DOMINIC MICHAEL LAURENTIUS

VISUALISIERUNG ARCHÄOLOGISCHER DATENSÄTZE

Diplomarbeit im Fach Informatik

Themensteller: Prof. Dr. U. Lang

Vorgelegt in der Diplomprüfung im Studiengang Wirtschaftsinformatik
der Wirtschafts- und Sozialwissenschaftlichen Fakultät der Universität zu Köln

Köln, 2014

Inhaltsverzeichnis

Abkürzungsverzeichnis....IV
Abbildungsverzeichnis....V
1 Einleitung....1
1.1 Problemstellung....1
1.2 Zielsetzung....2
2 Verfahren zur Erfassung und Archivierung von archäologischen Fundstücken....3
2.1.1 Standardisierte Zeichnungen....3
2.1.2 Fotografie....5
2.1.3 Flachbettscanner....6
2.1.4 Tachymetrie....6
2.2 Neue Möglichkeiten durch Einsatz von 3D-Scanning und virtueller Darstellung. 7
2.3 Überblick über NESPOS und dort im Einsatz befindliche Software....12
2.3.1 ArteCore....12
2.3.2 OptoView....14
2.3.3 Meshlab....16
3 Entwicklung einer neuen Software....18
3.1 Anforderungsdefinition....18
3.2 Qt-Bibliothek....22
3.3 RPly-Format und RPly-Bibliothek....23
3.4 Aufbau der Software....27
3.4.1 about.cpp / about.h / about.ui....28
3.4.2 geometry.cpp / geometry.h....28
3.4.3 glwidget.cpp / glwidget.h / resources.qrc....30
3.4.4 main.cpp....35
3.4.5 mainwindow.cpp / mainwindow.h / mainwindow.ui....36
3.5 Benutzeroberfläche und Bedienung....40
3.5.1 Menü....41
3.5.2 3D-Anzeigebereich....41
3.5.3 Lichter....42
3.5.4 Rotation....43
3.5.5 Visuelle Verbesserungen....45
3.5.6 Anmerkungen und Punkte....50

3.5.7 Informationen zum Artefakt....53
4 Test....54
4.1 Aufgaben....55
4.2 Kriterien....56
5 Ergebnisse....57
6 Diskussion der Testergebnisse....61
7 Ausblick & Fazit....64
7.1 Große Datenmengen....64
7.2 Kantenfindung und Texturen....65
7.3 Datenbankintegration....65
7.4 Ausbessern von Löchern....65
7.5 Kombinieren von Artefakten....66
7.6 Sprachen & Dokumentation....66
Literaturverzeichnis....IV
Anhang A – Aufgabenliste für den Test....VIII
Anhang B - BSD-2-Clause-Lizenz....XII
Erklärung....XIII
Lebenslauf....XIV

Abkürzungsverzeichnis

CTR	ConTaineR – ein Format für 3D-Daten der Firma Breuckmann
LGPL	Lesser General Public Licence
NESPOS	Neanderthal Studies Professional Online Service
PLY	PoLYgon – ein an der Universität Stanford entwickeltes Format für 3D-Daten mit Farbinformationen und anderen Zusatzinformationen
STL	Surface Tesselation Language, auch Standard Triangulation Language – ein Dateiformat zur Speicherung von 3D-Daten
UI	User Interface

Abbildungsverzeichnis

1 Exemplarisch ein Scanner der Firma Breuckmann (Bild von Firmenwebsite)
2 Vierfachsicht in ArteCore
3 Flächenmesswerkzeug in ArteCore
4 OptoView
5 Meshlab
6 Die Dateien des Projects Artefact Viewer
7 Benutzeroberfläche des Artefact Viewers
8 Beleuchtung
9 Rotation
10 Visuelle Verbesserungen
11 Dark-Edges-Algorithmus
12 Erkennung von Berg und Tal
13 Anmerkungen schreiben und Vermessungen vornehmen
14 Winkelmessung

1 Einleitung

1.1 Problemstellung

Diese Arbeit befasst sich mit der Aufgabe, eine Software zu entwickeln, die von Archäologen dazu genutzt werden kann, 3D-Scandaten von Artefakten in vielfältiger Art zu verwenden – einerseits als jederzeit und überall verfügbarer Ersatz für das Originalfundstück, andererseits als Hilfsmittel, bestimmte Eigenschaften präziser, schneller oder leichter wahrzunehmen als am Original.

Der Anstoß dazu kam vom Neanderthal-Museum in Mettmann, wo bereits seit Anfang des Jahrtausends mit 3D-Modellen von Artefakten gearbeitet wird. Dazu kamen zunächst vor allem Dateien im STL-Format zum Einsatz, für die im Rahmen eines EU-Projekts eine spezielle Software entwickelt wurde, die ausschließlich STL-Dateien lesen konnte und auf die später noch eingegangen wird.

Mit dem Fortschreiten der Technik werden 3D-Scans heute jedoch vor allem im PLY-Format erzeugt und gespeichert, weil dieses Format zusätzlich Farbinformationen aufnehmen und die Geometrie deutlich kompakter speichern kann. Da die ursprüngliche Software keine PLY-Dateien lesen konnte, wurden im Lauf der Zeit zusätzlich andere frei verfügbare Programme genutzt, um mit dem neuen Dateiformat arbeiten zu können. Diese sind jedoch von ihrer Funktionalität her nicht an die Bedürfnisse der Archäologen angepasst, sondern stellen lediglich beliebige 3D-Modelle - unter anderem im PLY-Format - dar. Es gibt keine Vermessungs- und Kommentarfunktionen und häufig ist es nicht möglich, aus den Anwendungen brauchbare Bilder zur Publikation zu generieren. Im wissenschaftlichen Bereich und zum Einsatz in der Lehre ist es zudem sehr wünschenswert, wenn die Software kostenlos, frei verfügbar und erweiterbar ist sowie leichtgewichtig in dem Sinne, dass sie im Gegensatz zu allgemeinen 3D-Programmen nur genau die Funktionen zur Verfügung stellt, die auch benötigt werden. Auf diese Weise bleibt sie leicht erlernbar. Diese Anforderungen werden von keinem der eingesetzten Programme gänzlich erfüllt.

Die ursprünglich eingesetzte Software zu modernisieren, so dass sie auf aktuellen Betriebssystemen wieder lauffähig gewesen wäre und alle zusätzlich genutzten Funktionen gehabt hätte, erschien aufgrund des Alters ihrer selbst sowie der zahlreichen Bibliotheken von über 10 Jahren und der Menge an Details, die zusätzlich implementiert werden mussten, wenig effizient.

1.2 Zielsetzung

Die Herausforderung bestand darin, eine völlig neue Software zu schreiben, die all jene Funktionen der alten Software zusammenfasste, die noch als nützlich empfunden wurden, und dazu die vielen neuen Möglichkeiten bot, die von den Wissenschaftlern gewünscht waren. Diese wurden in Form einer Anforderungsdefinition nach Wichtigkeit gestaffelt festgehalten und im Laufe der Entwicklung durch zahlreiche Gespräche erweitert und konkretisiert.

Neben den funktionellen Anforderungen sollte es außerdem möglich sein, die Software in Zukunft jederzeit unentgeltlich zu verbreiten und zu erweitern, weswegen sie quelloffen und frei verfügbar sein sollte. Um die leichte Erweiterbarkeit möglich zu machen, die Einfachheit zu wahren und bestmöglich zu garantieren, dass die Software auch in Zukunft möglichst lange weiterentwickelt und verbessert werden kann, sollte die Software auf der Qt-Bibliothek aufgebaut werden, die einerseits viele Werkzeuge zur Arbeitsersparnis bereithält und andererseits ständig weiterentwickelt wird.

Die vorliegende Arbeit beschreibt die Ausgangslage, den Entwicklungsprozess der neuen Software und schließlich den Testlauf und die daraus zu ziehenden Schlussfolgerungen und möglichen zukünftigen Verbesserungen.

2 Verfahren zur Erfassung und Archivierung von archäologischen Fundstücken

Wenngleich es für die Archäologie kein Gründungsdatum gibt, kann doch zumindest gesagt werden, dass sie etwa im 18. Jahrhundert aufkam und im Laufe des 19. Jahrhunderts zu einer anerkannten Wissenschaft wurde[1]. In den über hundert Jahren Entwicklung, die die Archäologie bis heute hinter sich hat, haben sich stets neue Anforderungen gestellt, für die es adäquate Lösungen zu finden galt. So wuchs mit wachsender Zahl an Fundstücken und den Forschern, die sich damit beschäftigten, gleichermaßen die Notwendigkeit, Archive zu schaffen und diese schneller durchsuchbar zu machen, um möglichst effizient Fundstücke zu sichten und miteinander vergleichen zu können. Außerdem zeigte sich bald, dass die Archäologie sehr eng mit anderen Wissenschaften in Verbindung steht, beispielsweise der Geologie und der Anthropologie, da neben den Fundstücken selber auch der Fundort interessante Informationen über die Entwicklung des Menschen enthalten kann. So kamen weitere Forschergruppen hinzu.

Die Erkenntnisse von Geologen halfen dabei, spezielle Gebiete zu finden, an denen besonders alte Artefakte geborgen werden können, wie beispielsweise den großen afrikanischen Grabenbruch. Dort konnten tiefere Erdschichten deutlich leichter untersucht werden als an anderen Stellen, wo im Laufe der Jahrtausende stetig neue Erdschichten hinzukamen und die Zeugnisse der Vergangenheit zunehmend tiefer unter sich begruben, was Ausgrabungsarbeiten entsprechend teurer machte.

2.1.1 Standardisierte Zeichnungen

Zu jedem Aufbewahrungsort von Fundstücken hin zu reisen und bestimmte Fundstücke im Original in Augenschein zu nehmen, war für Forscher sehr zeit- und kostenaufwändig. Daher entwickelte sich Ende des 19. Jahrhunderts[2] eine standardisierte Form der (Hand-) Zeichnung von Artefakten, die sogar Teil der Ausbildung von Studenten wurde[3]. Ziel war es, das Objekt einerseits in seiner Ästhetik möglichst korrekt

---

1 de.wikipedia.org, Artikel »Archäologie«

2 Adkins, Adkins 1989, S. 5

3 Pastoors, Weniger 2011

abzubilden und gleichzeitig die wissenschaftlich relevanten Merkmale der Beschaffenheit des Materials und der Spuren von Bearbeitung durch standardisierte zeichnerische Darstellung und einheitliche Symbole hervorzuheben.

Diese zeichnerische Darstellung hat sich bewährt, obgleich sie einige offensichtliche Probleme aufwirft. Ein grundlegendes Problem ist die Kombination der Darstellung von Ästhetik, die die Darstellung des Umrisses und der hervorstechendsten Merkmale der Oberfläche umfasst, mit der Darstellung der physischen Beschaffenheit, also etwa Haarrissen, Abnutzungs- und Bearbeitungsspuren, da diese die Zeichnung anders aussehen lassen als das reale Fundstück – insbesondere für den ungeübten Betrachter.[4]

So wird durch Schraffierungen und andere Linienformen die Art des Steines und seiner Bearbeitung dargestellt. Sie zeigen an, ob es sich beispielsweise um Kalkstein, Sandstein oder Quarz handelt, ob die Oberfläche durch einen Schlag entstanden ist oder durch einen Schliff. Zusätzliche Symbole geben weitere Informationen, darunter Andeutungen, welche Stellen wohl durch einen Gebrauch als Werkzeug einer Abnutzung unterlegen haben und wie diese ursprünglich ausgesehen haben könnten und wo Beschädigungen vorliegen[5].

Um den räumlichen Eindruck zu verstärken, werden Schatten so gezeichnet, als käme das Licht von links oben, etwa über der linken Schulter des Zeichners. Dabei bleibt die Darstellung aber stets schematisch, nie realistisch.[6]

All dies verdeutlicht, dass besagte Zeichnung eine subjektive Darstellung des Gezeichneten durch den Zeichner ist, was naturgemäß mit einem Verlust von Informationen einher geht. So stellt die Zeichnung also die Schnittmenge aus den im Fundstück enthaltenen Informationen und dem Wissensstand beziehungsweise Können des Zeichners dar. Der spätere Betrachter kann sich nie sicher sein, dass alles für ihn Relevante auch mit der nötigen Genauigkeit erfasst und so interpretiert wurde, wie auch er es getan hätte. Eine spätere Sichtung unter Berücksichtigung neuer Erkenntnisse oder Verfahren erfordert darüber hinaus die erneute Untersuchung des Originalfundes.

Insgesamt ist die Handzeichnung ein aufwendiges Verfahren, insofern als der Zeichner

---

4 Pastoors, Weniger 2011

5 Inizan, Reduron-Ballinger, Roche, Tixier 1999

6 Lindinger, Hörr 2006/2007

entsprechende archäologische Fachkenntnisse und zusätzlich Kenntnisse sowie Übung im Zeichnen haben muss und schließlich jede Handzeichnung auch digitalisiert und nachbearbeitet werden muss, um publiziert zu werden. Die gesamte Verarbeitungszeit pro Objekt liegt dabei zwischen einer Stunde und einem Tag.[7] Trotzdem lohnt der Aufwand, denn dem späteren Betrachter fallen relevante Merkmale des Objekts sofort ins Auge – zumindest die, die der Zeichner für darstellenswert hielt.

2.1.2 Fotografie

Mit Hilfe von Fotos wurde versucht, die Objektivität bei der Beurteilung eines Artefakts zu steigern. Was auf den ersten Blick wie die offensichtliche Lösung des Problems aussieht, führt bei näherer Betrachtung zu neuen Unzulänglichkeiten. So teilt sich die Fotografie mit der Zeichnung die Eigenschaft der Zweidimensionalität. In beiden Fällen geht zunächst eine Dimension bei der Abbildung verloren und dies erschwert beispielsweise das Erkennen von Strukturen auf der Oberfläche. Eine wichtige Rolle spielt zur Kompensation dieses Problems die Beleuchtung, die so eingestellt werden muss, dass ein gewisser Schattenwurf wieder einen räumlichen Eindruck entstehen lässt.

Bei hoher Auflösung lassen sich so die Details des Fundstückes gut dokumentieren, jedoch werden dafür bisweilen, um alle Einzelaspekte richtig in Szene zu setzen, viele Aufnahmen mit unterschiedlichen Beleuchtungen benötigt, was einerseits letztlich zeitaufwändiger ist, als eine Zeichnung anzufertigen, und andererseits die Publikation erschwert, da auf nur einem oder zwei häufig kleinen Bildern nicht alle Merkmale sichtbar sind, auf die der zugehörige Text eingeht. Auch ist der Kostenaufwand für den Druck von Fotos höher als für den Druck von Schwarzweiß-Skizzen.[8] Daher haben sich Fotos zur Dokumentation unter Experten bislang nur einen gewissen Anteil sichern können[9]. Sie finden vor allem bei besonders bedeutsamen Fundstücken Anwendung. Zur Präsentation von Artefakten für ein nichtfachkundiges Publikum sind Fotografien allerdings deutlich besser geeignet als Zeichnungen.

---

7 Gilboa, Tal, Shimshoni, Kolomenkin 2013

8 Lindinger, Hörr 2006/2007

9 Adkins, Adkins 1989, S. 6

2.1.3 Flachbettscanner

Eine relativ einfache und kostengünstige Möglichkeit für bestimmte Fundstücke ist die Erfassung via Flachbettscanner, die zumindest am Landesamt für Denkmalpflege und Archäologie in Sachen-Anhalt zum Einsatz kommt.[10] Aufgrund der festen Brennweite des Scanners und mangelnder Schärfentiefe ist dieses Verfahren jedoch auf sehr flache Objekte beschränkt. Die Größe der Fläche des Scanners schränkt die Größe der Objekte zusätzlich ein. Während Knochen(-stücke) weicher als Glas sind und somit keine Gefahr für die Glasplatte des Scanners darstellen, könnten Scherben und vor allem Steine ihre Lebensdauer selbst bei vorsichtigem Umgang deutlich einschränken.

2.1.4 Tachymetrie

Nicht alle Fundstücke sind kleine Objekte, die man einsammeln und mitnehmen kann. Auch beispielsweise Mauern oder Höhlenmalereien müssen erfasst werden können. Mithilfe der Tachymetrie kann eine Mauer steingenau vermessen und kartografiert werden.[11] Dafür werden vom Standpunkt des Tachymeters Winkel und Entfernungen zu markanten Punkten gemessen und daraus durch trigonometrische Berechnungen die Lage der Punkte und somit auch die Größe der von ihnen umschlossenen Objekte ermittelt. Bei den ursprünglichen Tachymetern mussten die Werte manuell abgelesen werden, heutige Systeme arbeiten mit deutlich genauerer elektrooptische Entfernungsmessung und erfassen die Daten automatisch. Die Tachymetrie kann auch in Verbindung mit den im nächsten Abschnitt beschriebenen 3D-Scannern verwendet werden, um einzeln gescannte Bereiche zu vermessen und zusammenzufügen. Um eine Vielzahl von Scans über eine große Fläche mit ausreichender Präzision zusammenzufügen, ist sie sogar unerlässlich.[12]

---

[10] Lindinger, Hörr 2006/2007

[11] Fichtmüller, Kießling 2008

[12] Fichtmüller, Kießling 2008

2.2 Neue Möglichkeiten durch Einsatz von 3D-Scanning und virtueller Darstellung

Seit den ersten Versuchen in den 90er Jahren des 20sten Jahrhunderts, Computer für die Erfassung und Archivierung archäologischer Objekte nutzbar zu machen, hat die Technik enorme Fortschritte gemacht. Laserscanner ermöglichten nun die Erfassung von großen Wandflächen aber auch kleinen Objekten wie steinernen Artefakten, Knochen, Gefäßen oder Schmuck.

Trotz verschiedener Gerätehersteller ist das Verfahren zur Erfassung stets ähnlich: Ein Laser überstreicht mit einer oder mehreren Linien das zu erfassende Objekt und wird dabei von verschiedenen Kameras gefilmt. Aus den Bildinformationen kann dann auf die Oberflächenbeschaffenheit geschlossen werden. Folgerichtig hängt die Auflösung der späteren 3D-Daten ganz wesentlich von der Auflösung der verwendeten Kameras ab.[13] Um die Farben an den jeweiligen Messpunkten korrekt erfassen zu können, wird zusätzlich eine gute Beleuchtung benötigt, die farbneutral alle Seiten gleichmäßig erhellt.

Abbildung 1: Exemplarisch ein Scanner der Firma Breuckmann (Bild von Firmenwebsite)

[13] Lindinger, Hörr 2006/2007

Laserscanner können jedoch kaum sinnvoll ohne auf die Situation angepasste Hilfsmittel benutzt werden. So werden für Wandscans in der Regel mehrere Messpunkte benötigt, die nur mit Hilfe von Tachymetrie so vermessen werden können, dass die späteren Einzelscans wieder zu einem Gesamtbild zusammengefasst werden können.[14] Für kleine Objekte dagegen wird ein Drehteller benötigt, um das Objekt von allen Seiten erfassen zu können. Allerdings können Einbuchtungen im Objekt, die die Kamera aus keinem Blickwinkel sehen kann, grundsätzlich nicht erfasst werden und bleiben später als Lücken im 3D-Modell bestehen. Eine Schließung dieser fehlenden Teile durch Spezialaufnahmen aus anderen Blickwinkeln ist bislang noch nicht wirtschaftlich möglich[15].

Die Präzision der Scanner hat sich seit deren Einführung stetig erhöht, so dass es nunmehr möglich ist, Objekte mit einer Auflösung von 10µm[16] bis 50µm[17] zu scannen. Dies ermöglicht theoretisch eine Erfassung von allen Details, die das menschliche Auge erfassen könnte, so dass der Informationsverlust deutlich geringer ist, als bei einer Handzeichnung.

Nicht erfassbar sind derzeit nur sehr kleine Objekte. Steine mit Quarzeinlagerungen oder Gefäße mit glänzenden Oberflächen, die den Laser reflektieren und so Löcher im späteren 3D-Modell entstehen lassen, müssen vor dem Scanvorgang mit einem Spray behandelt werden, welches die Oberfläche mattiert und später wieder restlos und ohne Beschädigung der Oberfläche entfernbar ist.[18] Es ist davon auszugehen, dass kommende technische Verbesserungen noch höhere Auflösungen ermöglichen und die Bandbreite scanbarer Objekte weiter vergrößern werden[19].

---

[14] Fichtmüller, Kießling 2008

[15] Lindinger, Hörr 2006/2007

[16] Lindinger, Hörr 2006/2007

[17] Schaich 2008

[18] Lindinger, Hörr 2006/2007

[19] Pastoors, Weniger 2011

Für weitere Hinterlassenschaften des Menschen, wie seine Knochen und Zähne, können andere moderne Methoden wie die Computer-Tomographie zum Einsatz kommen, durch die sogar volumetrische Ansichten des Inneren generiert werden, ohne das Fundstück selber zu zerstören[20]. Dies ist ein Beispiel dafür, dass der elektronische Datensatz sogar mehr Informationen enthält, als man dem Originalfundstück mit dem bloßen Auge ansehen kann.

Ein weiterer wichtiger Vorteil, der aus der Präzision und Schnelligkeit der Erfassung von Artefakten via Laserscanner beruht, ist die Möglichkeit, mehrere Abbilder eines Gegenstands in verschiedenen Stadien zu erfassen und dauerhaft für die Nachwelt festzuhalten. So kann Material bei der Restauration verloren gehen[21] oder bei der Probenentnahme kleiner Teile zwecks wissenschaftlicher Analyse[22], daher kann es nützlich sein, den Zustand vor diesen Arbeiten festzuhalten. Und schließlich können Fundstücke generell in politisch weniger stabilen Regionen leicht Opfer von Zerstörung werden[23]. Selbst in vermeintlich sicheren Ländern wie Deutschland kann ein komplettes Gebäude zusammenstürzen und ganze Sammlungen zerstören, beziehungsweise schwer beschädigen, wie beispielsweise beim Einsturz des Kölner Stadtarchives[24], auch wenn dort vor allem Schriften verschüttet wurden. Daher bietet die digitale und verteilte Speicherung von 3D-Scans einen großen Sicherheitsgewinn und tritt nicht nur in Konkurrenz zu anderen Abbildern, sondern sogar zu dem Originalfund.

Auch wenn diese Arbeit und die im Folgenden beschriebene Software derzeit primär auf Fundstücke ausgerichtet ist, die in der Hand gehalten werden können, so sei wegen der durchaus großen Bedeutung des Laserscannens für große Funde wie Bauwerke an dieser Stelle auch darauf hingewiesen, dass das virtuelle Abbild bisweilen die einzige realisierbare Möglichkeit darstellt, den Fund entweder an einem anderen Ort auszustellen oder überhaupt für spätere Untersuchungen möglichst realitätstreu zu erhalten – schließlich kann sich die Wichtigkeit eines Fundes oder einzelner Merkmale auch erst später herausstellen.

---

[20] Bayle, Bondioli, Macchiarelli, Mazurier, Puymerail, Volpato, Zanolli 2011

[21] Lindinger, Hörr 2006/2007

[22] Lindinger, Hörr 2006/2007

[23] Schaich 2008

[24] de.wikipedia.org, Artikel »Historisches Archiv der Stadt Köln«

Derzeit (2014) zwar noch recht neue, aber bereits durchaus nutzbare Technologien wie der 3D-Druck[25] oder spezielle Anzeigegeräte wie 3D-Brillen, 3D-Bildschirme oder 3D-Leinwände, die den Anwender in virtuelle 3D-Welten eintauchen lassen, dürften zusammen mit den gescannten 3D-Daten in Zukunft interessante Möglichkeiten des Replizierens und virtuellen Besichtigens eröffnen.

Passend dazu sind Rechner allgemein immer leistungsfähiger geworden, so dass auch größere Datenmengen, wie sie bei der derart detaillierten Darstellung von Artefakten anfallen, keine Spezialhardware mehr benötigen, sondern von normalen Arbeitsplatzrechnern verarbeitet werden können, ggf. auch von tragbaren Geräten, was die Verwendung direkt an der Ausgrabungsstätte ermöglicht. Bei leistungsfähigen mobilen Rechnern sind zwei Gigabyte Speicher für die Grafikkarte keine Seltenheit mehr (Stand 2014), was es ihnen ermöglicht, problemlos 3D-Modelle mit mehreren Millionen Knoten mit akzeptabler Bildrate anzuzeigen, was mit der im Rahmen dieser Arbeit entstandenen Software auch demonstriert wird.

Und nicht zuletzt hat sich die Größe von Festplatten immens vergrößert, während der Preis pro Gigabyte dabei stark gefallen ist. Ähnliches gilt für Bandbreite im Internet, so dass die Bewegung großer Datenmengen und somit der schnelle Zugriff auf umfangreiche Datenbanken mit detaillierten 3D-Scans kein Problem mehr darstellt.

Die größere Herausforderung ist es nunmehr, für eine zuverlässige Speicherung von Daten zu sorgen, damit diese dauerhaft für die Menschheit insgesamt und die Forscher im Speziellen erhalten bleiben. Denn nur dann ist die digitale Speicherung und Sichtung von Objekten eine echte Alternative zu Katalogen mit Handzeichnungen und der persönlichen Inaugenscheinnahme. Dies ist jedoch nicht nur ein technisches, sondern vor allem ein organisatorisches Problem, da hierzu erhebliche finanzielle Mittel benötigt werden, auch um das Vertrauen zu schaffen, dass eine digitale Speicherung ebenso sicher und zugänglich ist, wie konventionelle Archive. Ein Ansatz dazu ist das NESPOS-Projekt (www.nespos.org).

---

[25] Fastermann 2012

Die Speicherung und Typisierung der Fundstücke nach allgemein anerkannten Kriterien und Merkmalen[26] ist unerlässlich, um einen großen durchsuchbaren digitalen Datenbestand zu schaffen und sinnvoll mit 3D-Daten zu arbeiten, umfasst aber einen eigenen Themenbereich, der nicht Teil dieser Arbeit sein soll.

Im Fokus dieser Arbeit steht das Betrachten, Analysieren, in Szene setzen und Kommentieren von einzelnen Fundstücken, welches mit verschiedener Software bereits teilweise ermöglicht wird.

[26] Eggert 2001, S. 133-142

2.3 Überblick über NESPOS und dort im Einsatz befindliche Software

Im Rahmen des NESPOS-Projekts, welches ins Leben gerufen wurde, um 3D-Datensätze über das Internet zugänglich zu machen[27], existieren bereits einige Programme, die für einzelne Aufgabenstellungen zu gebrauchen sind. Auf diese Programme und Funktionen wird im Folgenden eingegangen.

2.3.1 ArteCore

ArteCore entstand eigens für das NESPOS-Projekt im Rahmen des Projektes TNT (The Neanderthal Tools)[28], welches ein Gemeinschaftsprojekt des Neanderthal Museums mit unter anderem dem Königlichen Institut für Naturwissenschaften in Brüssel, dem kroatischen Museum für Naturgeschichte in Zagreb und der Universität Poitiers war[29]. Es wurde von der Firma Art+Com entwickelt und die letzte Version stammt aus dem Jahr 2003.

Das für Software hohe Alter von zehn Jahren erwies sich bei der Evaluation der Software sogleich als Problem, da sich das Programm auf einem mit Microsofts Windows 7 ausgestatteten Rechner nicht installieren lässt. Glücklicherweise wurde ein Rechner mit Windows XP zur Verfügung gestellt, auf dem die Funktionen demonstriert werden konnten, so dass sich ein genaueres Bild der Anforderungen und der aktuell vorhandenen und genutzten Möglichkeiten ergab.

ArteCore kann keine PLY-Dateien einlesen, sondern nur STL-Dateien. Da diese keine Farbinformationen speichern können (bis auf einige proprietäre Ansätze, wenigstens rudimentäre Farbunterstützung einzubauen, die jedoch nicht genutzt werden), ist die Ansicht der Artefakte grundsätzlich einfarbig. Der Anwender kann dabei einstellen, ob er den Bildschirm aufteilen und das Objekt in einer Vierfachansicht (4-View) betrachten will (von oben, von vorne, von der Seite und perspektivisch gleichzeitig, jeweils konfigurierbar) oder sich auf eine perspektivische Vollbilddarstellung beschränken will.

---

[27] Gröning, Kegler, Weniger 2007

[28] Semal, Kirchner, Macchiarelli, Mayer, Weniger 2007

[29] Macchiarelli, Weniger 2011

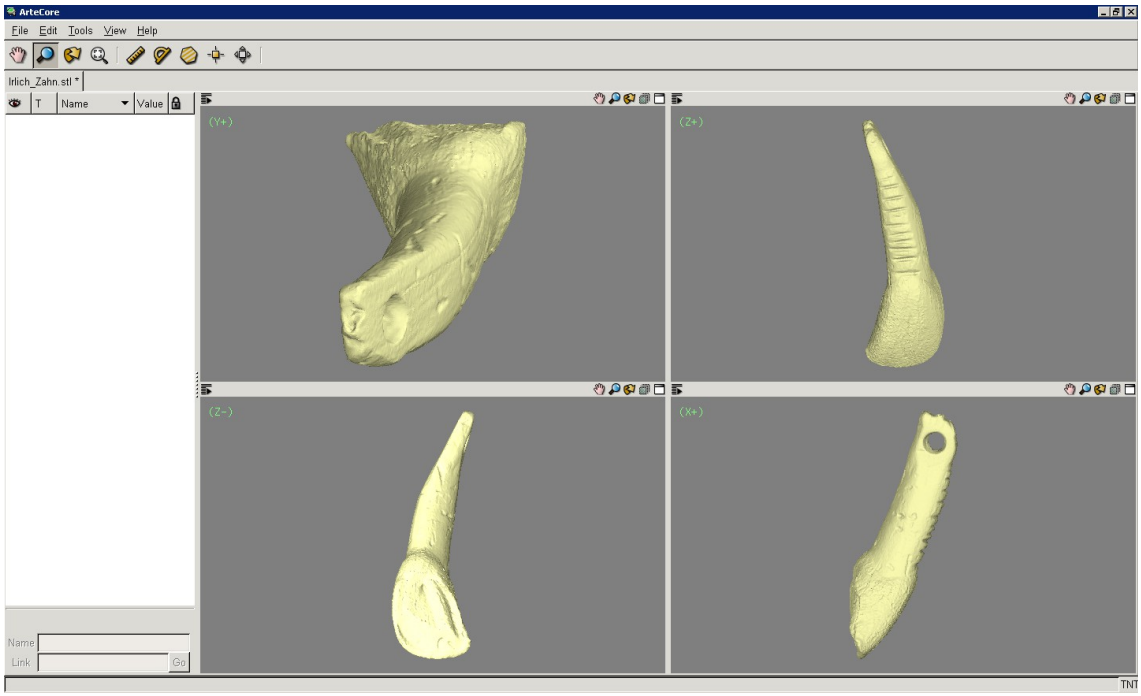

Abbildung 2: Vierfachsicht in ArteCore

Das Objekt kann im Raum verschoben und gedreht werden, sowie die Ansicht vergrößert. Jeder dieser Modi muss jedoch separat ausgewählt werden. Icons dafür befinden sich jeweils am oberen Rand jeder 3D-Ansicht.

Die Beleuchtung ist fest und kommt aus verschiedenen Richtungen.

Das Verhältnis von Millimetern zu Einheiten im 3D-Raum ist einstellbar, was für die Messwerkzeuge relevant ist. Derer gibt es drei:

- Streckenmesswerkzeug mit der Möglichkeit, mehrere Punkte anzugeben.
- Winkelmesswerkzeug durch setzen von 3 Punkten
- Flächenmesswerkzeug durch Aufspannen eines Dreiecksfächers, wobei der erste Punkt den Drehpunkt des Fächers definiert.

Außerdem können noch einzelne Punkte markiert werden. All diese Punkte und Punktsammlungen werden am linken Rand tabellarisch aufgeführt und können beschriftet werden.

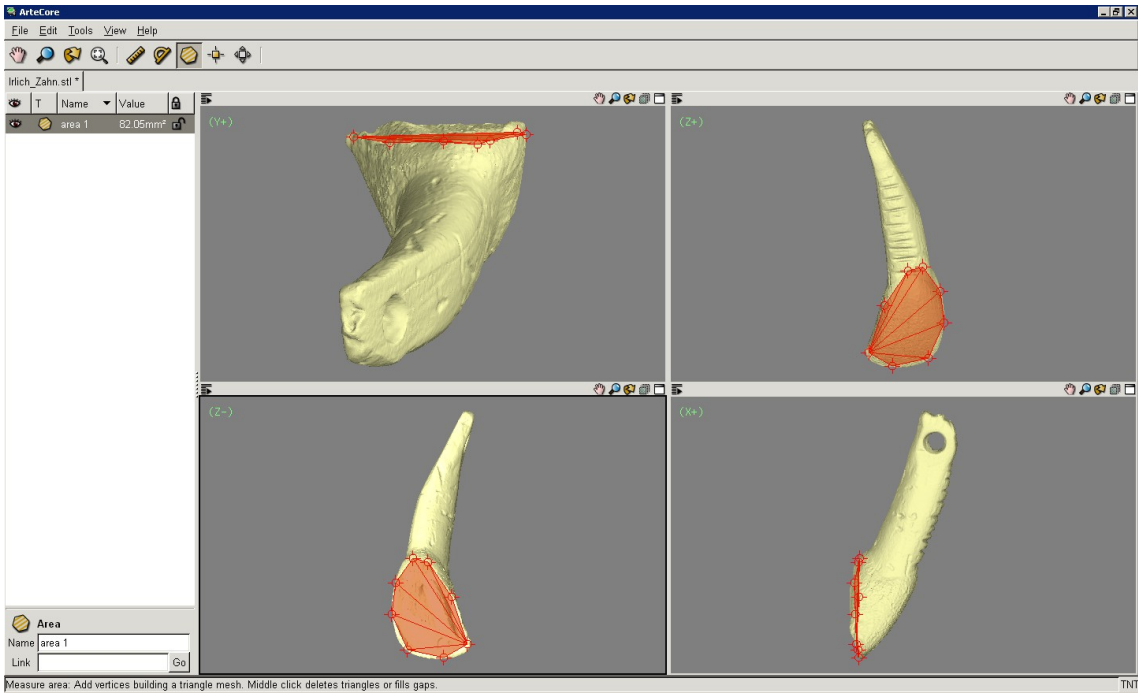

Abbildung 3: Flächenmesswerkzeug in ArteCore

Da ArteCore mit Microsofts Visual Studio 2003 entwickelt wurde, erschien es in Hinblick auf die begrenzte Zeit wenig aussichtsreich, zu versuchen, die Software zu erweitern, da Probleme mit nicht mehr aktualisierbaren Bibliotheken bereits absehbar waren.

2.3.2 OptoView

OptoView wird von der Firma Breuckmann entwickelt und ist kostenfrei. Das Programm wird als passende Software zu den verschiedenen Scannern, die die Firma Breuckmann anbietet, zur Verfügung gestellt. Es verfügt über die Möglichkeit, Lücken im Objekt automatisch zu schließen. Dabei wird nur die Geometrie hinzugefügt, jedoch keine Farben. Außerdem entspricht die hinzugefügte Geometrie mangels Scanner- Informationen nicht dem Original, sondern stellt nur eine mathematische Annahme dar, die primär eine kosmetische Verbesserung sein soll, für den wissenschaftlichen Nutzer jedoch keinen wirklichen Mehrwert bietet.

OptoView ist – wie der Name schon vermuten lässt – ein reiner Betrachter, verfügt also über keinerlei Mess- und Kommentarfunktionen. Einzig ein Hintergrundgitter kann

eingeblendet werden. Damit dieses auch Nutzen entfalten kann, ist die Projektion stets orthografisch, Abstände werden also nicht immer kleiner, je weiter sie von der Kamera entfernt sind und somit können Strecken ungefähr anhand der Linien im Hintergrund abgeschätzt werden. Passend dazu gibt es eine Möglichkeit, die Skalierung der Einheiten festzulegen.

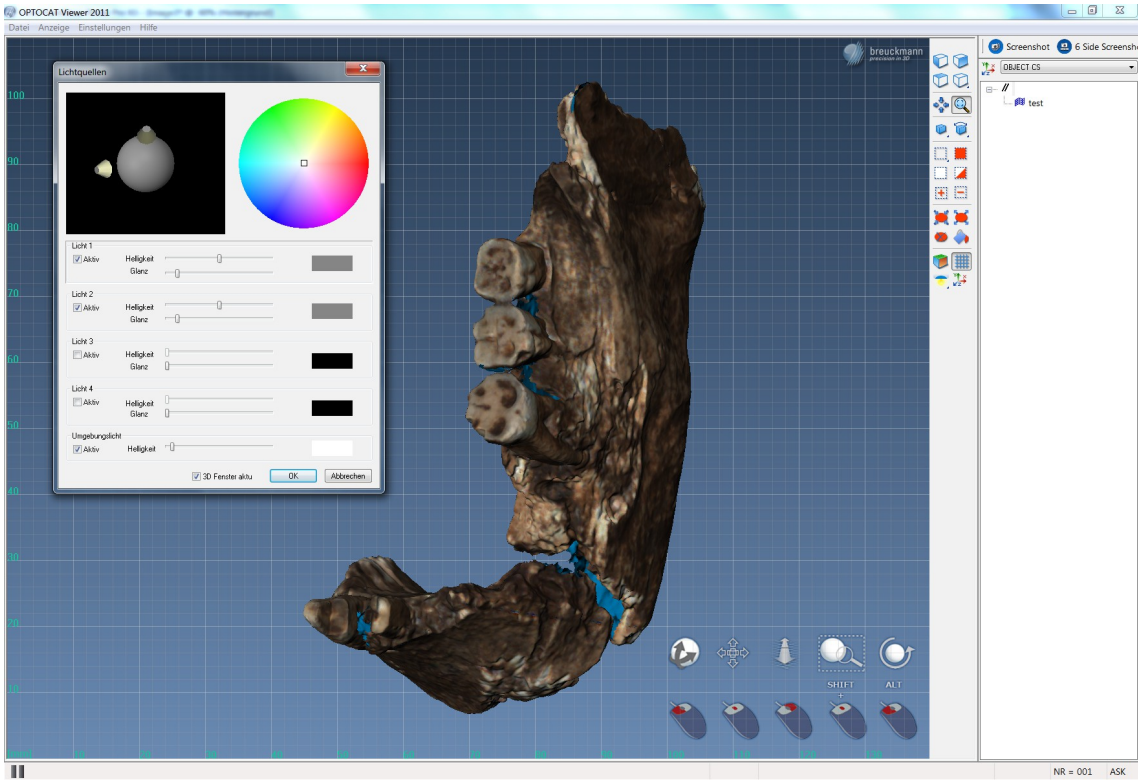

Abbildung 4: OptoView

Primär ist das Programm für CTR-Dateien geschrieben, andere Formate wie PLY und STL können jedoch importiert werden.

Es gibt verschiedene Einstellmöglichkeiten zur Art der Beleuchtung (flach, Gouraud, ...), die getrennt festgelegt werden können, jeweils für das Bewegen des Objektes und die darauf folgende unbewegliche Ansicht, wenn die Maustaste wieder losgelassen wird. Bemerkenswert dabei ist, dass die Darstellung stets mit einer im Vergleich zu anderen Programmen niedrigeren Bildrate geschieht.

Verschieben, Rotieren und Vergrößern des geladenen Objekts sind möglich, ohne besondere Modi auswählen zu müssen. Es werden verschiedene Maustasten (ggf. in Verbindung mit Shift und Alt) genutzt. Am unteren rechten Rand des Bildes können Symbole eingeblendet werden, die den Anwender daran erinnern, welche Tastenkombi-

nation welche Funktion hat. Das Rotieren erfordert allerdings Übung, da ein Zusammenhang zwischen der Bewegungsrichtung der Maus und der des Objekts schwer zu erkennen ist.

Die Beleuchtung des Objektes erfolgt standardmäßig mit zwei Lampen, die auf bis zu vier erweitert werden können. Jede Lampe kann dabei separat hinsichtlich ihrer Farbe, Leuchtkraft und der Stärke der verursachen Glanzlichter konfiguriert werden. Außerdem kann die globale Beleuchtung reguliert werden. Dabei fällt auf, dass es nicht möglich ist, Lampen ganz hinter das Objekt zu verschieben.

OptoView ist zwar kostenlos, aber nicht quelloffen. Insofern ist es nur zur Beurteilung dessen relevant, was eine für Archäologen nützliche Software können soll, kann jedoch nicht weiterentwickelt und auf die eigenen Bedürfnisse angepasst werden.

2.3.3 Meshlab

Meshlab ist ein Betrachter für eine breite Palette an 3D-Formaten und hat keinen speziellen Bezug zur Archäologie. Dafür ist es aber quelloffen und kostenlos im Internet bei www.Sourceforge.net verfügbar.

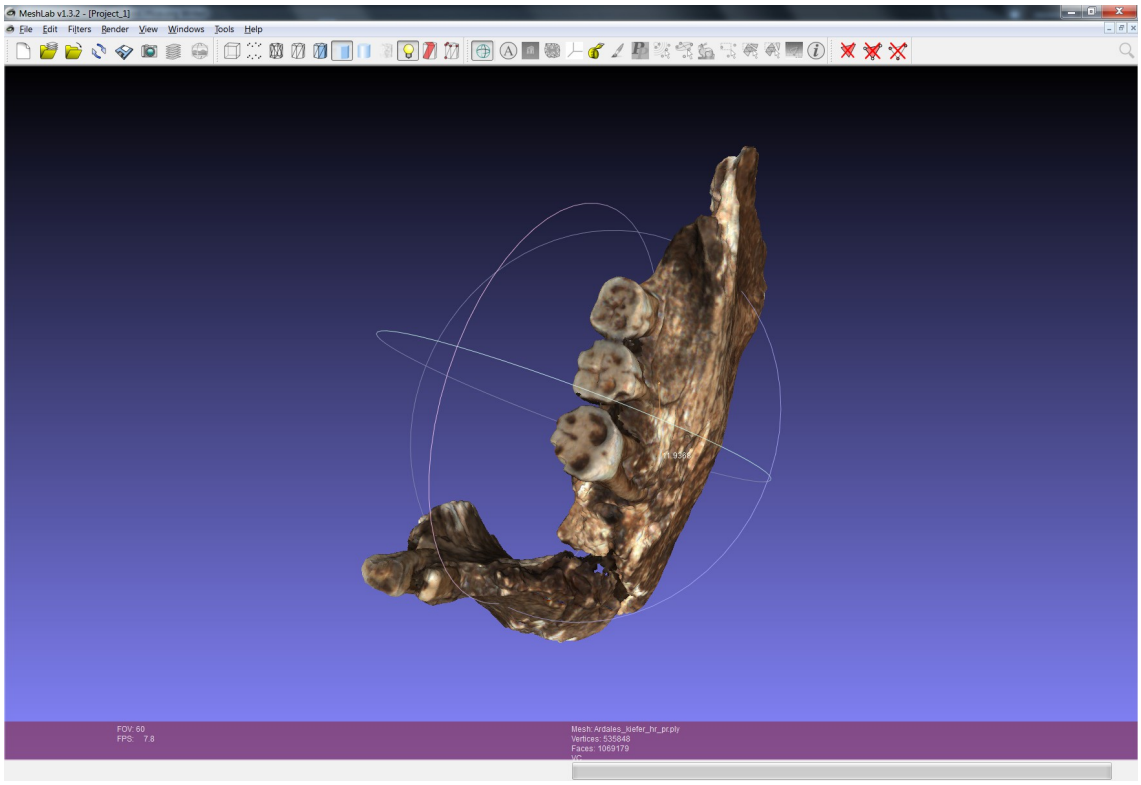

Abbildung 5: Meshlab

Die Ansicht kann geteilt werden, so dass auch per 4-View gearbeitet werden kann. Die Funktionen Verschieben, Rotieren und Vergrößern können mit verschiedenen Maustasten erwirkt werden.

Als einziges Messwerkzeug ist ein Maßband vorhanden, welches jedoch nur die Strecke zwischen zwei Punkten messen kann und zudem lediglich eine kaum sichtbare Linie einzeichnet.

Die Beleuchtung kann aktiviert und deaktiviert werden, die einzelnen Lampen jedoch sind nicht veränderbar.

Das Programm konzentriert sich dem Namen entsprechend auf verschiedene Darstellungsformen für 3D-Gitter wie Anzeige von Punkten, Gitterlinien, flachen und schattierten Flächen und deren Auswahl mit verschiedenen Werkzeugen. Möglichkeiten für Anmerkungen oder komplexere Messfunktionen sind nicht gegeben. Dafür kann auch Meshlab automatisch Lücken im Gitter schließen.

Meshlab benötigt eine ganze Reihe von Bibliotheken[30], was schlussendlich eine Erweiterung sehr schwierig erscheinen ließ. Da Meshlab unter anderem auf der Qt-Bibliothek basiert, entstand aber auf diese Weise die Idee, lediglich auf dieser neu aufzubauen.

---

[30] http://sourceforge.net/apps/mediawiki/meshlab/index.php?title=Compiling

3 Entwicklung einer neuen Software

Aufgrund der Tatsache, dass die Erweiterung der bestehenden Software sich schwierig gestaltete und sich die Qt-Bibliothek mit dem Qt-Creator, die später noch beschrieben werden sollen, zunehmend als geeignetes Werkzeug darstellte, um die geforderte Funktionalität in der gegebenen Zeit zu erreichen, wurde nun der Weg der Eigenentwicklung eingeschlagen.

Nach weiteren Gesprächen und Abwägungen bezüglich der Umsetzbarkeit spezieller Wünsche wurde in Zusammenarbeit mit Mitarbeitern des Neanderthalmuseums eine Anforderungsdefinition erstellt. Diese wurde gemäß der MoSCoW-Priorisierung strukturiert. Bei dieser Vorgehensweise werden die Anforderungen in vier Kategorien eingeteilt, die diesen eine Wichtigkeit bezogen auf das Endresultat zuordnen und somit indirekt auch eine Wahrscheinlichkeit der Erfüllung dieser Anforderung implizieren. Das Akronym MoSCoW leitet sich in etwa von den vier Kategorien ab, die da wären »Must«, »Should«, »Could« und »Won't«. Diese stehen in dieser Reihenfolge für die Anforderungen, die unbedingt umgesetzt werden **müssen**, die umgesetzt werden **sollten**, die umgesetzt werden **könnten** und zuletzt die, die vermutlich **nicht** umgesetzt **werden**. Die letzte Kategorie dient dabei vor allem dazu, aufzuzeigen, in welche Richtung die Entwicklung einmal fortgesetzt werden könnte.[31]

3.1 Anforderungsdefinition

Essenzielle Standardfunktionen für die Anwendung (**MUST**):

1. Gescannte 3D-Objekte können im ply-Format eingelesen werden.
2. Das Objekt sowie die Kamera sind in allen Richtungen drehbar.
3. Die Darstellung des Objekts ist mit Hilfe des Mausrades vergrößerbar durch Verringerung des Abstands der Kamera zum Objekt.
4. Die Darstellung erfolgt perspektivisch.
5. Die Hintergrundfarbe ist standardmäßig schwarz, kann aber geändert werden, gegebenenfalls auch in einen Farbverlauf.

---

[31] de.wikipedia.org, Artikel »MoSCoW-Priorisierung«

6. Die Darstellung des Objekts erfolgt wahlweise mit oder ohne Farbinformationen aus der PLY-Datei.
7. Die Beleuchtung erfolgt mit vier Punktlichtern, deren Intensität und Position in allen drei Raumachsen einstellbar ist, um den gewünschten Schattenwurf zu erzeugen, der zur maximalen Wahrnehmbarkeit von Details des Objekts nötig ist.
8. Die Ansicht des Objekts kann gespeichert werden. Das betrifft insbesondere die Rotation des Objektes und der Kamera, den Zoomfaktor und die Ausrichtung und Parameter der Lampen.
9. Die Kamera kann durch spezielle Schaltflächen gradgenau gedreht werden, um die für wissenschaftliche Publikationen übliche 6-fach Ansicht (vorne, hinten, links, rechts, oben, unten) erfassen zu können.
10. Die Software soll kostenfrei weiterverteilt werden können und der Quellcode soll jedem zugänglich sein und abgeändert werden können, was durch ihre eigene Lizenz, aber auch die der enthaltenen Bibliotheken sichergestellt wird.
11. Der Code wird ausreichend kommentiert, um eine spätere Erweiterung durch beliebige andere Programmierer zu ermöglichen.

Weitere wichtige Funktionen, die teilweise schon über das Leistungsspektrum der vorhandenen oben beschriebenen Anwendungen hinaus gehen (**SHOULD**):

1. Breite, Höhe und Dicke des Objekts in Millimetern können angezeigt werden.
2. Wahlweise können das Objekt überlagernde Ebenen mit zusätzlichen Informationen angezeigt werden, beispielsweise automatisch gefundenen Kanten oder Vertiefungen im Objekt.
3. An beliebiger Stelle auf der Oberfläche des angezeigten Objekts kann eine Anmerkung angeheftet werden. Ihr textlicher Inhalt steht dabei am linken Fensterrand und eine Linie führt von dort zu dem Punkt auf der Oberfläche, der die Anmerkung zugeordnet ist.
4. Anmerkungstexte können, so denn zu dem angezeigten Objekt überhaupt Anmerkungen existieren, ein- und ausgeblendet werden.
5. Der Inhalt des 3D-Anzeigebereichs ist jederzeit als Bilddatei abspeicherbar (Screenshot).

Sinnvolle Funktionen, deren Machbarkeit (teilweise aus Zeitgründen) nicht garantiert werden kann (**COULD**):

1. Anmerkungen können hinzufügt, editiert, gelöscht und ein- und ausgeblendet werden.
2. Auf dem Objekt kann gezeichnet werden zwecks Erzeugung einer handskizzenähnlichen Darstellung des Objekts mit Hilfe von gängigen Zeichenhilfsmitteln, als da wären:
    - Wahl der Pinseldicke, -farbe und -deckkraft
    - Wahl der Radiergummigröße
    - Wahl des Betriebsmodus (Zeichnen oder Radieren)
3. Alles Zeichnen auf dem Objekt erfolgt in Ebenen, die in ihrer Zahl beliebig sein können und alle einzeln ein- und ausgeblendet werden können.
4. Zur Erzeugung eines Umrisses kann durch je einen Knopf eine Schnittebene in x/y-, x/z-, y/z-Richtung angezeigt werden, welche alsdann mit der Maus verschoben werden kann, bis die linke Maustaste geklickt wird. Statt des Objekts wird dann der Schnitt in Weiß auf Schwarz angezeigt.
5. Das Volumens des Objekts kann berechnet und angezeigt werden.
6. Die Oberfläche des Objekts kann berechnet und angezeigt werden.
7. Durch Angabe von zwei Punkten A und B auf der Objektoberfläche kann der Abstand zwischen diesen Punkten als Länge des Vektors von A nach B errechnet und angezeigt werden.
8. Durch Angabe dreier Punkte A, B und C auf dem Objekt, durch die ein Dreieck aufgespannt wird, kann der Winkel zwischen den Vektoren AB und AC angezeigt werden. Es besteht die Möglichkeit, für den Punkt A eine Anmerkung zu generieren, die schon automatisch den gemessenen Winkel enthält und ermöglicht, weiteren erklärenden Text hinzuzufügen.
9. Für n angegebene Punkte (n > 2) kann ein Mittelpunkt berechnet werden, von dem aus ein Dreiecksfächer zu allen n Punkten aufgebaut wird, der die von den Punkten umschlossene Fläche umfasst und dessen Fläche somit der von den Punkten umschlossenen Fläche entspricht. Dabei müssen die Punkte sequenziell in einer kreisförmigen Bewegung gesetzt werden, da sich sonst Dreiecke des Fächers überlagern und nicht das gewünschte Ergebnis berechnet wird.

Ähnliches kann passieren, wenn zu konkave Formen umschlossen werden, so dass eine Linie vom Mittelpunkt bis zum äußeren Punkt durch einen nicht in der Flächenberechnung zu berücksichtigenden Bereich verläuft.

10. Alle Ebenen, Anmerkungen und die Standardrotation können in einer separaten Datei gespeichert werden und bei der nächsten Verwendung des 3D-Objektes wieder geladen werden.

Funktionen, die in zukünftigen Versionen der Software umgesetzt werden sollten (**WON'T**):

1. Möglichkeit der Aktivierung eines Algorithmusses, der eine neue Texturebene aus dem Objekt erzeugt, in die er selbsttätig in Form von Linien die Kanten des Objekts einzeichnet, die er findet.
2. Anzeige einer Sechsfachansicht des Objekts, die nicht interaktiv sein muss, sondern nur dazu dient, davon ein Abbild für Publikationen zu speichern. Die Sechsfachansicht zeigt das Objekt neben der Standardrotation auch jeweils 90° nach links und rechts um die z-Achse gedreht, sowie 180° gedreht (also Rückseite), sowie eine Ober- und Unteransicht, die eine Drehung um jeweils 90° aus der Standardrotation um die x-Achse darstellt.
3. Universeller Flächenmesser für alle Formen.
4. Texturebenen für Farbinformationen des Objekts, die auf UV-Koordinatenzuordnung der Pixel basieren, nicht auf der Zuweisung von Farben zu Knoten des Gitters, wodurch die Auflösung der Textur unabhängig wird von der Auflösung des Gitters.
5. Möglichkeit zur Reduzierung der Knoten im Gitter, um Objekte anzeigen zu können, deren Vielzahl von Knoten eigentlich die Speicherkapazität des Rechners übersteigt.

3.2 Qt-Bibliothek

Die Qt-Bibliothek ist eine C++-Bibliothek zur Vereinfachung der Entwicklung von grafischen Anwendungen auf unterschiedlichen Plattformen und hat bereits eine jahrelange Entwicklung hinter sich. Qt wurde von Trolltech entwickelt und im Jahr 2008 von Nokia aufgekauft, 2011/2012 an Digia weiterverkauft und steht zur Zeit unter der LGPL[32], einer Lizenzform, die die geforderte Offenheit der Software gemäß der MUST-Anforderung Nr. 10 gewährleistet.

Die Qt-Bibliothek kommt heute in vielen sehr bekannten Programmen zum Einsatz, wie beispielsweise dem KDE-Desktop, Google Earth oder auch dem VLC-Mediaplayer. Da die gerade erschienene Version 5.0.0 noch nicht voll ausgereift war und Probleme mit der OpenGL-Unterstützung hatte, wurde die Vorversion 4.8.4 verwendet.

Der Qt-Creator ist die zugehörige passende Entwicklungsumgebung für Qt und ist ebenfalls kostenlos erhältlich. Die Wahl fiel auf die Version 2.7.0.

Für die Implementierung der Anforderungen schien Qt gut geeignet, da eine komplett neues Bedienoberfläche entworfen werden musste, um die zu implementierenden Funktionen nutzbar zu machen. Mit Hilfe des Qt-Creators ist es mit relativ wenig Aufwand möglich, Fenster zu designen, Schaltflächen, Texteingabefelder und Ähnliches zu platzieren und diesen Elementen mit wenigen Zeilen Programmcode die gewünschte Funktion zu geben. Viele Parameter dieser Elemente können bequem im Qt-Creator geändert werden und werden im XML-Format in einer speziellen UI-Datei (user interface) gespeichert, die beim Kompilieren mitverarbeitet wird, so dass beim Ausführen des Programms die gewünschten Werte gesetzt sind. Davon unabhängig können alle Parameter auch selber innerhalb des eigenen Codes zur Laufzeit verändert werden.

Alle grafischen Elemente in Qt sind sogenannte Widgets, unter denen sich auch ein GLWidget befindet. Dieses ermöglicht es, mit OpenGL-Funktionen (und speziellen Klassen von Qt) in dem vom Widget belegten Bereich zu zeichnen. Dies war ein sehr entscheidender Faktor dafür, eine Neuentwicklung in der gegebenen Zeit in realistische Nähe zu rücken.

Da das PLY-Format nicht die laut Anforderungsdefinition geforderten zu speichernden

---

[32] http://www.gnu.org/licenses/lgpl-3.0

Informationen, wie beispielsweise Anmerkungen zu bestimmten Stellen, aufnehmen kann und es zudem generell wünschenswert ist, die darzustellenden Rohdaten von den persönlichen Einstellungen zu trennen, bot es sich an, zu jeder PLY-Datei eine gleichnamige XML-Datei erstellen zu lassen, die die nötigen Informationen aufnehmen konnte und der Philosophie von XML folgend jederzeit erweiterbar ist. Dies ist wichtig für die Erfüllung der MUST-Anforderung Nr. 11. Um XML-Dateien nicht selber auf Textebene lesen zu müssen, sondern gleich auf automatisch generierte Objekte zuzugreifen, wurde auf die entsprechende innerhalb der Qt-Bibliothek bereitgestellte Funktionalität zurückgegriffen.

Zusammenfassend wurden somit also folgende Bibliotheken dynamisch verlinkt:

- QtCore4.dll
- QtGui4.dll
- QtOpenGL4.dll
- QtXml4.dll

Eine statische Verlinkung ist der kommerziellen Version der Qt-Bibliothek vorbehalten.

3.3 RPly-Format und RPly-Bibliothek

Die Vielseitigkeit des PLY-Formats, welches zunächst als einziges einzulesendes Format vorgegeben war, macht die Verarbeitung von PLY-Dateien aufwändig, weswegen es sich anbot, auch hierfür eine fertige Bibliothek zu verwenden. Die RPly-Bibliothek von Diego Nehab[33] erwies sich als tauglich. Sie öffnet eine ihr übergebene Datei und interpretiert den Inhalt. Damit dieser in die eigene Software übernommen werden kann, müssen eigene Callback-Funktionen implementiert werden, die diese Werte entgegennehmen und in den geeigneten Variablen speichern. Dies wurde einmal für die Punkte (Koordinaten und Farben) und einmal für die Dreiecke (Indizes) getan.

---

[33] http://w3.impa.br/~diego/software/rply/

Im Folgenden wird der Aufbau einer PLY-Datei im Allgemeinen und der zur Verfügung gestellten Dateien im Speziellen erläutert:

Die erste Zeile enthält dabei nur die Zeichenfolge

```
ply
```

um anzuzeigen, dass es sich um eine PLY-Datei handelt.

Die zweite Zeile enthält das Schlüsselwort »format«, gefolgt von »ascii« oder »binary_little_endian« bzw. »binary_big_endian«, gefolgt von der Versionsnummer, beispielsweise »1.0«. Dies ist dem Umstand geschuldet, dass PLY-Dateien grundsätzlich in zwei Gruppen eingeteilt werden können, nämlich die im Textformat und die im Binärformat. Während das Textformat den Vorteil hat, dass die gespeicherten Knoten und Kanten menschenlesbar sind, hat es auch den Nachteil, dass diese Kodierung deutlich mehr Speicherplatz verbraucht als die Kodierung im Binärformat. Die Darstellung im Textformat kann also Sinn machen bei kleineren Datenmengen und zu Testzwecken, um den Inhalt der Datei leichter überprüfen zu können. Für die sehr fein aufgelösten 3D-Daten aus den Laserscannern ist sie jedoch ungeeignet. Alle zu Testzwecken vorgelegten Dateien lagen daher auch im Binärformat vor. Dennoch sollte es natürlich – wenn schon die Möglichkeit besteht, PLY-Dateien zu öffnen – auch jede mögliche Variation des Dateiformats gelesen werden können. Durch die Verwendung der RPly-Bibliothek beherrscht die resultierende Software nun beides.

Bei der Binärdarstellung muss weiterhin unterschieden werden, in welcher Reihenfolge die Bytes gespeichert sind, die eine Fließkommazahl darstellen (Big- oder Little Endian). Die konkret vorliegenden Dateien enthielten alle die Zeile

```
format binary_little_endian 1.0
```

wobei anzumerken ist, dass 1.0 die einzige existierende Versionsnummer[34] ist.

Danach können Zeilen folgen, denen die Schlüsselwörter »comment« für Kommentare und »obj_info« für Metainformationen zum Objekt vorangestellt sind. Der auf die Schlüsselwörter folgende Inhalt der Textzeile ist frei wählbar und nicht vom PLY-Format festgelegt. Ein konkretes Beispiel hierfür war »comment PLY by OPTOCAT«

[34] http://en.wikipedia.org/wiki/PLY_(file_format)

als Hinweis auf die Software, mit der die Datei generiert wurde.

Danach folgt die Definition der Struktur der Daten. Diese ist beim PLY-Format sehr flexibel, was das Format recht vielseitig macht, den Importvorgang jedoch aufwändig.

Die Zeile

```
element vertex 478189
```

definiert ein Element namens »vertex« und gibt an, dass die Datei davon 478.189 Stück enthält. Es folgt die Definition der Eigenschaften dieses Elements in Form der Zeilen:

```
property float x
property float y
property float z
property uchar red
property uchar green
property uchar blue
```

Es werden also drei Eigenschaftswerte x, y und z als Fließkommazahlen definiert, die die Koordinaten des Vertex in den drei Raumachsen repräsentieren und drei Werte red, green, blue als vorzeichenlose Char-Variablen, die den Farbwert des Punktes festlegen. An dieser Stelle könnten beliebige andere Eigenschaften wie Temperatur, Gewicht oder Druck ebenfalls festgelegt werden. Die zur Verfügung gestellten Beispieldaten enthielten jedoch genau diese Daten, weswegen auch nur sie von der Software verarbeitet werden. Sollten später weitere Eigenschaften erfasst werden, so ist eine Erweiterung des Software nötig, die jedoch problemlos möglich ist. Die RPly-Bibliothek kann unverändert bleiben, lediglich die Callback-Funktion muss so angepasst werden, dass sie mehr Werte lesen und in neue Variablen schreiben kann.

Darauf folgt die Definition eines weiteres Elements durch:

```
element face 956358
```

In der Datei sind demnach 956.358 Face-Elemente enthalten. Auch hier folgt die Definition, was genau ein Face ausmacht:

```
property list uchar int vertex_indices
```

Es handelt sich demnach um eine Liste von Werten, bei der der erste Wert wieder eine

vorzeichenlose Char-Variable ist, die angibt, wie viele zu diesem Element gehörenden Werte folgen werden. In den vorliegenden Dateien wurden stets Dreiecke abgebildet, so dass der Wert stets 3 war. Die drei folgenden Werte waren vertex_indices und vom Typ int, so dass nun also nur noch das Schlüsselwort:

```
end_header
```

folgen musste, um den Header zu beenden und anzuzeigen, dass alle folgenden Bytes so zu interpretieren waren, wie im Header angegeben. Das bedeutet, es folgen nun jeweils 6 Eigenschaften pro Vertex-Element, derer es 478.189 gab, insgesamt sind also 2.869.134 Fließkomma-Werte einzulesen. Danach folgen nahtlos noch die Face-Elemente, deren Größe auf dieselbe Weise ermittelt wird. Danach sollte das Ende der Datei erreicht sein.

Wie in Binärdateien üblich, folgen alle Datenbytes ohne Trennzeichen aufeinander und müssen abgezählt werden, um zu erkennen, wo eine Zahl aufhört und die nächste beginnt und wie diese zu interpretieren ist, daher ist es auch essenziell, zu wissen, wie viele Bytes die verschiedenen (Fließkomma-)zahlen belegen. Auch dies ist im PLY-Format nicht festgelegt. Dies alles automatisch zu erkennen und zu verarbeiten gehört zu den Leistungen der RPly-Bibliothek und bewirkt die große Arbeitsersparnis.

3.4 Aufbau der Software

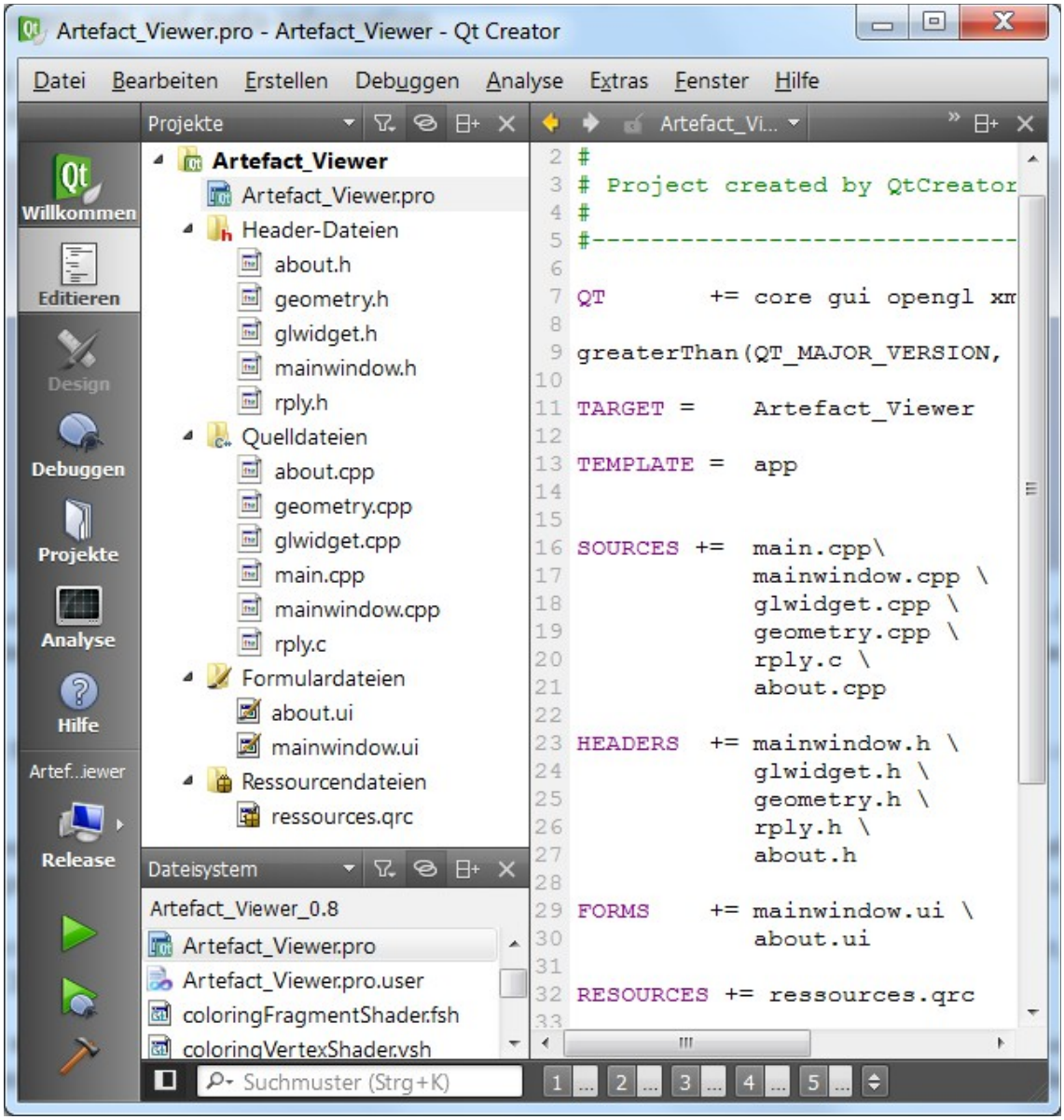

Abbildung 6: Die Dateien des Projects Artefact Viewer

Das obige Bild zeigt die verschiedenen Quelldateien, aus denen der Artefact Viewer - so der Arbeitsname des Programms - aufgebaut ist. Auf die Funktionen und Aufgaben der einzelnen Dateien soll im Folgenden eingegangen werden. Dabei sind die Namen von Funktionen jeweils kursiv geschrieben. Auf die Parameter wird der Übersichtlichkeit halber verzichtet.

3.4.1 about.cpp / about.h / about.ui

Hiermit wird ein QDialog-Widget angezeigt, ein kleines Fenster, welches die Programmversion und den Autor enthält. In der Datei about.ui ist das Design des Fensters festgelegt.

3.4.2 geometry.cpp / geometry.h

Diese Klasse speichert:

- die Position aller Knoten des Artefakts als sogenannter QVector von QVector3Ds. QVector ist ein dynamisches Array und QVector3D speichert ein Tripel von Fließkommazahlen, die einen Vektor im dreidimensionalen Raum repräsentieren. Beides sind Klassen, die die Qt-Bibliothek zur Verfügung stellt, wie man an dem vorangestellten Buchstaben Q erkennen kann.
- die Indizes der drei Knoten, die jeweils ein Dreieck bilden, als GLuint, was einem vorzeichenlosen Integer mit der Länge der auf dem Rechner installierten OpenGL-Installation entspricht. Es bietet sich an, genau diese Länge zu verwenden, da auf diese Weise spätere Probleme beim Übertragen der Werte an die Grafikkarte vermieden werden.
- die Normalen für jeden Knoten, die sie selber mittels der Methode *calculate-Normals* errechnet. Die Funktion wird einmalig beim Öffnen einer PLY-Datei aufgerufen. Sie iteriert über alle Dreiecke, berechnet aus den drei Knoten und dem Wissen, dass die Normale einer Ebene dann auf den Betrachter zeigt, wenn für ihn die drei Punkte der Reihenfolge nach gegen den Uhrzeigersinn angeordnet sind, die Normalen für die Ebene und nimmt diese Normale als Grundlage, um für alle Knoten die gemittelte Normale zu errechnen, denn OpenGL benötigt die Normalen in den Knoten, nicht im Mittelpunkt der Dreiecke.
- die Farbe der Knoten (»Vertexcolors«) genau wie die Position der Knoten.
- eine Shadowmap in Form eines QVectors, der für jeden Knoten einen float-Wert speichert, der einen Verdunklungsfaktor zwischen 0 und 1 enthält. Dieser Faktor wird einmalig beim Laden der Datei initialisiert und kann zur Laufzeit des

Programms ständig geändert werden, um die visuellen Verbesserungen zu optimieren. Jedes Mal wird dabei die Methode *calculateShadowMap* aufgerufen und somit für hunderttausende Knoten neue Werte berechnet, was zu leichten Verzögerungen zur Laufzeit führen kann.

- die Nachbarschaft von Knoten in einem QVector von QVectors von Gluints. Diese wird einmalig beim Laden der Datei erfasst, indem zu jedem Knoten seine Dreieckspartnerknoten als Nachbarn gespeichert werden. Dies wird benötigt, um später die Werte der Shadowmap zu glätten.
- eine Flatcolor, also ein einziger Farbwert, mit dem das ganze Objekt eingefärbt wird, wenn keine Farbinformationen in Form von Vertexcolors vorhanden sind oder sie nicht genutzt werden sollen.
- die Positionen der Knoten einer Pyramide, die schematisch einen simplen Lampenschirm darstellen soll sowie die passenden Farbinformationen für die Knoten für die beiden Zustände der Lampe »an« und »aus«. Diese Daten werden einmalig von der Methode *setupLightGeometry* erzeugt.
- einen Mittelpunkt. Dieser wird einmalig beim Öffnen der Datei mittels der Methode *calculateCenterPoint* ermittelt, und zwar als Mittelwert aller Knoten des Artefakts. Dies ist wichtig, damit man das Artefakt später um seinen »natürlichen« Schwerpunkt drehen kann. Der tatsächliche Nullpunkt der gegebenen Koordinaten kann aus technischen Gründen überall liegen, theoretisch sogar weit weg vom Objekt.
- eine QList von pointClouds, wobei pointCloud ein struct ist, der wiederum einen Text, eine Farbe, einen Typ und einen QVector von QVector3Ds enthält. Darin werden die Anmerkungstexte und deren zugehörige Geometrie gespeichert. Es handelt sich also um eine Liste von Anmerkungen, die wiederum aus einem Text bestehen, einem bestimmten Typ angehören und eine unterschiedliche Menge von Punkten haben (je nach Typ), die in einer vorgegebenen Farbe dargestellt werden.

Und schließlich gibt es noch einige Funktionen, die Berechnungen mit Knoten anstellen, so kann mit *getArtefactWidth*, *-Height* und *-Depth* die Breite, Höhe und Tiefe abgefragt werden, sowie mit *isInTriangle* geprüft werden, ob ein übergegebener Punkt in einem ebenfalls übergebenen Dreieck liegt, was vor allem nützlich ist, um zu prüfen,

wo ein Klick mit der Maus das Artefakt getroffen hat.

Als experimentell ist die Funktion *readStlFile* anzusehen. Diese kann STL-Dateien (statt PLY) lesen, allerdings nicht deren Geometrie optimieren.

3.4.3 glwidget.cpp / glwidget.h / resources.qrc

Wie zuvor erwähnt, dient das GLWidget (dessen einzige Instanz glWidget genannt wurde) der Anzeige des 3D-Raumes. Manche Variablen müssen einmalig initialisiert werden, andere jedes Mal, wenn eine neue PLY-Datei geladen wird, denn dabei bleibt das glWidget erhalten. Daher gibt es neben dem Konstruktor noch eine *initialize*-Methode, die wieder Standardbedingungen in der Szene herstellt, also alle Rotationen bezüglich der Kamera und der Lichter auf vorgegebene Werte stellt, die Lichter selber mit Standardeinstellungen belegt, den Hintergrund wieder schwarz macht und noch Einiges mehr.

Die *fillBuffers*-Methode wird benötigt, da es bei der Vielzahl von Geometriedaten des Artefakts angesichts der Tatsache, dass diese sich zur Laufzeit nicht ändern, zur Leistungssteigerung Sinn macht, diese im Grafikkartenspeicher puffern zu lassen, um sie nicht immer wieder hochladen zu müssen. Dabei wird je ein vertex-, color-, normal- und indexbuffer angelegt, wobei die Größe angegeben werden muss und sich je aus der Größe einer Variablen multipliziert mit der Zahl der zu speichernden Variablen dieses Typs ergibt. Auch beim Beschreiben des Puffers muss stets angegeben werden, wo genau geschrieben werden soll, was es nötig macht, die geschriebenen Elemente selber in einer Zählvariablen nachzuhalten.

Je nachdem, ob der Benutzer die Anzeige der Farben an den Knoten wünscht, wird der Farbpuffer mit diesen Werten gefüllt oder nur mit einem Einheitswert. Außerdem wird an dieser Stelle die Shadowmap berücksichtigt. Wenn beispielsweise Vertiefungen im Artefakt hervorgehoben werden sollen, wird der Farbwert der tiefer gelegenen Knoten an dieser Stelle abgedunkelt.

Der Vollständigkeit halber werden auch Puffer für die Geometrie der Lichter generiert, was allerdings die Leistung nicht merklich erhöht und eher der Einheitlichkeit dient.

Die *initializeGL*-Methode ist von Qt vorgegeben und muss somit (re)implementiert werden. In diesem Fall dient sie dazu, backface culling, also das Nichtzeichnen von dem Betrachter abgewandten Flächen, und depth test, also das Nichtzeichnen von verdeckten Teilen von Objekten, zu aktivieren. Außerdem werden drei Shader eingerichtet. Qt stellt zu diesem Zweck die Klasse QGLShaderProgram bereit. Es wird je ein Vertex- und ein Fragmentshader aus der Ressourcendatei gelesen. Die Ressourcendatei fasst im wesentlichen Bild- und Textdateien zusammen, die man ansonsten in Unterordnern zur exe-Datei dazugeben müsste, ähnlich wie die dynamisch verlinkten Bibliotheken. Die Shader basieren auf Standard-Beispielprojekten von Qt, welche angepasst wurden, vor allem, um vier Lampen verarbeiten zu können.

Die *paintGL*-Methode ist die größte und wichtigste Methode der Klasse, da in ihr die eigentliche Darstellung des 3D-Raumes stattfindet, und zwar immer, wenn *updateGL* (von Qt vorgegeben) aufgerufen wird, denn darin wird auch *paintGL* aufgerufen.

Zunächst wird der Hintergrund gezeichnet. Dazu wird der depth test abgeschaltet, da der Hintergrund alles bisher Gezeichnete verdecken soll und von allem später Gezeichneten selber überdeckt werden soll. Ohne depth test ist nur die Reihenfolge des Zeichnens maßgeblich dafür, was später zu sehen ist. Der Hintergrund besteht aus einem Rechteck, das den gesamten Querschnitt des Sichtbereichs der Kamera abdeckt. Die oberen beiden Knoten erhalten die eine vom Benutzer eingestellte oder voreingestellte Farbe, die unteren beiden Knoten die andere. Wird zweimal dieselbe Farbe verwendet, ist der Hintergrund somit nachher einfarbig, werden zwei verschiedene Farben verwendet, wird durch den Shader automatisch interpoliert, so dass ein Farbverlauf entsteht. Zum Einsatz kommt der coloringShader, da dieser pro Knoten eine Farbe entgegennimmt und Beleuchtung nicht berücksichtigt.

Als nächstes wird die Kamera an die richtige Stelle verschoben und gedreht, indem entsprechende Matrizenmultiplikationen stattfinden, und daraus die passende view matrix errechnet. Die projection matrix wird nur geändert, wenn sich die Größe des glWidgets ändert, also wenn das Fenster in seiner Größe verändert wird. Dies geschieht gegebenenfalls in der Methode *resizeGL*. Die projection matrix leistet die Umrechnung von 3D-Koordinaten in 2D-Koordinaten. Nachdem Objekte (also letztlich die Knoten, aus denen sie bestehen) durch die model matrix in ihrer Form und Lage modifiziert wurden und durch die view matrix bestimmt wurde, aus welcher Lage die Szene

betrachtet wird, kann durch die projection matrix nun wie bei einer echten Kamera berechnet werden, an welcher Stelle des Bildschirms welche Knoten abgebildet werden müssen, damit der Benutzer den Eindruck eines 3D-Raumes hat.

Wenn nun also das Artefakt gezeichnet werden soll, wird noch unterschieden, ob mit oder ohne Beleuchtung gezeichnet wird. Ohne Beleuchtung kommt der bereits bekannte coloringShader zum Einsatz und zeichnet das Artefakt mit den im Farbpuffer gespeicherten Farben. Dies entspricht in etwa einem Objekt der Realität, welches rundum vom gleichen hellen diffusen Licht angestrahlt wird und keinerlei Unterschiede in der Helligkeit seiner Oberfläche zeigt.

Ist die Beleuchtung dagegen aktiviert, wird der lightingShader verwendet. Dieser berücksichtigt die aktivierten Lampen und zeichnet gemäß Phong-Shading[35] Schattierungen auf das Artefakt. Dabei handelt es sich um eine von zahlreichen mathematischen Möglichkeiten, dem Betrachter ein an die Realität angenähertes Bild zu zeigen, welches so aussieht, als gäbe es in der gezeigten Szene Licht und Schatten. Eine realistische und physikalisch korrekte Berechnung von Licht und Schatten ist zum derzeitigen Zeitpunkt (2014) nicht in Echtzeit möglich, sondern nur bei Einzelbildern, die teilweise stundenlange Rechenzeit erfordern.

Nur im Falle der aktivierten Beleuchtung wird auch die Repräsentation der Lampen im 3D-Raum gezeichnet und auch das nur, wenn das entsprechende Häkchen gesetzt ist.

Um den Speicherplatzverbrauch zu minimieren, wird zum Zeichnen des Artefakts stets die *glDrawElements*-Funktion verwendet. Im Gegensatz zu *glDrawArrays*, bei dem die Dreiecke an die Grafikkarte in Form von Tripeln von Knoten übermittelt werden, wird bei *glDrawElements* zuerst eine Liste aller Knoten übergeben und dann die Dreiecke in Form von Tripeln von Indizes der Knotenliste. Da Knoten in der Regel in einigen verschiedenen Dreiecken vorkommen, wird damit eine deutliche Reduktion des benötigten Speicherplatzes erreicht, was vor allem bezüglich des Grafikkartenspeichers wichtig ist. Der Unterschied entspricht im Prinzip dem Unterschied zwischen den Dateiformaten PLY und STL.

Am Ende folgt noch das Zeichnen der eventuell vorhandenen Punkte, Linien und Flächen, die zu den Anmerkungen gehören. Damit diese stets korrekt zu sehen sind,

---

[35] Phong 1975

werden sie überlagernd gezeichnet, das heißt ähnlich wie beim Hintergrund wird der depth test wieder deaktiviert, außerdem wird GL_BLEND aktiviert, damit halbtransparent gezeichnet werden kann. Anhand des Typs der Anmerkung wird nun ausgewählt, wie die Punkte, die in der Anmerkung gespeichert sind, gezeichnet werden sollen, entweder als

- Punkt, um genau zu einer Stelle etwas anzumerken,
- Verkettung von beliebig vielen Linien zu Streckenmessung,
- genau zwei Linien, die einen Winkel aufspannen, oder
- Dreiecksfächer, der eine gemessene Fläche aufspannt.

Hierfür wird der singleColorShader benötigt, der **eine** Farbe als RGBA-Wert in Form eines QVector4D entgegennimmt, also mit Alpha-Wert, und damit das aktuelle geometrische Objekt zeichnet, ohne Beleuchtung zu berücksichtigen.

Da die Klasse QPainter komfortableres Rendern von Text ermöglicht als die GL-eigenen Funktionen, wurde zum Zeichnen des Anmerkungstextes, des Rahmen darum und der Linie auf den Punkt auf dem Artefakt, dem die Anmerkung zugeordnet ist, darauf zurückgegriffen.

Wichtig ist an dieser Stelle, dass OpenGL-Befehle nicht einfach mit QPainter-Befehlen gemischt werden können. Da zwischen den vom QPainter gezeichneten Ziffern an den Lampen und den Anmerkungsboxen noch die Linien von den Boxen zu den Punkten auf dem Objekt mit OpenGL-Befehlen gezeichnet werden, sind diese durch die beiden Methoden *beginNativePainting* und *endNativePainting* von QPainter gekennzeichnet.

Ein weiteres Problem an dieser Stelle ist, dass von einem zweidimensionalen Objekt - wie dem Rahmen um die Anmerkung - eine Linie gezeichnet werden soll zu einem dreidimensionalen Objekt – dem Artefakt. Zur Lösung wurde der dreidimensionale Punkt auf dem Artefakt mit Hilfe der ModelViewProjection-Matrix in zweidimensionale Koordinaten umgerechnet, wobei beachtet werden muss, dass OpenGL mit normalisierten Koordinaten im Fenster arbeitet, also pro Achse von -1.0 bis +1.0, während Qt den Koordinatenursprung in der oberen linken Fensterecke annimmt und von da aus die Pixel zählt. Eine entsprechende Umrechnung ist also in beide Richtungen erforderlich, wofür die beiden Hilfsmethoden *QVector3DUnnormalizedToQPoint* und *QPointNorma-*

lizedToVector3D in GLWidget implementiert wurden, die nicht nur die Koordinaten konvertieren, sondern auch die beiden Variablentypen QPoint und QVector3D in den jeweils anderen konvertieren.

Der letzte Schritt in der paintGL-Methode ist schließlich das Umschalten des Framebuffers. Beim Framebuffering wird das Ergebnis von Berechnungen, wie sie zuvor beschrieben wurden, nicht in den Speicherbereich geschrieben, der von der Grafikkarte für das aktuell ausgegebene Bild genutzt wird, sondern in einen zweiten Bereich gleicher Größe. Erst, wenn alle Berechnungen durchgeführt wurden und das Bild fertig ist, wird umgeschaltet, die Grafikkarte zeigt das neue Bild an, das nicht mehr benötigte alte Bild wird gelöscht und der Speicher steht somit für die Aufnahme neuer Daten bereit. So muss also für jedes neue angezeigte Bild einmal der Framebuffer umgeschaltet werden. Dadurch werden immer nur vollständig berechnete Bilder angezeigt und keine Teile davon.

Die Methoden *mousePressEvent, mouseReleaseEvent, mouseMoveEvent* und *wheelEvent* verarbeiten alle Mauseingaben und ermöglichen

- das Bewegen der Kamera mittels Rechtsklick+Mausbewegung sowie dem Mausrad,
- das seitliche Bewegen beziehungsweise Auf und Ab der Kamera, wenn zusätzlich zur rechten Maustaste auch die Shift-Taste gedrückt und die Maus bewegt wird,
- das Setzen von Punkten mit der linken Maustaste, aber nur, sofern die Anmerkungen aktiviert sind, denn sonst würde der Anwender die Punkte nicht sehen und möglicherweise völlig unkoordiniert eine Vielzahl von Punkten setzen, und dies erst später bemerken,
- das Verschieben von Punkten, wenn nahe genug an einen bestehenden Punkt geklickt und gezogen wird, und
- das Auswählen einer Anmerkung, wenn diese generell sichtbar sind.

Die *getIntersectionPoint*-Methode prüft, welcher Punkt auf dem Artefakt gemeint ist, wenn der Benutzer mit der linken Maustaste klickt. Bekannt sind zunächst nur die Widgetkoordinaten des Klicks. Diese müssen für OpenGL zunächst normalisiert werden. Durch umgekehrtes Anwenden der ProjectionView-Matrix, wird nun ein Strahl von der Kamera ausgehend erzeugt, dessen Schnittpunkte mit allen Dreiecken des Artefakts bestimmt werden. Dies ist jedoch nicht in einem Schritt möglich. Zunächst können nur jeweils die Schnittpunkte mit den Ebenen berechnet werden, in denen die Dreiecke liegen. Dann muss bestimmt werden, ob der Schnittpunkt auf der richtigen Seite der drei das Dreieck begrenzenden Geraden liegt. Wenn dies der Fall ist, wird der Punkt gespeichert. Unter den gespeicherten Punkten wird am Ende des Algorithmusses derjenige mit dem kleinsten Abstand zur Kamera bestimmt. Damit der neu gesetzte Punkt auch angezeigt wird, muss *updateGL* ausgeführt werden, um die gesamte Ansicht zu aktualisieren.

3.4.4 main.cpp

Normalerweise dient die Datei main.cpp bei Qt-Anwendungen nur dazu, eine QApplication und ein mainWindow zu erstellen und gegebenenfalls ein paar Parameter zu setzen.

Darüber hinausgehend sind bei diesem Projekt die Callbackfunktionen hier untergebracht, da die verwendete RPly-Bibliothek in der Programmiersprache C geschrieben ist und daher keine Methoden übergeben werden können, sondern nur Funktionen.

Die Funktion *storeVertex_cb* wird für jeden Knoten, der aus der PLY-Datei geladen wird, aufgerufen. Übergeben wird jeweils ein p_ply_argument, woraus ersichtlich ist, welcher Wert gerade gelesen wurde. Da die Reihenfolge der Werte stets **x, y, z, r, g, b** ist, wird bei jedem x-Wert ein neues Element in den vertices und colors des geometry-Objekten erzeugt, dann wird der Wert an die passende Stelle geschrieben, wobei die Farben umgerechnet werden müssen, da diese als Integer-Werte von 0 bis 255 in der Ply-Datei vorliegen, für OpenGL jedoch als Float zwischen 0 und 1 liegen müssen.

Die Funktion *storeTriangle_cb* ist dagegen einfacher und schreibt die übergebenen Indizes in den triangles-QVector des geometry-Objekts.

Damit die Callback-Funktionen auch genutzt werden, müssen sie durch die Funktion

readPlyFile noch gesetzt werden. Außerdem wird das geometry-Objekt (re)initialisiert, indem die QVectors für vertices, colors und triangles zurückgesetzt (also geleert) werden, sowie die Normalen, die Nachbarschaft, darauf aufbauend die Shadowmap und auch der Zentrumspunkt neu berechnet werden.

3.4.5 mainwindow.cpp / mainwindow.h / mainwindow.ui

Die drei Dateien mit dem Namen mainwindow definieren Aussehen und Verhalten des Hauptfensters der Anwendung. Dabei enthält die Datei mit der Endung UI das im Qt-Creator grafisch definierte User Interface, also hauptsächlich das Aussehen. Dieses ist in Form von XML-Code gespeichert. Die hinter allen Eingabeelementen liegende Funktionalität wird hingegen durch die CPP-Datei definiert.

Ähnlich wie beim glWidget werden auch beim mainwindow einige Werte, die nur einmal gesetzt werden müssen, im Konstruktor festgelegt und andere, die für jedes angezeigte Artefakt wieder zurückgesetzt werden sollen, werden in einer *initialize*-Methode zusammengefasst. Erwähnenswert ist dabei der Umstand, dass an dieser Stelle eine noch leere Punktewolke erzeugt wird, die eventuell später vom Benutzer gesetzte Punkte auf dem Artefakt so lange speichert, bis er sich entschließt, wie er mit den Punkten verfahren will.

Wird eine der Schaltflächen up, down, left oder right angeklickt, wird die Spalte von Radiobuttons rechts daneben abgefragt und die dort eingestellte Gradzahl mittels der Methoden *setCamHorizontalRotation* bzw. *setCamVerticalRotation* vom glWidget genutzt, um die Kamera um den gewünschten Betrag um das Artefakt zu drehen. Die Kamera bewegt sich dabei auf einer gedachten Kugeloberfläche, deren Radius mit dem Distance-Regler eingestellt werden kann und in deren Zentrum standardmäßig das Artefakt liegt.

Die beiden Schaltflächen zum Drehen des Artefakts im bzw. gegen den Uhrzeigersinn (»clockwise« und »counterclockwise«) sind durch die beiden Methoden *on_pushButton_clock_clicked* und *on_pushButton_cclock_clicked* realisiert, die ebenfalls die eingestellte Gradzahl berücksichtigen, um das Artefakt zu drehen. Als Drehachse wird die Blickrichtung der Kamera bzw. des Betrachters verwendet. Bei jeder Drehung wird die Methode *updateInfoGroup* aufgerufen, die dafür sorgt, dass die durch die Drehung geänderten Artefaktdimensionen im Infobereich angezeigt werden.

Die Methode *on_pushButton_rearrange_clicked* verbessert in der Regel die Sortierung der Anmerkungstexte am linken Rand, so dass es weniger Überschneidungen zwischen den Linien zu den Punkten auf dem Artefakt gibt. Die eigentliche Arbeit macht dabei die aufgerufene Methode *sortPointClouds*, die sich aus dem glWidget die mvpMatrix holt und sie auf alle jeweils ersten Punkte der Punktwolken anwendet, um in Form der sich ergebenden y-Koordinate des Punktes auf dem Bildschirm die »Höhe« im Verhältnis zu den anderen Punkten zu ermitteln. Die Annahme ist hierbei, dass Punkte, die auf dem Bildschirm höher liegen, auch höher liegenden Anmerkungstexten zugeordnet werden sollen. Die x-Koordinate bleibt unberücksichtigt. Trotzdem werden in der Praxis gute Ergebnisse erzielt, wie der spätere Test zeigt. Das Drehen des Artefakts zerstört die Ordnung ohnehin relativ schnell wieder, so dass die Funktion erneut aufgerufen werden muss. Da die Zahl der Punkte so gering ist (ca. 1 bis 7), wurde zur Sortierung ein einfacher Vertauschungsalgorithmus in $O(n^2)$-Schritten angewendet.

Die Methode *on_pushButton_note_clicked* erstellt eine neue Anmerkung. Dazu werden zwei Komponenten genutzt, die Qt zur Arbeitsersparnis bereitstellt. Der QInputDialog zeigt eine Aufforderung an, eine Anmerkung einzugeben und bietet entsprechend ein Eingabefeld an. All dies kann in einer Zeile erzeugt werden. QColorDialog präsentiert gar eine große Auswahl an Farben zur Auswahl und zusätzlich Möglichkeiten, die gewünschte Farbe numerisch einzustellen oder für später zu speichern. Schließt der Benutzer den Vorgang ab, wird die aktuell letzte Punktewolke, die in mygeometry gespeichert ist, vom Typ her als Anmerkung markiert, ihre Farbe entsprechend der Auswahl der Benutzers eingestellt und der Anmerkungstext dazu gespeichert. Im Anschluss daran wird wieder eine neue leere Punktewolke erzeugt, die eventuell später vom Benutzer gesetzte Punkte wieder aufnehmen kann.

Die Methode *on_pushButton_distance_clicked* arbeitet ganz ähnlich; ihr vorangestellt ist jedoch eine Schleife, die alle aktuell gesetzten Punkte durchläuft, Vektoren zwischen den Punkten errechnet und deren Länge bestimmt. Die Summe der Vektorlängen wird dem Benutzer dann zunächst in einer QMessageBox - einem weiteren vorgefertigten Widget von Qt - als Ergebnis der Streckenberechnung präsentiert, bevor sie als Standardtext für eine Anmerkung verwendet wird, die der Benutzer genau so abspeichern kann, als hätte er manuell eine Anmerkung generiert.

Die Methode *on_pushButton_angle_clicked* ist grundsätzlich sehr ähnlich, berechnet

statt der Strecke jedoch den Winkel zwischen den beiden Vektoren AB und AC, die sich aus den drei gegebenen Punkten A, B und C ergeben. Dazu werden die Vektoren normalisiert und das Skalarprodukt gebildet. Der Arkuskosinus dieses Skalarproduktes ergibt den Winkel zwischen den Vektoren als Wert zwischen 0 und Pi, wird also mit 180 multipliziert, um dem Benutzer stets einen Winkel zwischen 0° und 180° in einer QMessageBox zu präsentieren. Im Übrigen wird wieder eine Anmerkung generiert, lediglich der Typ ist verschieden. Zur optischen Verschönerung werden noch 5 weitere Punkte in die pointCloud geschrieben, die einen Teilkreis um A zwischen den Vektoren AB und AC ergeben, der anzeigt, dass dort der angezeigte Winkel gemessen wurde.

Die Methode *on_pushButton_area_clicked* nimmt zunächst alle vom Benutzer gesetzten Punkte und bildet daraus einen Mittelwert, den sie allen anderen Punkten voranstellt, denn auf ihn wird später die Linie vom Anmerkungstext aus zeigen. Um nun den gewünschten Dreiecksfächer zu erzeugen, wird der erste vom Benutzer gesetzte Punkt als Referenzpunkt genommen, um von diesem aus den Winkel (bezüglich des Mittelpunkts) zu allen anderen Punkten zu messen. Die Tupel aus Punkt und Winkel werden als eine QList von vectorAngles gespeichert und mit der Methode *sortVectorAngleList* sortiert. vectorAngle ist dabei ein Struct aus einem QVector3D und einem float-Wert. Um den Kreis zu schließen, wird der erste Punkt nochmal als letzter Punkt angehängt. Dann wird zu allen Dreiecken des Fächers die Fläche berechnet und als Gesamtsumme wiederum in einer QMessageBox ausgegeben. Dabei wird pro Dreieck lediglich eine Grundfläche g festgelegt, die Höhe h darauf bestimmt und die einfache Formel »Grundfläche mal Höhe durch 2« angewendet sowie durch 100 geteilt, um den Wert in Quadratzentimetern zu erhalten.

Die Methoden *on_actionOpen_triggered* und *on_actionSave_triggered* nutzen das Qt-eigene QDomDocument und QDomElement sowie QTextStream, welche es erlauben, objektorientiert beliebige Daten in eine XML-Datei zu schreiben. Das heißt, es muss kein Text geschrieben oder interpretiert werden, sondern die Bezeichner und Werte können direkt abgefragt beziehungsweise gespeichert werden. Die Formatierung der Datei wird automatisch beim Schreiben erzeugt oder beim Lesen interpretiert. Um die zu lesende Datei vom Benutzer abzufragen, kommt wiederum ein Standarddialog von Qt zum Einsatz: QFileDialog. Dieser ist in diesem Fall auf die Dateitypen PLY und STL limitiert. Beim Öffnen wird der Pfad zur aktuellen Datei gespeichert und geprüft, ob bereits eine XML-Datei existiert, deren Inhalt gelesen werden muss. Beim Speichern

stehen diese Informationen folgerichtig schon zur Verfügung und dienen dazu, zu entscheiden, wohin die XML-Datei geschrieben werden muss und ob eine Sicherheitsabfrage nötig ist, die versehentliches Überschreiben von bereits vorhandenen Einstellungen verhindert. Da die XML-Datei unweigerlich im gleichen Verzeichnis wie die PLY- / STL-Datei liegen muss, wurde auf einen Dialog zur Angabe des Speicherorts verzichtet.

Die Methode *on_actionScreenshot_triggered* dagegen nutzt den QFileDialog zur Abfrage des Speicherorts für ein Abbild des glWidgets. Dabei wird von der Methode *grabFrameBuffer* Gebrauch gemacht, die bereits in der Klasse GLWidget enthalten ist. Sie liefert ein QImage zurück, welche den aktuellen Inhalt des Framebuffers enthält. Um ein QImage in eine Datei zu speichern, kann es an eine QPixmap übergeben werden. Diese stellt unter anderem die Methode *save* zur Verfügung, die komfortabel anhand des übergebenen Dateinamens das Format erkennt und die QPixmap entsprechend als Datei speichert.

3.5 Benutzeroberfläche und Bedienung

Im Folgenden werden die einzelnen Bereiche besprochen, in die das Hauptfenster eingeteilt ist, sowie deren Bedienelemente und Funktion beschrieben.

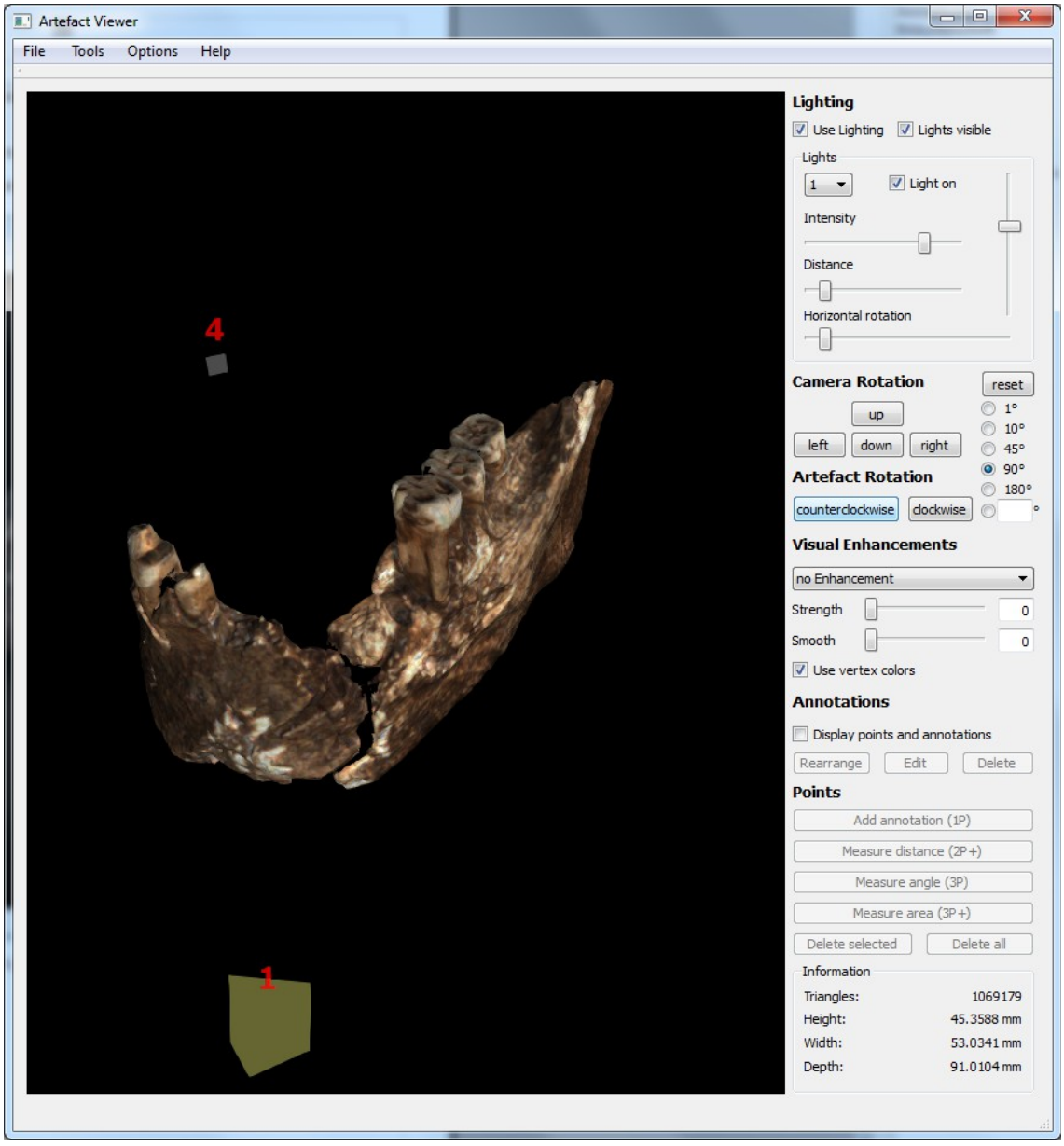

Abbildung 7: Benutzeroberfläche des Artefact Viewers

3.5.1 Menü

File → Open erlaubt es dem Benutzer, eine PLY-Datei (rudimentär auch STL) zu öffnen. Dabei wird automatisch nach einer XML-Datei gleichen Namens gesucht und diese gegebenenfalls geöffnet und deren Einstellungen übernommen.

File → Save speichert die aktuellen Programmeinstellungen in einer XML-Datei mit gleichem Namen wie die aktuell geöffnete Datei im gleichen Verzeichnis.

Tools → Screenshot öffnet einen Datei-Speichern-Dialog, in dem der Benutzer angeben muss, welchen Dateinamen und welches Bildformat er verwenden will. Tut er dies erfolgreich, wird der Inhalt des 3D-Anzeigebereichs in die gewünschte Datei geschrieben. Es stehen diverse von Qt unterstützte Formate zur Verfügung.

Options → Background color öffnet zwei Farbauswahldialoge. Wie die Titel der Fenster andeuten, kann zuerst die untere Farbe, dann die obere ausgewählt werden. Zwischen diesen beiden Farben wird alsdann ein Farbverlauf in den 3D-Anzeigebereich gezeichnet. Ist nur eine Farbe gewünscht, muss zwei Mal die gleiche Farbe ausgewählt werden. Schon nach dem Auswählen der ersten Farbe wird der untere Bereich neu gezeichnet, so dass die Änderung sofort sichtbar ist.

Options → Artefact color Bietet die Möglichkeit, eine einheitliche Farbe für das Artefakt auszuwählen. Diese wird verwendet, wenn »Use vertex colors« deaktiviert ist.

Help → About zeigt Informationen zum Programm an wie Autor und Programmversion.

Help → User Manual (german) zeigt eine mitgelieferte HTML-Datei an, die auf Deutsch in das Programm einführt.

3.5.2 3D-Anzeigebereich

Hier wird das Artefakt dargestellt, sofern eine Datei geladen wurde. Beim Programmstart wird derweil ein Dreieck angezeigt, damit der Benutzer auch ohne eine Datei mit echten Artefaktdaten eine Möglichkeit hat, das Programm auszuprobieren.

Innerhalb des 3D-Bereichs können verschiedene Interaktionen mit der Maus vorgenommen werden:

Drücken und Halten der rechten Maustaste und gleichzeitiges Bewegen der Maus bewirkt eine Drehung der Kamera um deren Bezugspunkt, der standardmäßig in der Mitte des Artefakts liegt. Dabei bleibt das Artefakt immer in aufrechter Position, weil es selber nicht bewegt wird und die Kamera stets mit ihrer Oberseite nach oben schaut und nicht gekippt werden kann.

Durch zusätzliches Drücken der Shift-Taste beim Ziehen mit der rechten Maustaste kann der Bezugspunkt der Kamera und die Kamera selber verschoben werden. Somit kann die Kamera horizontal oder vertikal am Objekt entlang schweben. Dies kann zur Erkundung komplex geformter Artefakte oder zur Ansicht ganzer Höhlenwände sehr nützlich sein.

3.5.3 Lichter

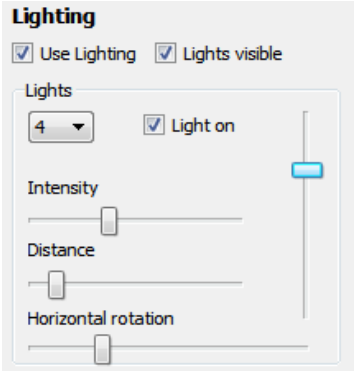

Abbildung 8: Beleuchtung

Ganz oben in der Werkzeugleiste auf der rechten Seite befinden sich die Einstellungen zur Beleuchtung (»Lighting«). Über die Checkbox »Use Lighting« kann die Beleuchtung generell ein- und ausgeschaltet werden. Im Endeffekt wird hier das Phong-Shading ein- und ausgeschaltet. Standardmäßig ist die erste Lampe eingeschaltet, was einerseits zu sehen ist, wenn die 1 aus der Dropdownliste ausgewählt wird, da das Häkchen an der Checkbox »Light on« gesetzt ist. Andererseits kann auch im 3D-Anzeigebereich überprüft werden, dass die Lampe an ist, denn aktivierte Lampen sind gelb, während die ausgeschalteten Lampen grau sind. Um die Lampen zuordnen zu können, sind sie mit roten Ziffern beschriftet.

Dies gilt allerdings nur, wenn die Lampen überhaupt sichtbar sind, denn sie können über die Checkbox »Lights visible« unsichtbar gemacht werden. Sie bleiben dabei als Lichtquelle erhalten, die Lampen selber sind jedoch unsichtbar. Diese Funktion dient dazu, die Lampen für einen Screenshot zu verstecken. Schließlich soll in einer Publikation nur das Artefakt zu sehen sein, nicht die Lampen.

Vier Schiebregler ermöglichen es, die jeweils ausgewählte Lampe genau an die Bedürfnisse anzupassen. »Intensity« regelt dabei die Helligkeit der Lampe. Je mehr Lampen verwendet werden, desto eher muss eine niedrigere Intensität ausgewählt werden, um eine Überbelichtung bestimmter Bereiche zu vermeiden.

Der Regler »Distance« ermöglicht es, den Abstand der jeweiligen Lampe zum Artefakt einzustellen.

Die verbleibenden Schieber, von denen nur der eine mit »horizontal Rotation« beschriftet ist, sind zuständig für die Rotation der Lampen um die Mitte des Artefakts. Dies kann man sich so vorstellen, dass die Lampen generell nur auf der Oberfläche einer Kugel um das Artefakt bewegt werden können. Mit dem schon erwähnten Distance-Regler wird der Radius der Kugel verändert, mit dem Regler für horizontale Rotation kann die Lampe nun auf einer horizontalen Kreisbahn auf der Kugel verschoben werden, mit dem Regler für vertikale Rotation wird die Lampe dagegen auf einer Halbkreisbahn zwischen höchstem und niedrigstem Punkt der Kugel verschoben. Dabei kann das Licht – wie die Kamera auch – nicht über den höchsten bzw. niedrigsten Punkt verschoben werden, da sich dann die Steuerung umkehren würde und der Benutzer verwirrt würde. Um auf die andere Seite der Kugel zu gelangen, muss die horizontale Rotation genutzt werden.

3.5.4 Rotation

Im Bereich »Rotation« kann sowohl die Kamera als auch das Objekt rotiert werden. Hier muss klar unterschieden werden, was gemacht werden soll. In der Regel sollte zunächst das Objekt rotiert werden, um eine natürliche Position im 3D-Raum einzunehmen, eine Vase sollte also aufrecht stehen, mit der Öffnung nach oben und gerade.

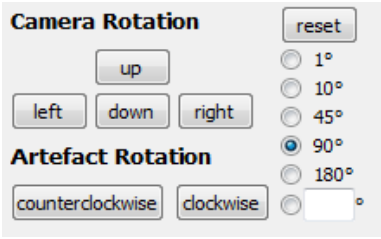

Abbildung 9: Rotation

Zum Erkunden des Objekts wird dann nur noch die Kamera um das Objekt gedreht. Der Unterschied liegt darin, dass die Lichter stets die gleichen Bereiche des Artefakts bescheinen, wenn dieses selber nicht gedreht wird. Dies ist in der Regel der gewünschte Zustand, denn falls andere Bereiche des Artefakts beleuchtet werden sollen, sollte dies über die Einstellungen der Lampen erwirkt werden, nicht durch eine Drehung des Artefakts.

Zunächst muss mit den Radiobuttons auf der rechten Seite eingestellt werden, um wie viel Grad gedreht werden soll. Standardmäßig stehen zur Auswahl 1, 10, 45, 90 oder 180 Grad. Außerdem besteht die Möglichkeit, selber einen Wert einzugeben.

Danach können die Schaltflächen auf der linken Seite genutzt werden. Die vier Flächen »up«, »down«, »left« und »right« sind dabei für die Drehung der Kamera zuständig. Sie bewirken nichts Anderes als das Rechtsklicken und Ziehen der Maus im Anzeigebereich, dienen jedoch dazu, präzise Drehungen möglich zu machen. So können durch Ziehen der Maus keine exakten Drehungen um nur eine Achse ausgeführt werden, wie es für eine klassische 6-fach-Ansicht von Artefakten nötig ist. Mit den beschrieben Schaltflächen ist dies problemlos möglich.

Die beiden Schaltflächen »clockwise« und »counterclockwise« drehen das Objekt in der aktuellen Ansicht im bzw. gegen den Uhrzeigersinn. Die Drehachse ist also eine Gerade von der Kamera aus zum Bezugspunkt der Kamera. Dieser liegt standardmäßig in der Mitte des Artefakts. In der Regel macht es somit Sinn, das Artefakt zunächst zu drehen, solange der Bezugspunkt in dessen Mitte liegt. Erfahrungsgemäß wird diese Aufgabe am besten in drei Teilaufgaben zerlegt, nämlich die Drehung in die korrekte Position bezüglich jeder der drei Raumachsen einzeln. Zur Vereinfachung diese Aufgabe können die Zifferntasten verwendet werden, um die Kamera in bestimmte Positionen zu drehen, ohne mit der Maus immer die Schaltfläche klicken zu müssen. Folgende Tastenbelegungen existieren zu diesem Zweck:

- 1 – Frontalansicht
- 3 – Ansicht von rechts
- 7 – Ansicht von oben

Darüber hinaus können auch die Tasten 2, 4 und 5 gedrückt werden, um je einen Zwischenzustand zu erreichen, der sich aus der Lage der Taste ergibt. So liegt die 4 auf den Nummernblock zwischen 1 und 7 und führt zu einer Ansicht »halb vorne, halb oben«. Ebenso verhält es sich mit der 2. Die Taste 5 wiederum führt zu einer Ansicht von schräg rechts oben, soll einer isometrischen Sicht ähneln und dient dazu, mit einem Tastendruck in eine Position zu kommen, aus der ein Artefakt möglichst schnell von allen Seiten ein bisschen betrachtet werden kann – eine Art kurzer Überblick. Sie könnte aber auch als Standard verwendet werden, wenn ein Artefakt nicht in 6-fach-Ansicht gespeichert werden soll, sondern beispielsweise ein einziges Vorschaubild für eine Datenbank oder einen Katalog benötigt wird.

Um auch später noch Korrekturen vornehmen zu können, kann die Schaltfläche »reset« benutzt werden. Sie fördert ein Abfragefenster zutage. Wird in diesem die Schaltfläche

»Cancel« geklickt, passiert erwartungsgemäß nichts. Wird »No« angeklickt, wird die Kamera wieder in die Ursprungsposition gebracht, die sie auch beim Start des Programms hat. Wird »Yes« angewählt, wird zusätzlich auch die Rotation des Artefakts zurückgesetzt und es wird wieder so angezeigt, wie es in der (PLY-)Datei gespeichert ist.

3.5.5 Visuelle Verbesserungen

In diesem Punkt war die Festlegung von Anforderungen am schwierigsten konkret zu fassen. Erwünscht war, Vorteile der Handzeichnungen wie die Hervorhebung von Merkmalen und die leichte Erfassbarkeit auch in 3D nutzbar zu machen. Allerdings sollte der Prozess vorzugsweise automatisch sein. Dies implizierte, dass vieles an schematischer Darstellung nicht nutzbar war. Bestimmte Schraffierungen für verschiedene Materialien konnten unmöglich verwendet werden, da aus dem 3D-Modell nicht auf das Material geschlossen werden kann. Auch automatische Erkennung der Art der Bearbeitung liegt mit hoher Sicherheit außerhalb des derzeit Möglichen. Im Wesentlichen blieb also die automatische Findung von Kanten übrig.

Hierzu gab es schon sehr fortgeschrittene und aktuelle Ansätze[36], die sich allerdings nicht ohne Weiteres umsetzen ließen. Neben ihrer Komplexität kam vor allem dazu, dass die Struktur der Datenspeicherung im Artefact Viewer darauf ausgelegt ist, Daten pro Punkt zu speichern. Um zusätzlich Kurven, die gefundene Strukturen nachzeichnen sollen, auf das bestehende 3D-Modell zu bringen, hätten diese in Texturen umgewandelt werden müssen. Da ansonsten jedoch keine Texturen verwendet wurden (die Einfärbung der Artefakte erfolgt nur nach Farbangaben an jedem Knoten des Gitters, welches das Artefakt darstellt), existierte diese Funktionalität gar nicht. Als alternative Lösung wäre denkbar, statt Texturen weitere Gitter zu generieren, die an die Oberfläche des Artefakts angepasst würden und minimal darüber lägen, so dass sie als Linie auf dem Artefakt sichtbar wären. Beides könnte für die weitere Entwicklung des Artefact Viewers überdacht werden.

Als wesentlich einfachere Lösung, die zwar etwas weniger gut Linien zeichnen, dafür aber auch künstliche Schatten erzeugen und selbst kleinste Details hervorheben kann,

---

[36] Gilboa, Tal, Shimshoni, Kolomenkin 2013

wurde ein eigener Weg beschritten, der sich die hohe Auflösung der Objekte zunutze macht, um alleine durch Einfärbung entsprechender Knoten ebenfalls Linien zu zeichnen und Schattenwurf zu simulieren. Kernidee dabei war, für alle Knoten zu errechnen, wie erhöht oder vertieft sie gegenüber ihrem Umfeld liegen. Eine Erläuterung hierzu folgt weiter unten, zunächst soll jedoch das Resultat gezeigt werden.

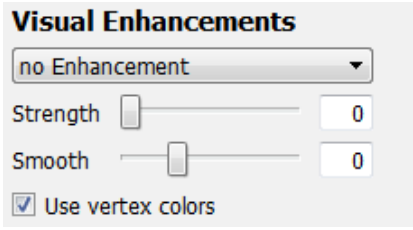

Abbildung 10: Visuelle Verbesserungen

Unter dem Punkt »Visual Enhancements« finden sich vier Optionen, die es erlauben, zusätzlich zu allen anderen Ansichtsoptionen das Objekt an bestimmten Stellen abzudunkeln. Damit soll die optische Wahrnehmbarkeit von Details verbessert werden, ohne dafür Beleuchtung zu nutzen, da diese stets zur Hervorhebung einzelner Details eingerichtet werden kann, jedoch nie alle Stellen gleich optimal beleuchten kann. Darüber hinaus müssen die Details bisweilen erst einmal gefunden werden. Ziel ist es nun also, zusätzlich zur Beleuchtung - oder auch vollkommen ohne sie - künstliche Verdunklungen zu schaffen, die den räumlichen Eindruck verbessern und Unregelmäßigkeiten an der Oberfläche hervorheben.

Für ein optimales Ergebnis ist es möglich, die Stärke des Effekts und die Glättung einzustellen. Darauf wird weiter unten bei der Erläuterung des Algorithmus eingegangen. Die Optionen im Einzelnen sind:

- no Enhancement – die Standardansicht bei Programmstart. Es werden keine Verdunklungen vorgenommen. Schatten resultieren allenfalls aus der Beleuchtung, sofern diese aktiviert ist.
- dark edges – scharfe Kanten werden geschwärzt. Die Stärke stellt ein, wie stark eine Kante sein muss, um überhaupt verändert zu werden. Das heißt, es kann eingestellt werden, ob nur sehr wenige markante Kanten schwarz dargestellt werden sollen oder auch etwas weniger scharfe Kanten in dunkelgrau ebenfalls dargestellt werden sollen. So ergibt sich ein mehr oder weniger kontrastreiches Bild.

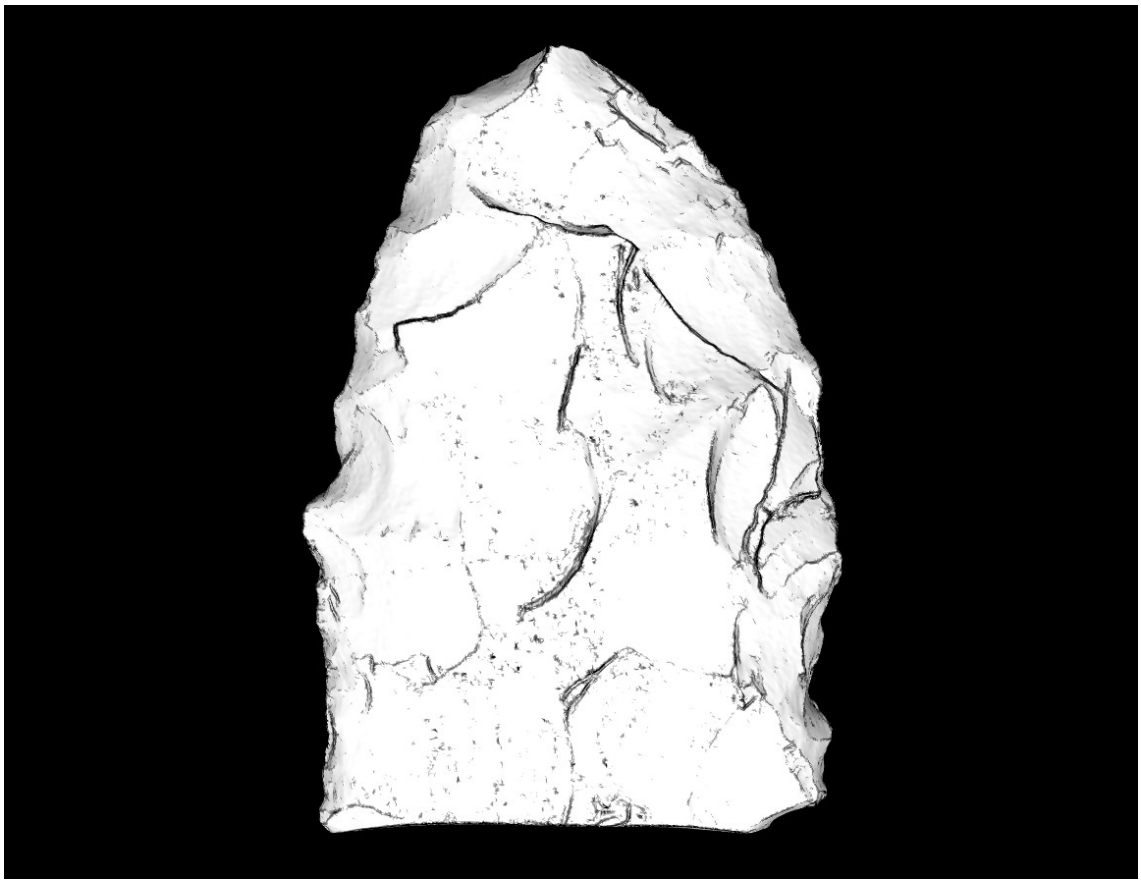

Abbildung 11: Dark-Edges-Algorithmus

- bright edges – Entspricht genau dem Negativ von dark edges. Hierbei werden die flachen Bereiche abgedunkelt und die scharfen Kanten aufgehellt. Diese Option war leicht umzusetzen und diente der Überprüfung, ob ein Negativbild möglicherweise einen zusätzlichen Nutzen bringen konnte.
- dark valleys – Vertiefungen im Artefakt werden dunkler dargestellt. Dies funktioniert am besten in Verbindung mit einigen Iterationen an Glättung.

Im Folgenden soll beschrieben werden, wie die beiden Algorithmen die scharfen Kanten und Vertiefungen finden, zunächst der »dark valleys«-Algorithmus:

Wie schon beschrieben, wird bei der Berechnung der Normalen für jeden Knoten der Oberfläche des Artefakts ein Mittelwert aus den Normalen der Dreiecke gebildet, die den Knoten umgeben.

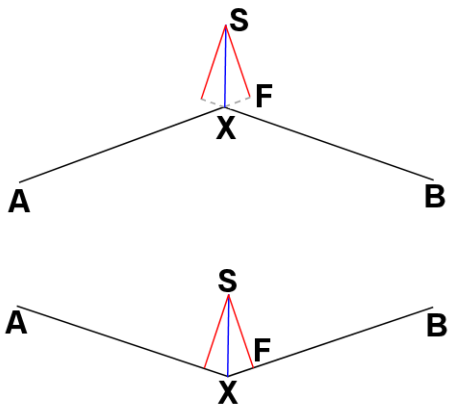

Abbildung 12: Erkennung von Berg und Tal

Zum besseren Verständnis ist auf der Abbildung 12 die Situation im zweidimensionalen Raum illustriert. Man betrachte zunächst den oberen Fall, bei dem zwei Geraden (schwarz) zusammentreffen und einen Berg formen. Sie entsprechen den Dreiecken, die im dreidimensionalen Raum in einem Knoten zusammenlaufen. Blau eingezeichnet ist die nach außen zeigende Normale am Punkt X, der die Bergspitze darstellt. Sie resultiert aus dem Mittel der beiden Normalen zu den Geraden. Im dreidimensionalen Raum können theoretisch beliebig viele Dreiecke einen gemeinsamen Knoten haben. In der Praxis sind es häufig drei bis fünf. Und ebenso viele einzelne Normalen gehen auch in die Berechnung der gemittelten Normalen in Punkt X ein. Im Algorithmus wird die gemittelte Normale normalisiert, also auf eine Länge von 1 gebracht.

Fällt man nun das Lot von der Spitze der Normalen (S) auf jedes der beteiligten Dreiecke, oder – wie in der 2D-Abbildung – auf jede der beteiligten Geraden, so kann man errechnen, ob dieser Fußpunkt (F) innerhalb oder außerhalb des Dreiecks, respektive der Strecke AX oder BX liegt. Wenn man die beiden Fälle vergleicht, fällt sofort auf, dass die Entscheidung innerhalb/außerhalb direkt darauf schließen lässt, ob es sich um einen Berg oder ein Tal handelt. Anders als im zweidimensionalen Fall, kann es im dreidimensionalen Raum allerdings leicht vorkommen, dass bei manchen Dreiecken der Fußpunkt innerhalb liegt und bei anderen außerhalb. Spätestens an dieser Stelle, aber auch, um genauer abbilden zu können, wie stark eine Vertiefung oder Erhebung ist, muss zusätzlich zur qualitativen Aussage, dass ein Berg oder Tal vorliegt, auch eine quantitative Aussage getroffen werden. Da die Normalen stets normalisiert sind, kann der Abstand von F zu X für die Quantifizierung herhalten. Zu beachten ist allerdings, dass jedes in sich geschlossenes Gitter von Knoten nach dieser Definition naturgemäß vor allem Berge hat, da sich sonst keine gekrümmte und dadurch geschlossene Oberfläche darstellen ließe.

Daher gibt es die Möglichkeit, mit dem Strength-Regler auf der Benutzeroberfläche die Verstärkung der Verdunklung an diesen Stellen einzustellen. Steht dieser weit links,

wird ausgehend von einem Helligkeitswert von 1 (keine Verdunklung) die Strecke zwischen F und X abgezogen. Diese ist jedoch bei kleinen Bergen fast 0 und führt daher zu keiner merklichen Verdunklung. Nur durch Verschieben des Reglers nach rechts wird die Verdunklung überhaupt sichtbar. Er funktioniert in gewisser Weise wie ein Kontrastregler bei einem Bildbearbeitungsprogramm.

Ähnlich funktioniert der »dark edges«-Algorithmus. Anstatt alle Winkel zwischen den Dreiecken zu berücksichtigen und dem Benutzer durch den Strength-Regler die Möglichkeit zu geben, die Verdunklung einzustellen, wird hierbei mit dem Regler angegeben, wie groß der Winkel (nicht in Grad) sein muss, damit der Punkt überhaupt geschwärzt wird. Wenn er diese Prüfung besteht, wird er automatisch recht stark geschwärzt. Anders als bei zu schwärzenden Tälern, bei denen recht großzügige Verdunklung gewünscht war, ist das Ziel hier nun dünne Linien an markanten Kanten zu zeichnen, ähnlich wie es bei handgefertigten Skizzen geschieht. Mit dem Strength-Regler kann der Benutzer nun entscheiden, wie viele Details er zu sehen wünscht – je nachdem, welche Kanten er für darstellenswert hält.

Um zu untersuchen, ob eine negative Darstellung vorteilhaft sein kann, wird das Ergebnis für den »bright edges«-Algorithmus invertiert. Ob dies in Bezug auf den Hintergrund oder die gleichzeitige Nutzung von Vertexcolors sowie Beleuchtung einen Vorteil bringen kann, sollte der Test zeigen.

In allen Algorithmen gibt es eine nachgeschaltete Schleife, die über alle Knoten iteriert und dabei die Nachbarschaftsbeziehungen nutzt, die vom mygeometry-Objekt generiert und gespeichert werden. Diese kann – wie bereits angedeutet – über den Smooth-Regler hinsichtlich der Anzahl ihrer Durchläufe gesteuert werden.

Da die Shadowmap stets berücksichtigt wird, wenn der Puffer der Grafikkarte mit Farbwerten für das Artefakt gefüllt wird, ist es problemlos möglich, jederzeit neue Algorithmen zur Dropdown-Liste hinzuzufügen, die bis zu zwei Parameter akzeptieren und als Ausgabe eine Liste von Fließkommazahlen zwischen 0 und 1 ausgeben, deren Länge der Anzahl der Knoten des Artefakts entspricht.

3.5.6 Anmerkungen und Punkte

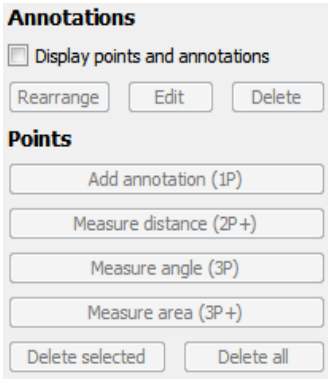

Abbildung 13: Anmerkungen schreiben und Vermessungen vornehmen

Ein weiteres Merkmal, welches den Artefact Viewer deutlich von den bereits im Einsatz befindlichen Softwarelösungen abhebt, sind die Möglichkeiten der Vermessung von Artefakten und deren Kommentierung.

Durch das Setzen des Häkchens bei »Display points and annotations« können diese eingeschaltet werden.

Um eine Anmerkung zu erstellen, muss im einfachsten Fall das Artefakt an der zu kommentierenden Stelle angeklickt werden, wodurch an der Stelle ein blauer Punkt gesetzt wird. Ein gesetzter Punkt führt dazu, dass die Schaltfläche »Add annotation« aktiviert wird. Bei Betätigung der Schaltfläche wird in einem kleinen Fenster die Möglichkeit geboten, einen Text einzugeben. Hat der Benutzer diesen bestätigt, kann er sich noch eine Farbe für den Punkt aussuchen. Ist diese ebenfalls bestätigt, ist der Vorgang abgeschlossen. Die Anmerkung steht nun links oben im 3D-Anzeigebereich und zeigt mit einer Linie auf den betreffenden Punkt. Alle diese als 2D-Overlay gezeichneten Elemente können nicht vom Artefakt verdeckt werden. Sie bleiben demzufolge auch sichtbar, wenn sie sich auf der dem Benutzer abgewandten Seite des Artefakts befinden.

Soll eine Strecke gemessen werden, müssen dafür mindestens zwei Punkte angegeben werden. In dem Fall wird die direkte Verbindung zwischen den Punkten, also der Vektor, in seiner Länge gemessen und das Ergebnis in einem Fenster angezeigt, welches wiederum die Möglichkeit eröffnet, daraus eine Anmerkung zu generieren. Ist dies nicht gewünscht, kann der Wert einfach abgelesen und dann abgebrochen werden. Es wird dann keine Anmerkung generiert und auch keine Linie eingezeichnet. Soll eine komplexe Strecke gemessen werden, können dazu beliebig viele Punkte angeklickt werden, die Strecke wird als Summe der Teilstrecken berechnet und angezeigt.

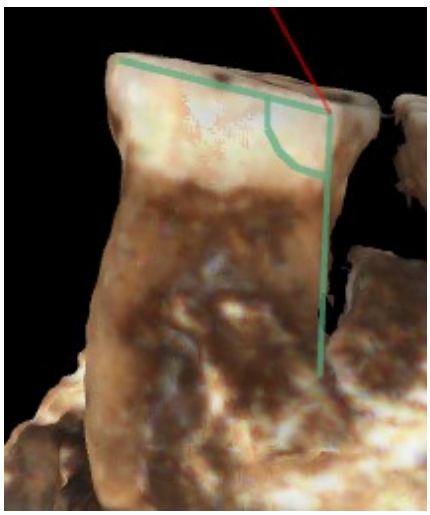

Abbildung 14: Winkelmessung

Exakt drei Punkte müssen gesetzt werden, um einen Winkel zu messen. Dabei wird der Winkel für drei

Punkte A, B und C am Punkt A bestimmt. Auch hier wird der Wert in einem Fenster angezeigt und die Möglichkeit der Generierung einer Anmerkung geboten. Die Reihenfolge der Punkte B und C ist unerheblich, da nur Winkel gemessen werden können, die kleiner als 180 Grad sind.

Darüber hinaus können drei und mehr Punkte auch zur Berechnung einer Fläche genutzt werden. Dabei muss bedacht werden, dass nur konvexe Flächen sicher gemessen werden können. Ein gewisses Maß an Einbuchtungen des Umrisses stellt jedoch kein Problem dar. Die Vorgehensweise, nach der der Dreiecksfächer aufgespannt wird, der für die Flächenberechnung genutzt wird, stellt sicher, dass es keine überschneidenden Dreiecke gibt, um stets eine korrekt berechenbare Fläche zu erhalten. Dabei wird aus allen Punkten, die vom Benutzer gesetzt werden, bei Betätigung des Flächenberechnungsknopfes ein Mittelwert gebildet und daraus ein zusätzlicher Punkt generiert. Er bildet den Mittelpunkt eines Dreiecksfächers, an dem all die vom Benutzer gesetzten Punkte teilnehmen. Hieraus ergibt sich die Notwendigkeit, die Punkte in der richtigen Reihenfolge entlang des Umrisses des zu messenden Bereiches zu sortieren. Werden die Punkt willkürlich durcheinander gesetzt, käme es sonst zu sich überschneidenden Dreiecken und somit zu einem falschen Ergebnis der Flächenmessung. Die automatische Sortierung funktioniert hierbei wie ein Uhrenzeiger, der um den Mittelpunkt läuft, wobei er beim ersten vom Benutzer gesetzten Punkt aus startet und dann die Punkte in der Reihenfolge notiert, in der er sie überstreicht, bis er wieder am Startpunkt angekommen ist. Werden die Punkte in dieser Reihenfolge zum Rand des Dreiecksfächers verbunden, entsteht zwingend eine ausreichend konvexe Form, die keine Überschneidungen von Dreiecken zulässt.

Wird diese Vorgehensweise nicht berücksichtigt, indem versucht wird, eine konkave Form zu zeichnen, **können** möglicherweise unerwünschte Formen berechnet werden. Von jedem Punkt auf dem Rand muss – vereinfacht gesprochen – der Mittelpunkt zu sehen sein. Oder anders ausgedrückt, zwischen jedem Punkt und dem Mittelpunkt muss eine Linie gezogen werden können, die nicht durch Gebiet läuft, welches bereits wegen anderer Punkten berücksichtigt wird.

Sind alle Punkte korrekt gesetzt, wird zu jedem Dreieck der Flächeninhalt bestimmt, die Flächen addiert und die Summe wieder in einem Fenster angezeigt, welches die Möglichkeit bietet, eine Anmerkung zu generieren.

Hierbei ist anzumerken, dass die gemessene Größe, sei es nun Länge, Winkel oder

Fläche, stets als Standardtext vorgegeben ist, damit diese komfortabel übernommen werden kann. Sie kann selbstverständlich beliebig ergänzt, geändert oder gänzlich gelöscht werden.

Um leichter mit den verschiedenen Anzahlen von benötigten Punkten zurecht zu kommen, ist als Gedächtnisstütze für den Benutzer auf jeder Schaltfläche die notwendige Punktzahl in Klammern erläutert. (3P+) heißt also beispielsweise, die Schaltfläche kann benutzt werden, wenn drei oder mehr Punkte gesetzt wurden.

Solange die Punkte noch gesetzt werden und keine Schaltfläche angewählt wurde, sind sie noch veränderbar. So können sie mit der linken Maustaste geklickt und gezogen werden, wenn ihre Position korrigiert werden muss. Der angeklickte Punkt wird weiß markiert und kann durch Betätigung der Schaltfläche »Delete selected« auch gelöscht werden. Es können alternativ auch mit »Delete all« alle noch nicht einer Anmerkung zugewiesenen Punkte gelöscht werden.

Ist die Anmerkung erst einmal bestätigt, sind die Punkte fixiert und können nicht mehr geändert werden. Allerdings kann die Anmerkung selber noch bearbeitet werden. Dazu ist die betreffende Anmerkung selber anzuklicken und die Schaltfläche »Edit« zu verwenden oder die Anmerkung einfach zu doppelklicken. Sie führt wieder zum Eingabefenster für Anmerkungen und zur Farbauswahl. Ebenso verhält es sich mit »Delete«, was die Anmerkung und die zugehörigen Punkte löscht.

Wenn viele Anmerkungen existieren, bleibt es - spätestens wenn das Objekt gedreht wird - nicht aus, dass sich die Linien von den Texten zu den Punkten überschneiden. Daher kann, wenn das Objekt – beispielsweise für einen Screenshot – fertig gedreht wurde, die Schaltfläche »Rearrange« angeklickt werden. Dies bewirkt, dass die Höhen der relevanten Punkte auf dem Bildschirm ermittelt werden und die Anmerkungen dementsprechend umsortiert werden. Danach sind die Linien meistens entwirrt. Dies ist jedoch keine permanente Lösung. Wird das Objekt gedreht, können die Linien sich erneut überschneiden. Eine automatische Sortierung der Anmerkungen würde jedoch die Bildrate beim Drehen mindern, eventuell durch die Unruhe den Benutzer verwirren und letztlich kann auch nicht mit Sicherheit gesagt werden, dass die Reihenfolge der Anmerkungen nicht trotz Überschneidungen genau so bleiben soll, wie sie ist. Daher wurde die Rearrange-Funktion als manuelle Schaltfläche realisiert.

3.5.7 Informationen zum Artefakt

Ganz unten rechts im Fenster befindet sich ein Informationsbereich, der die Größe des betrachteten Artefakts anzeigt. Dabei ist die erste Zeile ein Maß für die Größe als digitales Objekt. Die Anzahl der Dreiecke sagt etwas über den Bedarf an Speicher und Rechenleistung aus, die benötigt werden, um das Artefakt ansehen zu können.

Die drei verbleibenden Zeilen geben Höhe, Breite und Tiefe des Artefakts an, welches repräsentiert wird. Diese Originalmaße sind dann korrekt, wenn eine Einheit in OpenGL einem Millimeter in der Realität entspricht. Die Maße sind jedoch auf die aktuelle 3D-Ansicht bezogen. Wird das Objekt selber gedreht und nicht die Kamera, ändern sich dadurch auch seine Maße. Es muss also flach gelegt oder hochkant gestellt werden, um die richtigen Maße ablesen zu können.

4 Test

Im Folgenden wird der Testaufbau mit den gestellten Aufgaben und den Testkriterien vorgestellt.

Um einerseits die Einfachheit der Bedienung und Nützlichkeit zu testen und andererseits die fehlerfreie Funktion der Software zu prüfen, wurde auf eine heuristische Evaluation zurückgegriffen. Diese Methode des Testens ist besonders geeignet, wenn innerhalb kurzer Zeit und mit relativ wenig Personaleinsatz eine große Anzahl kleinerer Fehler und natürlich nahezu alle großen Fehler beseitigt werden sollen.[37]

Der zentrale Gedanke hierbei ist, dass bereits wenige Experten für Benutzerschnittstellen aufgrund ihrer Erfahrung im Umgang mit verschiedensten Programmen und Eingabekonzepten einen Großteil der Mängel in der Benutzeroberfläche finden können, wenn sie diese nach eigenem Ermessen durchforsten können. Vorteilhaft wäre dabei gewesen, Personen zu finden, die gleichzeitig Experten im Bereich Benutzerschnittstellen **und** Archäologie sind. Da dies leider nicht möglich war, wurden zwei Testgruppen gebildet mit Experten auf dem jeweiligen Gebiet. Auch dieses Vorgehen führt zu guten Ergebnissen beim Erkennen von Mängeln. Eine Gruppe Informatiker beurteilte die Software nach anerkannten Standards für Benutzerfreundlichkeit und eine zweite Gruppe von Archäologen wurde gebeten, die Software vor allem in Bezug auf die Nützlichkeit für ihre Aufgaben zu beurteilen, indem sie versuchten, damit möglichst realitätsnah zu arbeiten. Auf diese Weise fand jede der Gruppen unterschiedliche Mängel und zeigte Verbesserungsmöglichkeiten auf, wenn Aufgaben offenkundig nicht oder nur umständlich zu bewältigen waren.

Der Art des Tests entsprechend, konnte jede Testperson ihrer Meinung nach wichtige Kriterien testen. Da die meisten Personen die Software zum ersten Mal sahen und benutzten, wurde dennoch ein Satz von Kriterien festgelegt, der mindestens berücksichtigt werden sollte. Vor allem zum besseren und strukturierteren Kennenlernen der Software und ihrer Funktionen wurde außerdem ein Aufgabenkatalog festgelegt, der durchgearbeitet werden sollte, um sicher zu sein, alle Funktionen einmal benutzt zu haben. Darüber hinausgehend konnte jede Testperson zwischendurch oder nachträglich alles ausprobieren, was ihr aus ihrer Sicht sinnvoll erschien.

---

[37] Nielsen 1992

Alle Anmerkungen, die die Testpersonen machten, wurden protokolliert. Außerdem wurde vermerkt und diskutiert, wenn Aufgaben scheinbar oder tatsächlich nicht gelöst werden konnten. Entweder stellte dies einen Hinweis dar, eine bestimmte Funktion besser zu dokumentieren, so dass der Benutzer sie besser finden und verwenden konnte, oder es zeigte auf, welche zusätzlichen Funktionen noch implementierenswert waren. Folgende Aufgaben wurden gestellt – eine detailliertere Beschreibung findet sich im Anhang A.

4.1 Aufgaben

- Eine Datei öffnen
- Ein bereits ausgerichtetes Artefakt frei erkunden
- Die Kamera präzise ausrichten
- Ein verdrehtes Artefakt ausrichten
- Den Hintergrund einstellen
- Die Artefaktfarbe einstellen
- Vertexcolors nutzen
- Fertig ausgerichtetes Artefakt beleuchten
- Eine Anmerkung zu einem Punkt schreiben
- Eine Strecke messen und dabei manche Punkte nachträglich korrigieren und löschen (ohne Anmerkung)
- Einen Winkel messen und dabei eine Anmerkung erzeugen
- Eine Fläche messen und dabei eine Anmerkung erzeugen
- Anmerkungen sortieren lassen
- Eine Anmerkung editieren
- Eine Anmerkung löschen
- Eine adäquate Einstellung für »dark edges« finden

- Eine adäquate Einstellung für »dark valleys« finden
- Einen Screenshot machen
- Mit der Kamera seitwärts am Artefakt entlang fahren

4.2 Kriterien

Bei allen Aufgaben wurde – sofern dies an der Stelle sinnvoll erschien - folgende Aspekte entweder durch Beobachtung erfasst oder durch Befragung ergründet:

- Wie lange braucht der Proband, um die Aufgabe zu erfüllen?
- Wurden zunächst nicht zielführende Aktionen ausgeführt?
  - Konnte der falsche Weg problemlos wieder verlassen werden?
    - Warum nicht? Welche Verbesserungen sind nötig? Programmfehler?
- Brauchte der Benutzer Hilfe? Muss diese ins Handbuch aufgenommen werden?
- Wie intuitiv wurde die Bedienung empfunden?
- Entsprechen die benötigten Schritte allgemein anerkannten Standards?
- Kann der Proband sich einen besseren Lösungsweg denken?
- Findet er die entsprechende Funktion nützlich?
- Fehlt bestimmte Funktionalität?

Bei vielen Aufgaben konnten nur einige dieser Fragen sinnvoll gestellt werden. So mag niemand die Notwendigkeit und Nützlichkeit der Funktion »Öffnen« im Menü anzweifeln. Deren korrekte Funktion auch bei von der beabsichtigten Bedienung abweichendem Verhalten war jedoch durchaus wichtig.

An anderer Stelle mochten sogar weitere, bislang nicht aufgeführte Kriterien relevant werden, weswegen die Fragen nur den Charakter eines Denkanstoßes haben, kein festes Muster vorgeben sollten. Entsprechend frei konnten und mussten die Ergebnisse des Tests formuliert werden.

5 Ergebnisse

Die beiden Standardfunktionen des Öffnens und Speicherns von Dateien erwiesen sich als unproblematisch. Alle Probanden fanden und verstanden die Funktionen unverzüglich. Manche Probanden lasen dabei auch die anderen Menüpunkte und fanden daher auf Anhieb auch die Screenshot-Funktion und die beiden Menüpunkte zur Änderung der Artefakt- und Hintergrundfarbe. Andere suchten zunächst die Bedienelemente auf der rechten Seite flüchtig ab und wandten sich dann der Menüleiste zu. Für alle galt, dass sobald die Funktion im Menü gefunden wurde, ihre Bedienung intuitiv klar war.

Vermisst wurde von einem Probanden unter dem Menüpunkt »Help« die Anwesenheit eines Handbuches. Dies war in der Testversion noch nicht implementiert.

Der erste Versuch, das Objekt frei zu erkunden, wurde von allen Probanden mit der linken Maustaste begonnen. Die benötigte Zeit, um herauszufinden, dass die rechte Taste gedrückt werden musste, variierte zwischen unter einer Sekunde und kompletter Resignation, die sich darin äußerte, anklickbare Buttons für die Funktionalität zu suchen. Spätestens nach einem Hinweis auf die richtige Taste war jedoch das Rotieren der Kamera um das Artefakt für jeden Probanden unproblematisch. Die Zoomfunktion per Mausrad wurde von allen Probanden bis auf einen sofort entdeckt und intuitiv genutzt. Die betreffende Person war gewohnt, eine Zusatztaste auf der Tastatur in der Verbindung mit einer Maustaste für den Zoom zu drücken, konnte aber nach entsprechendem Hinweis auch mit dem Mausrad tadellos auskommen.

Das Verständnis darum, wie die Kamera mithilfe der Schaltflächen bewegt werden kann, differierte stark. Das Spektrum reichte hier von einem Zugang, so intuitiv, dass es schien, als hätten sie das Programm schon länger in Gebrauch bis totaler Verwirrung, was genau passierte und wo sich die Kamera im Raum hinbewegte oder befand. Damit korrelierte direkt, wie gut die jeweilige Person mit der Aufgabe des Ausrichtens des Artefakts zurecht kam.

Erwartungsgemäß war die Vorgehensweise zur korrekten Ausrichtung des Artefakts nicht intuitiv erkennbar. Alle Probanden benötigten eine mündliche Erklärung, dass das Artefakt relativ zum Betrachter gedreht werden muss und daher zunächst die Kamera in günstige Positionen verschoben werden musste. Dass diese Positionen sich in der Regel auf den drei Raumachsen befanden und mit welcher Position sie anfangen mussten, um schnellstmöglich zum Ziel zu kommen, musste durch Ausprobieren oder Vorführen

erlernt werden. Die Lerngeschwindigkeit differierte dabei zwischen guter Beherrschung des Systems innerhalb von etwa 2 Minuten und nicht optimalem, aber dennoch zielführendem Verständnis innerhalb von etwa 10 Minuten.

Alle Probanden waren jedoch nach dem Test der Ausrichtungsfunktion der Meinung, dass die Vorgehensweise sinnvoll sei und relativ leicht zu bedienen, wenn man denn wisse, wie es geht. Alternativvorschlag war oftmals, die Drehung des Artefakts mithilfe von einer »Trackball Rotation« zu ermöglichen. Ein Proband wünschte sich, die Artefaktrotation mit einer Checkbox deaktivieren zu können, um nach der Ausrichtung weiteres Verdrehen zu verhindern.

Beim Einstellen der Hintergrundfarbe wurde von keinem Probanden die Fensterüberschrift bemerkt und gelesen, in der beschrieben ist, was genau in dem Fenster auszuwählen ist, nämlich zuerst die untere, dann die obere Farbe. Die anfängliche Konfusion dauerte jedoch nicht lange, da direkt sichtbar wurde, was die ausgewählte Farbe bewirkte. So wurde die Farbauswahl sofort wiederholt und richtig benutzt, falls dies noch für nötig befunden wurde.

Das Einstellen der Artefaktfarbe war dagegen für niemanden ein Problem, nur ein Proband wunderte sich darüber, dass das Artefakt trotz Auswahl von weißer Farbe nur grau war. Dies lag an der aktivierten Beleuchtung und den damit verbundenen Graustufen durch Schattierungen.

Zu diesem Zeitpunkt hatten alle Probanden bereits die Bedienoberfläche so weit wahrgenommen und gemerkt, dass sie bei der Aufforderung, das Objekt nach ihren Vorstellungen zu beleuchten, gleich die dafür vorgesehenen Bedienelemente ausprobierten. Das Wechseln zwischen den Lampen wurde sofort intuitiv richtig genutzt. Die Nummerierung der Lampen war dabei hilfreich, alle Probanden mussten sich jedoch früher oder später damit beschäftigen, die Kamera so einzustellen, dass sie Lampen und Artefakt gleichzeitig im Blick hatten. Manche merkten an, dass der vertikale Regler nicht beschriftet sei, verwendeten ihn jedoch ohne Probleme. Als Verbesserungsvorschlag wurden die Anzeige von Kreisen als grafische Repräsentation der Bewegungsmöglichkeiten der Lampen um das Artefakt vorgeschlagen. Ein Proband hielt es für sinnvoll, einen Mindestabstand für Lampen zum Objekt vorzugeben, um zu verhindern, dass die Lampe im Objekt verschwindet.

Das Häkchen zur Umschaltung Vertexcolors/Artefaktfarbe wurde von manchen

Probanden zu diesem Zeitpunkt bereits bemerkt, bei den übrigen wurde nicht darauf gewartet, dass sie es finden, da nicht davon auszugehen war, dass sie den Begriff Vertexcolors kannten. Stattdessen wurden sie angewiesen, das Häkchen einfach zu verwenden und zu beobachten, was passiert. Damit kamen alle Probanden zurecht und verstanden den Unterschied in der Darstellung, spätestens nachdem probeweise die Beleuchtung abgeschaltet wurde.

Das System, erst Punkte zu setzen und dann zu entscheiden, was damit passieren soll, erschloss sich intuitiv keinem Probanden. Nach einem Hinweis diesbezüglich wurde schnell verstanden, dass unterschiedliche Punktzahlen unterschiedliche Optionen freischalten. Eine Anmerkung zu schreiben, wurde zumeist aus eigenem Antrieb ausprobiert, da dies ja auch eine der Hauptanforderungen an die Software war, und es wurden auf Anhieb die gewünschten Ergebnisse erzielt. Dieses Wissen zu benutzen, um die leicht erweiterte Aufgabe des Messens einer Strecke zu bewältigen, bereitete keinem Probanden Schwierigkeiten.

Bei der Winkelmessung musste vorab darauf hingewiesen werden, an welchem Punkt der Winkel gemessen werden würde. Dass drei Punkte benötigt wurden, war den Probanden intuitiv klar, da auch andere bereits benutzte Software so funktionierte. Mit diesem Vorwissen wurde spätestens im zweiten Anlauf das gewünschte Ergebnis erreicht. Als optische Verschönerung wurde gewünscht, dass nicht nur zwei Geraden den Winkel markieren, sondern ein kleiner Teilkreis zwischen den Geraden anzeigt, dass es um den Winkel zwischen den Geraden geht, nicht um eine gemessene V-förmige Strecke. Dies war in der Testversion noch nicht implementiert.

Die Flächenberechnung bereitete weniger Probleme als erwartet, da die Probanden entweder durch ihre Erfahrungen mit der Software ArteCore wussten, wie die Flächenberechnung in etwa funktionieren würde, oder durch ihr Vorwissen in Sachen Informatik/Mathematik über Dreiecksfächer Bescheid wussten. Die Punkte wurden reihum gesetzt und vorgebracht, dass es sehr praktisch wäre, die Punkte nicht in einer bestimmten Reihenfolge setzen zu müssen sowie einzelne Punkte editieren zu können, wie es auch in ArteCore möglich sei. Beides war in der Testversion noch nicht implementiert. Die sonstigen Aspekte wie Farbwahl und Kommentar waren durch die anderen Funktionen zur Erstellung von Anmerkungen schon ausreichend trainiert, um routiniert benutzt zu werden.

Durch die an dieser Stelle bei den meisten Probanden vorhandenen zahlreichen Anmer-

kungen wurde Notwendigkeit und Funktion der Schaltfläche »Rearrange« verstanden, wenn sie aufgefordert wurden, diese einmal auszuprobieren. Kein Proband empfand die »Unordnung« bis hier her bereits so störend, dass er darauf hingewiesen oder gefragt hätte, ob es eine Möglichkeit zur Umsortierung gibt.

Die Schaltflächen zum Löschen von Anmerkungen oder auch gesetzten Punkten wurden während der vorangegangenen Aktionen ohnehin schon gefunden und teilweise intuitiv genutzt. Auch den übrigen Probanden waren die Schaltflächen bereits ins Auge gefallen und konnten auf Aufforderung, eine Anmerkung zu löschen, auch sofort genutzt werden. Bisweilen wurde allerdings das Löschen von Punkten und Anmerkungen verwechselt. Ähnliches galt für das Editieren von Anmerkungen. Als Verbesserungsvorschlag wurde das Editieren von Anmerkungen durch Doppelklick gefordert. Dies war in der Testversion noch nicht implementiert.

Die Seitwärtsbewegung der Kamera stand als Letztes auf dem Programm, wurde von manchen Probanden jedoch schon vorher gesucht und nachgefragt. Die neuen Bewegungsmöglichkeiten bereiteten keine Schwierigkeiten und konnten sofort benutzt werden, um Wände zu erkunden. Allerdings hatte die Testversion in diesem Bewegungsmodus einige Fehler, die teilweise zu falschen Bewegungsabläufen führten. Während des Testen konnten diese Fehler durch Neustart des Programms umgangen und der Test weitergeführt werden. Alle gefundenen Fehler konnten in der Endversion 1.0 behoben werden.

Als allgemeine Anmerkung wurde von einem Probanden noch angeführt, dass es wünschenswert wäre, mehrere Punkte selektieren und verschieben zu können.

6 Diskussion der Testergebnisse

Viele Verbesserungsvorschläge waren relativ leicht umzusetzen und boten einen echten Mehrwert, so wurden Punkte nachträglich selektier- und verschiebbar gemacht und der selektierte Punkt statt dem letzten gesetzten Punkt löschbar gemacht. Bezüglich der Flächenberechnung wurde gegenüber ArteCore durch die Verwendung eines automatisch berechneten Mittelpunktes schon ein deutlicher Vorteil erzielt, da deutlich konkavere Formen problemlos berechnet werden können. Das zusätzliche automatische Sortieren der Punkte und die somit vorhandene Möglichkeit, jederzeit zusätzliche Punkte zu setzen, um den Umriss der zu berechnenden Fläche noch genauer zu gestalten, wurde als deutliche Verbesserung empfunden.

Dem Vorschlag, Visualisierung von Bewegungsmöglichkeiten oder Bedienelementen in Form eines Trackballs um das Artefakt zu schaffen, konnte nicht entsprochen werden. Sowohl zur Drehung des Artefaktes um sich selber als auch zur Drehung der Lampen um das Artefakt wären letztlich Kreise um das Artefakt selber nötig gewesen. Diese wären voneinander nicht zu unterscheiden. Einziger Ausweg wäre, diese Bedienelemente wiederum ein- und ausblendbar zu machen, was allerdings den Bedienkomfort wieder deutlich reduzieren würde und somit keinen intuitiveren Zugang gewähren würde, zumal Fehlbedienungen durch Verwechslung vorprogrammiert wären.

Der Vorteil wäre zum Zwecke der präzisen Ausrichtung ohnehin nicht so groß, wie es auf den ersten Blick scheint, da für eine präzise Ausrichtung exakt entlang der drei Raumachsen geschaut werden muss. Ist dies aber der Fall, können genauso gut die vorhandenen Bedienelemente genutzt werden. Einzig bei Benutzern, die Schwierigkeiten haben, intuitiv zu erkennen, welche Achse zunächst korrekt ausgerichtet werden muss, könnten es nützlich sein, einfacher ausprobieren zu können, ob sie dem gewünschten Ergebnis näher kommen. Es ist aber davon auszugehen, dass Benutzer nach der Ausrichtung einiger Artefakte ohnehin wissen, wie sie am schnellsten zum Ziel kommen.

Entsprechend der Vorgehensweise, dass Objekte eingescannt, ausgerichtet werden und mit diesen Ausrichtungsdaten im Internet veröffentlicht werden sollten, ist nur ein relativ kleiner und routinierter Teil der Benutzer überhaupt von der Ausrichtung betroffen. Die späteren Benutzer der Datenbank müssen keine weitere Ausrichtung des Artefakts vornehmen, wenn sie nicht wollen.

Bezüglich der Fehlerfreiheit des Programms war der Test sehr erfolgreich. Da vor allem die schwerwiegenderen Fehler teilweise auch von Testkandidat zu Testkandidat behoben wurden, nahm die Zeit, die es brauchte, einen weiteren Fehler zu finden, deutlich zu. Die Fehler führten grundsätzlich nicht zum Absturz des Programms, sondern lediglich zu mehr oder weniger störenden Fehlfunktionen. So war beispielsweise ein Punkt eingefärbt, der **nicht** ausgewählt war, was grundsätzlich die Funktion bezüglich (beispielsweise) des Messens von Strecken nicht einschränkte. Zum Ende des Tests kann somit zurecht behauptet werden, dass ein produktives Arbeiten problemlos möglich ist, da alle Funktionen mehrfach erfolgreich und fehlerfrei verwendet wurden.

Den Aussagen der Nutzer zufolge ist das Programm aufgrund der in der Endversion klar unterteilten Bedienelemente entweder durch eine kurze Einweisung oder Selbststudium des mitgelieferten kompakten Handbuchs in kurzer Zeit zu verstehen.

Schon das Grundgerüst der Software vor den ersten Tests wurde von den Nutzern positiv aufgenommen, die Änderungen während des Tests und im Anschluss daran verhalfen der Software zur echten Praxistauglichkeit, da sie nun alle positiven Eigenschaften der verschiedenen bisher eingesetzten Programme in sich zusammengefasst hat.

Bezüglich der MoSCoW-Priorisierung wurde sogar deutlich mehr erreicht, als zu Anfang gedacht, so dass Klagen über dringend benötigte, jedoch fehlende Funktionen ausblieben.

Alle MUST-Kriterien konnten so umgesetzt werden, wie sie auch ursprünglich gefordert wurden.

Bei den SHOULD-Kriterien konnten alle bis auf den Punkt 2 direkt umgesetzt werden. Der Punkt 2 sah vor, dass die Darstellung mehrere Ebenen vorsehen sollte. Angedacht war also eine Funktionalität ähnlich wie in gängigen 2D-Bildbearbeitungsprogrammen, wo Bilder auf verschiedene Ebenenen verteilt werden können, für die Einstellungen festgelegt sind, nach denen die Ebenen zu einem Gesamtbild zusammengefasst werden. Der wichtigste Aspekt davon wird erfüllt, nämlich, dass das Artefakt mit Vertexcolors überzogen werden kann und über diese nochmal die visuellen Verbesserungen eingeblendet werden können. Dies ist nicht ganz dasselbe, war aber nach Aussagen der Testpersonen vollkommen ausreichend. Wirklich beliebig viele Ebenen mit ebenfalls beliebigen Informationen zu füllen, würde eine grundsätzliche Überarbeitung der

Software nötig machen, deren Sinnhaftigkeit in Zukunft genau geprüft werden müsste.

Ähnlich verhält es sich bei den COULD-Kriterien. Hier wurden einige Punkte nicht implementiert, die wiederum Ebenen und Texturen erfordert hätten, wie die Zeichenfunktionen auf dem Artefakt. Andere Funktionen wurden von den Endbenutzern während der Entwicklung als weniger wichtig eingestuft, wie die Berechnung von Volumen und Oberfläche. Der Berechnung von Winkeln und Flächen und der Möglichkeit, auch im Programm Anmerkungen editieren und speichern zu können und nicht nur mit einem festen Set an Anmerkungen zu arbeiten, welches in einer XML-Datei vom Herausgeber der Datei vorgegeben war, wurde deutlich mehr Priorität eingeräumt, daher wurden auch diese Funktionen noch implementiert.

Selbst die WON'T-Kriterien wurden teilweise noch berücksichtigt. Aufgrund der Tatsache, dass diese zu Beginn der Entwicklung nur recht vage definiert werden konnten, sah die Implementierung allerdings etwas anders aus und orientierte sich mehr an den Wünschen, die die späteren Benutzer aufgrund des jeweils vorgelegten aktuellen Entwicklungsstands der Software äußerten. So wurde der Punkt 1 im Grunde erfüllt durch die visuellen Verbesserungen, die einen solchen Algorithmus darstellen, der Kanten findet oder zumindest für den Menschen besser sichtbar macht.

Der Punkt 2, die 6-fach-Ansicht, wird zwar nicht in einem Schritt generiert, es stehen aber alle Bedienelemente zur Verfügung, um die 6 Ansichten präzise einzeln anzufertigen. Ein zusätzliches Bildbearbeitungsprogramm wird daher benötigt, was allerdings – wie sich nachträglich herausstellte – sowieso so vorgesehen war.

Der universelle Flächenmesser für alle Formen wurde nicht erreicht, sehr wohl aber eine deutliche Verbesserung gegenüber der aktuell im Einsatz befindlichen Software, so dass in der Praxis wohl die allermeisten Flächen problemlos gemessen werden können sollten.

Der Punkt 4 fiel den fehlenden Texturebenen zum Opfer, der Punkt 5 erwies sich als unnötig, da sich herausstellte, dass bereits die Scannersoftware eine Ausgabe der Scans in verschiedenen Detailstufen erlaubt, so dass eine spätere Reduzierung unnötig ist.

7 Ausblick & Fazit

Trotz Erfüllung aller wichtigen vorab definierten Anforderungen und somit dem Erreichen der Zielsetzung, bleibt vielfältiges Entwicklungspotenzial, wie sich in Absprache mit den Forschern des Neanderthal-Museums während der Entwicklung gezeigt hat.

7.1 Große Datenmengen

Es existieren teilweise extrem große PLY-Dateien, vor allem von Wandscans, die mit der Version 1.0 der Software nicht eingelesen werden können. Bezüglich der konkret vorliegenden Dateien gab es jeweils passend auch niedriger aufgelöste Versionen, die stattdessen verwendet werden konnten – wobei niedrig in diesem Fall relativ ist bei immer noch mehreren Millionen Knoten. Für den Fall, dass keine niedrigere Auflösung zur Verfügung steht, wäre eine Funktion zur automatischen Reduzierung der Geometrie von Vorteil, welches möglichst viele Details erhält. Allerdings muss auch die Bandbreite bedacht werden. Schon die niedrig aufgelösten Versionen liegen im Bereich hunderter Megabytes. Die hochauflösenden Versionen liegen bei einigen Gigabytes und trotz technischer Machbarkeit muss an dieser Stelle natürlich auch die Kostenfrage für die Infrastruktur gestellt werden. Daher wäre es vorteilhaft, eine unabhängige vorgeschaltete Software zur Geometriereduzierung zu verwenden, die diese nicht vor dem Öffnen der Datei reduziert, sondern direkt nach dem Scannen, und es so möglicherweise ermöglicht, verschieden große Versionen eines einzigen Artefakts online zu stellen. Ob dies bereits von der Scannersoftware übernommen werden kann, wäre zu klären.

Ein weiterer Ansatz zur Verbesserung wäre eine Überarbeitung des Speichermanagements für die Geometriedaten dahingehend, je nach Zoomlevel nur dort die vollen Details zu laden, wo der Betrachter gerade hinsieht, und generell weniger Knoten anzuzeigen, wenn die Kamera weiter vom Objekt entfernt ist (»Level of Detail«). Denn neben der Speicherproblematik geraten selbst Grafikkarten der oberen Leistungsklasse bei derartigen Datenmengen an ihre Grenzen, was die Darstellung mit akzeptabler Bildrate angeht.

7.2 Kantenfindung und Texturen

Obgleich der Kantenfindungsalgorithmus trotz seiner Einfachheit auf sehr positive Resonanz gestoßen ist, wäre es wünschenswert, noch weitere Ansätze zur Kantenfindung zu verfolgen, insbesondere solche, die wirklich zusammenhängende Formen auf dem Artefakt finden und diese in Form einer eigenen Grafik darstellen können. Diese wäre unabhängig von der Auflösung der geladenen Datei, würde aber erfordern, dass die Anzeige von UV-Texturen auf dem Objekt ermöglicht wird. Neben der Anzeige automatisch generierter Linien wäre damit auch das Zeichnen auf dem Artefakt möglich. Allerdings war die Vorgabe für die aktuelle Entwicklung, dass automatisch und objektiv gefundene Formen den manuell gezeichneten vorzuziehen seien.

7.3 Datenbankintegration

Da alle vom Benutzer erzeugten Anmerkungen in XML gespeichert werden, liegt es nahe, diese Informationen in der Web-Datenbank zu nutzen, indem sie ausgelesen und beim Datensatz auf der entsprechenden Webseite angezeigt werden. Außerdem wäre es denkbar, standardisierte Stichwörter (Tags) zu vergeben, so dass Artefakte anhand dieser Merkmale ihrer selbst (»Unterkiefer«) oder Merkmalen der kommentierten Stellen (»Schädelbruch«, »Unfall«, etc.) gesucht werden können. Dazu müsste das Format der XML-Dateien allerdings nochmals etwas geändert werden. Eine Abwärts- und Aufwärtskompatibilität solcher Softwareversionen zu XML-Daten-Versionen wäre jedoch kein Problem.

7.4 Ausbessern von Löchern

Da Artefakte häufig Beschädigungen aufweisen, wäre es möglicherweise vorteilhaft, eine Funktion zu schaffen, mit der Löcher in der Geometrie geschlossen werden können. Dazu könnten die Randknoten automatisch gefunden oder manuell markiert werden und dazwischen automatisch zusätzliche Dreiecke generiert werden.

Ebenfalls denkbar wäre die Implementierung eines Sculpt-Werkzeuges, wie es in gängigen 3D-Modellierungsprogrammen existiert, bei dem mit dem Mauszeiger kleine virtuelle Knetgummikugeln an schon vorhandene Flächen angeheftet werden können

und somit Löcher ähnlich dem Verfüllen mit Lehm oder ähnlichen Werkstoffen geschlossen werden können. Dabei besteht generell mehr Möglichkeit zur Ausgestaltung von Details, gerade bei großen Löchern.

7.5 Kombinieren von Artefakten

Da beispielsweise Ober- und Unterkiefer nicht gleichzeitig gescannt werden sollten, weil der Scanner nur von außen scannen kann und somit ein Großteil des Mundinnenraumes vermutlich bezüglich des Lasers verschattet würde, wäre es vorteilhaft, zusammengehörige Einzelscans wieder zu einer Datei zusammenfügen zu können oder aber zwei Dateien gleichzeitig geöffnet haben zu können.

7.6 Sprachen & Dokumentation

Die Software wurde in englischer Sprache verfasst, um die Benutzung und Weiterentwicklung durch Menschen rund um die Welt zu ermöglichen. Dennoch wäre es angesichts der Unterstützung von Qt für verschiedene Sprachen wünschenswert, das Umschalten der Sprache zu ermöglichen. Eine entsprechende Übersetzungsdatei kann auch von Nichtprogrammierern erstellt werden.

Das Handbuch liegt außerdem – entsprechend der Sprache dieser Arbeit – nur in Deutsch vor. Dieses könnte übersetzt werden. Auch die Idee, ein Tutorialvideo zu machen, wurde positiv aufgenommen. Auch an dieser Stelle könnten verschiedene Sprachen zur Verfügung gestellt werden.

Abschließend kann gesagt werden, dass die Zielsetzung erreicht wurde. In der begrenzten Zeit wurde eine Software geschaffen, die die gestellten Anforderungen weitestgehend abdeckt und zudem wie gefordert quelloffen und erweiterbar ist, so dass die verbliebenen Anforderungen (ggf. nach einer Überarbeitung mit den neu gewonnenen Erkenntnissen) oder auch mögliche vollkommen neue Anforderungen in Zukunft ebenfalls implementiert werden können. Zu diesem Zweck wird sie unter der BSD-2-Clause-Lizenz veröffentlicht.

Literaturverzeichnis

L. Adkins, R. Adkins (1989)

Archaeological Illustration. Cambridge University Press

P. Bayle, L. Bondioli, R. Macchiarelli, A. Mazurier, L. Puymerail, V. Volpato, C. Zanolli (2011)

Three-dimensional imaging and quantitative characterisation of human fossil remains – examples from the NESPOS database. Pleistozäne Datenbanken: Datenerwerb, Speicherung, Austausch, S. 29-46

M. K. H. Eggert (2001)

Prähistorische Archäologie / Konzepte und Methoden. Francke-Verlag

P. Fastermann (2012)

3D-Druck / Rapid Prototyping. Springer-Verlag

D. Fichtmüller, H. Kießling (2008)

Hightech in der Archäologie – Laserscanning archäologischer Mauerbefunde und Fundstücke. Schriften des Bundesverbands freiberuflicher Kulturwissenschaftler, Band 2: Dokumentation und Innovation bei der Erfassung von Kultur gütern.

A. Gilboa, A. Tal, I. Shimshoni, M. Kolomenkin (2013)

Computer-based, automatic recording and illustration of complex archaeological artifacts. Journal of Archaeological Science 40, S. 1329-1339

F. Gröning, J. F. Kegler, G.-C. Weniger (2007)

Die digitale Welt der Neandertaler – NESPOS, ein Online-Archiv für die Neandertalerforschung. Archäologisches Korrespondenzblatt, Jahrgang 37, Heft 3, S. 321 - 333

M.-L. Inizan, M. Reduron-Ballinger, H. Roche, J. Tixier (1999)

Technology and Terminology of Knapped Stone. Préhistoire de la Pierre Taillée, Tome 5, Nanterre

E. Lindinger, C. Hörr (2006/2007)

Hightech meets handmade – Ein neu entwickeltes 3D-Scanverfahren für archäologische Objekte. Arbeits- und Forschungsberichte zur sächsischen Bodendenkmalpflege, Band 48/49, S. 9 - 18

A. Pastoors, G.-C. Weniger (2011)

Graphical documentation of lithic artefacts: Traditional hand craft versus 3-D mechanical recording. Wissenschaftliche Schriften des Neanderthal Museums 4, S. 9 - 17

R. Macchiarelli, G.-C. Weniger (2011)

Im Vorwort zu

Pleistozäne Datenbanken: Datenerwerb, Speicherung, Austausch. Wissenschaftliche Schriften des Neanderthal Museums 4

J. Nielsen (1992)

Finding usability problems through heuristic evaluation. CHI '92 Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, S. 373 - 380

B. T. Phong (1975)

Illumination for Computer Generated Pictures. Communications of the ACM, S. 311 – 317

M. Schaich (2008)

3D-Scanning-Technologien in der Bau- und Kunstdenkmalpflege und der archäologischen Feld- und Objektdokumentation. Schriften des Bundesverbands freiberuflicher Kulturwissen- schaftler, Band 2: Dokumentation und Innovation bei der Erfassung von Kulturgütern.

P. Semal, S. Kirchner, R. Macchiarelli, P. Mayer, G.-C. Weniger (2004)

TNT: The Neanderthal Tools. The 5th International Symposium on Virtual Reality, Archaeology and Cultural Heritage VAST, S. 1–2

Webressourcen

gnu.org - Abruf vom 2.3.2014

http://www.gnu.org/licenses/lgpl-3.0

opensource.org - Abruf vom 2.3.2014

http://opensource.org/licenses/BSD-2-Clause

praehistorische-archaeologie.de - Abruf vom 2.3.2014

http://www.praehistorische-archaeologie.de/wissen/grundlagen/evolution-des-menschen/

sourceforge.net - Abruf vom 2.3.2014

http://sourceforge.net/apps/mediawiki/meshlab/index.php?title=Compiling

de.wikipedia.org - Abruf vom 2.3.2014

http://de.wikipedia.org/wiki/MoSCoW-Priorisierung

http://de.wikipedia.org/wiki/Archäologie

en.wikipedia.org - Abruf vom 2.3.2014

http://en.wikipedia.org/wiki/PLY_(file_format)

Anhang A – Aufgabenliste für den Test

Eine Datei öffnen

- Menüpunkt »File« finden und »Open« anklicken
- Eine Datei geeigneten Formats auf der Festplatte finden und öffnen

Ein bereits ausgerichtetes Artefakt frei erkunden

- Rechte Maustaste drücken und Maus ziehen
- Mausrad drehen

Die Kamera präzise ausrichten

- Kamerarotation zurücksetzen
- Winkel einstellen
- Steuerkreuz betätigen

Ein verdrehtes Artefakt ausrichten

- Kamera nach oben (unten) drehen
- Objekt ausrichten
- Kamera zurückdrehen/zurücksetzen
- Kamera nach rechts (links) drehen
- Objekt ausrichten
- Kamera zurückdrehen/zurücksetzen
- Objekt ausrichten
- Ggf. Weitere Ausrichtungen aus verschiedenen Sichten

Den Hintergrund einstellen

- Menüpunkt »Options« finden und »Background Color« wählen
- Die untere Farbe auswählen und bestätigen
- Die obere Farbe auswählen und bestätigen

Die Artefaktfarbe einstellen

- Menüpunkt »Options« finden und »Artefact Color« wählen
- Eine Farbe auswählen und bestätigen

Vertexcolors nutzen

- Das Häkchen bei »use vertex colors« machen

Fertig ausgerichtetes Artefakt beleuchten

- Die Beleuchtung aktivieren
- Mindestens zwei Lampen anwählen und möglichst viele Parameter einstellen
- Beleuchtung verstecken

Eine Anmerkung zu einem Punkt schreiben

- Anmerkungen aktivieren
- Das Artefakt mit der linken Maustaste anklicken
- Den Knopf »Add Annotation« klicken
- Einen Text eingeben und bestätigen

Eine Strecke messen und dabei manche Punkte nachträglich korrigieren und löschen (ohne Anmerkung)

- Drei Punkte auf einer Strecke durch Linksklicks markieren
- Einen davon linksklicken und ziehen
- Einen löschen und wieder neu setzen
- Den Knopf »Measure Distance« anklicken
- Abbrechen

Einen Winkel messen und dabei eine Anmerkung erzeugen

- Drei Punkte setzen
- »Measure Angle« anklicken
- Den Text editieren und eine Farbe auswählen
- Bestätigen

Eine Fläche messen und dabei eine Anmerkung erzeugen

- Fünf Punkte in beliebiger Reihenfolge setzen
- Einzelne Punkte ziehen und klicken
- Einen Punkt löschen
- Einen neuen Punkt zwischen anderen hinzufügen
- »Measure Area« anklicken
- Den Text editieren und eine Farbe auswählen
- Bestätigen

Anmerkungen sortieren lassen

- Das Objekt drehen, so dass sich die Linien von Anmerkungen überschneiden
- Den Knopf »Rearrange« klicken

Eine Anmerkung editieren

- Eine vorhandene Anmerkung anklicken
- Den Knopf »Edit« drücken
- Den Text editieren
- Eine neue Farbe wählen
- Bestätigen

Eine Anmerkung löschen

- Eine vorhandene Anmerkung anklicken
- Den Knopf »Delete« drücken

Eine adäquate Einstellung für »dark edges« finden

- In der Dropdownliste »dark edges« auswählen
- Strength einstellen
- Smooth einstellen

Eine adäquate Einstellung für »dark valleys« finden

- In der Dropdownliste »dark valley« auswählen
- Strength einstellen
- Smooth einstellen

Einen Screenshot machen

- Menüpunkt »Tools« finden und »Screenshot« wählen
- Einen Dateityp und einen Speicherort bestimmen

Mit der Kamera seitwärts am Artefakt entlang fahren

- Shifttaste drücken
- rechte Maustaste drücken und Maus ziehen

Anhang B - BSD-2-Clause-Lizenz[38]

[38] opensource.org 2014

Erklärung

Hiermit versichere ich an Eides Statt, dass ich die vorliegende Arbeit selbstständig und ohne die Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten und nicht veröffentlichten Schriften entnommen wurden, sind als solche kenntlich gemacht. Die Arbeit ist in gleicher oder ähnlicher Form oder auszugsweise im Rahmen einer anderen Prüfung noch nicht vorgelegt worden.

Ort, Datum

Lebenslauf

Name: Dominic Michael Laurentius (geb. Wienand)

Geboren am: 18.2.1979 im Ev. Krankenhaus Köln, Weyertal

Eltern: Hans-Jörg und Gerda Wienand

- 1982 Kindergarten St. Alban in der Gilbachstraße
- 1985 Einschulung in die Katholische Grundschule Palmstraße
- 1989 Jahren Wechsel an das Gymnasium Kreuzgasse
- 1996 (Januar) Schülerbetriebspraktikum beim WDR (Abteilungen Rechnungswesen und Öffentlichkeitsarbeit)
- 1997 Führerschein Klassen B/C1/BE/C1E/CE/M/L
- 1998 Abitur nach 13 Jahren mit Gesamtnote 2,3
- 1998 – 1999 Zivildienst im Ev. Krankenhaus Köln-Kalk (Hauswirtschaft)
- 1999 Aufnahme des Studiums der Wirtschaftsinformatik an der WISO-Fakultät der Universität zu Köln
- 2000 – 2006 Studentischer Mitarbeiter im Bereich EDV beim Marktforschungsunternehmen IMW-Köln GmbH & Co KG
- 2004 Führerschein Klassen A/A1
- 2006 Heirat mit Jana Laurentius (Namensänderung erst 2009)
- 2006 – 2007 Selbstständige Tätigkeit als Webdesigner
- 2008 Aufnahme der Tätigkeit als Privatier im Vermietungsgeschäft für Wohn- und Gewerberäumlichkeiten im Raum Köln/Bonn
- 2010 Umzug von Köln nach Alfter-Witterschlick bei Bonn
- 2014 Start der Ausbildung zum Privatpiloten (PPL-A)