over

```

Die Funktion *over* ist im Prototypen definiert und enthält die Browserübergabe. Das wurde so bei allen Fundobjekten, die ein Symbol haben, verwirklicht.

Des Weiteren wurde ein ProximitySensor eingebaut. Damit kann man die aktuelle Position des Benutzers erfahren. Codebeispiel:

```

DEF ProxSens ProximitySensor
{
    center 0 0 0
    size 50000 50000 20000
    enabledTRUE
}

```

Als weiteres wurde bei der Stadmauer ein *Switch* eingeführt. Dies ist quasi ein Schalter, der die Zustände “an” oder “aus” kennt. Mit ihm kann man nun erreichen, dass der Stadmauer-Node und damit die Stadmauer in der VRML-Szene auf der Webseite entweder sichtbar ist oder nicht. Codebeispiel:

```
DEF      Stadtm Switch {
        whichChoice0
        choice [
DEF STADTMAUER Transform {
```

Die Funktion, die dies auslösen kann, befindet sich in der HTML-Datei „control.html“.

Darüberhinaus wird noch ein Skript eingefügt:

```
DEF      start Script {
        field MFString los "javascript:schlaufe()"
        url "vrmlscript:
        function initialize()
        {
                Browser.loadURL(los, new MFString());
        }
        "
}
```

Dieser wird dank **initialize()**, aktiv sobald die VRML-Szene vollständig geladen ist und startet die Funktion **schlaufe()**, die in der HTML-Datei „palairos.html“ definiert ist. Die Schlaufe wird deshalb auf diese auf den ersten Blick umständliche Art gestartet um sicherzustellen, dass die Funktion bei langsamen Internetverbindungen nicht gestartet wird bevor die VRML-Szene geladen ist, wie man es sonst mit **onLoad()** aus einer HTML-Seite heraus realisieren könnte. Denn dann würde man auf Funktionen, Nodes oder Fields zugreifen wollen, die noch gar nicht existieren.

Zu guter Letzt wird der Navigation-Node angepasst. Codebeispiel:

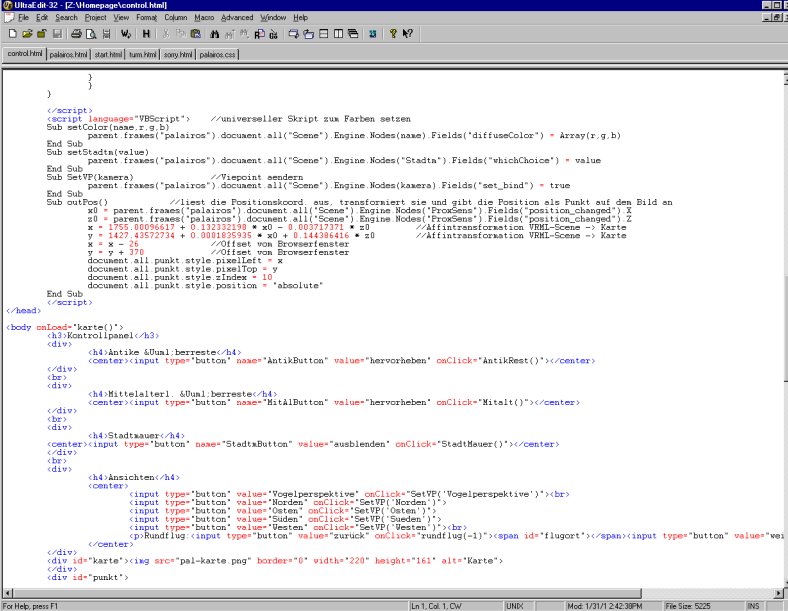
```
NavigationInfo {
        type "ANY"
        speed 10.0
        headlight TRUE
        avatarSize [0.5, 2.1, 0.75]
}
```

Von 3D Studio Max irrtümlich gesetzte PositionInterpolator und Timer bei den verschiedenen Viewpoints wurden gelöscht und der WorldInfo-Node gesetzt. Codebeispiel:

```
WorldInfo {
        title "Palairos"
        info [ "created by Michael Boller & Martin Fopp" ]
}
```

2.3.3 Bearbeitung in UltraEdit32

Im UltraEdit32 wird, wie schon weiter oben angesprochen, die Massenaarbeit gemacht. Denn dieser Editor bietet eine Replace-Funktion, die über mehrere Zeilen einen Pattern ersetzen kann.



```

}
}
</script>
<script language="VBScript"> //universeller Skript zur Farben setzen
Sub setColor(naame,r,g,b)
parent frames("palairos").document.all("Scene").Engine.Nodes(naame).Fields("diffuseColor") = Array(r,g,b)
End Sub
Sub setStadt(value)
parent frames("palairos").document.all("Scene").Engine.Nodes("Stadt").Fields("whichChoice") = value
End Sub
Sub SetVP(kamera) //Viewpoint oeffnen
parent frames("palairos").document.all("Scene").Engine.Nodes(kamera).Fields("set_bind") = true
End Sub
Sub outPos() //liest die Positionskoord aus, transformiert sie und gibt die Position als Punkt auf dem Bild an
z0 = parent frames("palairos").document.all("Scene").Engine.Nodes("ProxSens").Fields("position_changed") Z
z0 = parent frames("palairos").document.all("Scene").Engine.Nodes("ProxSens").Fields("position_changed") Z
z = 1755.0005617 + 0.12232249 * z0 + 0.00317371 * z0 //Affintransformation VRML-Szene -> Karte
y = 1427.43572734 + 0.0881855391 * x0 + 0.144386416 * z0 //Affintransformation VRML-Szene -> Karte
x = x * z
y = y * z
document.all punkt.style.pixelLeft = x
document.all punkt.style.pixelTop = y
document.all punkt.style.position = "absolute"
End Sub
</script>
</head>
<body onLoad="karte()">
<div id="Kontrollpanel">
<div>
<input type="button" name="AntikButton" value="hervorheben" onClick="AntikRest()"></div>
<div>
<input type="button" name="MittelButton" value="hervorheben" onClick="Mittel()"></div>
<div>
<input type="button" name="StadtButton" value="ausblenden" onClick="StadtMauer()"></div>
<div>
<input type="button" value="Vogelperspektive" onClick="SetVP('Vogelperspektive')"></div>
<input type="button" value="Morden" onClick="SetVP('Morden')">
<input type="button" value="Osten" onClick="SetVP('Osten')">
<input type="button" value="Suden" onClick="SetVP('Suden')">
<input type="button" value="Westen" onClick="SetVP('Westen')">
<input type="button" value="zurueck" onClick="rundflug(-1)"><span id="flugort"></span><input type="button" value="wes
</div>
<div id="karte">
</div>
<div id="punkt">

```

Abbildung 2.13. UltraEdit32-Screenshot

Deshalb wird er gebraucht um die überflüssig definierten material-Nodes bei den antiken und mittelalterlichen Überresten mit dem Pattern:

```
material USE Antikcolor
```

bzw.

```
material USE MiAlCol
```

zu ersetzen. Ähnlich wird bei den Bäumen verfahren, dort werden die Geometrien ersetzt durch:

```
USE Sphere02
USE Cylinder02
```

Dadurch wird die Datei merklich kleiner. Wobei zu guter Letzt das Ganze nochmals im VrmilPad geladen und dann definitiv gespeichert wird, neu aber in komprimierter Form¹⁴, was eine Dateireduktion um den Faktor 4 ergibt!

2.4 Programmierung der interaktiven Elemente der Webseite

An diesem Punkt wurde eine Hauptentscheidung in diesem Projekt getroffen. Es wurde festgelegt, dass die 3D-Visualisierung zuerst für den Internet Explorer 5+ erstellt und optimiert wird. In einem zweiten möglichen Schritt, der aber aus Zeitgründen nicht mehr gemacht werden konnte, wäre der Transfer auf andere Browser und Plugins gestanden. Der Internet Explorer wurde gewählt, da er einerseits heutzutage am weitesten verbreitet und andererseits am aktuellsten ist, von einigen Eigenheiten abgesehen. Zudem bietet er auch eine gute Performance beim Seitenaufbau.

Die verschiedenen Funktionen in JavaScript und VBScript wurden in den einzelnen Frame-Seiten der Webseite belassen und nicht in eine eigene Datei ausgelagert, damit ist der Zugriff auf die jeweilige Funktion von einem anderen Frame oder aus der VRML-Datei transparenter. Denn es hat sich gezeigt, dass in diesen Zugriffen eine Hauptfehlerquelle liegen kann.

Nachfolgend werden die einzelnen Funktionen anhand der dazugehörigen Schaltelelemente auf der Webseite erklärt.

¹⁴komprimieren mit GZip

2.4.1 Antike Überreste hervorheben

In diesem Element wird ausgenutzt¹⁵, dass alle antiken Überresten dieselbe Farbe besitzen und dass an einer einzigen Stelle diese geändert werden kann:

```
function AntikRest()           JavaScript in der Datei „control.html“
{
    name = "Antikcolor";
    if (switchAntik==0) {
        setColor(name,0.80784,0.65098,0.61961);
        switchAntik=-1;
        document.all.AntikButton.value="hervorheben";
    }
    else {
        setColor(name,0,1,0);
        switchAntik=0;
        document.all.AntikButton.value="Rückgängig";
    }
}
Sub setColor(name,r,g,b)      VBScript in der Datei „control.html“
    parent.frames("palairos").document.all("Scene").Engine.Nodes(name).Fields("diffuseColor") =
    Array(r,g,b)
End Sub
```

Wird nun der Knopf gedrückt, schaut das Skript nach, wie der Wert in **switchAntik** ist, beim Wert 0, sprich die Überreste sind momentan hervorgehoben, wird die alte Farbe gesetzt mittels der Funktion **setColor(name,r,g,b)**, switchAntik bekommt den Wert -1 und gleichzeitig wird der Text auf dem Knopf geändert. Ist der Wert ungleich 0 werden die Überreste hervorgehoben, der Variable switchAntik der Wert 0 zugewiesen und der Text auf dem Knopf nach „Rückgängig“ geändert.

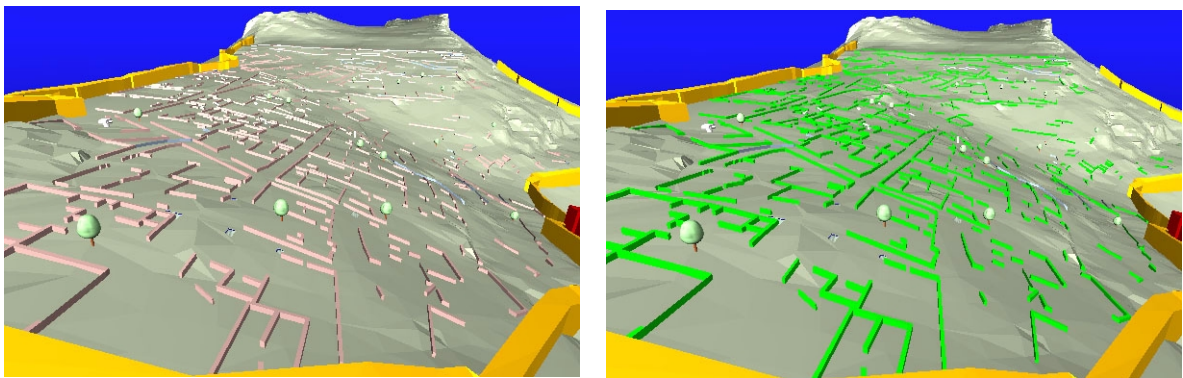


Abbildung 2.14. Antike Überreste nicht hervorgehoben und hervorgehoben

Der Zugriff auf die VRML-Szene muss über VBScript geschehen, mit JavaScript hat man im Internet Explorer keinen Erfolg.

2.4.2 Mittelalterliche Überreste hervorheben

Diese Funktion ist eigentlich praktisch die gleiche, wie die oben besprochene, einzig die JavaScript-Funktion ist leicht anders:

```
function Mitalt()           JavaScript in der Datei „control.html“
{
    name = "MiAlCol";
    if (switchMitAlt==0) {
```

¹⁵siehe Kapitel 2.3.2

```

        setColor(name,0.80784,0.65098,0.61961);
        switchMitAlt=-1;
        document.all.MitAlButton.value="hervorheben";
    }
    else {
        setColor(name,0,0,1);
        switchMitAlt=0;
        document.all.MitAlButton.value="Rückgängig";
    }
}

```

Die Funktion zum Setzen der Farbe ist diesselbe, da der Name des Nodes im jeweiligen Fall übergeben wird. Somit kann diese Funktion mehrfach verwendet werden.

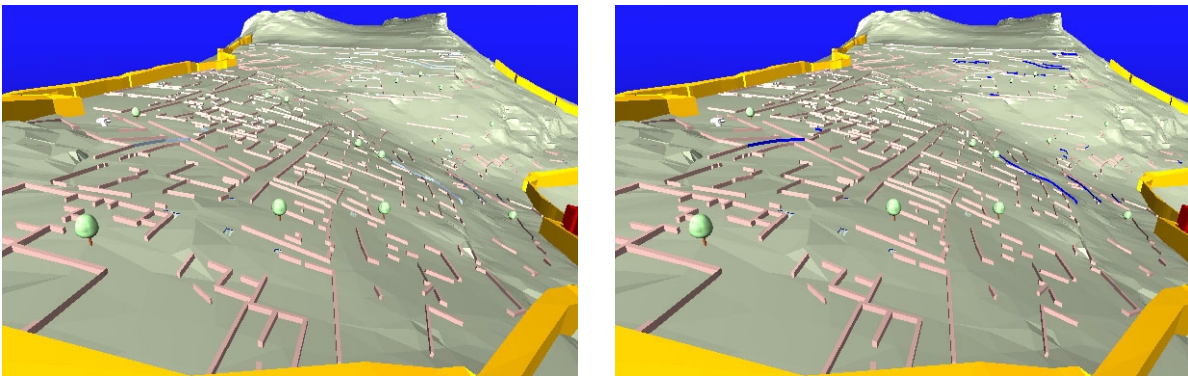


Abbildung 2.15. Mittelalterliche Überreste nicht hervorgehoben und hervorgehoben

2.4.3 Stadtmauer aus-/einblenden

Hier greift man nun auf den in der VRML-Datei definierten Switch¹⁶ zurück. Codebeispiel:

```

function StadtMauer()      JavaScript in der Datei „control.html“
{
    if (switchStadtm==0) {
        setStadtm(0);
        switchStadtm=-1;
        document.all.StadtmButton.value="ausblenden";
    }
    else {
        setStadtm(-1);
        switchStadtm=0;
        document.all.StadtmButton.value="einblenden";
    }
}
Sub setStadtm(value)      VBScript in der Datei „control.html“
    parent.frames("palairos").document.all("Scene").Engine.Nodes("Stadtm").Fields("whichChoice") = value
End Sub

```

Nach dem Drücken der entsprechenden Schaltfläche und somit dem Aufruf der Funktion **StadtMauer()** wird geprüft, was der Wert von **switchStadtm** ist, je nach Wert wird nun der Funktion **setStadtm(value)** der Wert 0 bzw. 1 übergeben, welcher die Stadtmauer einblendet bzw. ausblendet.

¹⁶siehe Kapitel 2.3.2

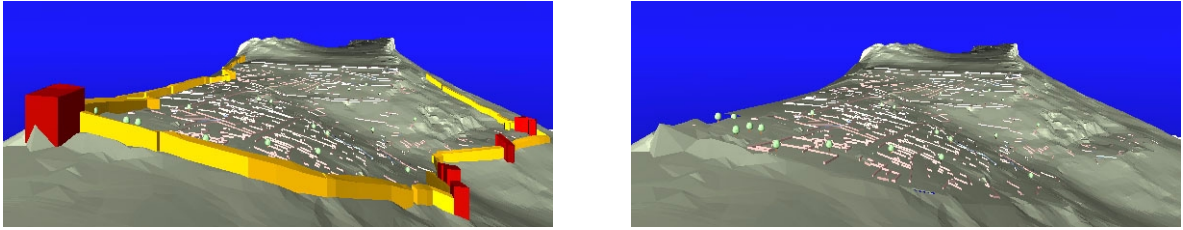


Abbildung 2.16. Stadtmauer eingebledet und ausgebledet

2.4.4 Ansichten wechseln

Die Ansichten entsprechen den im 3D Studio Max vordefinierten Kameras. Deren betreffende Nodes enthalten ein Field „set_bind“, welches entscheidet, welcher Viewpoint gerade aktiv ist. Das heisst, dieses Field wird bei der gewählten Kamera *true* gesetzt, dies geschieht über die Funktion **SetVP(kamera)**.

```
<input type="button" value="Vogelperspektive"
onClick="SetVP('Vogelperspektive')"> HTML-Code in der Datei „con-
trol.html“
Sub SetVP(kamera)          VBScript in der Datei „control.html“
    parent.frames("palairos").document.all("Scene").Engine.Nodes(kam-
era).Fields("set_bind") = true
End Sub
```

Zusätzlich gibt es noch einen manuellen Rundflug, bei dem der Benutzer um eine Station herum weiter oder einen zurück fliegen kann. Dabei interpoliert der Cortona Player die Flugstrecke zwischen den beiden Orten durch eine Fluganimation, welche ganz einfach Viewpoints sind. Codebeispiel:

```
<p>Rundflug:<input type="button" value="zurück" onClick="rund-
flug(-1)"><span id="flugort"></span><input type="button"
value="weiter" onClick="rundflug(1)"></p>HTML-Code in der Datei
„control.html“
function rundflug(n)      JavaScript in der Datei „control.html“
{
    var flug = ["Fahrt01", "Fahrt02", "Fahrt03", "Fahrt04",
"Fahrt05", "Fahrt06", "Fahrt07", "Fahrt08", "Fahrt09", "Fahrt10"];
    flugortnr = flugortnr + n;
    if (test==21) {
        transfer = flug[flugortnr];
        document.all.flugort.innerHTML = ("<b>&nbsp;" + trans-
fer+" </b>");
        test++;
    }
    else {
        if (flugortnr==10) {
            flugortnr = 0;
            transfer = flug[flugortnr];
            SetVP(transfer);
            document.all.flugort.innerHTML = ("<b>&nbsp;" + trans-
fer+" </b>");
        }
        else {
            if (flugortnr==-1) {
                flugortnr = 9;
                transfer = flug[flugortnr];
                SetVP(transfer);
                document.all.flugort.innerHTML = ("<b>&nbsp;" + trans-
fer+" </b>");
            }
        }
    }
}
```

```

        else {
            transfer = flug[flugortnr];
            SetVP(transfer);
            document.all.flugort.innerHTML = ("&nbsp;" + transfer + " </b>");
        }
    }
}

```

Dabei wird der neue Viewpoint wieder mit der Funktion **SetVP(kamera)** gesetzt.

2.4.5 Positionsanzeige

Diese wird realisiert über die Abfrage des ProximitySensor in der VRML-Szene. Damit aber die Positionsanzeige und die Übersichtskarte¹⁷ positionsmässig sich auf der Webseite richtig befinden, braucht es ein folgende Funktionen:

```

function karte()          JavaScript  in der Datei „control.html“
{
    parent.frames("control").document.all.karte.style.pixelLeft = 0;
    parent.frames("control").document.all.karte.style.pixelTop = 450;
    parent.frames("control").document.all.karte.style.zIndex = 3;
    parent.frames("control").document.all.karte.style.position =
    "absolute";
}
function schlaufe()      JavaScript  in Datei „palairos.html“
{
    parent.frames("control").setInterval("outPos()",1000);
}
Sub outPos()             VBScript    in der Datei „control.html“
x0 =
parent.frames("palairos").document.all("Scene").Engine.Nodes("Prox
Sens").Fields("position_changed").X
z0 =
parent.frames("palairos").document.all("Scene").Engine.Nodes("Prox
Sens").Fields("position_changed").Z
x = 1755.00096617 + 0.132332198 * x0 - 0.003717371 * z0
y = 1427.43572734 + 0.0001835935 * x0 + 0.144386416 * z0
x = x - 26           //Offset vom Browserfenster
y = y + 370         //Offset vom Browserfenster
document.all.punkt.style.pixelLeft = x
document.all.punkt.style.pixelTop = y
document.all.punkt.style.zIndex = 10
document.all.punkt.style.position = "absolute"docu-
ment.all.punkt.style.pixelLeft = x
document.all.punkt.style.pixelTop = z
document.all.punkt.style.zIndex = 10
document.all.punkt.style.position = "absolute"
End Sub

```

Die Funktion **karte()** wird beim Laden ausgeführt. Sie setzt die Karte absolut im Internet Explorer Frame „control“. Somit ist gewährleistet, dass beim Skalieren des Browserfensters die Position der Karte immer diesselbe ist. Die Befehle zur Positionierung sind CSS-Funktionen. Gleichzeitig wird, sobald die VRML-Szene geladen ist, aus dieser heraus die Funktion **schlaufe()** aufgerufen, diese fragt nun jede Sekunde über die Funktion **outPos()** ab. In dieser wird nun die momentane Position des Benutzers mittels des ProximitySensors ausgelesen. Achtung: Das Koordinatensystem in VRML ist anders definiert als gewohnt!

17. Diese erhielten wir freundlicherweise von unseren Mitstudentinnen Katrin Aeppli und Janine Sutter

Diese Werte werden nun mit einer Affintransformation gemäss folgender Gleichung transformiert:

$$Y = a + b*x + c*y$$

$$X = d + e*x + f*y$$

Tabelle 2.3. berechnete Parameter der Affintransformation

a	b	c	d	e	f
1427.43572734	0.0001835935	0.144386416	1755.0009667	0.132332198	0.003717371

Danach sind die Koordinaten im gleichen Massstab wie die der Karte, zudem werden sie noch umgerechnet, damit die Position im Browserfenster stimmt.

2.4.6 Legende zu den Fundstücken

Ein weiteres Feature ist die interaktive Legende. Diese ist realisiert in einem Zusammenspiel aus Funktionen in der VRML-Datei¹⁸ und einer Funktion in einer HTML-Seite. Der Code der HTML-Seite ist der folgende:

```
function writeDIV(text)    JavaScript in der Datei „palairos.html“
{
    document.all.description.innerHTML = text;
}
```

Beim Darüberfahren über ein Symbol eines Fundstücks wird eine Funktion ausgelöst, die schliesslich einen String an die Webseite übergibt und zwar an diese, in welcher die VRML-Datei per Plugin geladen wurde. Dieser String sieht folgendermassen aus „*writeDIV(<text>*““, das heisst es wird die Funktion **writeDIV(text)** aufgerufen, der ein Text übergeben wird. Der wird dann in das vordefinierte DIV „description“ als HTML ausgegeben, gegebenenfalls enthaltene HTML-Tags werden vom Browser interpretiert. Ein DIV definiert eine Blockstruktur, dem ein individueller Name gegeben werden kann.

2.4.7 Information zu einem Objekt

Als Beispiel wie man die Informationen zu einem Objekt darstellen könnte, wird an der Stadtmauer gezeigt. Dabei kommt wieder der PROTOTYPE zum Einsatz, diesmal aber übergibt er den Aufruf eines neuen Fensters an den Browser. Quellcode:

```
DEF sc-stadm _3125CD9D-968A-47C6-A771-B55AEB5C8763 {
  txt "javascript:window.open('turm.html','turm','resizable=no,width=470,height=375') "
}
ROUTE touchi.isActive TO sc-stadm.over
```

Mit dem Befehl **window.open** wird die HTML-Seite „turm.html“ aufgerufen, die ein Bild der Stadtmauer enthält.



Abbildung 2.17. Beispielfoto zur Stadtmauer als zusätzliche Information

18.siehe Kapitel 2.3.2

3 Technische Grundlagen

Im Folgenden wird die technische Umsetzung einer virtuellen Realität anhand von ausgesuchten Beispielen näher erläutert. Dazu nehmen wir Bezug zum Quellcode auf der beiliegenden CD-ROM im Anhang¹⁹ zu dieser Arbeit. Für ausführliche Informationen und Programmbeispiele suche man einfach die folgenden Webseiten auf:

- Spezifikationen ISO/IEC 14772-1:1997 VRML97²⁰
- ParallelGraphics Cortona Player²¹
- Microsoft²²
- ECMAScript²³

3.1 Einbinden von Cortona in HTML Dokumente

Das Cortona Plugin kann auf zwei verschiedene Arten in eine HTML-Seite eingebunden werden. Eine Möglichkeit ist das EMBED-Tag, die andere ist das OBJECT-Tag. Letztere Möglichkeit wird in unserer Arbeit benutzt. Das Object-Tag bietet viele Möglichkeiten, das Plugin mit bestimmten Voreinstellungen zu starten. Nachfolgend wird der benutzte Code gezeigt und erläutert:

```
object ID="Scene" TYPE="application/x-oleobject"
      CLASSID="CLSID:86A88967-7A20-11D2-8EDA-00600818EDB1"
WIDTH="640" HEIGHT="425" border="0"
      CODEBASE="http://www.parallelgraphics.com/bin/
cortvrml.cab#Version=3,0,0,49">
  <param name="Scene" value="palairos.wrz">
  <param name="BackColor" value="#FFFFFF">
  <param name="ShowLogo" value="1">
  <param name="NavigationMode" value="1">
  <param name="ColliderMode" value="1">
  <param name="NavigationBar" value="1">
  <param name="viewpoint_transition_mode" value="1">
</object>
```

Über die vergebene ID erhält das Objekt einen Namen und wird damit über Skriptsprachen und CSS angesprochen. Für die Webgestaltung kann auf diese Art auch die Grösse der Szene vorgegeben werden. Die CLASSID ist für jedes ActiveX-Objekt eindeutig, so können also mehrere VRML-Viewer installiert sein, doch es wird in diesem Fall nur der Cortona Player gestartet. Falls das Plugin nicht installiert ist, wird der Browser über CODEBASE aufgefordert, es unter obiger Adresse herunterzuladen.

Anschliessend gibt es noch die Möglichkeit verschiedene Parameter zu setzen, der erste davon ist unbedingt nötig, die anderen sind optional.

- Der erste Parameter gibt die zu ladende VRML-Datei an. Der Zweite die Hintergrundfarbe, falls in der VRML-Datei kein background-node vorhanden ist.
- Der dritte Parameter sagt in diesem Fall aus, dass das Cortona-Logo beim Start gezeigt wird.
- Der nächste Parameter gibt an, welche Fortbewegungsart am Anfang eingestellt ist. In unserem Beispiel haben wir WALK gewählt.
- Als nächstes wird die Kollisionsabfrage eingestellt, somit kann man nicht durch die VRML-Objekte hindurchfliegen.
- Der nächste Parameter enthält die Einstellung, dass die Toolbar standardmässig sichtbar ist.
- Zum Schluss ist noch festgelegt, dass die Bewegungsart zwischen verschiedenen Viewpoints eine Flugbewegung ist. Dies kommt zum Zug, wenn der Benutzer den Viewpoint wechselt.

All diese Parameter sind nur ein kleiner Ausschnitt der Möglichkeiten. Für weitergehende Informationen verweisen wir an dieser Stelle auf das Cortona SDK²⁴.

19.siehe Anhang B

20.VRML97 Spezifikationen <http://www.vrml.org/Specifications/VRML97>

21.ParallelGraphics <http://www.parallelgraphics.com>

22.Microsoft Corp. <http://www.microsoft.com>

23.ECMA 262 Spzifikationen E-262-iii.doc <http://www.ecma.ch>

24.Cortona SDK Dokumentation <http://www.parallelgraphics.com/products/sdk>

4 Präsentation der Daten

Im Folgenden möchten wir eine kurze Einleitung in die Bedienung unserer Homepage geben. Wir beschränken uns auf die für uns wichtigen Funktionen des Cortona VRML Players. Für ausführliche Informationen verweisen wir auf die Homepage des Herstellers ParallelGraphics²⁵.

4.1 Bedienungsanleitung und Benutzerführung

4.1.1 Bedienungselemente

Mit dem Cortona Player können VRML 3D Welten betrachtet und erforscht werden. Das Bedienungsfenster enthält zwei Teile:

- Toolbars:
- Das vertikale Bedienungselement enthält Knöpfe für Einstellungen der Navigation in einer Welt.
- Das horizontale Bedienungselement enthält Knöpfe mit vordefinierten Aktionen zum verändern Ihrer Position in einer Welt.
- Das 3D Fenster, welches die VRML Welt darstellt.
- Über die rechte Maustaste ist auch ein Popup Menü erreichbar, wenn der Mauszeiger über den Bedienungselementen steht.

4.1.2 Navigieren in Cortona

Die Navigation im dreidimensionalen Raum ist vergleichbar mit dem Bewegen einer Kamera. Die Bewegungen, Positionsverschiebungen und Rotationen, des Benutzers werden nun kontinuierlich umgesetzt in die VRML-Szene. Dabei gibt es verschiedene Fortbewegungsmöglichkeiten, die weiter unten erläutert werden. Kurz gesagt, man kann sich eine Person vorstellen, die der Benutzer steuern kann und dessen Sicht auf dem Monitor ausgegeben wird.

Benutzen von Viewpoints

Ein Viewpoint beschreibt eine Position und eine Orientierung in einer Szene. Ein VRML Autor möchte dem Benutzer die besten Ansichten in seiner Welt vorgeben oder eine Führung durch seine Welt definieren.

Um einen Viewpoint zu aktivieren:



- Auf VIEW in der horizontalen Toolbar klicken oder aus dem Popup-Menü einen Viewpoint auswählen.
- Mit dem Vor- bzw. Zurück-Knopf kann man zwischen den vordefinierten Viewpoints umschalten. Oder man drückt die Tastaturknöpfe Page Up bzw. Page Down.

Beachte: Falls keine Viewpoints vordefiniert wurden, erscheint die Nachricht "Empty".

Fortbewegungsarten: Walk, Fly und Study

Im Cortona Player gibt es 3 verschiedene Fortbewegungsarten: WALK, FLY und STUDY. Diese kann man ändern, in dem man auf den betreffenden Knopf in der vertikalen Toolbar drückt.



Jede Fortbewegungsart bietet verschiedene Bewegungsmöglichkeiten: PLAN, PAN, TURN und ROLL. Die Kombination aus gewählter Fortbewegungsart und Bewegung ergibt die mögliche Bewegung und Orientierung der virtuellen Kamera. Vom VRML-Autor wird dabei meistens eine Fortbewegungsart vordefiniert, im realisierten Projekt ist das der Walk-Modus.

Navigieren kann man mit der Maus, der Tastatur oder beidem. Nachfolgend die Erklärung, wie man mit der Maus navigiert:

1. Fortbewegungsart wählen.
2. Mauszeiger irgendwo in der VRML-Szene positionieren und die linke Maustaste drücken.

25. ParallelGraphics <http://www.parallelgraphics.com>

3. Mit gedrückter Maustaste die Maus bewegen, die gewählte Bewegungsrichtung wird dann umgesetzt in eine Bewegung in der VRML-Welt.

4. Beim Loslassen der Maustaste stoppt die Bewegung.

Beachte: Die gezogene Mausdistanz bestimmt die Fortbewegungsgeschwindigkeit. Wenn man stoppt die Maus zu bewegen, geht die Bewegung solange weiter, wie die Maustaste gedrückt bleibt. Um die Bewegungsgeschwindigkeit zu beschleunigen drückt man einfach SHIFT, CTRL oder SHIFT+CTRL.



Gebrauchen Sie WALK+PLAN um sich horizontal in einer Ebene zu bewegen.

Vorwärts – näher bewegen

Rückwärts – zurück bewegen

Rechts – nach rechts drehen

Links – nach links drehen

Beachte: Wird die Maus vorwärts oder rückwärts bewegt, während die Leertaste gedrückt wird, so kann man aufwärts und hinunter drehen.

Wird die Maus nach links oder rechts bewegt, während die ALT-Taste gedrückt wird, so bewegt man sich nach links oder rechts.



Der GOTO-Knopf kann gebraucht werden um direkt ein Ziel anzufliegen, auf das man geklickt hat.

Restore, Fit, und Align

Cortona bietet drei Möglichkeiten, die helfen können sich zu reorientieren, falls man die Orientierung verloren hat. Dabei werden vordefinierte Aktionen ausgelöst, sobald man einen der folgend beschriebenen Knöpfe gedrückt hat.



Mit RESTORE kehrt man automatisch zum zuletzt aktiven Viewpoint zurück.



Mit FIT erhält man eine Übersicht über die ganze Szene.



Mit ALIGN wird die Kameraposition horizontal und longitudinal parallel zur horizontalen Fläche der Szene ausgerichtet.

4.1.3 Optionen im Cortona Plugin

Die folgenden Optionen kann man über Popup-Menu im 3D-Fenster erreichen:

Viewpoints Präsentiert zur Auswähl die vordefinierten Viewpoints.

Headlight Cortona bietet die Möglichkeit einer Lampe für den Benutzer in jeder VRML-Szene. Diese scheint immer direkt in Front des des Benutzer. Man kann sie ein- oder ausschalten.

Navigation Wahlmöglichkeit des Navigationsmodus.

Speed Bestimmt die maximale Fortbewegungsgeschwindigkeit.

Full Screen Damit kann man alles ausser die VRML-Szene ausblenden, damit man diese auf dem ganzen Bildschirm geniessen kann. Falls man wieder zurück schalten möchte, einfach ESC oder F11 drücken.

Hide/Show Toolbars Wahlmöglichkeit die Toolbars ein- oder auszublenden.

Show/Hide Console Wahlmöglichkeit die Konsole anzuzeigen oder nicht, diese gibt Fehlermeldungen und Warnungen aus (gedacht für VRML-Entwickler)

Preferences.... Hier kann man die Einstellungen des Cortona Plugins ändern.

Renderer

Hier kann man die Art wählen wie das Betriebssystem die 3D-Szene rechnen soll. Dabei gibt es einerseits die Wahl zwischen 2 Hardware-Rendermodi: OpenGL und DirectX. Falls ihr System eine Art dieser Hardwaregrafikbeschleunigung davon unterstützt (beachte: es muss mindestens DirectX5 oder höher installiert sein), wählt man den passenden Modus. Je nach Grafikkarte wird nun aber nicht alles dargestellt, deshalb wird empfohlen auch mit dem dritten Modus, Softwarerenderer, zu vergleichen.

Renderer options

- Wireframe rendering mode. Das Objekt erscheint als Drahtmodell.
- Anti-aliasing. Diese Technik wird benutzt, um Treppenstufen bei Linien auf dem Monitor zu glätten und die Darstellungsqualität zu verbessern. Beim Realtime Antialiasing wird die Darstellungsqualität auf Kosten der Framerate verbessert.
- Automatic level of detail. Automatische Reduktion der Anzahl Polygone für ein Objekt die auf dem Monitor mit zunehmender Distanz dargestellt werden.
- Phong Schattierungsmethode. Eine Art die Farbe einer bestimmten Fläche an einem bestimmten Ort zu berechnen. Diese Methode verbessert die Qualität der Szene, aber auf Kosten der Geschwindigkeit. Diese Methode steht aber nur für R98 Renderer im Optimierungsmodus für Intel Pentium III Prozessoren zur Verfügung.
- Collision detection. In der Standardeinstellung, Cortona erlaubt, durch Objekte hindurch zu gehen. Diese Funktionalität kann auch detektiert werden, um einen realen Eindruck der Welt zu vermitteln.

4.1.4 Interaktivität mit der Welt

Einige Objekte in einer Szene können spezielle Effekte auslösen und dem Benutzer erlauben, mit der Szene in interaktiven Kontakt zu treten. Beim Berühren eines Objekts mit einem VRML Sensor, ändert sich der Mauszeiger:



- Touch Sensor. Bemerkt ein Darüberfahren oder gar ein Klicken auf ein Objekt, dies wird aber vom VRML-Autor vordefiniert.

4.2 Benutzerführung auf der Homepage

Dem Benutzer bieten sich zwei kombinierbare Wege an. Einerseits kann er alle vorgesehenen Funktionen des Cortona Players verwenden, andererseits gibt es noch spezielle Knöpfe auf der Webseite mit folgenden Funktionen:

- Antike Überreste hervorheben oder dies rückgängig machen
- Mittelalterliche Überreste hervorheben oder dies rückgängig machen
- Die Stadtmauer ein- bzw. ausblenden
- Verschiedene Ansichten wählen, mit der Möglichkeit einen manuellen Rundflug zu machen

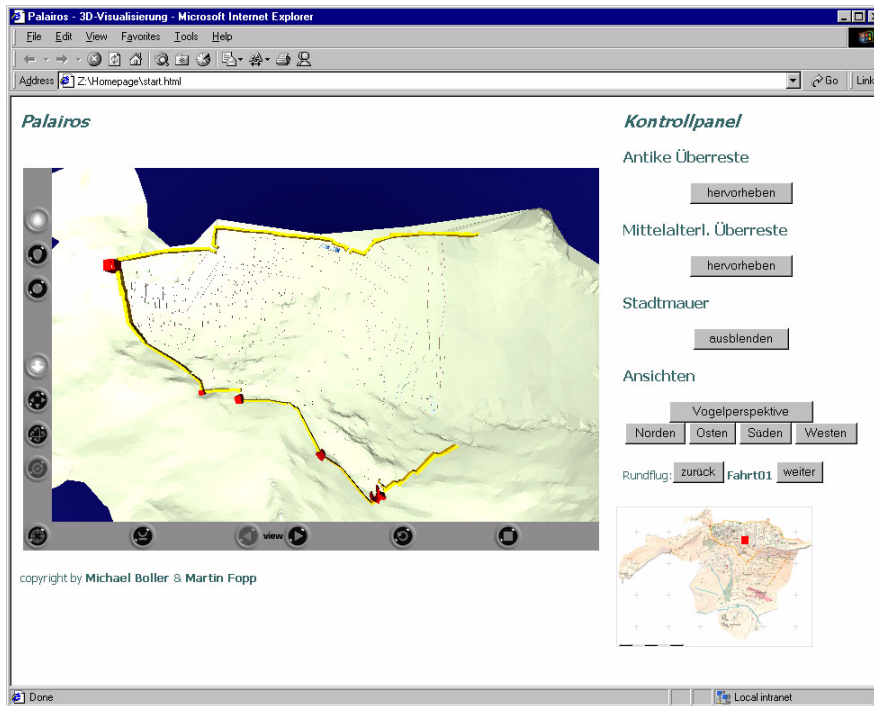


Abbildung 4.1. Bildschirmfoto Homepage

Einzig die verschiedenen Ansichten kann man sowohl über die Webseite, als auch über den Cortona Player erreichen. Nicht zu vergessen ist einerseits die Karte, welche die aktuelle Position angibt, und andererseits die Legende, die unter dem VRML-Szene erscheint, falls der Benutzer gerade über ein definiertes Objekt die Maus bewegt.

Schlusswort

Wir haben uns die Aufgabe gestellt, die archäologische Ausgrabungsstätte Palairos multimedial unter Zuhilfenahme der neuesten Techniken zu präsentieren. Dieses Ziel zu erreichen war viele Male eine Gratwanderung. Wir haben bei unseren Planungen der Kreativität freien Lauf gelassen und je weiter wir uns mit der Materie vertraut gemacht haben, um so kreativer und besser wurden unsere Ideen. Damit tauchte aber auch die Gefahr auf, dass wir uns in Nebensächlichkeiten verirrt und somit wertvolle Zeit verloren, die uns am Schluss fehlte.

Die Zeit wurde zusätzlich noch knapper, da wir uns zuerst in die uns unbekannte Sprache VRML einarbeiten mussten, was etwa vier Wochen in Anspruch nahm. Dennoch war dies keine verlorene Zeit, da die gelernten Fähigkeiten am Schluss bei der Optimierung der VRML-Datei und der Programmierung der einzelnen Funktionen gut genutzt werden konnten.

Für uns war es eine spannende Angelegenheit, sich ein ganzes Semester auf ein Projekt zu konzentrieren und somit auch zu einem ansprechenden Ergebnis zu gelangen. Dazu möchten wir an dieser Stelle noch einmal speziell Andreas Neumann und Uta Samaga danken, die uns während des Semesters begleitet haben und uns immer hilfreich zur Seite standen, wenn Probleme auftauchten.

Ausblick

Natürlich konnten nicht alle Ziele verwirklicht werden, zudem kamen während der Arbeit noch viele neue Ideen hinzu. Deshalb sehen wir vor allem in folgenden Punkten die Chance, die nun erstellte VRML-Szene einerseits aufzuwerten, andererseits auch einem grösseren Publikum zugänglich zu machen:

- Unterstützung weiterer Browser (Netscape, Opera)
- Unterstützung weiterer Plugins (v. a. Cosmo Player)
- Level of Detail (LOD)
- Übersichtskarte in SVG realisieren
- Integration der restlichen Stadtmauer
- Integration der Objekte und Gebäude des noch nicht bearbeiteten Gebietes
- Rekonstruktion der Überreste
- vordefinierte Pfade erstellen
- vordefinierte und automatisch ablaufende Rundflüge bzw. Rundgänge

Quellenverzeichnis

Hurni, L.(1999): Vorlesungs- und Übungsunterlagen ETHZ, Zürich

Oberhauser, N.(1999): Diplomarbeit Topographische Aufnahme Palairos...



VRML Spezifikationen ISO/IEC 14772-1 (1997): Virtual Reality Modeling Language (VRML97)
<http://www.vrml.org/Specifications/VRML97>

ECMAScript Language ISO/IEC 16262 Specification 3rd Edition December 1999 <http://www.ecma.ch/ecma1/stand/ecma%2D262.htm>

ParallelGraphics SDK Documentation

Koch, S. (1999): JAVA Script, dpunkt.verlag, ixEDITION

KINETIX (1999): 3D Studio Max Release 3 Reference Volume I und II, AutoDesk Inc.

KINETIX (1999): 3D Studio Max Release 3 Tutorials, AutoDesk Inc

Goodman, D. (1998): Dynamic HTML The definitive Reference, O' Reilly

Flanagan, D. (1998): JavaScript The definitive Guide, O' Reilly

Brandenbaugh, J. (2000): JavaScript Kochbuch für Web-Anwendungen, O'Reilly

Abbildungsverzeichnis

Übersichtskarte, Lage von Palairos	7
Siedlungsplateau und Südhang der antiken Stätte Palairos	8
Historische Lage von Palairos	9
Bildschirmfoto aus der Ebene Verbindung	15
Screenshot aus AutoCAD	15
Bildschirmfoto (die violetten Elemente stammen aus der Ebene "Felsen")	16
Stützpunktreihenfolge	18
Bildschirmfotos Bruchkanten	19
Bildschirmfotos Treppen	19
Bildschirmfoto Ausgangssituation	20
Bildschirmfoto modellierte Treppenform	20
Treppe im Gelände	20
Fundsymbole	21
Baumsymbol	21
VrmlPad-Screenshot	22
UltraEdit32-Screenshot	26
Antike Überreste nicht hervorgehoben und hervorgehoben	27
Mittelalterliche Überreste nicht hervorgehoben und hervorgehoben	28
Stadtmauer eingeblendet und ausgeblendet	29
Beispielfoto zur Stadtmauer als zusätzliche Information	31
Bildschirmfoto Homepage	36

Abkürzungen

CSS	Cascading Style Sheets
DAI	Deutsches Archäologisches Institut
D-BAUG	Departement für Bau, Umwelt und Geomatik
DHM	Digitales Höhenmodell
DTM	Digitales Terrainmodell
DXF	Data Exchange Format
ETHZ	Eidgenössische Technische Hochschule Zürich
HTML	Hyper Text Markup Language
IKA	Institut für Kartographie
Kult-Ing	Kulturingenieur
TIN	Triangulated Irregular Network
VTB	Vertiefungsblock
VRML	Virtual Reality Modeling Language

B Inhalt CD-ROM

- Homepage
- Präsentation (CD-ROM-Hüllen, Poster und Schlusspräsentation)
- Quellenliteratur
- 3D Studio Max Datei
- Technischer Bericht (Framemaker 6.0 -Dateien und PDF)
- VRML Grundlagen (Cortona Plugin, Cortona SDK, JavaScript Doc, VRML97 Specifications, VrmlPad)