ElKowmGIS
Ein neues System zur Dokumentation archäologischer Fundstellen

Inauguraldissertation

zur Erlangung der Würde eines Doktors der Philosophie

vorgelegt der

Philosophisch-Naturwissenschaftlichen Fakultät
der Universität Basel

von

Daniel Schuhmann

aus

Lörrach, Deutschland

Basel 2015

Genehmigt von der Philosophisch-Naturwissenschaftlichen Fakultät
auf Antrag von

Prof. Dr.ès sc. Jean-Marie Le Tensorer

Prof. N. Connard

Basel, 12.11.2013

Prof. Dr. Jörg Schibler

Dekan

Namensnennung-Keine kommerzielle Nutzung-Keine Bearbeitung 3.0 Schweiz
(CC BY-NC-ND 3.0 CH)

Sie dürfen: **Teilen** — den Inhalt kopieren, verbreiten und zugänglich machen

Unter den folgenden Bedingungen:

Namensnennung — Sie müssen den Namen des Autors/Rechteinhabers in der von ihm festgelegten Weise nennen.

Keine kommerzielle Nutzung — Sie dürfen diesen Inhalt nicht für kommerzielle Zwecke nutzen.

Keine Bearbeitung erlaubt — Sie dürfen diesen Inhalt nicht bearbeiten, abwandeln oder in anderer Weise verändern.

Wobei gilt:

- **Verzichtserklärung —** Jede der vorgenannten Bedingungen kann **aufgehoben** werden, sofern Sie die ausdrückliche Einwilligung des Rechteinhabers dazu erhalten.
- **Public Domain (gemeinfreie oder nicht-schützbare Inhalte) —** Soweit das Werk, der Inhalt oder irgendein Teil davon zur Public Domain der jeweiligen Rechtsordnung gehört, wird dieser Status von der Lizenz in keiner Weise berührt.
- **Sonstige Rechte —** Die Lizenz hat keinerlei Einfluss auf die folgenden Rechte:
    - Die Rechte, die jedermann wegen der Schranken des Urheberrechts oder aufgrund gesetzlicher Erlaubnisse zustehen (in einigen Ländern als grundsätzliche Doktrin des **fair use** bekannt);
    - Die **Persönlichkeitsrechte** des Urhebers;
    - Rechte anderer Personen, entweder am Lizenzgegenstand selber oder bezüglich seiner Verwendung, zum Beispiel für **Werbung** oder Privatsphärenschutz.
- **Hinweis —** Bei jeder Nutzung oder Verbreitung müssen Sie anderen alle Lizenzbedingungen mitteilen, die für diesen Inhalt gelten. Am einfachsten ist es, an entsprechender Stelle einen Link auf diese Seite einzubinden.

Quelle: http://creativecommons.org/licenses/by-nc-nd/3.0/ch/ Datum: 12.11.2013

„Da jede Ausgrabung die Fundstelle unwiederbringlich zerstört, muss man jede nur mögliche Information vor der Beendigung der Arbeit festhalten.“

Ian Tattersall 1999

Danksagung

Während der letzten Jahre konnte ich die Unterstützung vieler Leute geniessen, denen ich hier

danken möchte.

Zuallererst möchte ich Herrn Prof. Dr.ès sc. Jean-Marie Le Tensorer ganz herzlich danken. Er ermöglichte es mir, dieses spannende Thema als Rahmen meiner Doktorarbeit zu nutzen. Ausserdem unterstützte er mich in allen inhaltlichen sowie organisatorischen und administrativen Fragen betreffend dieser Arbeit.

Als Nächstes möchte ich Herrn Prof. N. Connard der Universität Tübingen herzlichst dafür danken, dass er sich bereit erklärte, die Zweitkorrektur der Arbeit zu übernehmen.

Ohne die Finanzierung und Unterstützung der Freiwilligen Akademischen Gesellschaft und des Schweizerischen Nationalfonds wäre diese Arbeit nicht möglich gewesen. Sie unterstützen beide das Projekt "Le Paléolithique d'El Kowm (Syrie)", in dessen Rahmen diese Arbeit entstand. Ohne dieses Projekt hätten die Grundlagen der Doktorarbeit gefehlt. Des Weiteren danke ich der Syrischen Antikenbehörde (General Directorate of Syrian Antiquities) und dem Museum in Palmyra für Ihre Unterstützung während der Grabungskampagnen.

Für spannende und lehrreiche Stunden in der syrischen Wüste danke ich auch sämtlichen Teilnehmern der ElKowm-Expeditionen. Ein besonderer Dank geht an Dorota Wojtczak, die immer ein offenes Ohr für mich hatte und mir mit Rat und Tat zur Seite stand. Weiter danken möchte ich folgenden Personen für Ihre Unterstützung und die anregenden Diskussionen über meine Arbeit: Hélène Le Tensorer, Vera von Falkenstein, Peter Schmid, Ahmed Taha, Mahmud Taha, Waleed Asa'ad, Fabio Wegmüller, Pietro Martini, Hani El Sued, Manar Kerdy, Mustafa Al Najjar, Christine Pümpin, Reto Jagher, Kristin Ismail-Meyer und Daniel Richter. Zusätzlicher Dank gilt den Einwohnern von El Kowm für Ihre Gastfreundschaft und die Hilfe auf der Ausgrabung.
Ein ganz besonderer Dank geht an Taha Taha, der mir etliche Male Einblicke in die Gedanken eines syrischen Staatsbürgers gewährt hat. Für die zahlreichen Diskussionen über das Leben und die Politik in Syrien und dem nahen Osten bin ich ihm äusserst dankbar.

In den letzten Jahren hatte ich mehrfach die Möglichkeit auf die Hilfe verschiedener Forscher zurückzugreifen. Speziell erwähnt seien hier Yuri Demidenko, Renate Ebersbach, Urs Brombach, Hans Sütterlin, Jürg Rychener, Bruno Magri, Jörg Schibler, Brigitte Röder, David Brönnimann und Angela Schlumbaum. Des Weiteren danke ich den MitarbeiterInnen und Studierenden der Integrativen Prähistorischen und Naturwissenschaftlichen sowie der Ur- und frühgeschichtlichen und provinzialrömischen Archäologie der Universität Basel.
Für die Unterstützung und Hilfe durch Prof. Dr. Peter-Andrew Schwarz und Prof. Dr. Martin Guggisberg sei hier ein herzlicher Dank untergebracht. Sie boten mir die Möglichkeit, mich während meiner Dissertation

zusätzlich in den beiden Bereichen Projektmanagement und Geoprospektion weiterzubilden.

Ein Dank privater Natur sei an dieser Stelle an meine Mitbewohner Sandra Ammann, Claudia Klausner und René Fisch gerichtet. Sie haben mich in den letzten Wochen meiner Arbeit kaum gesehen und jederzeit vollstes Verständnis dafür aufgebracht.

Abschliessend möchte ich meinem Vater, Hugo Schuhmann, aus tiefstem Herzen danken. Er unterstützte mich während der ganzen Arbeit in vollem Umfang. Ich durfte auch auf seine Hilfe bei der Korrektur der Schlussfassung der Arbeit zählen. Ausserdem besass er für all meine Fragen und Probleme ein offenes Ohr.

Inhalt

Zusammenfassung

Die Gegend um El Kowm ist bekannt durch seine zahlreichen paläolithischen Fundstellen. Auf einem Gebiet von ca. 1'300 km² finden sich ungefähr 177 paläolithische Fundstellen, welche sich vom Alt- bis zum Mittelpaläolithikum erstrecken. Der Grund für die grosse Anzahl an Fundstellen dürfte wohl hauptsächlich an den geographischen Gegebenheiten der Region liegen.
Das Dorf von El Kowm liegt etwa auf halbem Weg der Strecke von Palmyra nach Raqqah, ca. 100 km nordöstlich von Palmyra, auf einem Plateau, welches durch den Gebirgszug der Palmyriden und dem Gebirgszug des Djebel Al Bishri begrenzt wird. Zahlreiche Wasserstellen, von denen viele noch heute als Brunnen genutzt werden, zogen sowohl tierische und menschliche Besucher an. Daher auch die verhältnismässig grosse Anzahl an Fundstellen.

Unter diesen Fundstellen findet sich auch Hummal, deren Topographie und Stratigraphie die Grundlagen für diese Arbeit liefern. Die ersten Prospektionen um Hummal herum wurden bereits 1966 durch G. und M. Buccellati durchgeführt. Dabei entdeckten sie, dass Hummal eine bis zu 20 Meter mächtige Stratigraphie aufweist. Erste Grabungen wurden 1980 durchgeführt und bis 1987 weitergeführt. Im Winter 1987 führte eine starke Erosion zur Einstellung der Arbeiten in Hummal. Erst 1997 wurden die Grabungen in Hummal durch die Universität Basel in Zusammenarbeit mit S. Muhesen in kleinerem Massstab wieder aufgenommen. Seit 1999 finden nun jährlich Ausgrabungen in der Fundstelle statt.

In dem Brunnenschacht, welcher heutzutage nicht mehr direkt benutzt wird, findet sich, wie bereits erwähnt, eine mächtige Stratigraphie von über 20 Metern. Diese Stratigraphie kann in verschiedene archäologische Schichtkomplexe unterteilt werden. Darin finden sich die Kulturen des Oldowayen, Tayacien, Yabroudien, Hummalien und Mousterien. Innerhalb dieser Schichtkomplexe lassen sich verschiedene Niveaus erkennen, so dass man insgesamt 19 Schichten zählen kann.

Während der Grabungskampagne 2005 wurde die Vermessung auf der Grabung komplett auf die digitale Vermessung mittels Tachymeter und Feldcomputer umgestellt. Hierfür wurden 2005 ein Tachymeter des Herstellers Leica und das Programm AutoCAD mit der Erweiterung TachyCAD, welche die Kommunikation zwischen Tachymeter und Computer ermöglicht, angeschafft. Als Feldcomputer dienten zwei ausgediente Notebooks der urgeschichtlichen Abteilung der Integrativen und Naturwissenschaftlichen Archäologie (IPNA) der Universität Basel.
Im Laufe der letzten Jahre stellte sich jedoch immer mehr heraus, dass die auf dem Markt erhältliche Software vor Allem für die Projektabwicklung sehr eingeschränkt ist. Das heisst, dass die Software es kaum erlaubt, eine individuelle, an das jeweilige Projekt angepasste Datenstruktur, wie sie in einer solchen Fundstelle benötigt wird, zu verwenden. Ausserdem hat sich gezeigt, dass die Möglichkeiten der Kontrolle der Messungen, zum Beispiel mit Hilfe eines Protokolls der Messungen, nicht gegeben sind. Im Rahmen der Diplomarbeit "Digitale Modellierungen und Schichtrekonstruktionen der paläolithischen Fundstelle Hummal, Syrien" des Autors, war es nötig, sich näher mit der gesamten Datenstruktur des Projektes zu beschäftigen. Dabei wurde festgestellt, dass es mit der aktuellen Version von AutoCAD nicht möglich ist, digitale Modellierungen der Oberflächen oder der Grabung selbst zu erstellen.

Während den Grabungskampagnen 2006 und 2007 mussten bereits während der Grabung verschiedenste, an für sich grundlegende, Funktionen selbst programmiert werden, um den Grabungsalltag, respektive die Messabläufe zu ermöglichen.

Aus diesen Gründen heraus entstand die Idee eines eigenen Programms zur Vermessung archäologischer Fundstellen, welches es erlaubt, das Programm an das System der Grabung anzupassen.

ElKowmGIS stellt nun eine Softwarelösung dar, die in verschiedenen Fundstellen mit verschiedensten Dokumentationssystemen zum Einsatz kommen kann. Hierfür wurde die verwendete Datenbank so angelegt, dass der Benutzer keinerlei Einschränkungen bei der Eingabe der Daten erfährt. Zusätzlich bietet ElKowmGIS die Möglichkeit, mehrere Softwareanwendungen, beispielsweise ein GIS-Programm und eine Tabellenansicht der Datenbankeinträge, parallel zu nutzen, ohne das Programmfenster wechseln zu müssen. Ein weiterer Vorteil von ElKowmGIS liegt darin, dass es auf jedem Betriebssystem lauffähig ist und somit auch auf Unix- oder Mac-Systemen funktioniert. Dies wird durch den Einsatz der Programmiersprache Java erreicht.

Die hier vorliegende Arbeit verfolgt im Grunde zwei Ziele:

Zum einen soll das Konzept und die Programmierung hinter dem Programm aufgezeigt und erläutert werden. Hierfür finden sich in den Kapiteln “2 Einleitung“, “4 Die Datenbank von ElKowmGIS“ und “7 Die Technik hinter den Modulen“ die Ziele und die Ideen der Arbeit sowie die informatische Umsetzung dieser. Diese Kapitel besitzen verständlicherweise einen informatischen Schwerpunkt, da sie der Erläuterung der Programmstruktur dienen. Das Kapitel “3 Informatische Grundlagen“ dient der Vorstellung der verwendeten Programmiersprache Java, ihren Erweiterungen und dem Datenbankserver PostgreSQL.

Das zweite Ziel ist die Erstellung eines Handbuches für die Benutzung von ElKowmGIS. Hierzu zeigt das Kapitel “5 Installation und Deinstallation“ wie ElKowmGIS auf einem Rechner installiert wird. Da es für die Deinstallation von ElKowmGIS einige Punkte zu berücksichtigen gilt, wird im selben Kapitel ein eigener Abschnitt für die Deinstallation aufgeführt. Kapitel 6 “Das Programm“ ist als das eigentliche Handbuch zu verstehen. Hier wird erläutert, wie man ElKowmGIS verwendet, ohne dabei auf die informatischen Grundlagen detailliert einzugehen.

Als Abschluss der Arbeit wird in Kapitel 8 “Teststudie Hummal“ anhand der Grabungen in Hummal die Software getestet und mit kommerziellen Softwarelösungen verglichen. Gleichzeitig dient dieses Kapitel als Beispiel für die Verwendung von ElKowmGIS.

1 Darstellungskonventionen

Bei der hier vorliegenden Arbeit handelt es sich um eine Arbeit mit Elementen der Fachrichtung Informatik. Es werden im Text einige spezielle Darstellungskonventionen getroffen. Die speziellen Eigenheiten dieser Arbeit sollen im Folgenden kurz vorgestellt werden und gleichzeitig Anleitung gesehen werden, wie die einzelnen Formatierungen zu lesen sind.

Um die Lesbarkeit des Textes zu erhöhen, wurde darauf verzichtet, die weibliche und die männliche Form der Wörter Benutzer, Anwender, etc. zu verwenden. Es wurde vielmehr versucht, eine geschlechtsneutrale Formulierung einzusetzen. In Fällen, in denen dies jedoch nicht möglich war, sind selbstverständlich beide Geschlechter angesprochen.

Die Literaturzitate werden nach den Regeln der Zeitschrift Current Anthropology (The University of Chicago Press 2013) aufgeführt. Diese Zitierweise wird für alle Zitate aus Monografien, Zeitschriften, Grauer Literatur, Internetdokumenten und persönlichen Mitteilungen verwendet. Wie in naturwissenschaftlichen Arbeiten üblich, wird auf Fussnoten im Text gänzlich verzichtet.

Da der Text mehrfache Verweise auf die Dokumentation von Java und ElKowmGIS enthält, werden diese Verweise auf eine andere Art und Weise formatiert. Die Verweise auf die Java-Dokumentation erscheinen in hierbei in schwarzer, *kursiver Schrift*. Verweist der Text auf die Dokumentation von ElKowmGIS, so wird der Verweis in *blauer, kursiver Schrift* aufgeführt.
Begriffe, die erstmals im Text erscheinen und im Glossar aufgeführt sind, werden in **fetter Schreibweise** dargestellt.

Hinweis:

Im Text finden sich Hinweisfelder, die für das Verständnis der Programmierung oder die Benutzung dienlich sind. Sie sind in einem grauen Kasten mit abgerundeten Ecken untergebracht.

Warnung:

Neben den Hinweisen finden sich in der Arbeit Warnungen, die mit einem roten Ausrufezeichen in einem gelben Kasten versehen sind. Sie sollen dazu dienen, den Benutzer auf Fehler im Umgang mit der Software oder den Daten hinzuweisen.

1

2 Einleitung

2.1 Grundlagen der Arbeit

Das hier vorliegende Programm **ElKowmGIS** wurde im Rahmen einer Doktorarbeit innerhalb des Projektes “Le Paléolithique d’El Kowm (Syrie)“ (Le Tensorer et al. 2011) der Integrativen und Prähistorischen Archäologie (IPNA) der Universität Basel entwickelt.

Auslöser für die Arbeit war die Vermessungslösung, welche im Jahr 2005 eingeführt wurde. Während dieser Kampagne in Syrien wurde die Dokumentation von einer manuellen Vermessung auf die digitale Vermessung mittels Tachymeter und Computer umgestellt. Hierbei kamen kommerzielle Produkte zum Einsatz (**AutoCAD**, **TachyCAD** und **Photoplan**), welche es ermöglichen, Funde und Befunde direkt auf der Grabung digital zu erfassen (Brönnimann and Schuhmann 2006). Bereits zu Beginn der Kampagne zeigte sich, dass das gewählte System nicht an die Bedürfnisse der Grabungen in El Kowm anpassbar war. Dies äusserte sich besonders in der integrierten Befundverwaltung von TachyCAD und der Aufnahme der Objektinformationen während der Vermessung. So liess sich die Befundverwaltung nicht nach einzelnen Kampagnen trennen und die Objektinformationen mussten der Übersichtlichkeit halber auf die Angabe der Objektart und der Laufnummer reduziert werden. Weitere Informationen mussten separat erfasst und nach der Messung ausserhalb von AutoCAD mit den Messungen zusammengeführt werden. Durch dieses Vorgehen war es jedoch nötig mit mehreren Programmen gleichzeitig zu arbeiten. Auf diese Problematik wird im Abschnitt “2.2 Ziele dieser Arbeit“ noch detailliert eingegangen.

Während der Kampagnen 2006, 2007 und 2008 wurde weiterhin mit dieser Lösung gearbeitet. Dabei wurde das Dokumentationssystem der Grabungen in El Kowm weiter angepasst und die Softwarelösung auf die Bedürfnisse der Archäologie abgestimmt. Allerdings war dies nur in begrenztem Masse möglich.

Durch diese Defizite entstand die Idee, eine eigene Lösung im Rahmen dieser Doktorarbeit zu erarbeiten und zu entwickeln. Bei der Erarbeitung der Ziele der Doktorarbeit wurde auf verschiedene Punkte Wert gelegt. Die zwei wichtigsten Punkte hierbei lassen sich wie folgt zusammenfassen: Zum einen sollte die Arbeit ein System liefern, welches mehrere Programme, hier seien nur die Stichworte **GIS**, **CAD**, Datenbank und Fotoverwaltung genannt, in einer Lösung vereint. Zum anderen sollte das Programm die Verwaltung der Messungen und Dokumente selbstständig übernehmen.
Zusätzlich soll die Lösung nicht nur für die Grabungen in El Kowm, sondern generell für archäologische Ausgrabungen, ungeachtet ihrer zeitlichen und geografischen Positionierung, anwendbar sein. Ziel hierbei ist es, das sich die Software an die Grabung anpassen lässt und nicht die Grabung resp. die Dokumentation der Grabung sich an die Software anpassen muss. Des Weiteren soll die Softwarelösung plattformunabhängig sein, das heisst, dass es keine Rolle spielt welches, Betriebssystem verwendet wird. Gerade in der heutigen Zeit, in der viele verschiedene Betriebssysteme, z.B. Windows, Unix, Mac-OS, im Einsatz sind, ist dies ein wichtiger Aspekt.

Eine weitere Idee bestand darin, dass das Programm aus einzelnen Modulen zusammengesetzt ist. Dadurch lässt sich die Software jederzeit erweitern, indem ein neues Modul hinzugefügt wird. Mögliche Anwendungen dieser Vorgehensweise wären zum Beispiel die Entwicklung eines Moduls für die Auswertung von gesamten Grabungen oder Teilbereichen, spezifische Module für die Einbindung der Ergebnisse naturwissenschaftlicher Analysen oder Module für weiterführende geostatistische Auswertungen. Schliesslich sollte das ganze System mit einer benutzerfreundlichen resp. selbsterklärenden Oberfläche versehen sein. Eine ausführliche Beschreibung der Ziele findet sich im nächsten Abschnitt.

2.2 Ziele dieser Arbeit

Grundlegendes Ziel dieser Arbeit ist es, ein Vermessungssystem zu entwickeln, welches an die Bedürfnisse der Archäologie angepasst ist. Hierbei darf es nicht zu Einschränkungen durch das Messsystem, die zeitliche resp. räumliche Stellung des Projektes und der verwendeten Gerätschaften kommen. Um dies zu gewährleisten wurde bereits zu Beginn der Arbeit ein Funktionsumfang der Software definiert. In den folgenden Abschnitten soll auf die einzelnen Funktionen im Vergleich zu der bestehenden Lösung eingegangen werden.

2.2.1 Vereinigung mehrerer Anwendungen

Im bisherigen System sind für den Messablauf, die Kontrolle der Daten und die Ablage der gewonnenen Informationen mehrere verschiedene Anwendungen notwendig. Dabei werden Funde und Befunde zuerst mittels CAD eingemessen, wobei die grundlegenden Informationen Laufnummer und Objektgattung in der Eingabemaske eingegeben werden. Die Information über die Grabung, das Jahr, die Fläche und das Fundniveau werden im Dateinamen der CAD-Datei gespeichert. Zusätzlich dazu werden weiterführende Informationen, die Anzahl der Funde, die Dateinamen zugehöriger Fotos, das Aufnahmedatum und der Stand der Arbeiten in einem separaten Datenblatt aufgenommen. Für die Erstellung eines Fundjournals müssen die Daten aus dem CAD exportiert und anschliessend in eine Tabellenkalkulation importiert werden. Weitere Auswertungen, z.B. in einem GIS, erfordern den Import aller benötigten Daten in das entsprechende Programm. Stellenweise ist hierfür eine Umformatierung der Dateien von Nöten, um mit diesen zu arbeiten. Die Dateiverwaltung muss hierbei vom Benutzer übernommen werden. Sollen Fotos mit eingebunden werden, so müssen weitere Dateien und Dateiformate angelegt und eingebunden werden.

An diesem Punkt greift ElKowmGIS ein und vereint das CAD mit der Datenbank, dem GIS und der Dateiverwaltung. Im CAD-Modul werden die Daten und Informationen erfasst und direkt in eine Datenbank gespeichert. Dabei ist anzumerken, dass sämtliche Informationen, wie etwa die Grabungsnummer, der Stand der Bearbeitung etc., bereits als eigene Information während der Messung angegeben werden kann. Diese Werte werden als Eigenschaft jedem einzelnen Messpunkt zugewiesen. Dies bedeutet, dass die Information nicht versehentlich geändert werden kann.
Gleichzeitig kann sofort auf die Datenbank zugegriffen werden, um die Daten bei Bedarf nochmals zu überprüfen. Es ist nicht notwendig die Daten aus dem CAD zu exportieren. Durch die Wahl eines

Datenbankservers im Hintergrund der Vermessungslösung entfällt das Anlegen einer Dateistruktur und die Verwaltung mehrerer Dateien in verschiedenen Ordnern.

Die Funktionalitäten eines GIS sollten bereits in der CAD-Oberfläche vorhanden sein, so dass der Umgang mit nur einer grafischen Oberfläche für die Aufnahme und die Auswertung/Interpretation einer Grabung erlernt werden muss. Zusätzlich dazu sollte das GIS die Möglichkeit bieten die Daten dreidimensional darzustellen. Bisher verfügbare GIS-Lösungen bieten eine dreidimensionale Darstellung der Informationen nur eingeschränkt über den Einsatz weiterer Programme oder Programmteile, die nicht direkt in das Programm selbst integriert sind.
Die Lösung für eine Bildverwaltung sollte ebenfalls in dem System integriert werden, damit der zeitliche Aufwand der Beschriftung und Verschlagwortung der Bilder minimiert werden kann. Durch die Integration der Bildverwaltung in die Software lässt sich ein Thesaurus anlegen, welcher direkt mit der Datenbank verknüpft ist. Dadurch lässt sich das Auftreten von Tippfehlern reduzieren. Gleichzeitig lässt sich hier eine Vereinfachung einer notwendigen Dateistruktur erreichen.

2.2.2 “Künstliche Intelligenz“

Der Begriff “künstliche Intelligenz“ ist in diesem Zusammenhang bewusst in Anführungszeichen gesetzt, da es sich nicht um die künstliche Intelligenz (KI) handelt, wie sie in der Welt der Informatik verstanden wird. Der Begriff künstliche Intelligenz (aus dem Englischen: artificial intelligence), in der Informatik wurde in den 1950er Jahren in der USA geprägt und stellt eine eigene Forschungsrichtung in der Informatik dar (Fischer and Hofer 2008, S. 471). Sie untersucht dabei, wie Computer eingesetzt werden können, um komplexe Probleme zu lösen. Oft wird die KI in Verbindung mit Computerspielen gebracht, wo der Computergegner eine scheinbare (artificial) Intelligenz besitzt.

Die künstliche Intelligenz im Zusammenhang mit der vorliegenden Arbeit beschränkt sich auf kleinere Aufgaben, welche die Software von sich aus übernehmen soll, um Eingabefehler zu vermeiden. Dabei handelt es sich konkret um die automatische Fortführung der Laufnummer innerhalb einer Grabung, einer Fläche oder eines Fundkomplexes und die Überprüfung der Eindeutigkeit der Laufnummern. Die automatische Fortführung der Laufnummer ist in dem bisherigen System der Grabungen in El Kowm nicht vollständig gegeben. Die bisherige Lösung ermöglicht es nicht ohne weiteres, mehrere Punkte für ein Objekt aufzunehmen, da jeder Punkt eine eindeutige Nummer benötigt. Für das System in El Kowm hat man sich darauf geeinigt, ein Objekt, welches mit mehr als einem Punkt eingemessen wird, mit der gleichen Laufnummer und dem Anfügen eines Kleinbuchstabens aufzunehmen. Die Softwarelösung ändert bei der nächsten Messung jedoch nicht den Buchstaben, sondern fügt lediglich eine Ziffer an die Nummer an. So kann es geschehen, dass ein Fund, welcher durch zwei Punkte eingemessen wurde, die Punktnummern HU10-E11165a und HU10-E11165a1, statt HU10-E11165b, erhält. In der weiteren Handhabung kann dies zu Fehlern führen. Aus diesem Grund soll das System die Verwaltung der Laufnummern übernehmen und das Anfügen eines weiteren Zeichens selbst vornehmen.
Die Kontrolle der Eindeutigkeit einer Laufnummer ist im bisherigen System hingegen nicht vorhanden. Es ist für den Anwender möglich, mehreren Objekten die gleiche Laufnummer zuzuweisen. Dies kann durch

Systemfehler, z.B. Absturz des CADs, Stromausfall während der Messung, auftreten. Da sich dieses Problem während der Messung nicht bemerkbar macht, fällt dieser Fehler meist erst bei der Aufarbeitung der Funde auf. Dadurch und durch die systembedingte Aufteilung der Dateien muss die Laufnummer in einem zweiten, zeitaufwendigen Schritt nachträglich an mehreren Stellen geändert werden. Da die Kontrolle der Nummern nur über das Fundjournal vorgenommen werden kann, muss dieser Schritt manuell ausgeführt werden. Die hier vorliegende Arbeit soll dem entgegenwirken und bereits während der Messung den Benutzer darauf hinweisen, dass die eingegebene Nummer bereits vorhanden ist. Ferner soll die Software mehrere Vorschläge anbieten, welche Möglichkeiten nun bestehen. Dabei handelt es sich um folgende Möglichkeiten:

- Ändern der Koordinaten der bestehenden Nummer
- Ändern aller Informationen der bestehenden Nummer
- Anhängen eines weiteren Punktes zur bestehenden Nummer
- Vergabe der nächsten Laufnummer.

Zusätzlich zu den beiden bereits aufgeführten Punkten soll die Software die Dokumenten- und Bildverwaltung selbstständig übernehmen. Auf diesen Punkt wird im nächsten Abschnitt eingegangen.

2.2.3 Dokumentenstruktur und -verwaltung

Wie sich in der Beschreibung des Arbeitsablaufes auf der Grabung aus Abschnitt 2.2.1 bereits erahnen lässt, werden durch das eingesetzte System verhältnismässig viele Dateien produziert. Momentan beläuft sich die Dateianzahl für die Grabung in Hummal (El Kowm, Syrien) auf 25‘000 Dateien. All diese Dateien sind für die Dokumentation und Verwaltung der Fundstelle wichtig. Durch die Menge der Dateien war es zwingend notwendig, eine Ordnerstruktur (Abb. 1) (Brönnimann and Schuhmann 2006), (Schuhmann 2010) aufzubauen, in welcher die einzelnen Dateien thematisch abgelegt werden können. Hierbei stellte sich jedoch heraus, dass je grösser die Anzahl der Dateien wird, desto weniger übersichtlich wird die Dokumentation.

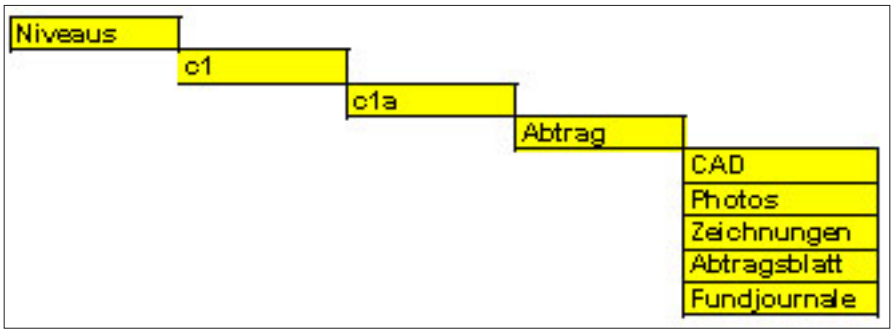


Abb. 1: Ausschnitt aus der Ordnerstruktur des Grabungsprojektes El Kowm

Deswegen soll das hier vorgestellte System die Verwaltung der Informationen und Dokumente selbstständig bewerkstelligen. Konkret heisst das, dass pro Grabung lediglich ein Ordner ausserhalb der Datenbank angelegt wird. In diesem Ordner werden dann vier weitere Unterordner angelegt, in welchen die

Dokumente gespeichert werden. Die Erstellung und Benennung der Ordner und der notwendigen Dateien sollen von der Software übernommen werden. Der erste Ansatz die Dateien mit im Datenbankserver abzuspeichern wurde bereits nach kurzer Zeit wieder verworfen. Der Grund hierfür lag in der mangelnden Geschwindigkeit mit der die Dokumente wieder abgerufen werden können. Die Tatsache, dass die Welt der Informatik sich ebenfalls von diesem Vorgehen abwendet (M. Guggisberg, mündliche Mitteilung, 27. Juni 2012), bestätigte diese Entscheidung.

Durch dieses Vorgehen ist es möglich, Dateien und Dokumente direkt mit den eingemessenen Informationen zu verknüpfen und mögliche Fehlerquellen auszuschliessen. Wie im Bereich der Bildverwaltung lassen sich durch die Anbindung an die Datenbank Thesauri automatisiert erstellen, welche eine Verschlagwortung und Verknüpfung ohne grossen Aufwand ermöglichen. Dennoch liefert dieser Ansatz eine Möglichkeit auf die Dokumente zuzugreifen, falls der Datenbankserver einmal nicht verfügbar sein sollte. Der Nachteil dieser Lösung liegt allerdings in der Tatsache, dass zusätzlich zum Datenbankserver eine Ablagemöglichkeit für die Dateien zur Verfügung stehen muss. Hierbei kann es sich um einen Dateiserver, eine externe Festplatte oder eine Netzwerkfestplatte handeln. Durch die vielfältigen Möglichkeiten bleibt eine Flexibilität beim Einsatz der Software gewährleistet.

2.2.4 Anpassung an die Grabung

Mit diesem Punkt soll das Ziel verfolgt werden, das neue System so flexibel wie nur möglich zu halten, um es für möglichst viele Grabungen verwendbar zu gestalten. Hierbei handelt es sich um die Eingabe der Grabungs- und Laufnummern, Flächenbezeichnungen, Darstellungsoptionen und Informationseingaben. Im bisherigen System gib es lediglich ein Feld für die Laufnummer und eines für die Informationen. So muss jegliche weiterführende Information in dieses Feld geschrieben werden. Erst in der Nachbearbeitung in einer Tabellenkalkulation oder Datenbank kann die Zusatzinformation getrennt abgespeichert werden. Als Beispiel sei hier ein verbrannter Silex mit der Nummer HU07-E9065 aus dem Niveau 10c angeführt. Der Silex besitzt die Nummer “E9065“ und die Information “sx -b“ (sx: Silex; b: verbrannt). Die Grabungsnummer (HU07) und das Niveau (10c) finden sich lediglich im Dateinamen “HU07_c10c_FL1_ab6.dwg“. Diese Informationen müssen nach der Messung wieder aufgetrennt werden, um eine Tabellenansicht wie in Abbildung 2 dargestellt zu erhalten.

Nr	qm	nø	Fundgattung	Status	X	Y	Z	Schicht	Kampagne
A32-1	A32	1	os	-	100.25	32.88	-6.52	5b1	2002
A32-2	A32	2	os	-	100.42	32.86	-6.53	5b1	2002
A32-3	A32	3	os	-	100.24	33	-6.5	5b1	2002
A32-4	A32	4	sx	-	100.26	32.72	-6.54	5b1	2002
A32-5	A32	5	sx	-	100.35	32.87	-6.53	5b1	2002
A32-6	A32	6	sx	-	100.63	32.87	-6.57	5b1	2002
A32-7	A32	7	sx	-	100.41	32.76	-6.55	5b1	2002
A32-8	A32	8	sx	-	100.17	32.65	-6.62	5b2	2002
A32-9	A32	9	sx	-	100.72	32.7	-6.71	5b2	2002
A32-10	A32	10	sx	-	100.48	32.62	-6.7	5b2	2002
A32-11	A32	11	sx	-	100.18	32.59	-6.62	5b2	2002

Abb. 2: Auszug aus dem Fundjournal der Grabung Hummal

Da dieses Vorgehen fehleranfällig ist, soll die neue Lösung die Informationen bereits bei der Messung getrennt abspeichern, um zu verhindern, dass die Informationen nachträglich einer Überarbeitung bedürfen. Hierfür soll die Datenbank eine eigene Tabelle besitzen, in welcher der Kontext einer Messung abgelegt wird.
Ein weiterer Punkt ist die Form resp. die Darstellung der Laufnummer. Jede Grabung besitzt bereits ein System, wie und aus was sich die Laufnummer zusammensetzt. Beispielsweise setzt sich die Laufnummer der Grabungen in Syrien aus einer Sektorbezeichnung (N: Nord, E: Ost, S: Süd, W: West) und einer mehrstelligen, fortlaufenden Nummer zusammen (siehe Beispiel oben). Für die Ausgrabungen der Römerstadt Augusta Raurica zum Beispiel setzt sich die Laufnummer aus der Grabungsnummer, Fundkomplex und Laufnummer zusammen. Als letztes Beispiel sei die Grabung in Mutzig (Elsass, FR) genannt. Hier setzt sich die Nummer aus den Komponenten Grabungsnummer, Quadratmeter, Niveau, Abtrag, Objektart und der Laufnummer zusammen (Bsp.: 5510-F6-C.2-UP1-RLT34).
Wie die Beispiele zeigen, können sich die eindeutigen **Identifikatoren** für Objekte auf verschiedene Weisen zusammensetzen. ElKowmGIS soll daher die Möglichkeit besitzen, verschiedene Formen als eindeutige Zuweisungen zu akzeptieren. Hierbei ist wichtig, dass der Benutzer die Zusammensetzung der Zuweisung frei definieren kann. Die Software soll keinerlei Einschränkungen besitzen.

2.2.5 Plattformunabhängigkeit

Dass ein Programm auf möglichst vielen Betriebssystemen (Windows, Unix, MAC, Android, etc.) funktionsfähig ist, wird heute beinahe schon als selbstverständlich vorausgesetzt. Dennoch bietet die in El Kowm verwendete Lösung nur eine native Unterstützung für Windows. Um die Programme unter Unix- oder MAC-Systemen nutzen zu können, ist es erforderlich eine sogenannte Virtualisierung oder einen Wrapper (http://www.winehq.org/, abgerufen: 28.03.2013), welcher die Windows-Umgebung simuliert, zu nutzen. Dadurch lassen sich Windows-Programme auch auf diesen Systemen ausführen. Dies bedeutet aber in gewisser Hinsicht einen Mehraufwand, da sich der Nutzer neben dem eigentlichen Betriebssystem um ein zweites Betriebssystem und zusätzlich um die Vermessungssoftware kümmern muss.

Aus diesen Gründen heraus soll ElKowmGIS plattformunabhängig sein, das heisst auf jedem Betriebssystem ohne weitere Aufwendungen betriebsfähig sein. Um dies zu gewährleisten, muss eine entsprechende Programmiersprache gewählt werden. Es muss ebenfalls darauf geachtet werden, dass allfällige Pfadangaben im Programm die einzelnen Betriebssysteme berücksichtigen. Erwähnt sei hier als Beispiel die unterschiedliche Verwendung der Zeichen “/“ und “\“ in den Systemen Windows und Linux. Während bei Windows ein Pfad mittels “\“ (Backslash) getrennt wird (“C:\Programme\ElKowmGIS“), wird bei Unix-Systemen der Pfad mit Hilfe des Slash (“/“) unterteilt (“../lib/ElKowmGIS“).

Ebenfalls ein Unterschied in den Systemen besteht in der Bezeichnung der Laufwerke. Während bei Unix-Systemen nicht direkt ersichtlich ist, auf welchem Datenträger der Ordner oder die Datei liegt, ist bei der absoluten Pfadangabe unter Windows ersichtlich, auf welchem Laufwerk gearbeitet wird. Um dieses

Problem zu umgehen muss die Software mit sog. relativen Pfaden arbeiten. Dabei wird ausgehend von der Position der ausführenden Programmdatei durch das Dateisystem navigiert. Als Beispiel dient hier der Zugriff auf die Datei config.txt, welche im Ordner conf innerhalb des Programmordners ElKowmGIS liegt. Mit Hilfe einer absoluten Pfadangabe würde der Pfad innerhalb des Programms "C:\Programme\ElKowmGIS\config\config.txt" lauten. Dieser Pfad ist aus den oben angeführten Gründen jedoch unter Unix-Systemen nicht auffindbar. Mittels eines relativen Pfades lautet der Pfad zur config.txt lediglich "config/config.txt". Dies bedeutet, dass sich im gleichen Ordner wie die ausführende Datei ein Ordner "config" befindet, in welchem sich die Datei "config.txt" finden lässt. Dieses Vorgehen bietet somit dem Benutzer die Möglichkeit, den Programmordner auf jedem beliebigen Datenträger abzulegen. Des Weiteren ist hierdurch die Ablage auf einem USB-Stick oder einer CD denkbar.

2.2.6 Modularer Aufbau

Die Idee des modularen Aufbaus rührt aus dem bisherigen System heraus. Hier ist es so, dass es sich bei AutoCAD um die Grundanwendung handelt und die Programme TachyCAD und Photoplan Erweiterungen dieser Anwendung darstellen. Sie sind nicht eigenständig ausführbar. Dieses Konzept soll auch in der neuen Lösung angewendet werden. Wie die Beschreibung aus 2.2.1 bereits zeigt, setzt sich die Grundanwendung bereits aus mehreren Modulen (Datenbank, GIS/CAD, Bildverwaltung und Tabellenkalkulation) zusammen. Sollte ein Benutzer ein weiteres Modul wünschen, um zum Beispiel eine detailliertere archäologische Auswertung der Feuersteinobjekte einer Grabung vorzunehmen, wünsche, so soll es möglich sein, ein entsprechendes Modul zu laden, ohne dass das komplette Programm neu geschrieben werden muss. Hierfür muss bereits bei der Erstellung der Grundanwendung darauf geachtet werden, dass die einzelnen Bereiche, auf die ein eventuell notwendiges Modul zugreifen muss, durch neue Module ansprechbar sind. Ferner muss die Ordnerstruktur für die Verwaltung der Software einen Ordner für die Module bereitstellen. Eine Routine zur Überprüfung, ob weitere Module vorhanden sind, muss ebenfalls mit implementiert werden.

2.2.7 Anwenderfreundlichkeit

Trotz der komplexen Aufgabenstellung verschiedene Systeme zuzulassen und der Kombination der einzelnen Module soll das neue System benutzerfreundlich bleiben. Dies umfasst eine einfach gehaltene Benutzeroberfläche und selbsterklärende Beschriftungen von Schaltflächen und Menüeinträgen. Ferner sollen die Projektverwaltung und die Steuerung der Datenbank über die Software erfüllbar sein. Dazu sollen für diese Bereiche eigene Fenster zur Verfügung stehen, in denen zum Beispiel die Layerstruktur der einzelnen Projekte gesteuert werden kann. Die Datenbank soll direkt aus der Software heraus gesteuert werden können, so dass der Benutzer sich lediglich in die Grundsteuerung der Datenbank einarbeiten sollte. Anlegen neuer Benutzer und die Vergabe von Zugriffsrechten soll aus ElKowmGIS heraus geschehen. Zusätzlich soll ein Beispielprojekt angelegt werden, welches die Möglichkeiten der Datenbank aufzeigt. Die in diesem Projekt vorhandenen Definitionen von Layern, Objekten, Funden, Proben und Dokumenten soll hierbei als Vorlage für neue Projekte dienen, aber dennoch veränderbar bleiben. Eine Auflistung der vordefinierten Einträge findet sich in Anhang B.3.

2

3 Informatische Grundlagen

In den folgenden Abschnitten soll grundlegend auf die verwendeten Softwarelösungen und deren Geschichte vorgestellt werden. Auf die Vor- und Nachteile dieser Anwendungen soll jedoch erst in der Schlussdiskussion eingegangen werden. Für die hier vorliegende Arbeit wurden hauptsächlich die Programmiersprache Java und ein PostgreSQL-Datenbankserver eingesetzt. Weitere Module oder Erweiterungen werden innerhalb der einzelnen Abschnitte separat vorgestellt.

3.1 Die Programmiersprache Java

3.1.1 Beschreibung

Bei der Programmiersprache Java handelt es sich um eine Programmiersprache, welche sich an die Programmiersprachen C und C++ anlehnt, jedoch mit deren Komplexität aufräumt. Grob gesagt sollte mit Java eine einfache, objektorientierte, aber dennoch vertraute Sprache geschaffen werden. Dabei wurde besonderer Wert auf die Einfachheit gelegt. Um dies zu erreichen, wurden bestimmte Funktionen aus den Sprachen C und C++ nicht, resp. nur in vereinfachter Form, übernommen. Im Grossen und Ganzen lassen sich die Ziele der Entwicklung von Java auf folgende fünf Punkte reduzieren (Gosling and McGilton 1996):

- Einfache, objektorientierte und vertraute Programmiersprache:

  Einfach bezeichnet hierbei die Möglichkeit für den Programmierer, ohne grossen Lernaufwand bereits von Beginn an qualitativ hochwertige Produkte zu erstellen. Dies wird zum einen durch die verhältnismässig einfach-gehaltene Grammatik der Programmiersprachen und durch die grosse Klassenbibliothek, auf die der Programmierer zurückgreifen kann, gewährleistet. Zum anderen ist die ganze Sprache logisch aufgebaut und besitzt nicht die Möglichkeit einer GOTO-Funktion. Dies bedeutet, dass der Quelltext Zeile für Zeile abgearbeitet wird.
  Für die Erklärung der Objektorientierung (oder: Objektorientierte Programmierung) sei hier die erste Definition für den Begriff “object-oriented“ von Alan Kay, dem Präger diesen Begriffs, angeführt:

  > Everything is an object
  >
  > Objects communicate by sending and receiving messages (in terms of objects)
  >
  > Objects have their own memory (in terms of objects)
  >
  > Every object is an instance of a class (which must be an object)
  >
  > The class holds the shared behavior for its instances (in the form of objects in a program list
  >
  > To eval a program list, control is passed to the first object and the remainder is treated as its message (Kay 1993, S. 78).

  2003 relativierte Kay im Rahmen einer E-Mail die Definition: „OOP to me means only messaging, local retention and protection and hiding of state-process, and extreme late-binding of all things.“ (Kay 2003).

  Heutzutage lässt sich der Begriff objektorientiert wie folgt definieren: Eine Programmiersprache ist

objektorientiert, wenn sie die Punkte Abstraktion, Vererbung, Kapselung und Polymorphie unterstützt (Ullenboom 2012).

Die Vertrautheit gilt in diesem Zusammenhang Programmierern, die bereits mit C++ gearbeitet haben. Für Sie sollte Java ähnlich aussehen, jedoch nicht so komplex strukturiert sein.

- Sicher und stabil

  Da Java auch für den Einsatz auf verschiedensten Systemen, sowie im Web- sowie auch Netzwerkbereich konzipiert wurde, steht der Punkt Sicherheit an oberster Stelle der Ziele. Durch den Aufbau der Sprache ist es nicht möglich von aussen auf den Quelltext zuzugreifen. Ein weiterer Punkt stellt die Appletsicherheit dar. Applets werden schwerpunktmässig im Web gebraucht, um dem Benutzer Inhalte zur Verfügung zu stellen, ohne dass dieser sie herunterladen muss. Die Applets besitzen aber keinerlei Zugriff auf das System des Benutzers, wodurch es nicht möglich ist, Viren zu programmieren, die die Festplatte des Benutzers zerstören oder auslesen.
  Durch die Überprüfung eines Programms während dem Kompilieren und als zweiter Schritt während der Ausführung können dem Programmierer während seiner Arbeit bereits Fehler und Unstimmigkeiten in seinem Programm aufgezeigt werden. Durch die Java-Interne automatische Speicherbereinigung kann es nicht zu einem Speichermangel kommen. In Java werden nicht mehr benötigte Objekte in regelmässigen Zeitabständen auf einen Ursprungszustand zurückgesetzt und somit Speicher an das System zurückgegeben. Diese Punkte erlauben es, eine stabile Anwendung zu erarbeiten.

- Plattformunabhängig

  Auch dieses Ziel beruht auf der Tatsache, dass Java für verschiedene Systeme konzipiert wurde. Durch diesen Einsatz ist dabei davon auszugehen, dass diverse Geräte und Betriebssysteme genutzt werden. Java garantiert diese Unabhängigkeit durch den Einsatz von **Bytecode**. Dieser wird durch den Java Compiler generiert und gesichert. Auf den Geräten, welche das Programm ausführen möchten, muss lediglich eine **Java Virtual Machine (JVM)** installiert sein, welche ein Teil der **Java Runtime Environment (JRE)** bildet. Diese virtuelle Maschine übersetzt nun den Bytecode in die für das jeweilige Gerät verständliche Maschinensprache.

- Hohe Geschwindigkeit

  Dieser Punkt ist in der heutigen Zeit besonders wichtig. Computer werden immer schneller und Festplatten immer grösser. Dieses Mehr an Geschwindigkeit lässt sich jedoch nur dann nutzen, wenn die Software, welche auf den Rechnern verwendet wird, performant geschrieben wurde. Das ganze System ist nur so schnell, wie die langsamste Komponente. Durch die Trennung des Interpreters und der Laufzeitumgebung erzielt Java eine hohe Geschwindigkeit in der Programmausführung. Auch hier spielt die automatische Speicherbereinigung eine wichtige Rolle. Durch die Freigabe nicht benötigten Speichers steht der gesamten Anwendung genug Speicher zur Verfügung. Optimierungen der Geschwindigkeit lassen sich dadurch erzielen, dass die automatische Speicherbereinigung nach

rechenintensiven Aufgaben manuell im Quelltext aufgerufen wird.

- Interpretiert, nebenläufig und dynamisch
  Im Punkt Plattformunabhängigkeit wurde bereits aufgezeigt, dass die JVM den Bytecode in Maschinensprache übersetzt. Dieses Vorgehen beschreibt den Begriff interpretiert sehr gut. Durch die Übersetzung des Quellcodes in Bytecode und erst auf dem entsprechenden Gerät zur Maschinensprache werden Zeit und Speicher gespart.
  Der Begriff Nebenläufigkeit lässt sich am einfachsten an einem Beispiel erklären. Ein Computernutzer sitzt vor seinem Rechner und bearbeitet in einer Textverarbeitung ein Manuskript. Gleichzeitig lädt er von einem Server Bilder herunter, die er später verwenden möchte. Als weitere Aktivität läuft im Hintergrund ein Musikstück, welchem der Benutzer zuhört. Somit finden drei Tätigkeiten gleichzeitig statt, die sich gegenseitig nicht direkt beeinflussen. Indirekt daher, da sie lediglich den Benutzer und seine Aufmerksamkeit beeinflussen, jedoch nicht die Vorgänge im Rechner. Um die Nebenläufigkeit in einem Satz zu definieren könnte man sie wie folgt umschreiben: Prozesse können als nebenläufig bezeichnet werden, wenn diese nicht direkt voneinander abhängen. Java ist daher so konzipiert, dass sich mehrere Prozesse gleichzeitig ausführen lassen. Durch die Umsetzung der Java-Klassenbibliotheken sind für jeden einzelnen Prozess die gleichen Routinen verfügbar, ohne dass ein Prozess einen anderen blockiert. Java lässt sich als dynamisch beschreiben, da alle Methoden und Routinen erst bei Bedarf aufgerufen werden. Nach der Verwendung werden die Aufrufe wieder geschlossen. Dies ermöglicht es dem Programmierer auch hier wertvollen Speicher zu sparen, da nicht alle Informationen bereits beim Programmstart in den Speicher geladen werden und dort bis zum Programmende gesichert werden müssen. Durch dieses Vorgehen ist es aber auch möglich, neue Module auf schnelle Art und Weise mit einem Programm zu verknüpfen, ohne dass das komplette Programm neu kompiliert werden muss.

3.1.2 Geschichte

Da die gesamte Geschichte von Java bereits ausführlich in etlichen informatischen Fachbüchern niedergeschrieben wurde, sei hier nur eine kurze Zusammenfassung der Geschichte erlaubt. Dem interessierten Leser werden an dieser Stelle jedoch der Artikel von Jon Byous (Byous 1998), der Abschnitt “1.1 Historie“ der Monografie von Krüger und Stark (Krüger and Stark 2011), sowie der Abschnitt “1.1 Historischer Hintergrund“ des Werkes von Ullenboom (Ullenboom 2012) empfohlen. In diesen Publikationen wird die Geschichte und Entwicklung der Java-Technologie ausführlich erläutert.
Die Vorgeschichte, bevor Java 1996 erstmals veröffentlicht wurde, reicht bis in die 1970er Jahre zurück. Damals hatte der Sun-Mitbegründer Bill Joy die Idee einer Programmiersprache, welche die Vorteile anderer Sprachen miteinander vereint. Diese Vision konnte er jedoch nicht umsetzen. In den frühen 1990ern verfasste Joy einen Artikel, in welchem er seine Vorstellungen beschreibt, was eine objektorientierte Programmiersprache ausmacht. Er beging jedoch den Fehler, die neue Sprache auf C++ beruhen zu lassen, da diese für grössere Projekte zu komplex ist.
Gleichzeitig zu dieser Entwicklung arbeitete James Gosling an einem Projekt, in welchem er die neue

Sprache “Oak“ entwarf. 1990 startete Patrick Naughton das Green-Projekt zusammen mit James Gosling und Mike Sheridan. Das Green-Projekt sollte zuerst den Bereich der Unterhaltungs- bzw. Konsumelektronik abdecken. Ein Beweis hierfür stellt das im Rahmen des Green-Projekts entwickelte Gerät *7 (Star Seven), welches im Prinzip eine Fernbedienung für Elektrogeräte darstellte. Das Aussergewöhnliche daran war die Tatsache, dass das Gerät über eine grafische Touch-Oberfläche gesteuert werden konnte.

Die darauf folgende Zeit gestaltete sich schwieriger als gedacht. Verträge mit grösseren Partnern zur Produktion und zur Vermarktung des *7 konnten nicht abgeschlossen werden oder wurden nicht erfüllt. In der Zwischenzeit entwickelte sich jedoch auch das World Wide Web weiter und Sun Microsystems konzentrierte sich nun auf die Entwicklung einer Programmiersprache, welche es ermöglichen sollte, Daten über das Internet auszutauschen, ohne dass ein Schaden dabei entstehen kann. Die Entwickler der Sprache Oak realisierten schnell, dass Ihre Sprache alle Eigenschaften aufwies, um solche Aufgaben sicher durchzuführen. Allerdings war in der Zwischenzeit der Name Oak bereits von einer anderen Programmiersprache benutzt worden: die Geburtsstunde von Java. Der Name Java stammt offenbar von der Kaffeesorte, die die Programmierer bevorzugten: Java.

In dieser Zeit wurde auch ein Browser, basierend auf Java, durch Naughton entwickelt, der HotJava Browser. Lediglich wenige Anwender nutzten jedoch diesen Browser. Das Glück für Java lag jedoch darin, dass Ende 1995 die Java Technologie im Netscape Browser eingesetzt wurde. Im Januar 1996 wurde dann auch das erste **Java Developement Kit** (JDK) veröffentlicht, welches es Programmierern ermöglichte, eigene Java-Programme zu schreiben. Kurz bevor das JDK veröffentlicht wurde, gründeten die Mitglieder des Green-Projekts die Firma JavaSoft, welche von Sun Microsystems auch mit der Betreuung und Weiterentwicklung von Java betraut wurde. Das Copyright für die komplette Java-Technologie weiterhin bei Sun Microsystems, bis diese im Jahr 2010 von der Oracle Cooperation übernommen worden. Oracle übernahm dabei sämtliche Rechte, auch die Rechte an Java.

Seit 1996 gibt es insgesamt 16 Java-Versionen, welche untereinander mehr oder weniger kompatibel waren. Momentan aktuell ist die Version 7.0, auch Dolphin genannt. Die Version 8 ist frühestens in der zweiten Hälfte 2013 zu erwarten.

In der hier vorliegenden Arbeit wurde die aktuelle Version 7 des Java Developement Kit und Runtime Environment in den 64bit-Varianten verwendet. Bei der **Kompilierung** wurde aber darauf geachtet, dass auch ältere Versionen das Programm ausführen können.

3.2 Klassen, die Java erweitern

Für die Umsetzung von ElKowmGIS werden neben der Programmiersprache Java insgesamt 6 erweiternde Klassen genutzt. Hierbei handelt es sich um Bibliotheken, welche den Funktionsumfang von Java erhöhen. Durch den Einsatz solcher Bibliotheken ist es zum Beispiel möglich, Bilder in verschiedenen Formaten schnell zu laden und zu bearbeiten. Weitere Möglichkeiten bestehen zum Beispiel auch in der Anbindung externer Geräte über die sog. **RS232-Schnittstelle**. Die folgenden Abschnitte stellen die einzelnen Bibliotheken vor und erläutern deren Funktionen.

3.2.1 Java GeoTools

Bei den Java GeoTools (Open Source Geospatial Foundation 2013) handelt es sich um ein OpenSource-Projekt, welches Bibliotheken für die Programmierung mit Geodaten unter Java zur Verfügung stellt. Hierbei werden verschiedene Ansätze verfolgt, wie etwa die Nutzung der JTS Topology Suite. Hierbei handelt es sich um eine weitere Bibliothek für Java. Ziel dieser Bibliothek ist es, die Darstellung räumlicher Gegenstände in die objektorientierte Programmierung abzubilden. Weiteres Ziel ist es, Funktionen welche für die Berechnung einzelner Geometrien angewandt werden können, zur Verfügung zu stellen.
Es ist wichtig zu erwähnen, dass die Java GeoTools versuchen, die Spezifikationen, wie sie von dem Open Geospatial Consortium erarbeitet werden, zu implementieren. Gerade in heutiger Zeit, in der immer mehr Daten produziert werden, ist es wichtig, eine gewisse Struktur resp. Einheitlichkeit in die Daten zu bringen.

Die Java GeoTools bieten neben der Unterstützung von Vektor- wie auch Rasterdaten eine grosse Bandbreite an Funktionen und unterstützter Formate. Wichtig hierbei ist vor allem, dass die gängigen Dateiformate, wie etwa Esri's shp-Dateien oder dxf-Dateien von Autodesk, unterstützt werden. Weitere Aspekte sind die Arbeit mit verschiedenen geografischen Projektionssystemen, die Erstellung grafisch aufwendiger Karten und Diagrammen, sowie das Filtern und Bearbeiten von Datensätzen.

Für die Programmierung von ElKowmGIS wurde zuerst die Version 2.7.4 der GeoTools gewählt. Da diese Version jedoch nicht alle Funktionen unterstützte, wurde während der Arbeit auf den 1. Release Candidate der Version 8 (Version 8.0-RC1) gewechselt.

3.2.2 RXTX

Bei der Bibliothek RXTX (RXTX 2013) handelt es sich um eine Sammlung von Klassen und Funktionen, die es ermöglichen mit den seriellen und parallelen Schnittstellen eines Rechners zu kommunizieren. Dieses Modul wird in der Arbeit für die Kommunikation mit externen Geräten, wie etwa einem Tachymeter oder einem GPS-Gerät, benötigt. Es lehnt sich an Spezifikationen der Java Communications API an ohne dabei komplexe Programmierstrukturen zu erfordern.

Bei RXTX handelt es sich mit 52 Klassen um eine vergleichsweise kleine Bibliothek, da der Funktionsumfang nur auf die Kommunikation mit den Schnittstellen beschränkt ist. Für das Auslesen der Daten gibt es verschiedene Reader, während das Schreiben auf ein Gerät über verschiedene Writer möglich ist. Dabei ist anzumerken, dass RXTX die Daten lediglich in dem Format annimmt, wie sie vom Gerät geliefert werden. Eine allfällige Übersetzung in eine für das restliche Programm verwendbare Form muss vom Programmierer erstellt werden. Beispiele hierfür finden sich in Abschnitt "7.11 Standpunktbestimmung".

3.2.3 Java ImageIO und Java Advanced Imaging (JAI)

Diese Klassen bieten, wie es der Name bereits vermuten lässt, Routinen und Funktionen zum Öffnen und der Bearbeitung von Bildern. Die beiden Klassenbibliotheken unterscheiden sich durch ihren Funktionsumfang: Das Öffnen, sowie auch das Speichern von Bildern wird mittels der Klassen aus dem Java ImageIO (Oracle 2013a) durchgeführt. Die Klassen der JAI-Bibliothek (Oracle 2013a) bieten Routinen und Funktionen zur Bearbeitung von Bildern. Durch die Nutzung einer dritten Bibliothek, der JAI ImageIO,

3

lassen sich weitere Formate mittels der Java ImageIO öffnen. Die ähnlich lautenden Namen der Klassen rühren aus den Tatsachen, dass zum einen die Bibliotheken eng miteinander verwandt sind und zum anderen die Autoren der einzelnen Bibliotheken die gleichen waren.

3.2.4 Log4j

Bei Log4j (Apache Software Foundation 2013) handelt es sich um eine freie Bibliothek der Apache Software Foundation. Sie dient vor allem der Erstellung eigener Log-Routinen, die es dem Programmierer ermöglichen, eigene Log-Dateien zu generieren. In einer Log-Datei werden die Vorgänge und Ereignisse während der Programmausführung gesichert. Mit Log4j lässt sich nun definieren, welche Ereignisse und Vorgänge gespeichert werden. Dadurch lassen sich zum Beispiel der Start des Programms, das erfolgreiche Laden eines wichtigen Treibers, der erfolgreiche Aufbau einer Verbindung zu einem Server oder sämtliche Fehler während der Programmausführung protokollieren. Mit der hier verwendeten Bibliothek kann der Programmierer bestimmen, wann und wo im Programm eine Meldung in das Protokoll geschrieben wird. Hierfür werden sog. Logger verwendet, die das Prozedere des Schreibens übernehmen. Jedem Logger sollte dabei eines der folgenden Level zugewiesen werden. Die Level definieren hierbei welche Meldungen ausgegeben werden.

TRACE: Hiermit werden detaillierte Meldungen ausgegeben an welcher Position innerhalb des Quelltextes sich das Programm gerade befindet. So kann zum Beispiel verfolgt werden, wann eine Funktion gestartet und diese wieder beendet wird.

DEBUG: Debug-Meldungen sind unpräziser als Trace-Meldungen, geben sie lediglich die Position im Quelltext wieder, welche den Fehler auslöst.
INFO: Wie der Name schon vermuten lässt, werden hiermit Informationen ausgegeben. Beispielsweise lässt sich hiermit die erfolgreiche Anmeldung an einem Server protokollieren.
WARN: Mit Warn-Meldungen lassen sich Warnungen ausgeben. Gründe für eine Warnmeldung können vielfältig sein. Beispielsweise könnte eine Warnmeldung ausgegeben werden, wenn der Benutzer eine falsche Zahl eingegeben hat, die in einer späteren Berechnung beispielsweise zu einer Division durch 0 führen könnte. Eine weitere Warnmeldung könnte ausgegeben werden, falls der Speicher eines Rechners übermässig belastet wird.

ERROR: Fehlermeldungen werden dann ausgegeben, wenn eine Eingabe oder ein Vorgehen zu einem Fehler führt, der aber die Ausführung des Programms nicht stoppt. Ein Beispiel hierfür wäre das Öffnen eines zu grossen Bildes, was dazu führt, dass mehr Speicher benötigt wird, als vorhanden ist. Dabei wird der Ladevorgang des Bildes zwar abgebrochen, jedoch läuft das Programm weiter.
FATAL: Die unangenehmsten Meldungen stellen Fatal-Meldungen dar. Sie werden ausgegeben, wenn ein Fehler so fatal ist, dass das gesamte Programm abgebrochen werden muss. Ein Grund hierfür können die Nichtverfügbarkeit einer Klassenbibliothek oder, im Fall von ElKowmGIS, die fehlende Verbindung zu einem Datenbankserver sein.
Des Weiteren gibt es noch die Level ALL und OFF. Sie führen dazu, dass der Logger alle (All) oder keine (OFF) Meldungen protokolliert. Theoretisch ist es auch möglich, eigene Level umzusetzen, jedoch raten

die Hersteller davon ab (Gülcü 2002).

Ein weiterer Grund, der für die Nutzung von Log4j spricht, ist die Möglichkeit, das Protokoll auf vielfältige Weise auszugeben. So lassen sich die Meldungen zum Beispiel auf ein Konsolenfenster oder in eine eigene Datei ausgeben. Dabei kann entschieden werden, ob die Daten überschrieben werden, oder ob sie an allfällig bereits bestehende Daten angehängt werden sollen. Hierbei können auch Routinen genutzt werden, die in bestimmten Zeitintervallen die Logdatei unter einem neuen Namen mit Datumsangabe abspeichern und die aktuelle Logdatei leeren. Dadurch kann verhindert werden, dass Logdateien zu gross werden. Dieses Vorgehen bot sich in modifizierter Weise auch für die Testphase von ElKowmGIS an. Die Modifikation lag hierbei im Rhythmus der Speicherung. Während der Testphase wurde mit jedem Programmstart eine neue Logdatei geschrieben. Somit liessen sich die einzelnen Sitzungen separat voneinander analysieren und eventuell vorhandene Schwachstellen beheben.
Ferner ist es mit Log4j möglich vordefinierte Layouts für die Ausgabe zu nutzen. Damit lässt sich das Aussehen der Logdatei konfigurieren. Im Falle von ElKowmGIS wird das sog. HTMLLayout verwendet. Dieses ermöglicht die Ausgabe der Meldungen in eine HTML-Datei, welche mit jedem Browser geöffnet werden kann. Der Nachteil im HTMLLayout liegt darin, dass die Anordnung der Spalten bei der Ausgabe nicht geändert werden kann. Ein Beispielprotokoll findet sich in Anhang C.

3.2.5 PostgreSQL JDBC Driver

Beim PostgreSQL JDBC Driver (The PostgreSQL Global Development Group 2011) handelt es sich prinzipiell ebenfalls um eine Klassenbibliothek, die gleich aufgebaut ist, wie die bisher vorgestellten Bibliotheken. Da sie jedoch die Verbindung zu einem PostgreSQL-Server erlaubt, wird sie als Treiber (Driver) bezeichnet. Auch wenn die Bibliothek weitere Funktionen zur Verfügung stellt, wird für die hier vorliegende Arbeit lediglich die Funktionalität als Treiber genutzt. Alle Abfragen und Änderungen der Datenbank können über die Java-interne java.sql-Bibliothek ausgeführt werden.

3.3 Der Datenbankserver: PostgreSQL

Die PostgreSQL-Datenbank bezeichnet sich selbst als “The world's most advanced open source database“ (The PostgreSQL Global Development Group 2013b) und stellt in der Tat ein freies, relationales und mächtiges Datenbanksystem dar. Das System verdankt seiner Eigenschaft, möglichst viele Plattformen zu unterstützen, eine weite Verbreitung in der EDV-Welt. So ist sie bereits in den meisten Linux-Systemen zu finden. Selbst Apple liefert die Datenbanklösung mit ihrem Betriebssystem “Mac OS X Lion Server“, einem Betriebssystem für Server als Standarddatenbank aus (The PostgreSQL Global Development Group 2013a).
Die Entwicklung des PostgreSQL-Servers begann bereits 1986, als Fortführung des Datenbankprojektes Ingres, welches von 1977-1985 lief. Michael Stonebraker, der Hauptverantwortliche des Ingresprojektes, verliess bereits 1982 die University of California in Berkeley (UCB), um das Projekt kommerziell zu vertreiben. 1985 kehrte er jedoch an die UCB zurück und startete ein Jahr später die Entwicklung eines Post-Ingres-Projektes. Während der nächsten Jahre entwickelte das Team um Stonebraker die Datenbank und alle Regeln, Datentypen, etc., die für die Datenbank notwendig waren.
Einen wichtigen Schritt unternahmen 1995 zwei Doktoranden Stonebrakers, als sie die Syntax der

Datenbank auf eine erweiterte **SQL-Syntax** umschrieben. Im gleichen Atemzug wurde das Projekt in Postgres95 umbenannt, da mittlerweile das ursprüngliche Projekt kommerzialisiert worden war.

Der eigentliche Durchbruch des Systems gelang eigentlich erst 1996, als das Projekt in ein OpenSource-Projekt ausserhalb der UCB umgewandelt wurde. Seither betreut eine grosse OpenSource-Community dieses System. Seither gewinnt die Datenbank regelmässig Preise als beste Datenbank oder als beste Serveranwendung. Den letzten Preis gewann das System auf der Cebit 2012 in Hannover. Dabei handelt es sich um den Linux New Media Award für die beste OpenSource Datenbank (Huber 2012).

PostgreSQL begann mit Version 6. Bis heute entwickelte das Team etliche neue Versionen. Die aktuellste Version ist Version 9.1.4. Für die hier vorliegende Arbeit wurde Version 9.0.2 verwendet.
Da die Datenbank sehr mächtig ist und diese Arbeit nur einen kleinen Teil des vollen Funktionsumfangs nutzt, seien hier nur die für die Arbeit wichtigen Funktionen aufgeführt. Auf der Homepage der Community (speziell unter: http://www.postgresql.org/about/featurematrix/) werden alle Eigenschaften des Systems ausführlich aufgelistet und beschrieben.
Folgende Gründe sprachen für den Einsatz des Systems in dieser Arbeit:

- Schnittstelle zur Programmiersprache Java (neben vielen anderen Sprachen)
- Mengenoperationen möglich (Berechnen von Werten innerhalb der Datenbank)
- Verbreitung auf vielen Plattformen
- Datenbankgrösse lediglich durch den Speicher, welches das System zugewiesen bekommt, begrenzt
- Daten können direkt oder über sog. **Schemata** ex- bzw. importiert werden.

Gerade der letzte Punkt war für die hier vorliegende Arbeit interessant, da es mittels der Schemata möglich ist, bestimmte Bereiche innerhalb der Datenbank voneinander zu trennen. Die Idee hierbei war es, ein Schema mit verschiedenen Tabellen für die Definitionen und ein Schema für die eigentlichen Messdaten anzulegen. Da die Schemata auch ex- resp. importiert werden können, ist es möglich, dem Benutzer Vorlagen für die Datenbank mit zu liefern.

Auch für den PostgreSQL-Server gibt es einige Erweiterungen, die den Funktionsumfang erweitern sollen. In dieser Arbeit wird lediglich die PostGIS-Erweiterung genutzt. Sie ermöglicht es, die Datenbank in eine Geodatenbank zu verwandeln, welche mit den gängigsten GIS-Programmen verwendet werden kann.

3.3.1 Aufbau eines PostgreSQL-Servers

Bevor im folgenden Kapitel 4 auf die Eigenschaften der Datenbank von ElKowmGIS eingegangen wird, soll an dieser Stelle kurz auf den Aufbau eines PostgreSQL-Servers eingegangen werden. Dadurch lassen sich das Konzept und die Struktur der Datenbank der hier vorliegenden Arbeit leichter verstehen.

Jeder PostgreSQL-Server kann mehrere Datenbanken enthalten, wobei die Benutzerkonten über alle Datenbanken hinweg verwendet werden können. Daten hingegen können nicht über mehrere Datenbanken hinweg getauscht werden. Ferner ist es nur möglich, sich mit einer Datenbank auf dem Server zu verbinden. Es ist nicht möglich mit einem Benutzer gleichzeitig in mehreren Datenbanken zu arbeiten.
Jede Datenbank beinhaltet wiederum ein oder mehrere Schemata. Ein Schema kann am ehesten mit einem Ordner im Dateisystem eines Rechners verglichen werden. In den Schemata werden Tabellen, Funktionen, Wörterbücher und weitere Objekte abgelegt, die bei der Benutzung der Datenbank benötigt werden. Der entscheidende Vorteil bei den Schemata liegt darin, dass Namen und Bezeichnungen nur innerhalb eines Schemas eindeutig sein müssen. Beispielsweise ist es möglich, dass die Schemata “def“ und “daten“ je eine Funktion mit dem Namen Summe enthalten, die in den beiden Schemata jedoch unterschiedliche Berechnungen durchführt. Ferner ist es durch Schemata möglich, mehrere Benutzer und Programme gleichzeitig mit der Datenbank arbeiten zu lassen, ohne dass es zu Konflikten kommt. Ein weiterer wichtiger Punkt ist die Möglichkeit durch die Nutzung von Schemata eine Gruppierung von Bereichen in der Datenbank umzusetzen, was eine einfachere Verwaltung der Daten und Definitionen ermöglicht.

Zu jeder Tabelle, die in einem Schema abgelegt wird, gehören weitere Informationen. So werden die Spaltenüberschriften und der Typ der Spalte (Zahl, Text, etc.), allfällige Bedingungen (sog. Constraints), Indizes, Regeln und Trigger mit abgelegt. Diese Informationen steuern das Verhalten der Datenbank, wenn Daten in eine Tabelle eingetragen, gelöscht oder aktualisiert werden.
Da ElKowmGIS lediglich die oben beschriebenen Funktionen nutzt, werden die weiteren Funktionen, wie zum Beispiel die Nutzung von Wörterbüchern oder Triggern, hier nicht vorgestellt. Für eine ausführliche Beschreibung und Benutzungshinweise wird hier auf das Handbuch zur PostgreSQL-Installation (The PostgreSQL Global Development Group 2013c) sowie auf das offizielle Handbuch (Eisentraut 2003, 31ff.) verwiesen.

3

4 Die Datenbank von ElKowmGIS

4.1 Konzept

Das Datenbankkonzept von ElKowmGIS umfasst prinzipiell gesehen vier Grundideen. Zuallererst sollten die Daten von den Definitionen getrennt sein, d.h. die eingefügten Daten (Funde, Befunde usw.) werden losgelöst von den Definitionen (Layer, Phasen, etc.) gespeichert. Dieses Vorgehen ermöglicht es, eine Art Anfangszustand von ElKowmGIS wiederherzustellen, ohne die Definitionen zu löschen. Andererseits lassen sich Definitionen schnell und einfach hinzufügen oder entfernen, ohne die Daten direkt zu beeinflussen.

Der zweite wichtige Aspekt war die eindeutige Identifikation (unique Key) der einzelnen Einträge in der Datenbank. Hierfür wird zwischen den Definitionen und den Daten selbst unterschieden. Für die Definitionen dienen zwei Felder der jeweiligen Tabelle als eindeutige Identifikation. Hierbei handelt es sich immer um die Grabungs- resp. Projektnummer und eine ID, welche für jede Tabelle separat generiert wird. Dadurch ist gewährleistet, dass innerhalb einer Grabung jede ID nur einmal pro Tabelle vorhanden sein kann. Somit ist jede Definition eindeutig identifizierbar. Auf der Seite der Daten wird ein etwas komplexeres System genutzt. Hier wird je nach Tabelle aus verschiedenen Feldern die eindeutige Kennung generiert. Ausser bei der Tabelle, die die Projektinformationen speichert, werden aber immer die Felder Projektnummer, unit, subunit und die Laufnummer verwendet, um die Einträge anzusprechen. In einzelnen Fällen wird ein weiteres Feld an die eindeutige ID angehängt.

Hinweis:

Die Begriffe unit und subunit sind hier ohne jeglichen Kontext zu sehen. Die Begriffe wurden gewählt, da sie als neutrale Bezeichnung für die Beschreibung mehrerer Gegebenheiten gesehen werden können. Die Interpretation der Bedeutung einer unit bleibt dem Benutzer überlassen. Einheiten könnten zum Beispiel einzelne Kampagnen oder verschiedene Fundkomplexe innerhalb einer Grabung sein.

Die Definitionen sollten so offen wie möglich sein, damit der Benutzer das Aussehen seiner Informationen weitestgehend selbstständig definieren kann. Hierfür wurden 12 Tabellen angelegt, für jeden Objekttyp eine. In diesen Tabellen werden beispielsweise die Darstellungsfarbe oder die Abkürzung der Bezeichnung gespeichert. Diese Einstellungen können direkt über ElKowmGIS eingesehen und geändert werden. Zusätzlich zu diesen Tabellen beinhaltet das Schema def eine weitere Tabelle, die die Benutzerinformationen speichert. Die genaue Funktion dieser Tabelle wird in Abschnitt “7.16 Der Verwaltungsbereich“ ausführlich beschrieben.

Schliesslich sollen die Daten in mehreren Tabellen aufgeteilt abgelegt werden, um die Tabellen an für sich übersichtlicher zu gestalten. Hierfür wurden insgesamt 5 Tabellen angelegt, welche die verschiedenen Informationen beinhalten. Eine genaue Beschreibung der Datentabellen findet sich im übernächsten Abschnitt.

4.2 Das Schema def

Das Schema def enthält insgesamt 13 Tabellen. Im Folgenden sollen nun die einzelnen Tabellen vorgestellt und ihre Funktion erläutert werden. Dabei werden für jede Tabelle die Anzahl der Spalten und die Bestandteile des eindeutigen Identifikator (unique Key) aufgelistet. Anschliessend folgt eine kurze Erklärung der Funktion der Tabelle. Eine Übersicht über die Spaltennamen und den Inhalt der Tabellen findet sich in Anhang B1. Bei allen Tabellen des Schemas def wurde vor dem Tabellennamen ein def_ vorangestellt, damit sofort ersichtlich ist, dass es sich um eine Definitionstabelle handelt.
Alle Tabellen des Schemas def, ausser der Tabelle def_user, verfügen über denselben Constraint. Sie beziehen den Schlüssel excavationid als Fremdschlüssel aus der Tabelle excavations im Schema elkowmgis. Die SQL-Syntax sieht wie folgt aus:

CONSTRAINT excavationid FOREIGN KEY (excavationid) REFERENCES elkowmgis.excavations (excavationid) MATCH SIMPLE ON UPDATE NO ACTION ON DELETE NO ACTION
Die Bedingung lässt sich wie folgt lesen:

Der Wert in der Spalte excavationid der jeweiligen Tabelle muss mit dem Wert in einer Zelle der Spalte excavationid der Tabelle excavations im Schema elkowmgis übereinstimmen. Andernfalls ist der Eintrag ungültig. MATCH SIMPLE erlaubt, dass auch NULL-Werte in der Spalte vorkommen. ON UPDATE und ON DELETE definieren das Verhalten der Datenbank, wenn in der Tabelle excavations ein Eintrag gelöscht wird. Da diese Aktionen bereits von ElKowmGIS gesteuert werden, ist der Wert jeweils NO ACTION. Dies bedeutet, die Datenbank führt keine Aktion aus.

Hinweis:

ElKowmGIS verwaltet die Vergabe des Schlüssels selbstständig. Sie als Benutzer oder Administrator des Programms müssen sich nicht um die Vergabe der Schlüssel in der Datenbank kümmern.

Hinweis:

Genauer gesagt handelt es sich bei einem unique Key ebenfalls um eine Bedingung, die gewährleistet, dass jeder Schlüssel nur einmal in einer Tabelle vorkommt. Die volle SQL-Syntax dazu lautet beispielsweise:
CONSTRAINT def_cats_excavationid_key UNIQUE (excavationid, catid)
Der Primärschlüssel der Tabelle def_cats setzt sich aus den Werten excavationid und catid zusammen und muss eindeutig (UNIQUE) sein. Beim Ausdruck def_cats_excavationid_key handelt es sich um den Namen der Bedingung.

Tabelle def_cats	
Anzahl Spalten: 13	unique Key: excavationid, catid

Funktion:

Speichert die Fundkategorien (Silex, Knochen, etc.) für jedes Projekt einzeln ab. Weiter werden die Abkürzung der Kategorie sowie die Darstellungsoptionen (Farbe, Linientyp, Symbol usw.) hier abgelegt. Zusätzlich werden die möglichen Statusmeldungen (ausgeschieden, verbrannt usw.) in einer Spalte gesichert. Schliesslich wird gesichert, auf welchem Layer die Objekte dargestellt werden.

Tabelle def_codes	
Anzahl Spalten: 12	unique Key: excavationid, codeid

Funktion:

In dieser Tabelle finden sich die Objekttypen (Punkt, Fund, Probe, Dokument usw.), die ElKowmGIS für die korrekte Darstellung aller Daten benötigt. In dieser Tabelle werden ebenfalls Darstellungsoptionen festgehalten. Dies ermöglicht es, alle Objekte des gleichen Typs in beispielsweise einer Farbe darzustellen, ohne diese Einstellung für jedes Objekt separat einstellen zu müssen.

Tabelle def_data	
Anzahl Spalten: 5	unique Key: excavationid, epochid

Funktion:

In den 5 Spalten der Tabelle def_data werden die Phasenbezeichnungen für die einzelnen Projekte abgespeichert. Dabei ist die Spalte epoch, die die eigentlichen Bezeichnungen enthält, als Textfeld angelegt. Somit kann eine Phasenbezeichnung auch aus einem Text bestehen.

Tabelle def_documents	
Anzahl Spalten: 12	unique Key: excavationid, docid

Funktion:

Die Tabelle def_documents enthält alle Dokumenttypen, die in ElKowmGIS vorkommen können. Dabei wird unterschieden, was das Dokument enthält. Momentan enthält die Tabelle die Typen drawing, photo, plan, text, table, video und audio. Mit diesen Datensätzen sollten alle Bedürfnisse einer archäologischen Dokumentation abgedeckt sein. Auch in dieser Tabelle finden sich Spalten für die Darstellungsoptionen. Somit ist es möglich, die Darstellungsoptionen in ElKowmGIS so zu wählen, dass sich verschiedene Dokumenttypen direkt im Programm optisch unterscheiden lassen.

Tabelle def_features	
Anzahl Spalten: 13	unique Key: excavationid, featureid
Funktion:	
Def_features ist prinzipiell gleich aufgebaut wie die Tabelle def_cats oder def_samples. In dieser Tabelle werden jedoch die einzelnen Befundtypen (Mauer, Brunnen, Grab usw.) und ihre Abkürzungen gesichert. Des Weiteren werden auch hier Darstellungsoptionen mit gesichert, um die optische Unterscheidung der Befunde in ElKowmGIS zu gewährleisten. Die Befunde werden für jedes Projekt separat abgelegt.	

Tabelle def_layer	
Anzahl Spalten: 12	unique Key: excavationid, layerid
Funktion:	
Für die Sicherung der verwendeten Layer in einem Projekt steht die Tabelle def_layer zur Verfügung. Wie in den anderen Tabellen auch werden hier neben dem Namen und der Abkürzung die Darstellungsoptionen abgespeichert. Im Unterschied zu den meisten anderen Tabellen werden in dieser Tabelle auch Statusinformationen abgelegt.	

Tabelle def_lines	
Anzahl Spalten: 8	unique Key: excavationid, lineid
Funktion:	
Für die Darstellung von Linien im GIS-/CAD-Modul werden hier verschiedene Linientypen gesichert. ElKowmGIS liefert bereits eine Auswahl an verschiedenen Linien, die im Grabungsalltag benötigt werden. Eigene Linien können über das Optionsmenu des Programms eingefügt werden.	

Tabelle def_phases	
Anzahl Spalten: 7	unique Key: excavationid, phaseid
Funktion:	
Speichert für jedes Projekt die Phasendefinitionen ab. Hierfür werden die Grabungsnummer, die ID der Phase, der Name, die Zeitspanne, die Epochen-ID (aus der Tabelle def_data) und die Kultur abgelegt. Die zwei Spalten der Zeitspanne (von- und bis-Spalte) sind als double precision-Felder angelegt. Dadurch ist es möglich, nicht nur Ganzzahlen zu speichern.	

Tabelle def_samples	
Anzahl Spalten: 13	unique Key: excavationid, sampleid

Funktion:

Hier werden die einzelnen Probentypen der Projekte gesichert. Die Definitionen umfassen eine ID, eine Bezeichnung, eine Abkürzung, Darstellungsoptionen und eine Statusspalte. Mit der Statusspalte soll ermöglicht werden, den aktuellen Bearbeitungszustand einer Probe festhalten zu können.

Tabelle def_states	
Anzahl Spalten: 4	unique Key: excavationid, stateid

Funktion:

Die Tabelle def_states beherbergt verschiedene Zustandsmeldungen für die einzelnen Objekte. Hierbei werden lediglich die Grabungsnummer, die eindeutige ID, der Name und die Abkürzung gespeichert.

Tabelle def_symbols	
Anzahl Spalten: 8	unique Key: excavationid, symbolid

Funktion:

Diese Tabelle ist vergleichbar mit der Tabelle def_lines. Hier werden jedoch die Informationen für die Punktdarstellungen gespeichert. ElKowmGIS liefert auch hier eine Vorauswahl. Weitere Symbole können über das Menu von ElKowmGIS hinzugefügt werden.

Tabelle def_topo	
Anzahl Spalten: 12	unique Key: excavationid, topoid

Funktion:

Sichert die unterschiedlichen topografischen Objekttypen, wie etwa Fixpunkte, Bildpunkte, Geländepunkte oder Niveaugrenzen. Auch hier werden die Darstellungsoptionen mit abgelegt. Für jedes Projekt werden diese Informationen in die Tabelle eingetragen.

4

Tabelle def_user	
Anzahl Spalten: 3	unique Key: username

Funktion:

Die Tabelle def_user ist nicht vergleichbar mit den bisher vorgestellten Tabellen. Sie besteht lediglich aus drei Spalten und speichert den Benutzernamen, die Funktion des Benutzers sowie die Berechtigungen in den einzelnen Projekten.

Die Funktion des Benutzers unterscheidet sich hierbei in User (U) oder Administrator (A). Diese Unterscheidung bezieht sich lediglich auf die Verwaltung der gesamten Datenbank, nicht die Verwaltung eines einzelnen Projektes. So kann ein User durchaus berechtigt sein, in einem Projekt Daten einzugeben und zu löschen, während er in einem anderen Projekt die Daten nur lesen kann. Administratoren hingegen können in allen Projekten lesen, schreiben und löschen. Zusätzlich können sie Projekte komplett löschen. Ferner können Administratoren neue Benutzer anlegen, resp. bestehende löschen. Sie können ausserdem die Berechtigungen der einzelnen User steuern.

Die Berechtigungen werden in einem Textfeld gespeichert. Dabei werden vordefinierte Zeichenketten gebildet. Diese umfassen immer den Grabungsnamen, ein Fragezeichen als Trennzeichen zwischen Projekt und Berechtigung, den Berechtigungen, sowie ein Stern als Trennzeichen zwischen den Projekten. Bei den Berechtigungen wurden die in Datenbanken und Dateisystemen üblichen Buchstaben r,w,d für das Lesen, Schreiben und Löschen, sowie der Buchstabe u für die Benutzerverwaltung verwendet.

Beispiel:

Ein Benutzer besitzt folgende Berechtigungen: „0?r*HU10?rw*MU09?rwdu“. Ausformuliert bedeutet dies:

Der Benutzer besitzt in dem Projekt mit der Nummer “0“ nur Leserechte (r).

Im Projekt “HU10“ besitzt er Lese- und Schreibrechte (rw), kann jedoch keine Einträge löschen.

Für das Projekt MU09 kann der Anwender alle Datensätze lesen (r), neue Einträge erstellen (w), Daten löschen (d) und die Benutzersteuerung des Projektes (u) übernehmen.

Hinweis:

PostgreSQL verfügt bereits über eine eigene Benutzerverwaltung. Die Tabelle def_user stellt daher eine Erweiterung der PostgreSQL-eigenen Strukturen dar. Dieser Schritt war notwendig, da die Daten aller Projekte in den gleichen Tabellen abgelegt werden. Um eine Zugriffssteuerung zu entwickeln, war es daher nötig, diese Lösung zu wählen. PostgreSQL bietet eine Zugriffssteuerung für ganze Tabellen, jedoch nicht für die einzelnen Einträge. Durch die hier gewählte Lösung ist es nun jedoch möglich, einzelne Zeilen mit einer Steuerung des Zugriffs zu versehen.

4.3 Schema elkowmgis

In diesem Schema befinden sich die 5 Tabellen, die die Daten speichern, welche zur Laufzeit generiert werden. Die Verteilung der Daten in 5 verschiedene Tabellen dient hierbei der Übersichtlichkeit, aber auch dazu, Redundanzen zu vermeiden. Hier spielt das Grundkonzept einer relationalen Datenbank, wie es PostgreSQL ist, eine wichtige Rolle. Bestimmte Datensätze müssen nur einmal abgelegt werden und können dann durch eine eindeutige Verknüpfung wieder aufgerufen werden. Im Vergleich zum Schema def sind die Constraints hier komplexer. Daher werden diese für jede Tabelle separat vorgestellt. Die Beschreibung der Constraints findet sich im vorhergehenden Abschnitt.

Tabelle context
Anzahl Spalten: 21 unique Key: excavationid, unit, subunit, id Constraints: CONSTRAINT excavationid FOREIGN KEY (excavationid) REFERENCES elkowmgis.excavations (excavationid) MATCH SIMPLE ON UPDATE NO ACTION ON DELETE NO ACTION
Funktion:
Speichert den Kontext zu einem Objekt. Dabei werden die Laufnummer, der Typ, die Zustandswerte, der Besitzer, die Geschichte des Objekts (Bsp.: Erstellt durch Benutzer DS am 25.05.2012), der Layer, das Niveau und die Darstellungsoptionen gesichert. Der Fremdschlüssel excavationid wird mit der Tabelle excavations verknüpft.

Tabelle datings
Anzahl Spalten: 10 unique Key: excavationid, unit, subunit, id, datingid Weitere Constraints: CONSTRAINT excavationid FOREIGN KEY (excavationid) REFERENCES elkowmgis.excavations (excavationid) MATCH SIMPLE ON UPDATE NO ACTION ON DELETE NO ACTION CONSTRAINT id FOREIGN KEY (unit, subunit, id, excavationid) REFERENCES elkowmgis.context (unit, subunit, id, excavationid) MATCH SIMPLE ON UPDATE NO ACTION ON DELETE NO ACTION
Funktion:
Tabelle für die Sicherung von Datierungsergebnissen. Hierfür wird neben der Laufnummer der Datierung auch die Laufnummer eines eventuell zugehörigen Objektes gespeichert. Ferner werden der Ersteller, die Geschichte des Eintrags sowie die Datierung selbst gespeichert. Der Primärschlüssel wird mit zwei Fremdschlüsseln verknüpft. Zum einen mit der excavationid der Tabelle excavations, sowie mit den Werten excavationid, unit, subunit und id der Tabelle context.

Tabelle documents

Anzahl Spalten: 10 unique Key: excavationid, unit, subunit, id, documentid

Weitere Constraints:

CONSTRAINT excavationid FOREIGN KEY (excavationid) REFERENCES elkowmgis.excavations (excavationid) MATCH SIMPLE ON UPDATE NO ACTION ON DELETE NO ACTION
CONSTRAINT id FOREIGN KEY (unit, subunit, id, excavationid) REFERENCES elkowmgis.context (unit, subunit, id, excavationid) MATCH SIMPLE ON UPDATE NO ACTION ON DELETE NO ACTION

Funktion:

Die erste Idee hinter dieser Tabelle, die Dokumente mit in der Datenbank zu sichern, musste aufgrund von Geschwindigkeitsproblemen verworfen werden (siehe auch Kapitel: Ziele der Arbeit). Die Idee, Dokumente separat aufzulisten, sollte jedoch nicht verworfen werden, da es somit möglich ist, sich schnell einen Überblick über die vorhandenen Dokumente zu verschaffen. Hierfür stehen nun insgesamt 10 Spalten zur Verfügung. Sie speichern neben der Grabungsnummer, der Einheit, der Untereinheit und der ID noch die aktuelle Laufnummer des Dokuments, den Dokumenttyp (siehe Tabelle def_documents) und den Speicherort auf dem Dateiserver. Die Verknüpfungen des Primärschlüssels sind gleich den Verknüpfungen der vorangehenden Tabelle.

Tabelle excavations

Anzahl Spalten: 10 unique Key: excavationid

Weitere Constraints:

keine

Funktion:

Bei der Tabelle excavations handelt es sich streng genommen um die wichtigste Tabelle der gesamten Datenbank. In ihr werden sämtliche Projekte mit den wichtigsten Informationen abgelegt. Ohne diese Tabelle ist ElKowmGIS definitiv nicht lauffähig. Gespeichert werden hier die Projektnummer, der Name des Projektes, der Ort, die Kampagne, das Darstellungsformat der Laufnummern, die Aufnahme der Orientierung, der Ersteller, ein Zustand, Bemerkungen und die Geschichte des Projektes. Die Geschichte des Projektes speichert lediglich Änderungen, die den Eintrag in der Tabelle excavations betreffen. Veränderungen in anderen Tabellen werden in der jeweiligen Tabelle gesichert. Als eindeutige Identifikation dient hier die Grabungsnummer. Dabei ist zu berücksichtigen, dass der Begriff Grabungsnummer nicht vollständig richtig ist. Es kann sich hierbei auch um eine Kombination von Buchstaben und Zahlen handeln (Bsp.: HU10 oder Basel-10-Rheingasse-2541). Durch den Unique-Constraint ist es nicht möglich, dass eine Projektnummer zweimal vergeben wird.

Für die fünfte Datentabelle, die Tabelle xyz bedarf es vorgängig einer Erklärung. Die Tabelle xyz enthält eine Spalte namens suffix. Bei diesem numerischen Wert handelt es sich um eine interne Laufnummer, die von ElKowmGIS vergeben wird, falls ein Objekt mit mehreren Punkten oder als Linie aufgenommen wird. Durch die Einführung dieses Suffix wird es beispielsweise ermöglicht, mit einem Objekt neben der Orientierung, auch die Rohdaten für die Berechnung der Orientierung mit abzuspeichern. Ein weiteres Beispiel stellt die Einmessung einer römischen Mauer dar. In der Regel werden bei Mauern mehrere Höhenpunkte aufgenommen. ElKowmGIS ist es durch diesen suffix möglich, mehrere Höhen der gleichen Mauer mit der gleichen Mauernummer aufzunehmen. Die Verwaltung dieses Zählers wird dabei automatisch von ElKowmGIS übernommen und dem Benutzer lediglich zur Information auf dem Bildschirm angezeigt.

Tabelle xyz
Anzahl Spalten: 10 unique Key: excavationid, unit, subunit, id, suffix Weitere Constraints: CONSTRAINT excavationid FOREIGN KEY (excavationid) REFERENCES elkowmgis.excavations (excavationid) MATCH SIMPLE ON UPDATE NO ACTION ON DELETE NO ACTION CONSTRAINT id FOREIGN KEY (unit, subunit, id, excavationid) REFERENCES elkowmgis.context (unit, subunit, id, excavationid) MATCH SIMPLE ON UPDATE NO ACTION ON DELETE NO ACTION
Funktion:
Die Tabelle xyz dient der Speicherung der Koordinaten der Einträge in der Tabelle context. Eindeutig müssen hierbei die Werte excavationid, unit, subunit, id und suffix sein. Die Koordinaten werden als Zahlen abgespeichert, getrennt nach X-, Y- und Z-Koordinate. Auch hier stehen ein Feld für Bemerkungen und die Geschichte des Eintrags (Bsp.: Spalte Z am 23.06.2012 durch Benutzer admin von -25 in +20 geändert) zur Verfügung.

Um dieses Kapitel abzuschliessen, sollen die Abhängigkeiten der beiden Schemata mit Hilfe der Abbildungen 3 und 4 auf der nächsten Seite nochmals grafisch aufgezeigt werden. Dabei wird deutlich, dass die Tabelle excavations die zentrale Rolle spielt und alle weiteren Tabellen mindestens über den Schlüssel excavationid mit ihr verknüpft sind. Weiter fällt auf, dass im Schema def keine weiteren Verknüpfungen unter den Tabellen zu sehen sind. Da diese Tabellen verschiedene Definitionen der unterschiedlichen Typen (Linien, Phasen, Layer, etc.), die definitiv nicht voneinander abhängen, enthalten, ist es nicht notwendig die Tabellen zu verknüpfen.

Dem Gegenüber steht nun das Schema elkowmgis. In Abbildung 4 sieht man deutlich, dass alle Tabellen, ausser der Tabelle excavations, untereinander verknüpft sind. Diese Verknüpfungen garantieren, dass gewisse Aktionen nur dann ausgeführt werden können, wenn sie sich auf alle Tabellen beziehen. Beispielsweise lässt sich ein Fund aus der Tabelle context nur dann löschen, wenn zuvor seine Koordinaten aus der Tabelle xyz gelöscht wurden. Umgekehrt lassen sich die Koordinaten eines Fundes erst eintragen, wenn in der Tabelle context der Fund eingetragen wurde. Durch diese Verknüpfungen kann somit ein unbeabsichtigtes Löschen oder ein Eintrag eines Objektes ohne **Metadaten** verhindert werden.

Abb. 3: Auszug aus dem Datenbankschema def

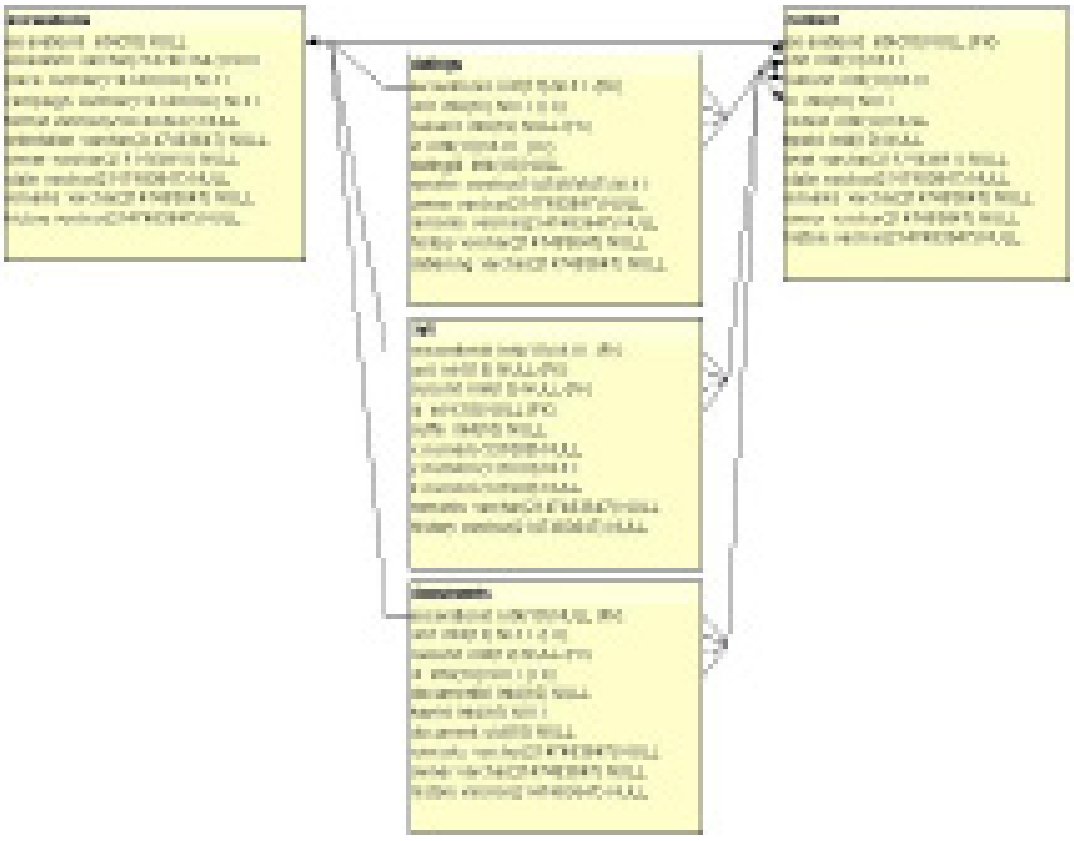


Abb. 4: Grafische Darstellung des Datenbankschemas elkowmgis

5 Installation und Deinstallation

5.1 Vorbereitende Massnahmen

Für die Nutzung von ElKowmGIS muss auf dem System, auf welchem es genutzt werden soll, mindestens eine Java-Laufzeitumgebung in der Version 1.7 oder höher installiert sein. Die aktuellste Version findet sich in der Literatur unter (Oracle 2013). Für die Nutzung von ElKowmGIS bedarf es keiner besonderen Einstellungen. Die Standard Java-Installation genügt allen Anforderungen.
ElKowmGIS nutzt als Datenbankserver einen PostgreSQL-Server. Die aktuellste Version finden Sie im Literaturverzeichnis unter (The PostgreSQL Global Development Group 2013b). ElKowmGIS wurde mit Version 9.0 entwickelt. Aus diesem Grund sollte die Version 9.0 oder höher genutzt werden. Für ElKowmGIS sind keine speziellen Einstellungen an der Installation des PostgreSQL-Servers vorzunehmen. Während der Installation werden Sie aufgefordert ein Passwort für den Benutzer “postgres“ anzugeben. Bitte vergeben Sie ein Passwort und merken sich dieses. Es wird für die Einrichtung und die Erstanmeldung zu einem späteren Zeitpunkt benötigt.

Sollte bereits ein PostgreSQL-Server auf Ihrem System installiert sein, oder möchten Sie einen Server über das Netzwerk verwenden, so müssen sie die Installation nicht auf Ihrem System vornehmen. ElKowmGIS ist standardmässig für eine lokale Installation eingerichtet. Diese Einstellung wird bei der ersten Anmeldung abgefragt und kann geändert werden. Ebenfalls standardmässig eingestellt ist der Port 5432 für den Server. Diese Nummer des Ports entspricht der Standardeinstellung des Datenbankservers. Falls Sie die Port-Nummer ändern, müssen Sie die geänderte Einstellung ElKowmGIS bei der Anmeldung mitteilen. Weitere Informationen hierzu finden sich im Kapitel Anmeldung.

5.2 Installation

Die Installation von ElKowmGIS gestaltet sich einfach. Extrahieren Sie die Datei ElKowmGIS.zip an einem beliebigen Ort auf Ihrer Festplatte. Wichtig hierbei ist es, dass die Ordnerstruktur erhalten bleibt, da ElKowmGIS sonst nicht lauffähig ist. Durch das Extrahieren wird ein neuer Ordner ElKowmGIS angelegt, in welchem sich die programmrelevanten Daten befinden.

5.3 Einrichten des Datenbankservers

Bevor Sie das Programm das erste Mal starten können, muss die Datenbank im Datenbankserver angelegt werden. Hierfür stehen Ihnen zwei Möglichkeiten zur Verfügung:

5.3.1 Automatische Einrichtung

Hierfür müssen Sie die Datei EG_createDB.jar ausführen. Hierbei handelt es sich um ein kleines Java-Programm, welches die Einrichtung der Datenbank übernimmt. Im sich öffnenden Fenster geben Sie bitte den Pfad zum PostgreSQL-Server, den Port des Servers (Standard: 5432) und das Passwort für den Benutzer postgres an.

Ein Klick auf die Schaltfläche “Ausführen“ startet die automatische Einrichtung des Servers. Hierbei wird als Erstes eine neue Datenbank namens ElKowmGIS angelegt. Zusätzlich werden die zwei Benutzergruppen eg_users und eg_admin angelegt. In einem nächsten Schritt werden der Datenbank zwei sog. Schemata angefügt.
Das Schema „def“ enthält alle Tabellen, die zur Verwaltung von ElKowmGIS notwendig sind. Hierin finden sich insgesamt 12 Tabellen für Verwaltung der einzelnen Funde, Proben, Layer, Benutzer, usw. In diesen Tabellen sind auch die Vorlagen von ElKowmGIS gespeichert.

Das Schema „elkowmgis“ enthält insgesamt 5 Tabellen, die die Speicherung der Daten der einzelnen Projekte übernehmen. Zur Einrichtung der Datenbank sind diese Tabellen leer. Sie werden erst mit dem Anlegen eines Projektes gefüllt.

5.3.2 Manuelle Einrichtung

Möchten Sie die Einrichtung des Servers lieber manuell vornehmen, so empfiehlt sich folgende Vorgehensweise:

- Anlegen der Datenbank ElKowmGIS
- Anlegen des Schemas def
- Anlegen der 15 Definitionstabellen (vgl. Anhang B.1: Schema def)
- Anlegen des Schemas elkowmgis
- Anlegen der Datentabellen (vgl. Anhang B.2: Schema elkowmgis)
- Anlegen der Benutzerrolle eg_users
- Setzen der Berechtigungen für die Gruppe eg_users: SQL-Befehl
- Anlegen der Benutzerrolle eg_admin
- Setzen der Berechtigungen für die Gruppe eg_admin: SQL-Befehl
- Benutzer postgres der Gruppe eg_admin hinzufügen

Warnung:

Beachten Sie dabei bitte, dass die Strukturen der Datenbank, der Schemata und der Tabellen zwingend der Vorgabe in Anhang B: Datenbankspezifische Anhänge entsprechen muss. Dem weniger geübten Benutzer wird daher empfohlen, die automatische Einrichtung zu wählen. Andernfalls kann keine Garantie für die Lauffähigkeit, die Sicherung und die Genauigkeit der Daten übernommen werden!

5.4 Erster Programmstart

Nachdem Sie das Programm und die Umgebung eingerichtet haben, können Sie das erste Mal ElKowmGIS starten. Hierfür führen Sie bitte die Datei ElKowmGIS.jar per Doppelklick aus. Nun sollte ein Anmeldefenster (Abb. 5) erscheinen, in welchem Sie die Verbindungsparameter angeben können.

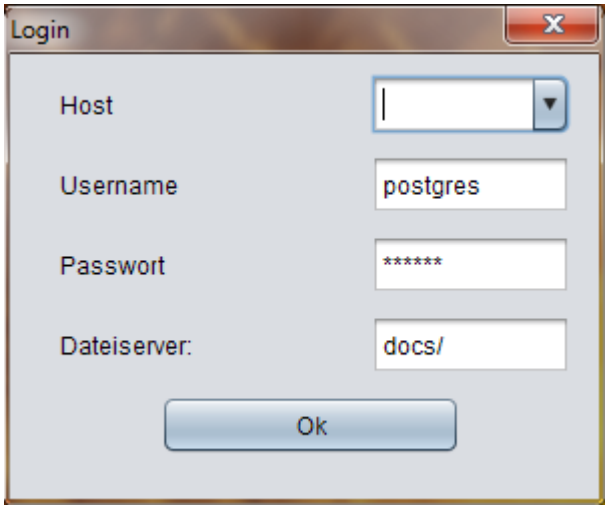


Abb. 5: Anmeldefenster von ElKowmGIS mit den Eingabefeldern für die Datenbank, den Benutzer und den Dateiserver

5.5 Deinstallation und Bereinigung des Servers

Falls Sie ElKowmGIS löschen und die Datenbank leeren möchten, so empfiehlt es sich, zuerst die Datenbank zu leeren und dann ElKowmGIS zu löschen. Möchten Sie ElKowmGIS lediglich von einem Rechner entfernen, so überspringen Sie bitte die Schritte 1 und 2.

1. Schritt: Leeren der Datenbank
2. Schritt: Löschen der Dokumente
3. Schritt: Entfernen von ElKowmGIS

Die einzelnen Schritte werden auf der nächsten Seite ausführlich erläutert.

Hinweis:

Es empfiehlt sich, vor dem Löschen eine Sicherung des kompletten Servers vorzunehmen. Hierzu schlagen Sie bitte im Handbuch Ihrer PostgreSQL-Installation nach (Stichwort **Dump**), wie Sie eine Sicherung durchführen können.

Schritt 1: Leeren der Datenbank

Um die Datenbank zu leeren, stellt ElKowmGIS das Programm EG_dropDB.jar zur Verfügung. Ähnlich der Einrichtung wird das Programm per Doppelklick ausgeführt. Sie müssen im sich öffnenden Fenster lediglich die Adresse zur Datenbank (inklusive des Ports) angeben (Bsp.: 192.168.1.25:5432 oder localhost:4914) und zusätzlich das Passwort für den Benutzer postgres. Durch Betätigen der Schaltfläche „Lösche DB“ wird auf dem Server die komplette Datenbank ElKowmGIS inklusive aller Schemata und Tabellen gelöscht. Ebenfalls entfernt werden die Benutzergruppen eg_users und eg_admin.

Warnung:

Die Benutzer, welche Sie via ElKowmGIS erstellt haben, werden durch das Programm EG_dropDB nicht gelöscht! Entweder löschen Sie die Benutzer vor dem Leeren der DB über ElKowmGIS oder nach dem Leeren der DB über die bordeigenen Mittel des Servers.

Schritt 2: Löschen der Dokumente

Um die Dokumente, welche mit ElKowmGIS erstellt oder verwaltet wurden, zu löschen, reicht es, den Ordner, welchen Sie als Dateiserver eingetragen haben, zu löschen.

Schritt 3: Entfernen von ElKowmGIS

Für die Deinstallation von ElKowmGIS gibt es keine Routine, da lediglich der Ordner ElKowmGIS von Ihrer Festplatte gelöscht werden muss. Da ElKowmGIS keinerlei Veränderungen am System oder der Registry Ihres Systems vornimmt, sind dadurch das Programm und alle seine Bestandteile gelöscht.

Warnung:

Im Ordner ElKowmGIS befindet sich der Ordner docs, welcher standardmässig als Speicherort für sämtliche Dokumente eingestellt wird. Sollten sich in diesem Ordner Dateien befinden, so empfiehlt es sich diesen Ordner vor dem Löschen zu sichern!

6 Das Programm

6.1 Anmeldefenster

Nach erfolgreicher Installation und dem Programmstart geht es nun daran ElKowmGIS das erste Mal zu nutzen. Das Anmeldefenster, wie wir es aus dem vorangegangenen Kapitel bereits kennen, verfolgt im Grunde genommen drei Ziele:

1. Auswahl des Servers

Das Feld Host (Abb. 6 Ziffer 1) dient zur Auswahl des Datenbankservers. Voreingestellt ist der Wert localhost. Localhost bezeichnet hierbei eine lokale Installation (auf dem gleichen Rechner, auf welchem auch ElKowmGIS läuft) des PostgreSQL-Servers. Möchten Sie einen anderen Server auswählen, so können Sie die IP-Adresse des Servers in die Liste eintragen. Falls Sie eine neue Adresse eintragen, wird diese in der Datei hosts.txt gespeichert und automatisch als Startwert ausgewählt. Wenn Sie möchten, können Sie mehrere Serveradressen auch direkt in die Datei hosts.txt eintragen und in ElKowmGIS nur noch die von Ihnen gewünschte Adresse auswählen.

Hinweis:

Falls Sie mit mehreren PostgreSQL-Installationen arbeiten möchten, so berücksichtigen Sie, dass auf allen Servern die Einrichtung wie in Abschnitt “5.3 Einrichten des Datenbankservers“ beschrieben durchgeführt worden sein muss. ElKowmGIS arbeitet standardmässig mit dem Port 5432 für den Datenbankserver. Dies ist der standardmässige Port, wie er auch von der PostgreSQL-Installation vorgeschlagen wird. Sollte Ihre PostgreSQL-Installation einen anderen Port als 5432 verwenden, so fügen sie die Portnummer via “:“ an die Adresse an (Bsp.: localhost:4914 oder 192.168.1.25:4914). Dadurch erkennt ElKowmGIS, dass der Port nicht dem Standardwert entspricht.

2. Anmeldung am Server

Im Feld Username (Abb. 6 Ziffer 2) können Sie den Benutzernamen eingeben, mit dem Sie sich anmelden möchten. Das Feld Passwort (Abb. 6 Ziffer 3) nimmt das Passwort für den Benutzer entgegen. Aus Sicherheitsgründen wird das Passwort bei der Eingabe nicht angezeigt.

Hinweis:

Bei der Erstanmeldung sind noch keine Benutzer angelegt, daher müssen Sie sich mit dem Benutzer postgres anmelden. Wie man weitere Benutzer für ElKowmGIS anlegt, erfahren Sie in Kapitel “6.12 Admin-Area“.

3. Auswahl des Dateiservers

Im letzten Feld, Dateiserver genannt, (Abb. 6 Ziffer 4) geben Sie den Speicherort für Dokumente, die mit ElKowmGIS verwaltet werden, an. Standardmässig wird der Ordner docs im Ordner ElKowmGIS verwendet. Falls der Ordner auf einen anderen Rechner oder FileServer verweist, müssen der Benutzername und das Passwort für diesen Speicherort gleich den Login-Daten auf dem Datenbankserver sein. Falls dies nicht der Fall sein sollte, muss zuerst eine Verbindung mit dem Ziel hergestellt werden, damit ElKowmGIS die Daten dort speichern kann. Selbstverständlich benötigen Sie die Berechtigung im Zielordner Dateien anzulegen. Falls Sie sich nicht sicher sind, welche Berechtigungen Sie besitzen, wenden Sie sich bitte an Ihren zuständigen EDV-Verantwortlichen.

Mit der Schaltfläche ok melden Sie sich am Server an und die Benutzeroberfläche von ElKowmGIS öffnet sich.

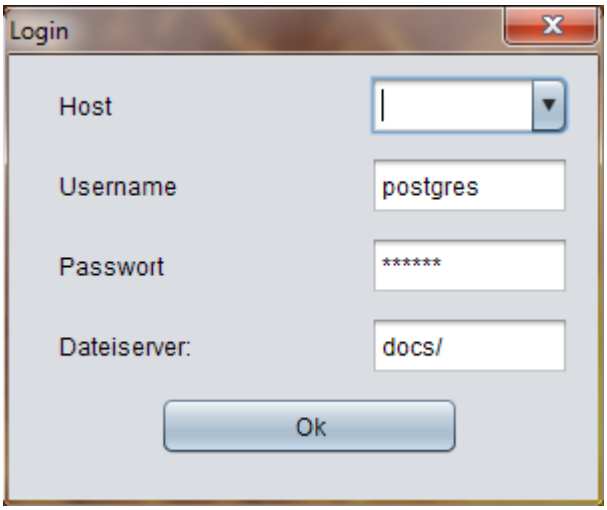


Abb. 6: Anmeldefenster von ElKowmGIS.

6.2 Die Benutzeroberfläche und Befehlsübersicht

Die Benutzeroberfläche von ElKowmGIS ist in die drei Bereiche Menu (Abb. 7 Ziffer 1), Seitenleiste (Abb. 7 Ziffer 2) und Anzeige (Abb. 7 Ziffer 3) gegliedert. Im Folgenden werden diese drei Bereiche detailliert beschrieben.

6.2.1 Menu-Leiste

Hier stehen Ihnen sämtliche Befehle zur Verfügung, die ElKowmGIS umfasst. Der Aufbau ist vergleichbar mit vielen Standardprogrammen. Ähnlich verhält es sich mit den Tastaturkürzeln. Hier wurden die allgemein üblichen Kürzel gewählt.

Name	Beschreibung	Tastaturkürzel
Menu Datei		
Neues Projekt	Öffnet das Dialogfenster zum Anlegen neuer Projekte	Strg+umschalt+N

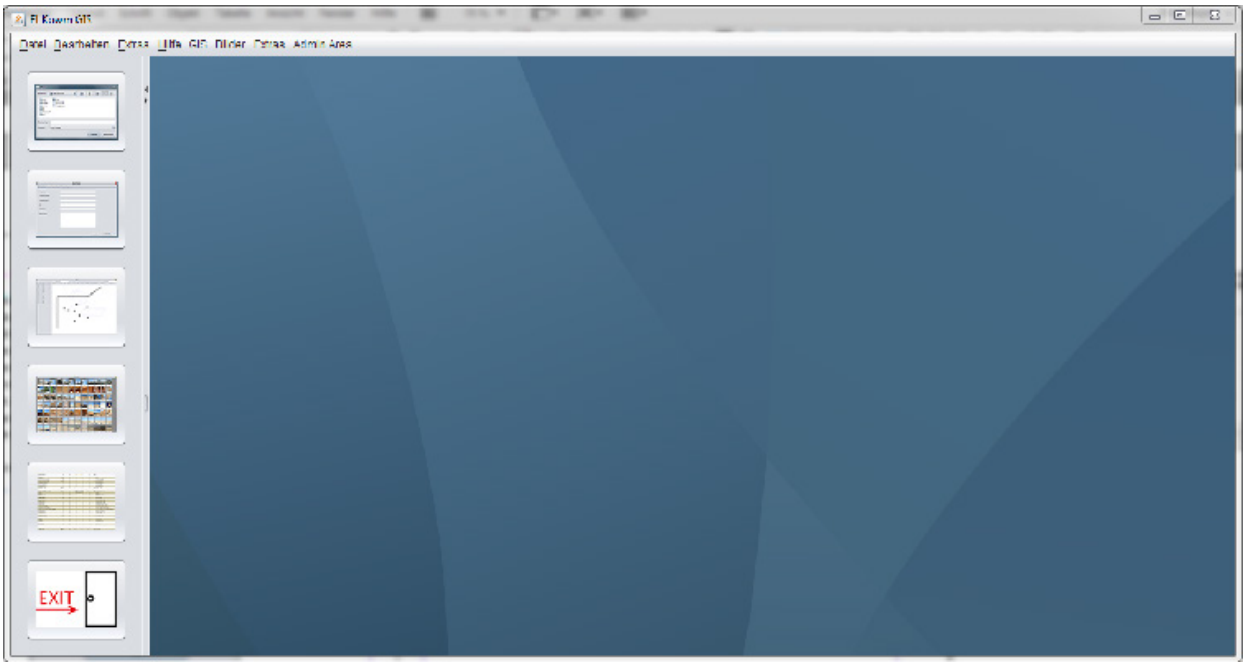


Abb. 7: Standardfenster von ElKowmGIS mit den Elementen Menu (1), Seitenleiste (2) und Anzeige (3)

Neu	Öffnet ein Dialogfenster zum Erstellen neuer Dokumente	Strg+N
Öffnen	Öffnet den Öffnen-Dialog zur Projektauswahl	Strg+O
Speichern	Speichert die Änderungen	Strg+S
Speichern unter	Öffnet den Export-Dialog	Strg+umschalt+S
Seiteneinstellung	Öffnet die Seiteneinstellungen	Strg+umschalt+P
Drucken	Öffnet den Druckdialog	Strg+P
Optionen	Öffnet die Programmeinstellungen	kein Kürzel
Beenden	Schliesst das Programm	Strg-Q
Menu Bearbeiten		
Ausschneiden	Schneidet das gewählte Objekt aus	Strg-X
Kopieren	Kopiert das gewählte Objekt	Strg-C
Einfügen	Fügt ein Objekt aus der Zwischenablage ein	Strg-V
Löschen	Löscht das gewählte Objekt	Entf
Alles Markieren	Markiert alle Einträge/Objekte	Strg-A
Suchen	Suchen innerhalb des Projektes	Strg-F
Ersetzen	Suchen und Ersetzen innerhalb des Projektes	Strg-H

Menu Hilfe

Hilfe	Öffnet dieses Handbuch als PDF	Alt-F1
Über ElKowmGIS	Informationen über das Programm	kein Kürzel

Menu GIS

GIS	Öffnet das GIS-/CAD-Fenster	kein Kürzel
Standpunkt	Öffnet ein Dialogfenster zur Bestimmung des Standpunkts	kein Kürzel
Geländemodell	Öffnet die Oberflächenmodellierung	kein Kürzel
Funddichte	Öffnet das Menu zur Berechnung der Funddichte	kein Kürzel

Menu Bilder

Bild	Öffnet die Bildverwaltung	kein Kürzel
Bild entzerren	Öffnet den Dialog zur Bildentzerrung / -referenzierung	kein Kürzel

Menu Extras

Importiere Tabelle	Importiert eine Tabelle im csv-Format	kein Kürzel
Importiere Bild	Öffnet die Bildverwaltung	kein Kürzel
Importiere Photoplan	Importiert ein Bild mit entsprechendem World-File aus Photoplan	kein Kürzel
Importiere DXF	Importiert eine DXF-Datei	kein Kürzel

Falls der angemeldete Benutzer Administrator ist, steht folgendes Menu zusätzlich zur Verfügung:

Menu Admin-Area

Verwaltung	Öffnet die Benutzer-/Projektverwaltung	kein Kürzel

6.2.2 Seitenleiste

Die Seitenleiste bietet einen Schnellzugriff auf die Funktionen „Projektauswahl“ (Abb. 8 Ziffer 1), „Neues Projekt“ (Abb. 8 Ziffer 2), „GIS“ (Abb. 8 Ziffer 3), „Bildverwaltung“ (Abb. 8 Ziffer 4) und „Tabellen“ (Abb. 8 Ziffer 5). Ferner befindet sich in dieser Leiste ein Beenden-Knopf (Abb. 8 Ziffer 6). Während der Benutzung verändert sich ihr Aussehen, da die Steuerelemente der Module in der Seitenleiste eingeblendet werden.

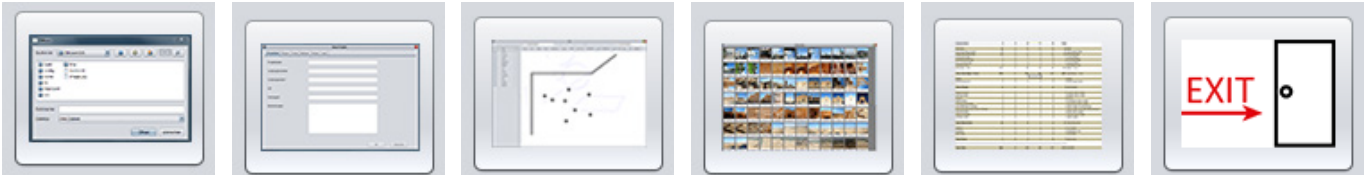
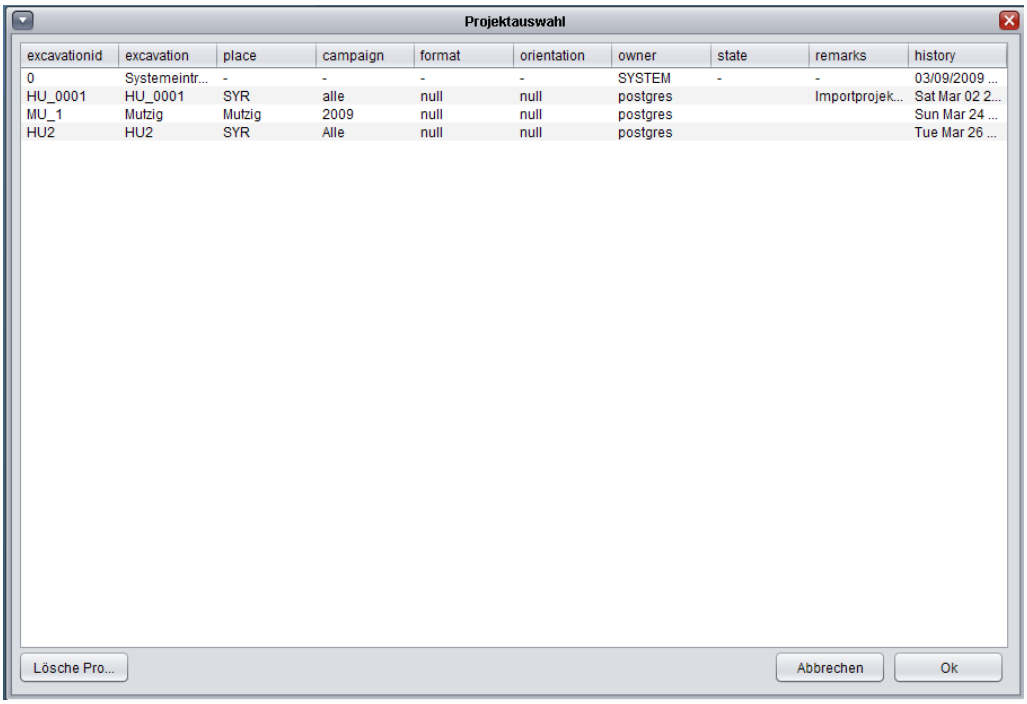


Abb. 8: Elemente der Seitenleiste von ElKowmGIS

6.2.3 Öffnen eines Projektes

Icon:  **Shortcut:** Strg+O **Menu:** Datei -> Öffnen

Fenster:



Funktion:

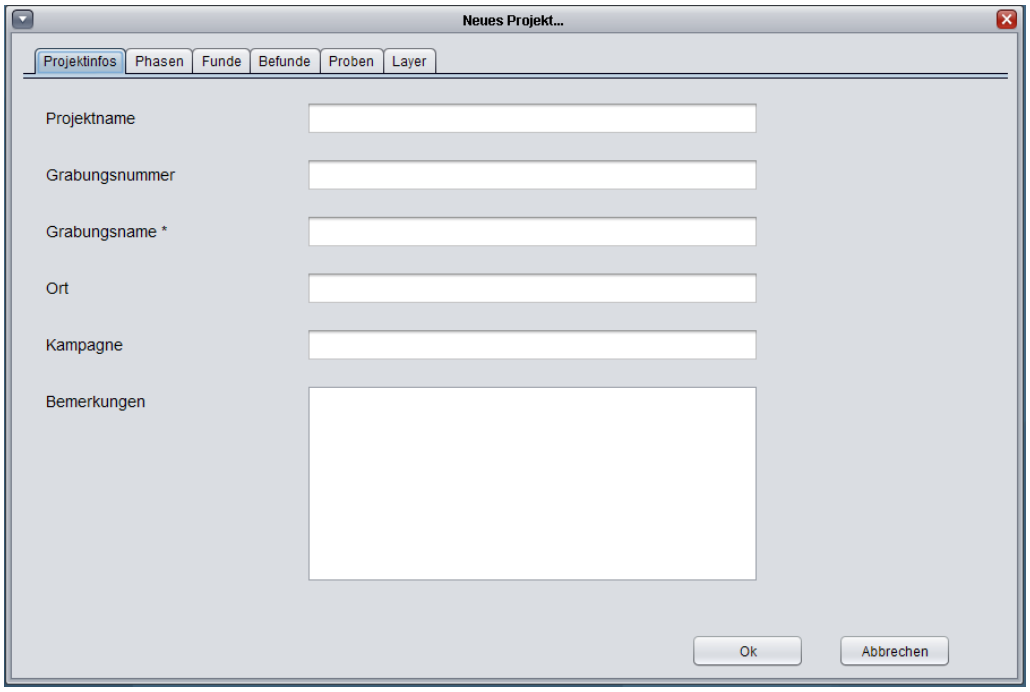
Durch Aufrufen der Projektauswahl öffnet sich ein Dialogfenster in der Anzeige, welches die in der Datenbank vorhandenen Projekte auflistet, für die der aktuell angemeldete Benutzer mindestens eine Leseberechtigung verfügt. In der Tabelle stellt jede Zeile ein eigenes Projekt dar. Die Spalten repräsentieren die Informationen über das jeweilige Projekt. Als Informationen angezeigt werden die Grabungsnummer, der Grabungsname, der Ort, die Kampagne, das Format der Laufnummer, die Orientierung, der Besitzer, der Status des Projektes, Bemerkungen und das Erstelldatum.
Den Administratoren wird zusätzlich eine Schaltfläche zum Löschen des ausgewählten Projektes angezeigt.

Ein Projekt kann durch das Auswählen der entsprechenden Zeile innerhalb der Tabelle und abschliessendem Drücken des Ok-Buttons geöffnet werden.

6.2.4 Anlegen eines neuen Projektes

Icon:  Shortcut: Strg+Umschalt+N Menu: Datei -> Neues Projekt

Fenster:



Funktion:

Im sich öffnenden Fenster können sämtliche Projektangaben für ein neues Projekt eingetragen werden. Hierfür stehen insgesamt 7 Karteikarten für die unterschiedlichen Angaben zur Verfügung.

Die erste Karte enthält Eingabemöglichkeiten für die allgemeinen Informationen des Projektes. Die Felder Grabungsnummer, Grabungsname, Ort und Kampagne müssen zwingend ausgefüllt werden. Zur Übersichtlichkeit der Projekte wird empfohlen, auch das Feld Projektname auszufüllen. Beim Feld Bemerkungen handelt es sich um ein Freitext-Feld, welches weiterführende Informationen zum Projekt enthalten kann.

Der Reiter Phasen enthält eine Tabelle zur Eingabe von sog. Phasen. Der Begriff Phase bezieht sich hierbei nicht zwingend auf eine absolute zeitliche Definition. Vielmehr kann eine Phase auch eine Bauphase o.ä. bezeichnen. Der Begriff Phase wurde in diesem Fall gewählt, da der Begriff einen Abschnitt oder eine Stufe innerhalb einer stetig verlaufenden Entwicklung oder eines zeitlichen Ablaufs (Drosdowski and Klosa 1996) beschreibt. Diese Beschreibung erschien als neutrale Lösung für die Beschreibung einer Entwicklung innerhalb einer Fundstelle. Für eine absolute Datierung steht in ElKowmGIS eine eigene Möglichkeit bei der Eingabe von Objekten zur Verfügung. Die erste Spalte der Tabelle enthält die Bezeichnungen der

einzelnen Phasen. Die nächsten beiden Spalten geben an, von wann bis wann eine Phase dauert. Diese Felder müssen nicht zwingend ausgefüllt sein. Die Spalte Epoche bietet eine Auswahl an vordefinierten Begriffen zur groben Einteilung an. Die Epochenbegriffe können in den Programmoptionen geändert werden. Zur Vervollständigung der Phasenangabe bietet die Tabelle die Spalte Kultur an. Hier kann eine allfällige Zugehörigkeit zu einer bestimmten Kulturgruppe eingetragen werden.

Die Reiter Funde, Befunde, Proben und Layer sind identisch aufgebaut. Alle enthalten eine neue Sammlung von Karteikarten für die Definition der einzelnen Kategorien. Für jede Kategorie sind die gleichen Felder auszufüllen. Das erste Feld enthält den Namen der Kategorie (z.B.: Silex, Graben, etc.). Das zweite Feld gibt die dazugehörige Abkürzung an. Ein Klick auf den farbigen Button öffnet die Farbauswahl, mit der die Darstellungsfarbe bestimmt werden kann. Die beiden Gruppen Darstellungssymbol und -grösse, sowie der Linientyp und die -stärke, bieten die Möglichkeit, das Symbol für die Darstellung als Punkt (Darstellungssymbol und -grösse) oder die Linieneigenschaften der jeweiligen Kategorie zu ändern. Über das Feld Layer kann ausgewählt werden, auf welchem Layer die Kategorie im GIS-Modul dargestellt wird. Die Liste Status auf der rechten Seite enthält Begriffe, die die jeweilige Kategorie noch weiter beschreiben können. Einträge können aus der Liste entfernt werden, indem sie einzeln markiert werden und dann das Minus-Symbol oben links angeklickt wird. Um Einträge hinzuzufügen, klicken Sie auf das Plus-Symbol oberhalb der Liste. Dadurch öffnet sich ein Fenster, indem entweder eine vordefinierte Beschreibung aus

Hinweis:

ElKowmGIS beinhaltet bereits komplette Sätze fertiger Einträge. Diese dienen zum einen als Beispiel, zum anderen sollen sie als Vorschläge für eine Vereinheitlichung der Dokumentation gesehen werden. Durch eine Vereinheitlichung der Daten bereits bei der Aufnahme kann verhindert werden, dass Datensätze nachträglich bearbeitet oder standardisiert werden müssen.

der Datenbank geladen wird oder eine neue Beschreibung erstellt wird. Eine neu erstellte Beschreibung wird per Klick auf die Schaltfläche hinzufügen automatisch in der Datenbank gesichert.

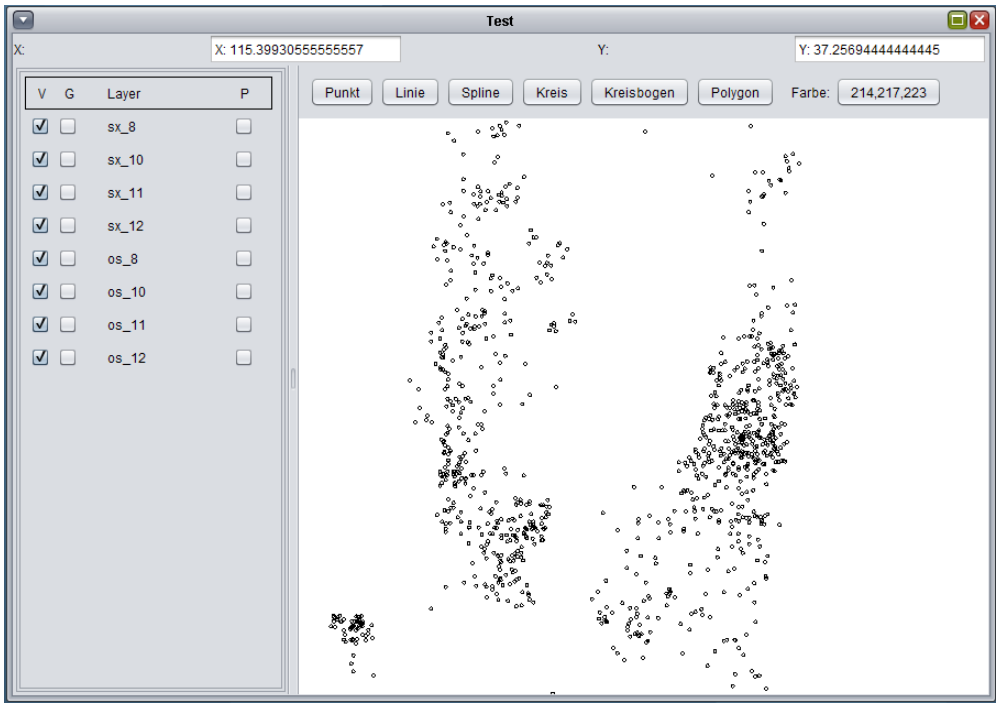
Im letzten Reiter lassen sich die Darstellung der Laufnummer und die Art der Aufnahme der Orientierung von Objekten einstellen. Für die Laufnummer steht ein einfaches Textfeld zur Verfügung, in welches die Form der Laufnummer eingetragen werden kann. Hierfür stehen die Kürzel $g für die Grabungsnummer, $u für die sog. Unit, $s für die subunit und $i für die Laufnummer des Objektes. Die Orientierung lässt sich auf die drei unterschiedlichen Weisen “Orientierung nicht berücksichtigt“, “Orientierung wird manuell bestimmt“, “Orientierung wird über 2 Punkte aufgenommen“ sichern. Die Orientierungsmessung über zwei Punkte (McPherron 2005) erfolgt hierbei über die Aufnahme von zwei Punkten auf dem Objekt. Anschliessend berechnet ElKowmGIS die Orientierung.
Bei Klick auf die Schaltfläche Ok prüft ElKowmGIS, ob die Felder Grabungsnummer, Grabungsname, Ort und

Kampagne ausgefüllt wurden. Sollte dies der Fall sein, speichert das Programm die Informationen in der Datenbank und wählt das neue Projekt als aktuelles Projekt aus. Andernfalls erscheint eine Warnmeldung, welches Feld nicht ausgefüllt wurde und das Programm kehrt zum Dialogfenster zurück.

6.2.5 Das kombinierte GIS- und CAD-Modul

Icon:  **Shortcut:** - **Menu:** GIS -> GIS

Fenster:



Funktion:

Das Modul GIS umfasst Funktionen und Routinen zur Darstellung der geografischen Informationen von Objekten und Dokumenten. Im sich öffnenden Fenster findet sich über der Zeichenfläche eine zusätzliche Menuleiste sowie eine Positionsangabe (Abb. 9). Die Positionsangabe gibt die X- und Y-Koordinaten der Position des Mauszeigers auf der Karte an. In der Menuleiste finden Sie Schaltflächen zum Einzeichnen

Hinweis:

Snapping bedeutet, dass der Anfangspunkt eines zu zeichnenden Objektes auf den nächsten bereits bestehenden Punkt eines geografischen Objektes gesetzt wird. Dabei werden alle Punkte innerhalb eines bestimmten Radius um die Kartenposition des Mauszeigers gesucht. Der Punkt, der am nächsten zum Mauszeiger liegt, wird als Anfangspunkt verwendet. Der Snapping-Radius kann im Optionsmenu von ElKowmGIS verändert werden.

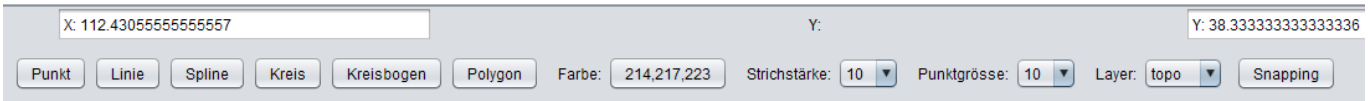


Abb. 9: Ausschnitt aus dem GIS-Fenster mit Koordinatenanzeige und zusätzlicher Menuleiste.

von Punkten, Linien und weiteren Objekten. Ferner können hier die Darstellungsoptionen der Objekte verändert werden. Des Weiteren befindet sich in der Leiste die Schaltfläche, um das sog. Snapping ein oder auszuschalten.

Die Navigation im Kartenfenster kann entweder mittels der Tastatur oder der Maus erfolgen. Um den Kartenausschnitt mittels der Tastatur zu verschieben dienen die Pfeiltasten. Um den Ausschnitt in grösseren Sprüngen zu verschieben, verwenden Sie die Taste Bild auf, resp. Bild ab, um das Bild vertikal zu verschieben. Eine horizontale Verschiebung erfolgt, wenn Sie gleichzeitig die STRG-Taste mit der Bild auf-Taste oder Bild ab-Taste gedrückt halten. Die Zoomfunktionen befinden sich auf den Tasten + und -.

Die Steuerung mit der Maus orientiert sich an den gängigen GIS- und CAD-Lösungen. Der Kartenausschnitt wird durch gedrückt halten der mittleren Maustaste und gleichzeitigem Bewegen der Maus verschoben. Das Zoomen erfolgt durch Drehen des Mausrades. Die Steuerung mit der Maus umfasst auch das Auswählen von Objekten. Hierfür klicken Sie mit der linken Maustaste in einen beliebigen Bereich der Karte. Wenn Sie nun den Mauszeiger bewegen, erscheint zwischen dem ersten Punkt und dem Mauszeiger ein eingefärbtes, transparentes Viereck. Je nach Richtung, in die Sie den Mauszeiger bewegen, ändert das Viereck die Farbe. Bewegt sich der Mauszeiger rechts vom ersten Punkt, so ist das Viereck blau eingefärbt. Links vom ersten Punkt ist das Viereck grünlich eingefärbt. Die Farbe dient zur optischen Unterscheidung der Auswahlmethode. Beim blauen Viereck werden lediglich diejenigen Elemente ausgewählt, die vollständig von dem Viereck umschlossen sind (Abb. 10 rechts). Das grüne Viereck zeigt an, dass alle

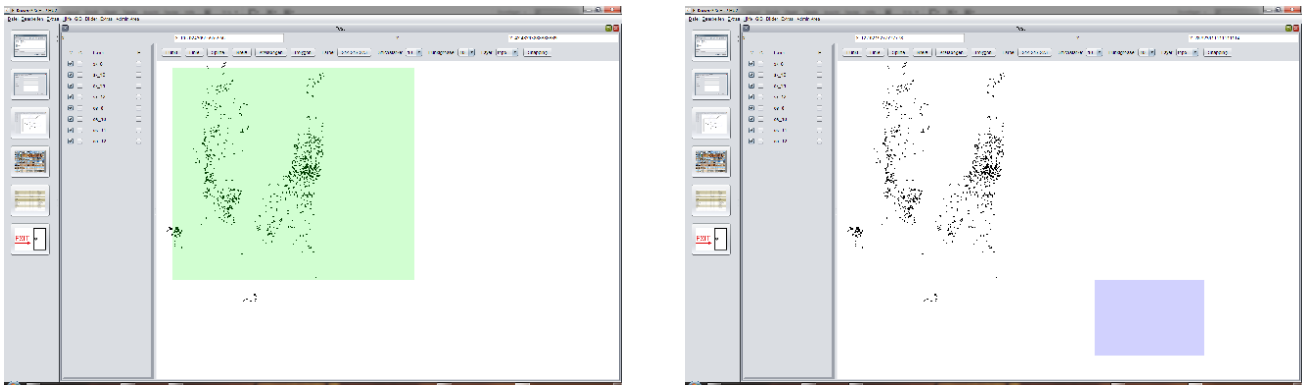


Abb. 10: Auswahlmethoden in ElKowmGIS: links die Auswahl, in der alle berührten Elemente gewählt werden. Im rechten Beispiel werden lediglich die vom Viereck umschlossenen Objekte ausgewählt.

Elemente ausgewählt werden, die vom Viereck umschlossen oder geschnitten werden (Abb. 10 rechts). Ein zweiter Klick mit der linken Maustaste wählt die betroffenen Objekte aus. Die Auswahl lässt sich mittels eines Klicks mit der rechten Maustaste aufheben. Markierte Objekte werden in einer anderen Farbe als die nicht ausgewählten Objekte dargestellt. Die Farben der Auswahl und der Vierecke können im Optionsmenu umgestellt werden.

Gleichzeitig werden in der Seitenleiste die Anzahl und die Objekttypen der ausgewählten Objekte angezeigt. Die Liste am oberen Rand des Seitenfensters erlaubt es, gezielt Informationen zu bestimmten Objektarten (Punkt, Linie, etc.) auszuwählen. Sind mehrere Objekte gleichzeitig ausgewählt, so zeigt ElKowmGIS in den Eigenschaftsfeldern den Wert „Mehrfachauswahl“. Ist nur ein Objekt ausgewählt, so werden in den Eigenschaftsfeldern die Eigenschaften des ausgewählten Objektes angezeigt.

Um ein neues Objekt in die Karte einzutragen stehen Ihnen im linken Teil der Menuleiste insgesamt 6 Schaltflächen zur Verfügung. Sie können zwischen den geometrischen Objekten Punkt, Linie, Spline, Kreis,

Warnung:

Mit ElKowmGIS ist es nicht möglich, Hilfslinien oder –punkte zu erstellen, um zu verhindern, dass Objekte in der Datenbank gesichert werden, die keinerlei Zuweisungen oder Identifikation besitzen. Möchten Sie dennoch Hilfslinien und Punkte nutzen, empfiehlt es sich, einen eigenen Objekttyp (beispielsweise „Hilfsobjekt“ genannt) in den Projekteigenschaften zu definieren. Zusätzlich wird empfohlen, einen eigenen Layer für Hilfslinien etc. anzulegen. Dadurch werden die Objekte zwar in der Datenbank gesichert und erhalten auch eine eigene Laufnummer, sie sind aber deutlich von anderen Objekten zu unterscheiden. Es ist ebenfalls ratsam, eine eigene unit resp. subunit für Hilfszeichnungen anzulegen. Dadurch können diese bei der Suche ausgeblendet werden.

Kreisbogen und Polygon wählen. Um einen Punkt zu zeichnen, wählen sie die Schaltfläche Punkt. Unter dem Mauszeiger erscheint das aktuelle Punktsymbol als Vorschau. Bewegen Sie den Mauszeiger an die Stelle, an die der Punkt gesetzt werden soll. Klicken Sie anschliessend die linke Maustaste. Nun erscheint ein Eigenschaftenfenster, welches Sie vollständig ausfüllen müssen, um den Punkt in die Zeichnung einzutragen.

Die restlichen Objekte werden prinzipiell gleich gezeichnet, wobei hier mehr Punkte erforderlich sind.

Für eine Linie wählen sie die Schaltfläche Linie und bewegen den Mauszeiger an die Stelle, an der die Linie beginnen soll. Auch hier erscheint die Vorschau des Punktsymbols unter dem Mauszeiger, bevor der erste Punkt gesetzt ist. Setzen Sie den ersten Punkt, so zeigt ElKowmGIS beim Bewegen der Maus eine Vorschau der Linie auf der Karte. Ein zweiter Klick zeichnet die Linie und öffnet das Eigenschaftenfenster.

Um eine gebogene Linie zu zeichnen, steht die Schaltfläche Spline zur Verfügung. Um die Linie zu zeichnen,

benötigen Sie den Anfangspunkt, einen Mittelpunkt und einen Endpunkt. Sobald Sie den Anfangs- und den Mittelpunkt gesetzt haben, sehen Sie eine Vorschau der Linie. Der dritte Punkt beendet die Linie.

Beim Zeichnen eines Kreises müssen Sie insgesamt zwei Punkte in der Karte wählen. Der Anfangspunkt ist ein beliebiger Punkt auf dem Kreis. Den zweiten Punkt des Kreises können Sie über die Vorschau auswählen.

Für einen Kreisbogen werden insgesamt 3 Punkte benötigt. Der erste Punkt repräsentiert den Anfangspunkt des Bogens, während der Zweite einen beliebigen Punkt auf dem Kreisbogen zwischen dem Anfangs- und dem Endpunkt des Bogens definiert. Der dritte Punkt bildet den Endpunkt des Bogens.

Während die vorhergehenden geometrischen Objekte alle mit einem einfachen Klick beendet wurden, bildet das Zeichnen eines Polygons eine Ausnahme. Das Polygon wird im Prinzip aus einzelnen Linien zusammengesetzt, wobei der Startpunkt einer Linie gleich dem Endpunkt der vorherigen Linie gleicht. Um ein Polygon zu zeichnen gehen Sie wie folgt vor:

- Wählen Sie die Schaltfläche Polygon und bewegen Sie den Mauszeiger in die Karte. Es erscheint wiederum die Punktvorschau unter dem Mauszeiger.
- Setzen Sie den Anfangspunkt auf die gewünschte Position. Ab sofort wird eine Linie als Vorschau angezeigt. Die Linie erstreckt sich hierbei vom zuletzt gesetzten Punkt bis zur Position des Mauszeigers.
- Sobald Sie alle Seiten des Polygons gezeichnet haben, führen Sie einen Doppelklick mit der linken Maustaste aus. Dadurch erkennt ElKowmGIS, dass das Polygon vollständig gezeichnet wurde und zeichnet automatisch eine Linie vom letzten geklickten Punkt zum Anfangspunkt des Polygons. Dieser Schritt wird hier eingefügt, da Polygone geschlossene Objekte sein müssen. Um zu verhindern, dass das Polygon nicht geschlossen wird, wird die letzte Verbindungslinie automatisch gezogen.

Bei allen Zeichenobjekten gilt: Solange der Zeichenvorgang aktiv ist (d.h. das Eigenschaftenfenster noch nicht aufgerufen wurde) setzt ein Rechtsklick die Zeichnung zurück und unter dem Mauszeiger erscheint wieder das Punktsymbol.

Ein Umschalten der Geometrie durch Drücken einer der restlichen Schaltflächen ist nicht möglich. Sollten Sie die falsche Geometrie ausgewählt und bereits zu zeichnen begonnen haben, so setzen Sie bitte die Zeichnung mittels der rechten Maustaste zurück. Danach können Sie eine andere Geometrie auswählen und zeichnen.

Das Ausschalten der aktivierten Zeichenschaltfläche (Punkt, Linie, etc.) versetzt das Kartenfenster wieder in den Navigationsmodus.

6.2.6 Erstellen von Geländemodellen

ElKowmGIS besitzt die Funktion, Geländemodelle aus Punkten aus der Datenbank zu erstellen. Hierzu

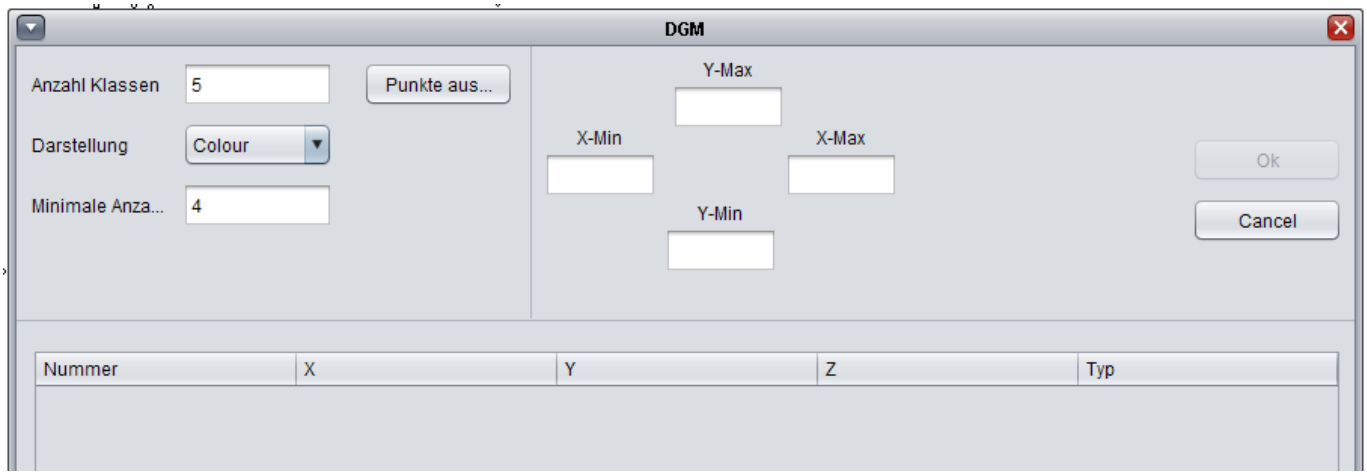


Abb. 11: Dialogfenster zur Erstellung eines digitalen Geländemodells (DGM) in ElKowmGIS.

klicken Sie im Menu GIS auf den Eintrag Geländemodell. Hinweis: Sie müssen sich in einer aktiven GIS-Sitzung befinden um dieses Menu aufrufen zu können.

Es öffnet sich das Dialogfenster (Abb. 11) mit den Angaben der Variablen. Die Anzahl der Klassen gibt an, in wie viele Intervalle der Wertebereich unterteilt wird. Dies dient der Färbung der Karte nach dem sie berechnet wurde. Die Auswahl der Darstellung umfasst die beiden Einträge Colour und Grey. Mit der Einstellung Colour erhalten Sie ein eingefärbtes Geländemodell, mit Grey ein Geländemodell in Graustufen. Die minimale Anzahl an Punkten gibt an, wie viele Punkte sich im Suchradius befinden sollen.

Die Schaltfläche Punkte auswählen führt Sie ins GIS-Fenster, in welchem Sie nun die Punkte, die zur Generierung des Geländemodells dienen sollen, auswählen können. Hierzu ziehen Sie einfach ein Rechteck um alle Punkte, die das Modell enthalten soll. Hierzu klicken Sie mit der Maus auf einen Eckpunkt des gewünschten Rechtecks, ziehen es mit der Maus auf und klicken auf die gegenüberliegende Ecke des Rechtecks. Mit dem zweiten Klick öffnet sich wieder das Dialogfenster und die Werte X-Min, X-Max, Y-Min, sowie Y-Max sind eingetragen. Mit diesen Werten können Sie die Region anpassen, für die das Modell berechnet wird. Es empfiehlt sich jedoch, die voreingestellten Werte zu belassen, da die Werte die maximale Ausdehnung der Punktwolke darstellen. Eine Vergrösserung der Region würde bewirken, dass das Modell an den Rändern falsch dargestellt wird, da dort keine Informationen zur Verfügung stehen. Eine Verkleinerung der Region hingegen stellt kein Problem dar. Die Tabelle im unteren Teil des Dialogs zeigt nun die Nummer, die Koordinaten und den Typ der ausgewählten Punkte an. Dies kann hilfreich sein, um zu überprüfen, ob die gewählten Punkte sich zur Modellierung eignen. Mit einem Klick auf Ok wird das Modell berechnet, der Karte hinzugefügt und als Bild im Ordner “img“ auf dem FileServer abgelegt. Gleichzeitig werden die benötigten Datenbankeinträge vorgenommen. Dabei wird der Wert der id automatisch vergeben. Als Wert für die Unit wird der Wert dokument gesetzt.

6.2.7 Erstellen von Funddichten

Das Dialogfenster zum Erstellen einer Funddichte erreichen Sie über den Befehl Funddichte. Der Aufbau des

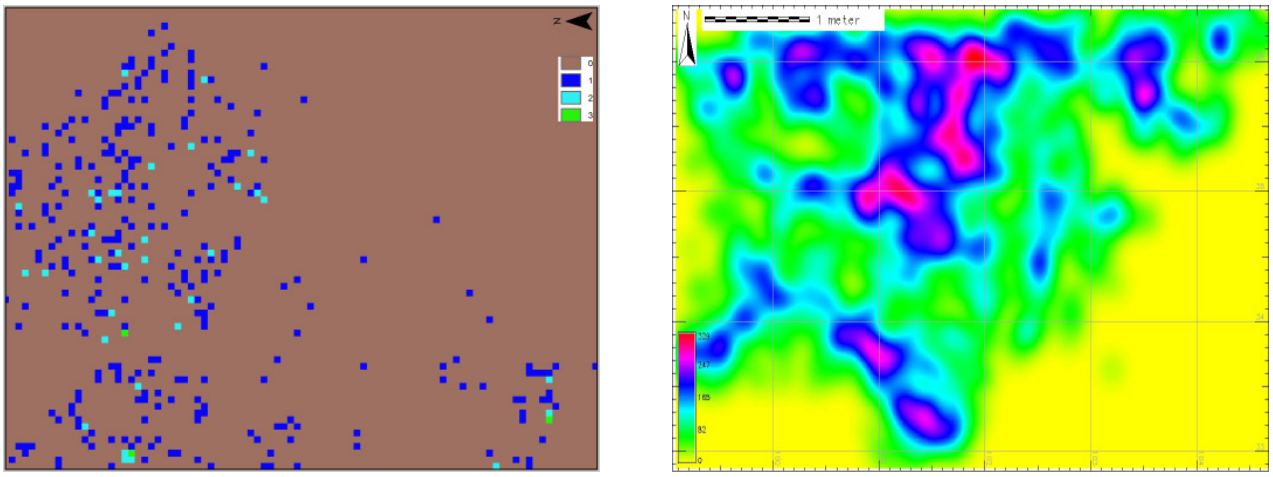


Abb. 12: Darstellung der Funddichte in klassischer Form (links) und in interpolierter Form (rechts).

Fensters ist ähnlich dem Fenster für das Geländemodell. Der Unterschied besteht in der untersten Auswahl. Im Falle der Funddichte geben Sie hier an, ob eine klassische Variante (Abb.12) oder eine interpolierte Variante gezeichnet werden soll. Das restliche Verfahren ist gleich der Erstellung eines Geländemodells.

6.2.8 Kommunikation mit einem Tachymeter

ElKowmGIS unterstützt auch die Vermessung mittels Tachymeter. Bevor Sie jedoch messen können, muss zuerst ein Standpunkt definiert werden. Dies geschieht in ElKowmGIS mittels einer freien Stationierung über drei Passpunkte. Hierbei müssen nacheinander drei Passpunkte anvisiert und gemessen werden. Die Koordinaten der Passpunkte müssen bekannt sein, jedoch müssen Sie nicht als Punkte in der Karte vorhanden sein. Um die Stationierung aufzurufen, rufen Sie bei geöffneter Karte den Menupunkt Standpunkt auf. Im sich öffnenden Fenster können Sie nun entweder die Angaben manuell eintragen oder einen Punkt von der Karte wählen, indem Sie auf die Schaltfläche Von Karte -> klicken und den Punkt auswählen. Nach erfolgter Auswahl werden die Koordinaten und die Punktnummer in den dafür vorgesehenen Feldern angezeigt. Anschliessend rufen Sie das Messfenster über die Schaltfläche Messe Punkt auf. Im Messfenster stehen Ihnen nun zwei Möglichkeiten zur Verfügung. Die erste Möglichkeit stellt die Messung via Tachymeter dar. Hierzu müssen sie lediglich die richtigen Geräteeinstellungen auswählen.

Warnung:

ElKowmGIS arbeitet, wie in der Vermessungstechnik üblich, bei Winkelangaben in Gon! Überprüfen Sie daher die Einstellungen in Ihrem Gerät, falls Sie sich nicht sicher sind, in welcher Einheit Ihr Gerät arbeitet.

Andere Einstellungen können der Datei edms.txt im Ordner config hinzugefügt werden. Der com-Port wird vom Betriebssystem bestimmt und gibt an, wo Ihr Gerät mit dem Computer verbunden ist. Als Nächstes geben Sie noch die Instrumenten- und Prismenhöhe manuell ein und betätigen den Auslöseknopf. ElKowmGIS fordert dann vom Tachymeter die benötigten Werte an.

Sollte es nicht möglich sein, das Gerät mit dem Computer zu verbinden oder Sie arbeiten nicht mit einem Tachymeter, so können Sie die benötigten Werte auch von Hand eingeben. Hierfür wählen Sie lediglich die zweite Option Manuelle Eingabe aus und füllen die Felder aus.

Mit Ok bestätigen Sie die Messwerte und kehren zum Stationierungsfenster zurück. ElKowmGIS hat nun die Tabelle im unteren Teil des Fensters um einen Eintrag ergänzt. Sobald Sie 2 oder mehr Punkte eingemessen haben, beginnt ElKowmGIS mit der Berechnung des Standpunktes. Dies erkennen Sie auch

Hinweis:

Die Toleranz findet sich unter dem Wert dS in der Datei config.properties. Sie ist standardmässig auf 1 cm eingestellt.

daran, dass die Felder für die Koordinaten des Standpunktes ausgefüllt sind. Ab 3 Punkten erscheint in den Feldern “Gen. X“ und“Gen. Y“ auch der jeweilige Fehlerwert. Das Textfeld und die zwei Felder in der Mitte des Fensters geben den Zustand der Stationierung an. Das Textfeld kann die Werte Stationierung nicht bestimmt (nicht genügend Punkte gemessen), Stationierung bestimmt (3 Punkte gemessen) oder Stationierung überbestimmt (mehr als 3 Punkte gemessen) annehmen. Sollten Sie bereits 3 oder mehr Punkte eingemessen haben, die Anzeige aber weiterhin auf Stationierung nicht bestimmt stehen, so ist die berechnete Abweichung grösser als die von Ihnen eingestellte Toleranz. In diesem Fall hilft es, entweder einzelne Messungen zu korrigieren oder weitere Punkte aufzunehmen.

Um eine Messung zu korrigieren können Sie einfach den gewünschten Punkt auf der Karte nochmals auswählen oder nochmals die Informationen eingeben und die Messung erneut durchführen. ElKowmGIS korrigiert die Standpunktberechnung selbstständig. Um einen Punkt aus der Liste zu löschen, wählen Sie ihn in der Tabelle aus und klicken auf die Schaltfläche Löschen. Haben Sie die gewünschte Genauigkeit erreicht, so können Sie das Fenster mit Ok schliessen. ElKowmGIS speichert die Standpunktinformationen für diese Sitzung und im Menu GIS ist die Option Messen verfügbar.

6.2.9 Messen von Objekten

Um mit ElKowmGIS Objekte einzumessen, klicken Sie nach erfolgreicher Stationierung auf das Menu Messen. ElKowmGIS öffnet damit ein Fenster in dem Sie die Einstellungen für den Punkt vornehmen können. Hierzu gehören die Attributangaben, wobei ElKowmGIS überprüft ob das Objekt nicht schon in der Datenbank vorhanden ist. Ausserdem können Sie eine Prismenhöhe mit eingeben. Dies ist notwendig, wenn die Messung nicht direkt auf das Objekt erfolgen kann und/oder standardmässig ein Prisma im Einsatz ist. Über die Auswahl der Geometrie und je nach Einstellung der Orientierung können Sie nun eine oder mehrere Messungen durchführen um das Objekt einzumessen. Nach Abschluss der Messungen klicken Sie auf Daten übertragen und ElKowmGIS überträgt das Objekt in die Datenbank und die Karte. Das

Optionsfeld in der untersten Zeile des Fensters dient dazu mehrere Objekte nacheinander einzumessen. Ist es aktiviert, so öffnet sich das Messfenster nach der Übertragung der Daten erneut mit denselben Einstellungen wie zuvor. Lediglich die ID wird um eins erhöht.

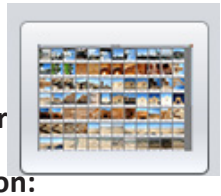
6.2.10 Bilder referenzieren

Bei geöffneter GIS-Sitzung erreichen Sie über das Menu Bild entzerren das Dialogfenster für eine Georeferenzierung. Sie werden zuerst aufgefordert, ein Bild auszuwählen. Dieses Bild wird Ihnen dann in einer Vorschau angezeigt, in der Sie sich mit den Schaltflächen +, -, <, v, > und ^ bewegen können. Die Schaltflächen + und – vergrössern oder verkleinern das Bild, während die übrigen vier Schaltflächen den sichtbaren Bildausschnitt verschieben. Um den Bildausschnitt zu verschieben können Sie alternativ auch die Scroll-Balken am Rand des Bildes bewegen. ElKowmGIS benötigt zur Referenzierung von Bildern mindestens 5 Punkte, die sowohl im Bild wie auch in der Karte markiert werden. Dabei wird zuerst der Punkt im Bild ausgewählt, anschliessend schliesst sich das Fenster, der entsprechende Punkt wird auf der Karte ausgewählt und das Fenster öffnet sich wieder. Wurde der fünfte Punkt in der Karte markiert, zeigt ElKowmGIS neben dem Fenster das referenzierte Bild auch in der Karte an. Nun können Sie die Genauigkeit der Referenzierung noch verbessern, indem Sie weitere Punkte angeben oder das Bild mit einem Klick auf Ok übernehmen. ElKowmGIS schliesst dann das Fenster, speichert das transformierte Bild auf dem Server ab und legt die Datenbankeinträge an.

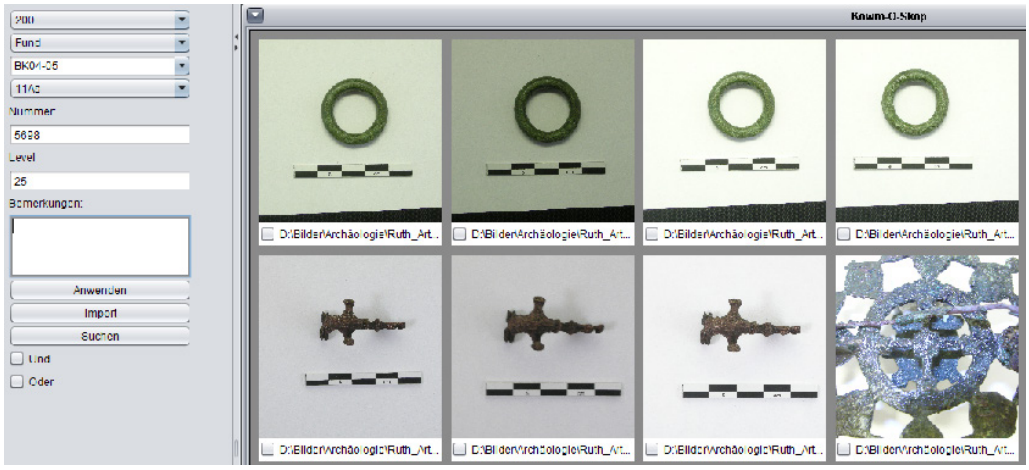
6.2.11 Die Bildverwaltung

Icon: **Shortcut:** - **Menu:** Bild -> Bildverwaltung

Fenster



Funktion:



Die Bildverwaltung erlaubt das Laden und Verwalten von Bildern. Bilder können von externen Quellen bezogen, verschlagwortet und im Dokumentenordner abgelegt werden. Über die Schaltfläche Import

gelangen Sie zum Auswahldialog, in welchem Sie die zu importierenden Bilder auswählen können. Das Programm lädt die Bilder anschliessend in den Zwischenspeicher und zeigt sie in der Anzeige von ElKowmGIS verkleinert an. Die Grösse der Bilder kann mit der Auswahl oben links in der Seitenleiste verstellt werden.

Die so eingestellte Grösse bezieht sich nur auf die Ansicht in ElKowmGIS. Die Ablage der Bilder erfolgt in der vollen Grösse und Auflösung. Um Bilder zu verschlagworten, füllen Sie bitte die Felder in der Seitenleiste aus und markieren alle Bilder, die diese Schlagworte erhalten sollen. Anschliessend klicken Sie auf die Schaltfläche anwenden. Mit dem Klick auf Anwenden werden die Bilder im Dokumentenordner abgelegt und die Informationen zu den Bildern in der Datenbank gespeichert.

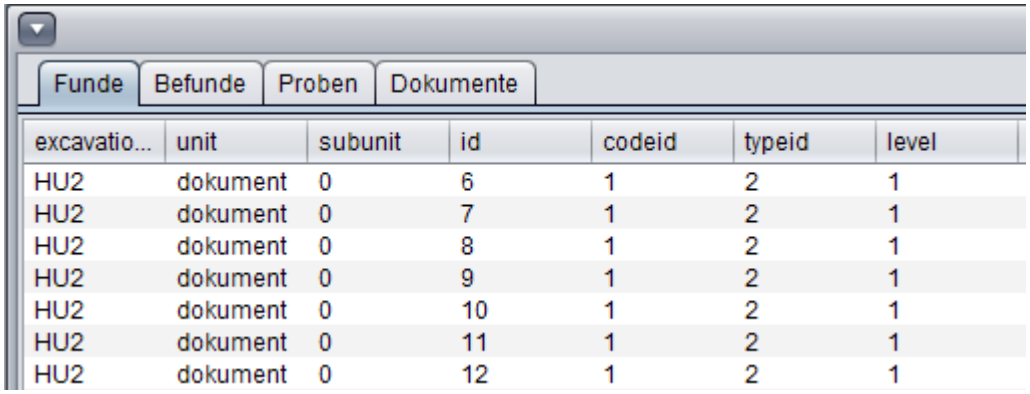
Für die Suche nach bestimmten Bildern fügen Sie die Begriffe, nach denen Sie suchen möchten, in die Seitenleiste ein. Wählen Sie zunächst, ob eine Suche über mehrere Felder mittels einer “AND“- oder einer “OR“-Suche durchgeführt werden soll. Mittels einer AND-Suche werden nur die Datenbankeinträge wiedergegeben, welche alle Suchbegriffe enthalten. Eine OR-Suche liefert hingegen alle Einträge die mindestens einen Suchbegriff enthalten. Um die Suche auszuführen, klicken Sie nach der Auswahl des Suchmodus auf die Schaltfläche suchen.

6.2.12 Die Tabellenansicht

Icon: **Shortcut:** - **Menu:** -

Fenste 

Funktion:



excavatio...	unit	subunit	id	codeid	typeid	level
HU2	dokument	0	6	1	2	1
HU2	dokument	0	7	1	2	1
HU2	dokument	0	8	1	2	1
HU2	dokument	0	9	1	2	1
HU2	dokument	0	10	1	2	1
HU2	dokument	0	11	1	2	1
HU2	dokument	0	12	1	2	1

Die Datentabellen dienen der Übersicht über alle aufgenommenen Informationen zum aktiven Projekt. Der erste Reiter beinhaltet die Suchfunktion, während die hinteren Reiter die gesamte Datenbank widerspiegeln. Wird eine Suche ausgeführt, wird ein neuer Reiter mit den Ergebnissen angezeigt.

6.2.13 Seiteneinrichtung

Über das Menu Seiteneinstellung erreichen Sie das Dialogfenster zur Einrichtung der Seite für die Druckausgabe. In diesem Dialogfenster können Sie die Grösse und Ausrichtung des Papiers einstellen,

sowie die Seitenränder verändern.

6.2.14 Drucken

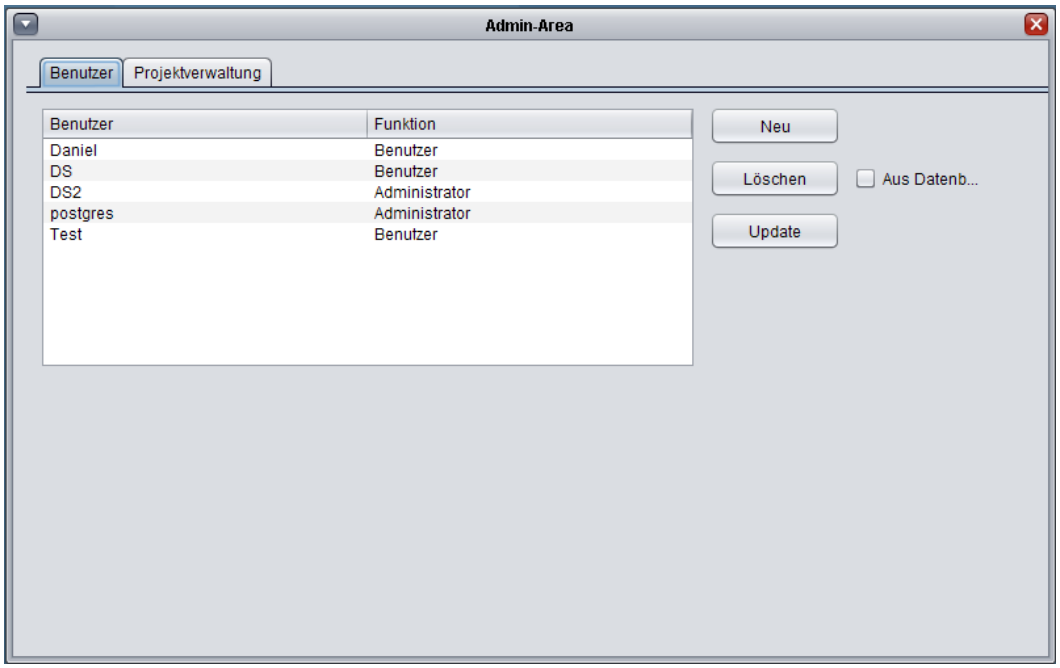
ElKowmGIS bietet Ihnen zwei verschiedene Optionen zum Drucken. Der Befehl Drucken bietet die Möglichkeit die Datentabellen direkt an einen beliebigen Drucker, der auf Ihrem System installiert ist, zu senden. Hierzu öffnen Sie zuerst die Tabelle die Sie drucken möchten und wählen dann den Druckbefehl aus. Es öffnet sich der altbekannte Druckdialog. Mit einer Bestätigung des Dialogs wird das Dokument gedruckt.

Der Befehl “Drucke Karte“ bietet die Option, den aktuellen Kartenausschnitt zu drucken. Hierzu eröffnen Sie eine GIS-Sitzung und verschieben die Karte auf den gewünschten Ausschnitt. Anschliessend wählen Sie das Menu Drucke Karte an und bestätigen den Druckdialog.

6.2.15 Admin- Area

Icon: - Shortcut: - Menu: Admin-Area -> Verwaltung

Dialogfenster:



Funktion:

Die Admin-Area dient der Verwaltung der Datenbank im Hintergrund von ElKowmGIS. Die Funktionen die ElKowmGIS zur Verfügung stellt ermöglichen es, die Benutzer zu verwalten, die Projektverwaltung zu erledigen und Backups des kompletten Servers anzufertigen. Hierzu ist das Dialogfenster in drei separate Tabs unterteilt.

Der Tab Benutzer dient, wie es der Name bereits vermuten lässt, der Benutzerverwaltung. Bei der Auswahl

eines Benutzers werden unterhalb der Nutzerliste die Projekte in der Datenbank angezeigt und die Berechtigungen, die der Benutzer besitzt. Dabei gibt es einen Unterschied zwischen Administratoren und Benutzern. Bei Administratoren ist der Eintrag in der ersten Zeile mit dem Vermerk ALL versehen und die restlichen Grabungen sind nicht ausgefüllt. Dies bedeutet, dass ein Administrator Zugriff auf alle Projekte besitzt. Durch Ändern der jeweiligen Berechtigung kann dem Benutzer das Recht vergeben werden (CheckBox aktiv) oder entzogen werden. Durch Klick auf die Schaltfläche Update wird die Berechtigung verändert. Ein Klick auf die Schaltfläche Neu öffnet ein Dialogfenster zum Erstellen eines neuen Benutzers. In diesem Fenster geben Sie den Benutzernamen, sein Passwort und die Wiederholung des Passworts ein. Sollten Passwort und Wiederholung nicht übereinstimmen kann der Benutzer nicht erstellt werden. Leere Passwörter werden nicht akzeptiert! Mit der Schaltfläche Administrator können Sie festlegen, dass der Benutzer als Administrator geführt wird, d. h. Zugriff auf alle Projekte hat. Andernfalls können Sie in der Tabelle die Berechtigungen für die einzelnen Grabungen getrennt festlegen. Ok erstellt den Benutzer und meldet ihn an der Datenbank an. Mit der Schaltfläche Löschen können Sie den aktuell markierten Benutzer vom Zugriff auf ElKowmGIS ausschliessen. Er verliert sämtliche Berechtigungen und kann sich nicht mehr im Programm anmelden. Den Zugriff auf die restliche Datenbank besitzt er aber weiterhin. Möchten Sie den Benutzer aus der kompletten Datenbank löschen, so aktivieren Sie das Häkchen neben der Schaltfläche Löschen. Somit verliert der Benutzer sämtliche Berechtigungen auf dem Datenbankserver und kann sich auch an diesem nicht mehr anmelden.

Die Projektverwaltung stellt im Prinzip das umgekehrte Verhalten dar. Zum einen können Sie hier pro Projekt, den Nutzern von ElKowmGIS die Berechtigungen erteilen. Hierzu wählen Sie das Projekt in der Liste aus und setzen im unteren Teil der Tabelle die Berechtigungen. Dadurch ist es nicht nötig für mehrere Benutzer die gleichen Schritte zu erledigen. Dies erledigt ElKowmGIS für Sie. Des Weiteren können Sie hier ein Projekt erstellen. Die Schaltfläche Neu öffnet hierzu das Dialogfenster für ein neues Projekt. Zu guter Letzt können Sie hier auch Projekte löschen. Hierbei ist jedoch Vorsicht geboten. ElKowmGIS löscht sämtliche Daten aus der Datenbank. Es wird keine Sicherungskopie zuvor erstellt. Dies muss zuvor manuell im nächsten Reiter ausgelöst werden.

Der letzte Tab des Fensters bietet Ihnen zwei Schaltflächen und ein Textfeld. Mit der Schaltfläche Datei können Sie den Pfad festlegen, in den die Datei des Backups gespeichert wird. Standardmässig zeigt dieser Pfad in den Ordner backup auf Dateiserver. Im Textfeld daneben können Sie einen Dateinamen eingeben, wie das Backup benannt werden soll. Wenn Sie das Feld leer lassen, wird als Dateiname die Zeichenfolge “bu_Erstellungsdatum_Benutzer.sql“ verwendet. Mit der Schaltfläche Los starten Sie das Backup.

Hinweis:

Falls Sie keine automatische Sicherung Ihres gesamten Systems besitzen, ist es ratsam in festgelegten Intervallen und vor grossen Änderungen (Erstellen oder Löschen eines Projektes) ein Backup vorzunehmen. Alternativ können auch die Routinen des PostgreSQL-Servers verwendet werden.

7 Die Technik hinter den Modulen

7.1 Aufbau des Programms

Der Quelltext von ElKowmGIS ist in 9 Java Packages unterteilt, welche es ermöglichen, die benötigten Klassen nach ihrer Funktion abzulegen. Dieses Vorgehen ist für den eigentlichen Programmablauf irrelevant, dient aber bei der Programmierung der Übersichtlichkeit. Auch für das nachträgliche Hinzufügen von Modulen ist diese Methode sehr hilfreich. Durch die Aufteilung in Funktionsgruppen muss bei der Programmierung neuer Module nicht das komplette Paket des Quelltexts referenziert werden. Die in diesen Paketen abgelegten Klassen werden durch das Voranstellen des Paketnamens und einer eindeutigen Bezeichnung - Java erlaubt es nicht, dass innerhalb eines Projekts zwei Klassen denselben Namen besitzen - benannt (Bsp.: Dialogs_Login.java). Abbildung 13 zeigt die einzelnen Pakete und deren Inhalt, sowie die Funktionalität der Gruppe. Die vollständige Java-Dokumentation findet sich in Anhang A.

Paket	Funktion
definitions	Enthält eigene Typdefinitionen, die in Java so nicht vorhanden sind
dialogs	Die Klassen der einzelnen Dialogfenster
inframes	Fenster der Benutzeroberfläche
listener	Enthält die Klassen zum Abfangen von sog. Actions (beispielsweise Maus- und Tastatureingaben, Bei Änderung eines Wertes in einer Auswahlliste)
main	Enthält die Hauptklasse, mit der ElKowmGIS gestartet wird
panel	Die Inhalte der verschiedenen Fenster
queries	Alle Verbindungen und Abfragen der Datenbank
threads	Befehlsabfolgen, die ElKowmGIS für verschiedenste Berechnungen benötigt
windows	Enthält das Hauptfenster von ElKowmGIS

Abb. 13: Übersicht über die Packages von ElKowmGIS.

In den folgenden Abschnitten sollen nun die technischen Details von ElKowmGIS aufgezeigt werden. Da es sich bei dieser Arbeit nicht um ein Handbuch zur Java-Programmierung handelt, werden die Eigenschaften und Regeln der Programmiersprache hier nicht aufgeführt. Für eine Einführung in die Java-Programmierung wird die Monografie von Krüger und Stark (Krüger and Stark 2011) empfohlen, welche einen guten Einstieg in die Programmierung mit Java liefert. Der Aufbau der folgenden Abschnitte hält sich dabei an den Programmablauf, wie er auftritt, wenn ElKowmGIS ausgeführt wird.

7.2 Die main-Klasse

Bei der Ausführung der Datei ElKowmGIS.jar wird als Erstes die sogenannte main-Klasse aufgerufen. Die *main()*-Methode wird von der JVM benötigt, um das Programm ausführen zu können (Ullenboom 2012, S. 328). In der Main-Klasse werden alle Variablen initialisiert und gespeichert, die ElKowmGIS während der Ausführung verwendet. Dazu gehören etwa der Benutzername, die Eigenschaften des Programms, die Adresse des Datenbank- und des Fileservers, sowie die Spracheinstellungen. Zusätzlich wird in dieser Klasse der **Logger** definiert, der die Ereignismeldungen, wie in Abschnitt "3.2.4 Log4j" beschrieben, speichert. Alle anderen Klassen legen keine neuen Logger an, sondern beziehen sich auf diesen.

Die Main-Klasse legt zuerst einen neuen Logger an und stellt die Eigenschaften des Logfiles ein. Im nächsten Schritt wird das Aussehen der Anwendung eingerichtet. ElKowmGIS nutzt dabei die Methode *UIManager.setLookAndFeel("com.sun.java.swing.plaf.nimbus.NimbusLookAndFeel")*. Beim UIManager handelt es sich um eine Java-Klasse, welche die Verwaltung des Aussehens der einzelnen Komponenten eines Java-Programms übernimmt. Durch die Methode setLookAndFeel kann das Aussehen der Oberflächen verändert werden. ElKowmGIS verwendet ein vorgefertigtes Aussehen, sog. LookAndFeel, welches in Java bereits enthalten ist. Hierbei handelt es sich um das sog. NimbusLookAndFeel (kurz Nimbus). Nimbus wurde mit der Version Java SE 6 update 10 release eingeführt und verspricht auf allen Plattformen lauffähig zu sein (http://docs.oracle.com/javase/6/docs/technotes/guides/jweb/otherFeatures/nimbus_laf.html). Nimbus bietet ein moderneres und runderes Aussehen, als die üblichen LookAndFeels von Java.

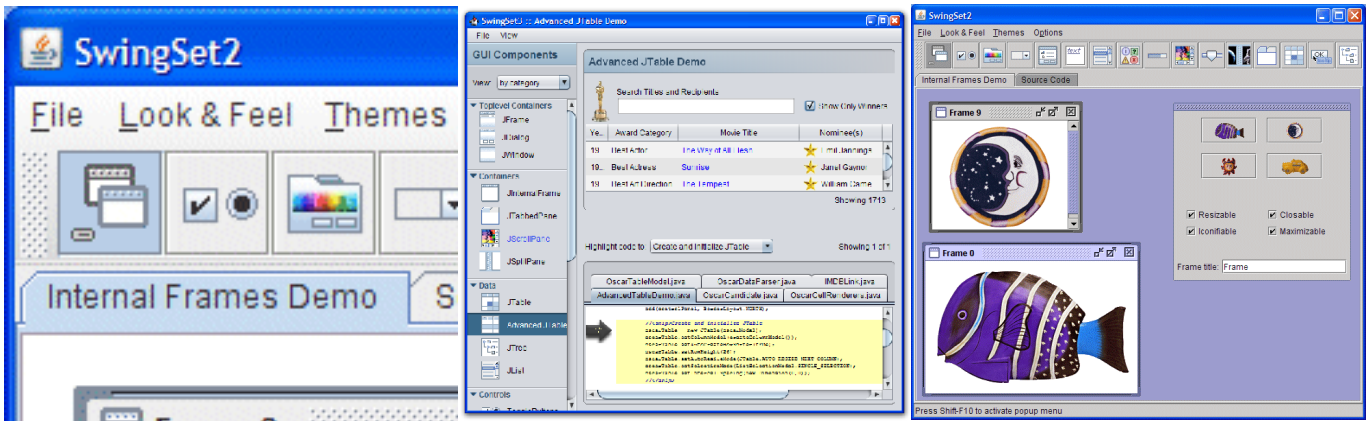


Abb. 14: Die drei Darstellungsvorlagen von Java im Vergleich: links das standardmässige Layout, mittig das Nimbus-LookandFeel und rechts das Steel-Aussehen. (Alle Abbildungen aus (Oracle 2013)).

Durch die Tatsache, dass Nimbus mittlerweile in den neuesten Java-Versionen enthalten ist, ist auch gesichert, dass das Programm die Darstellung mittels des NimbusLookAndFeel unterstützt. Zur Sicherheit wird in ElKowmGIS jedoch eine Überprüfung, ob Nimbus vorhanden ist, eingebunden. Ist Nimbus nicht verfügbar, so wird das Aussehen von ElKowmGIS durch das Standard LookAndFeel, welches in jeder JVM verfügbar ist, festgelegt.

Im letzten Schritt ruft ElKowmGIS den **Konstruktor** der Klasse Dialogs_login im Paket Dialogs auf. Dadurch wird das Anmeldefenster generiert und geöffnet.

7.3 Anmeldefenster

Die einzige Aufgabe, die der Konstruktor der Klasse *Dialogs_login* besitzt, liegt darin, die Methode *initComponents* aufzurufen. Diese Methode initialisiert einen neuen *JDialog* und alle Objekte, die im Anmeldefenster benötigt werden. Als Letztes wird ein JButton generiert, mit dem die Eingaben an das Programm übermittelt werden. Hierfür wird dem *JButton* ein *Listener_ButtonAction* als *ActionListener* hinzugefügt. Dieser reagiert primär auf das Klicken mit der Maus auf den Button selbst. Damit der Button aber auch bei Drücken der Enter-Taste reagiert, wird der JButton als *DefaultButton* des *RootPanes* des JDialogs definiert. Der DefaultButton ist der Button, der standardmässig aufgerufen wird, wenn beispielsweise die Enter-Taste gedrückt wird (Oracle 2013b).
Um mittels dem *Listener_ButtonAction*, welcher für mehrere Schaltflächen Verwendung findet, die richtige Aktion auszuführen, wird dem JButton des Anmeldefensters das *ActionCommand* "dia_login" zugewiesen. Schliesslich wird der JDialog via *setVisible(true)* auf dem Bildschirm angezeigt.

7.4 Die Klasse Listener_ButtonAction.java

Diese Klasse implementiert das Java-Interface *java.awt.event.ActionListener*. Diese enthält lediglich die Methode *actionPerformed(ActionEvent e)* und beinhaltet die Aktionen, die ausgeführt werden sollen, wenn ein bestimmtes Ereignis auftritt. Wird beispielsweise die Schaltfläche Ok im Anmeldefenster gedrückt, so wird ein sogenannter *ActionEvent* abgefeuert. Durch Abfrage des sog. ActionCommands kann innerhalb des ActionListeners bestimmt werden, welche Aktion ausgeführt wurde und welche Aufgaben nun auszuführen sind. Bei Klick auf Ok im Anmeldefenster registriert der *ActionListener* somit ein Event mit dem Command "dia_login" und führt somit die nachfolgenden Schritte aus. Zuerst werden die Informationen aus den Eingabefeldern des Anmeldefensters ausgelesen und den entsprechenden Variablen der main-Klasse zugewiesen. Sollte ein Feld nicht ausgefüllt sein, zeigt ElKowmGIS eine Fehlermeldung. Diese Überprüfung findet ebenfalls im ActionListener statt. Sind alle Felder ausgefüllt, so werden die zwei Klassen *Queries_load_db_driver* und *Queries_connect_db* geladen. Die beiden Klassen dienen dazu, den PostgreSQL-Treiber zu laden und sich mittels diesem mit der Datenbank zu verbinden. Im nächsten Schritt werden die Optionseinstellungen von ElKowmGIS geladen. Als Nächstes werden über einen Thread die Definitionen aus der Datenbank gelesen und als eigene Listen abgespeichert. Sobald der Thread beendet ist, wird das Hauptfenster generiert und das Anmeldefenster ausgeblendet.

Hinweis:

Es wäre möglich, auf diesen Schritt zu Beginn zu verzichten und die Definitionen erst bei Bedarf zu laden. Jedoch kann es dabei, je nach Verbindungsgeschwindigkeit zur Datenbank, zu Verzögerungen im Programmablauf kommen. Die Lösung, die Daten zwischenzuspeichern, erlaubt hingegen einen flüssigeren Programmablauf.

7.5 Die Benutzeroberfläche

Das Hauptfenster wird über die Klasse *Windows_frame*.java im Paket Windows definiert. Der Konstruktor der Klasse ruft hier ebenfalls lediglich die Methode *initComponents* auf. Die Methode initialisiert dann die Hauptbestandteile des Fensters. Dabei handelt es sich um eine Menuleiste des Typs *JMenuBar* und eine *JSplitpane*, die den Anzeigenbereich in ein *JPanel*, welches die Seitenleiste beinhaltet, und ein *JDesktopPanel*, welches die Fenster und die Karte beinhaltet, unterteilt. Weiter werden hier die Einträge der Menuleiste generiert und mit einem *ActionListener* versehen. Um die Befehle des Menus separat abzufangen wurde hierfür die Klasse *Listener_MenuAction* implementiert. Sie funktioniert prinzipiell gleich wie der *ButtonActionListener*, behandelt jedoch andere *ActionCommands*, die den Schaltflächen des Menus zugewiesen wurden.

Im nächsten Schritt werden die Eigenschaften des Hauptfensters gesetzt. Hierzu wird festgelegt, welches Ereignis auftritt, wenn das Fenster geschlossen wird, wie der Titel des Fensters lauten soll und wie das Fenster erscheinen soll. Mit der Methode *setDefaultCloseOperation* wird bestimmt, was geschieht, wenn das Fenster geschlossen wird. Im Fall von ElKowmGIS wird das komplette Programm beendet, wenn das Hauptfenster geschlossen wird. Der Titel des Fensters lautet, solange noch kein Projekt ausgewählt wurde, “El Kowm GIS“. Für die Bildgrösse des Fensters wurde entschieden, es zu Beginn in der maximalen Ausdehnung zu zeigen.

Schliesslich wird dem Hauptfenster noch eine Grösse zugewiesen. ElKowmGIS wird mit einer Fenstergrösse von 1024x748 Pixel ausgeführt. Dabei handelt es sich auch um die Mindestauflösung eines Bildschirms, auf dem ElKowmGIS ausgeführt werden soll.

7.6 Die Projektauswahl

Über den Menubefehl Datei-Öffnen oder die Schaltfläche Wähle Projekt der Seitenleiste wird ein *Actionevent* mit dem Command “open“ abgefeuert, welches von der Klasse *Listener_MenuAction*.java abgefangen wird. Diese Klasse legt bei Erhalt des *Events* eine neue Instanz der Klasse *Inframe_Projects()* im Paket Inframes an. Diese Klasse erzeugt ein neues Fenster, welches eine *JScrollPane* und zwei, resp. drei, Schaltflächen enthält.

Die *JScrollPane* wiederum beinhaltet eine *JTable*, welche die einzelnen Projekte auflistet. Die Projekte und die dazugehörigen Informationen werden beim Öffnen des Fensters über eine Abfrage direkt aus der Datenbank geladen. Dadurch ist gewährleistet, dass das Programm immer auf dem aktuellsten Stand ist. Die *JTable* zeigt nur die Projekte, für die der Benutzer mindestens die Lese-Berechtigung besitzt.

Bei den Schaltflächen handelt es sich um die üblichen Ok- und Abbrechen-Schaltflächen sowie um eine Löschen-Schaltfläche. Die Schaltfläche Löschen wird allerdings nur angezeigt, sofern der angemeldete Benutzer Administrator ist. Allen Schaltflächen wurde ein eigenes *ActionCommand* und ein *ActionListener* der Klasse *Listener_ButtonAction* zugewiesen. Wird ein Eintrag ausgewählt und die Schaltfläche Ok gedrückt, so schreibt der *ActionListener* die ausgewählte Projektnummer als aktuelle Projektnummer in die Variable projid der Main-Klasse und ändert den Fenstertitel in “El Kowm GIS: Projektnummer & Projektname“.

Danach wird die Projektauswahl ausgeblendet. Drückt der Benutzer die Schaltfläche Abbrechen, so wird lediglich das Auswahlfenster geschlossen und es werden keine weiteren Aktionen durchgeführt. Drückt der Benutzer auf Lösche Projekt, so muss er nochmals bestätigen, dass das Projekt gelöscht werden soll. Wird diese Abfrage bestätigt, so wird das Projekt aus der Tabelle entfernt und im Hintergrund die Methode *removeProject(Projektnummer)* der Klasse *Queries_all_queries*.java aufgerufen. Diese Methode führt zuerst eine Abfrage auf die Datenbank aus, in der alle Tabellennamen der Definitionstabellen zurückgegeben werden. Eine Schleife arbeitet nun die Ergebnisse ab und löscht in jeder Definitionstabelle die Einträge des zu löschenden Projektes. Im nächsten Schritt wird die gleiche Abfrage über das Schema elkowmgis durchgeführt. Auch hier arbeitet eine Schleife die Ergebnisse ab und löscht in jeder Datentabelle die zugehörigen Einträge.

Warnung:

In einigen Datenbanken ist es üblich, die Daten lediglich als gelöscht zu markieren und nicht mehr anzuzeigen. In ElKowmGIS werden die Daten allerdings definitiv gelöscht! Es wird daher empfohlen, vor dem Löschen eine Sicherung aller Daten vorzunehmen!

7.7 Erstellen eines neuen Projektes

Das Menu “Neues Projekt“ und der Button „Neues Projekt“ der Seitenleiste geben ein *Event* mit dem *Command* “new project“ aus. Dieses wird vom *Listener_MenuAction* entgegengenommen. Dieser generiert das Projektfenster, welches eine *JTabbedPane* enthält. Die *JTabbedPane* ist mit einem Karteikasten vergleichbar. Sie enthält im Fall von ElKowmGIS insgesamt 7 verschiedene Karteikarten. Der Inhalt jeder Karteikarte wird jeweils von einer eigenen Klasse aus dem Paket *panel* generiert. Die Klassen funktionieren mehrheitlich nach dem gleichen Schema und erweitern alle die Klasse *JPanel*. Der Konstruktor der Klasse ruft jeweils die Methode *initComponents* auf, in der die einzelnen Bestandteile des Panels erzeugt werden. Die Panels für die Funde, Befunde, Proben und Layer sind dabei optisch gleich aufgebaut. Sie enthalten alle eine weitere *JTabbedPane*, wobei für jede Kategorie eine neue Karteikarte generiert wird. Die Generierung erfolgt dabei automatisch anhand der Datenbank. Dabei werden die einzelnen Kategorien der Datenbank abgefragt und für jede das gleiche Set an Eingabefeldern angelegt. Lediglich die Beschriftung der Felder unterscheidet sich zwischen den einzelnen Karten. Zusätzlich wird eine neue Karteikarte angelegt, die es ermöglicht, weitere Kategorien einzufügen. Neu eingefügte Kategorien werden direkt in der Datenbank gespeichert.

Für die Liste der Zustandswerte, die ebenfalls direkt aus der Datenbank importiert werden, wurden zwei Schaltflächen, + und -, eingefügt, um Zustände ein- oder auszublenden. Um Zustände einzublenden öffnet sich ein neues Dialogfenster, das die zur Verfügung stehenden Zustände anzeigt. Hier ist es auch möglich, neue Zustände zu definieren.

Der letzte Reiter dient der Definition der Laufnummer und der Orientierungsmessung. Für die Definition

der Laufnummer steht ein einfaches Textfeld zur Verfügung mittels dem man die Form der Laufnummer selber erstellen kann. Bei der späteren Anzeige der Laufnummer wird die so erstellt Regel berücksichtigt. Die Angabe der Orientierungsmessung gibt später vor, wie die Orientierung eines Objektes im Projekt berechnet wird.

Durch Drücken der Schaltfläche Ok wird überprüft, ob alle Felder ausgefüllt wurden und übergibt das *ActionCommand* "np_ok" an den Listener_ButtonAction. Dieser ruft alle Eigenschaften des neuen Projektes

Hinweis:

Die Orientierung der Objekte wird nicht gespeichert. Sie wird bei jedem Aufruf neu berechnet.

aus dem Dialogfenster ab und bereitet die Daten für die Übertragung an die Datenbank vor. Dafür werden Schritt für Schritt die Informationen der einzelnen Karteikarten der ersten *JTabbedPane* gelesen und mittels mehrerer Insert-Abfragen in die Datenbank eingefügt. Schliesslich wird das Fenster geschlossen. Das so hinzugefügte Projekt kann nun über die Projektauswahl als aktives Projekt gewählt werden.

7.8 Die GIS-Funktionen

Die GIS-Funktionen umfassen, wie der Name des Programms ElKowmGIS bereits vermuten lässt, den grössten Teil der Software. Ferner ist anzumerken, dass die Berechnungen des Standpunkts, der Oberfläche und der Punktdichte erst funktionieren, wenn eine GIS-Sitzung eröffnet wurde.

Um eine GIS-Sitzung zu eröffnen muss lediglich das GIS-Modul über die Seitenleiste oder das Menu mittels des Befehls "GIS" gestartet werden. Es öffnet sich sogleich ein neues Fenster, das sogenannte Kartenfenster. Dabei löst der Listener *Listener_MenuAction* bei Erhalt des *ActionCommands* "GIS" die Generierung eines neuen Fensters aus. Das neue Fenster ist ein *Inframe* der Klasse *Inframe_gis*. Mit ihm werden mehrere Komponenten für die Darstellung und die Navigation in der Karte generiert. Zuerst wird ein *JPanel* angelegt, welches 2 Label und 2 Textfelder enthält. Diese dienen der Ausgabe der aktuellen Mausposition in den **Weltkoordinaten** (möglich wäre hier auch eine Ausgabe der Position in **Bildschirmkoordinaten**).

Im nächsten Schritt wird ein weiteres *JPanel* erstellt, welches die Layerverwaltung enthält. Die einzelnen Elemente werden dynamisch generiert. Hierfür werden zuerst über die Methode *return_Layer(Projektnummer)* alle Layer abgefragt, die zum aktuell geöffneten Projekt gehören. Hierfür wird die Anfrage "SELECT * FROM def.def_layer WHERE excavationid = 'Projektnummer';" an die Datenbank gesendet. Das resultierende *ResultSet* enthält nun alle Ebenen, die für das aktuelle Projekt erstellt wurden. Anschliessend wird dieses *ResultSet* zeilenweise abgearbeitet und für jeden Eintrag die Information id und Name des jeweiligen Layers ausgelesen. Gleichzeitig werden 3 *JCheckBoxes* generiert. Die erste *JCheckBox* dient der Sichtbarkeit des Layers. Über sie lassen sich die Elemente, die auf dieser Ebene liegen, ein- und ausblenden. Hierfür wird die *JCheckBox* mit der id des Layers benannt.

Da ElKowmGIS nicht mit den Namen der Elemente arbeitet, sondern mit deren ID, ist es möglich den entsprechenden Layer ohne erneute Datenbankabfrage ein- oder auszublenden. Würde man der *JCheckBox* den Namen des Layers als Namen übergeben, so müsste man bei einer Änderung des Status der CheckBox zuerst in der Datenbank nachschlagen, welche ID der soeben geänderte Layer besitzt und anschliessend erst die Anzeige aktualisieren. Um diese Zeit und Rechenleistung zu sparen, wurde die o. a. Variante gewählt.

Die zweite *JCheckBox* dient der Einstellung, ob ein Layer bearbeitbar oder nicht (nicht gesperrt, gesperrt) ist. Dies dient beispielsweise dazu, gewisse Grundebenen, etwa den Katasterplan einer Region o.ä., vor Änderungen zu schützen. Falls die Box aktiviert ist, ist die Ebene gesperrt, d.h. es können keine Objekte hinzugefügt, entfernt oder verändert werden. Hierfür überprüft die Zeichenroutine, ob die CheckBox mit dem Namen “G-Layerid“ aktiviert ist, oder nicht. Ist sie aktiviert, wird der Befehl verworfen.

Die dritte *JCheckBox* bestimmt, ob ein Layer gedruckt wird oder nicht. Hierfür ermittelt die Druck-Methode, ob die einzelnen Layer für den Druck freigegeben sind oder nicht. Ist eine CheckBox deaktiviert, so ist der Layer für den Druck freigegeben. Andernfalls ignoriert die Druck-Funktion die Elemente des aktivierten Layers.

Schliesslich wird zwischen die zweite und die dritte *JCheckBox* ein *JLabel* eingefügt, welches den Namen des jeweiligen Layers enthält. Dieses Label hat keine weiteren Funktionen, es dient lediglich der Beschriftung. Abschliessend werden die ID und der Name des Layers in zwei Listen gespeichert.

Anschliessend wird das eigentliche Kartenfenster generiert. Vorbereitend werden zuerst alle Objekte in der Datenbank gesucht, die erstens zum gewählten Projekt gehören, und zweitens Koordinaten besitzen. Hierfür wird mittels der Methode *return_All_Context(Projektnummer)* ein *ResultSet* generiert, das alle Objekte des Projektes enthält. Dieses ResultSet wird nun zeilenweise abgearbeitet und alle Informationen in eigenen Variablen zwischengespeichert. Aus den Elementen unit, subunit und id wird ein eindeutiger Schlüssel generiert, der für die nächste Abfrage benötigt wird. Dabei werden über die Methode *return_Objects(Projektnummer, Schlüssel)* die Koordinaten der Objekte aus der Datenbank ausgelesen. Die Ergebnisse werden wiederum in einem *ResultSet* gespeichert. Falls das *ResultSet* nicht leer ist, werden die Koordinaten, getrennt in X, Y und Z, in eigene Listen gespeichert. Dieser Schritt ist für alle Objekte ausser den Punktinformationen notwendig, da die Objekte über mehrere Punkte definiert sind. Beispielsweise wird eine Linie über zwei Punkte, den Start- und den Endpunkt definiert. Eine Polylinie kann beliebig viele Punkte besitzen. Obwohl die Punktinformation nur ein Koordinatentripple umfasst, wird die Geometrie eines Punktes ebenso über diese drei Listen weitergegeben. Dieses Vorgehen wurde lediglich zur Vereinheitlichung der Methodik gewählt. Es besitzt jedoch auch keine Vor- oder Nachteile gegenüber der separaten Behandlung von Einzelpunktinformationen.

Anschliessend wird geprüft, ob es sich beim aktuellen Objekt um ein Rasterbild (sog. Coverage) handelt. Hierzu wird die Methode *return_cov_file(Projektnummer, Schlüssel)* aufgerufen. Enthält das Ergebnis einen Eintrag, so wird der Pfad zu dem Rasterbild als Variable gespeichert. Andernfalls bleibt diese Variable leer.

Schliesslich wird ein neues Objekt der Klasse *DBEntry* generiert und in einer Liste gespeichert.

Sind alle Objekte des ersten ResultSets abgearbeitet, wird das Panel, welches die Karte enthält, generiert. Das Kartenpanel ist ein Objekt der Klasse *Panel_map*. Mittels des Konstruktors *Panel_map(Liste1, Liste2, Liste3)* kann eine neue Karte generiert werden. Hierbei beinhaltet Liste1 die Namen der Layer des aktuellen Projektes, Liste2 deren IDs und Liste3 die Objekte der Datenbank. Mit dem Aufruf des Konstruktors werden zuerst die Listen den Klassen-eigenen Variablen zugewiesen. Als Nächstes wird eine Menuleiste über die Methode *createMenuBar(Liste1)* generiert. Diese Methode erstellt ein *JPanel*, welches mehrere Buttons und Auswahllisten enthält. Über diese Elemente können neue Objekte (Punkt, Linie, Spline, Kreis, Kreisbogen und Polygon) direkt in die Karte gezeichnet werden. Zur Funktionsweise des Zeichnens siehe Abschnitt 7.9. Mit den Auswahlmenus können die Eigenschaften Strichstärke, Strichart, Punktgrösse, Punktformat und Layer neuer Objekte bereits vor dem Zeichnen eingestellt werden.
Schliesslich wird mittels der Methode *createMap()* ein neues *JMapPane* erstellt. Die Methode erstellt zuerst einen neuen *MapContent*, abhängig vom eingestellten Koordinatensystem. In einem nächsten Schritt werden 6 *SimpleFeatureCollections*, für jeden Geometrietyp eine, erstellt. Bei den Geometrietypen handelt es sich um Punkte, Linien, Splines, Kreise, Polygone und Kreisbögen. Schliesslich wird in einer Schleife jedes Element der Datenbankeinträge (Liste3) wie folgt behandelt:

- Lese das Objekt aus der Liste
- Ermittle den Layer, auf dem das Objekt abgelegt ist
- Vergleiche, ob der Layer in der Liste der Layer-IDs (Liste2) vorhanden ist.
- Falls ja, suche die shape_id (Code für die Geometrie des Objektes)

Daraufhin wird in einer *switch*-Anweisung das Objekt mittels eines Konstruktors an die Klasse *Draw* übergeben, welche eine *SimpleFeatureCollection* zurückliefert. Dabei wird für jeden Geometrietyp eine eigene Collection zurückgegeben. Ausnahme hierbei bilden Rasterdaten. Diese werden nicht an die Klasse *Draw* übergeben, sondern direkt in der Methode *createMap()* dem *MapContent* hinzugefügt. Die Rasterdaten werden mit zwei Punkten in der Datenbank gespeichert, dem unteren linken und dem oberen rechten Eckpunkt des Rasterrechtecks. Diese zwei Punkte werden innerhalb der *switch*-Anweisung aus der Datenbank gelesen. Daraufhin wird ein *ReferencedEnvelope* mittels dieser zwei Punkte aufgespannt. Dieser *ReferencedEnvelope* gibt an, welche Fläche von der Rasterdatei bedeckt wird. Anschliessend wird aus dem Dateinamen der Rasterdatei ausgelesen, wie das Bild dargestellt werden soll. Hierbei wird zum einen ermittelt ob das Bild farbig oder in Graustufen abgebildet wird. Zum anderen wird ermittelt, ob die Werte zwischen den Punkten der Dichtekartierung interpoliert wurden, oder ob die Anzahl der Punkte innerhalb eines bestimmten Perimeters angezeigt werden soll (beispielsweise die Anzahl der Silices innerhalb eines Quadratmeters). Für die Unterscheidung zwischen interpoliert und Anzahl siehe auch das Kapitel über die Modellierung (Abschnitt 7.12).
Im nächsten Schritt wird das Bild über die JAI-Methode *create("fileload", Dateipfad)* als *RenderedImage* angelegt. Mit Hilfe der *GridCoverageFactory* der Java GeoTools kann nun ein *GridCoverage2D* erstellt werden, welches nun direkt in ein *GridCoverageLayer* umgewandelt werden kann, um es zum

MapContent hinzuzufügen. Zuerst wird in ElKowmGIS jedoch ein Farbstil für das *GridCoverage* erstellt, der abhängig von den bereits zuvor ermittelten Farbeinstellungen ist. Hierzu wurden die Methoden *createRGBStyle(GridCoverage2D)* und *createGrayStyle(GridCoverage2D)* implementiert.

Anschliessend werden für die nicht-leeren FeatureCollections einfache Darstellungsstile erstellt. Mit Hilfe der *FeatureCollections* und der zugehörigen Stile wird für jeden Geometrietyp ein eigener Layer erstellt und dem *MapContent* hinzugefügt. Abschliessend wird ein *JMapPane* mit dem *MapContent* als Inhalt erstellt. Die letzten zwei Schritte setzen die Koordinaten des Kartenausschnitts auf die maximale Ausdehnung des Karteninhalts und fügen dem Kartenfenster einen *MouseListener*, der Mausbewegungen und Eingaben erfasst, hinzu.
Die Klasse *Panel_map* erweitert die Klasse *JPanel* und besitzt somit die gleichen Eigenschaften wie ein *JPanel* auch. Aus diesem Grund kann das Kartenfenster auch wie ein normales *JPanel* verarbeitet werden. Das Kartenfenster teilt sich, wie bereits beschrieben, in drei Teile. Zu oberst über die gesamte Breite des Fensters befindet sich das Koordinaten-Panel. Dieses wird mit der Methode *add(Panel, BorderLayout.PAGE_START)* dem Inframe hinzugefügt. Die Konstante *BorderLayout.PAGE_START* bewirkt, dass das Panel immer am Anfang des Fensters positioniert wird. Die zwei weiteren Bestandteile, die Layerliste und das Kartenfenster, liegen zusammen auf einer *JSplitPane*. Diese ermöglicht es, Bestandteile nebeneinander oder übereinander anzuordnen. Die Besonderheit hierbei ist jedoch, dass die Grenze zwischen den beiden Objekten, in unserem Fall zwei *JPanel*, nicht statisch fix sein muss. Die Begrenzung lässt sich verschieben. Somit lassen sich grössere Kartenausschnitte oder auch längere Layernamen verwenden. Durch die Methode *add(JSplitPane, BorderLayout.CENTER)* wird das *JSplitPane*, und dadurch auch die beiden *JPanel*, in der Mitte des Inframes platziert.

7.9 Die Klasse Draw

Die Klasse *Draw* dient der Erstellung einzelnen *SimpleFeatureCollections* zur Darstellung der einzelnen geometrischen Objekte aus der Datenbank. Sie umfasst insgesamt 6 Methoden, für jeden Geometrietyp eine, die unterschiedliche Funktionalitäten besitzen. Die einzelnen Schritte innerhalb der Methoden sind zwar technisch gesehen prinzipiell für jede Methode gleich, jedoch finden sich inhaltliche Unterschiede. Im Folgenden sollen die technischen Schritte kurz erläutert werden.

Zuerst wird eine neue, leere *FeatureCollection* erstellt. Anschliessend wird ein *SimpleFeatureType* für das zu zeichnende Objekt erstellt. Ein *SimpleFeatureType* beschreibt dabei das Objekt an für sich. Es beinhaltet neben der Angabe der Geometrie, sämtliche Attribute, die eine Beschreibung und Identifikation des Objektes ermöglichen können. Der Aufbau eines *SimpleFeatureTypes* ist dabei immer gleich. Zuerst wird dem Typ ein Name gegeben, bevor mit einem zweiten String die Attribute definiert werden. Zur Definition des Typs gehört auch die Definition der Geometrie. Hierfür stehen in den Java GeoTools mehrere Werte zur Verfügung. Ein Teil der Werte wird in diesem Abschnitt der Arbeit vorgestellt. Die Definition der Attribute setzt sich aus zwei Bestandteilen zusammen. Zuerst wird der Name des Attributs gesetzt, anschliessend, durch einen Doppelpunkt getrennt, die Form des Attributes. Die Form sagt hierbei aus, ob es sich um eine Geometrie, einen Text oder eine Zahl handelt. Als Beispiel seien hier die Definition der Attribute Punkt (Geometrie) und X (Kommazahl) aufgelistet: “Punkt:Point“ und “X:0.0“.

Die verschiedenen Typdefinitionen sind in der **Java-API** zu den Java GeoTools unter dem Abschnitt *DataUtilities*, Methode *createType* ausführlicher beschrieben. ElKowmGIS verwendet lediglich die Typdefinitionen für die Geometrie (Point, LineString und Polygon), Text(“”), Ganzzahlen (0) und Kommazahlen (0.0). Die Attribute in ElKowmGIS stimmen mit den Spaltennamen der Datenbanktabelle context überein. Mit dem Typ kann ein *SimpleFeatureBuilder* erstellt werden, der am Ende der Routine das Feature erstellt. Anschliessend werden die X-, Y- und Z-Koordinaten des Objektes in den Zwischenspeicher geladen. Die folgenden Anweisungen fügen dem *SimpleFeatureBuilder* die einzelnen Informationen hinzu. Zuerst wird die Geometrie des Objektes mittels einer der *create*-Methoden der Klasse *GeometryFactory* aus den Java GeoTools hinzugefügt. Dieser Schritt ist notwendig, um das Feature später in der Karte korrekt darstellen zu können. Die weiteren Befehle fügen die Attribute, die nicht zur Geometrie gehören, z. B. die Projektnummer oder die Schicht in der ein Objekt gefunden wurde, zum Builder hinzu. Schliesslich kann das Feature vom *FeatureBuilder* mittels *buildFeature()* erstellt und zur *Collection* hinzugefügt werden.
Im Folgenden sollen nun für die 6 Methoden erläutert werden, worin sie sich inhaltlich unterscheiden und welche Auswirkungen diese Unterschiede für den technischen Ablauf bedeuten. Begonnen wird dabei mit der einfachsten Methode *drawPoint(Datenbankeintrag, Form-ID)*.

7.9.1 Methode drawPoint(Datenbankeintrag, Form-ID)

Die Typdefinition der Punkte beinhaltet als Geometrietyp den Punkt (Point). Zusätzliche Attribute bilden die Werte X, Y und Z. Da ein Punkt lediglich ein Koordinatentripple besitzt, können diese Werte direkt hier als Attribut hinzugefügt werden. Dies erleichtert die Darstellung dieser Werte im Seitenmenu. Zudem können somit die Koordinaten auch als Label für die Punkte verwendet werden.
Die Koordinaten für einen Punkt finden sich, wie bei den anderen Geometrietypen in den drei Listen X, Y und Z. Da ein Punkt jedoch nur eine Koordinate besitzt, wird beim Setzen der Koordinaten keine Schleife benötigt. Es reicht, den ersten Wert der Liste mittels Liste.*get(0)* abzurufen. Mit diesen Koordinaten kann im nächsten Schritt dem *FeatureBuilder* die Geometrie zugewiesen werden. Hierzu wird die Methode *createPoint(Koordinate)* der Klasse *GeometryFactory* ausgeführt. Das Ergebnis wird mittels *add(Geometrie)* dem *FeatureBuilder* hinzugefügt. Die folgenden add-Anweisungen fügen der Reihe nach alle Attribute hinzu. Dabei ist die Reihenfolge der Attribute durch die Definition des Typs am Anfang der Methode festgelegt. Schliesslich wird das neue Feature, welches man durch die Methode *buildFeature()* erhält, der *Collection* hinzugefügt.

7.9.2 Methode drawLine(Datenbankeintrag, Form-ID)

Diese Methode unterscheidet nicht, ob es sich um eine Polylinie (Linienzug mit mehreren Abschnitten) oder um eine einzelne Linie (als Verbindung zwischen zwei Punkten) handelt.
In dieser Methode wird der GeometrieTyp Linienzug (Linestring) verwendet. Zu den üblichen Attributen, wie sie in der Datenbank vorhanden sind, werden hier die Länge und die Fläche des Linienzugs mit aufgenommen. Für die Koordinaten werden alle Punkte berücksichtigt, die in der Datenbank unter der Nummer der Linie abgelegt sind. Prinzipiell unterscheidet sich dieses Vorgehen nicht von dem einer

Polylinie. Um die Geometrie der Linie zu erzeugen, wird ein **Array** mit den Koordinaten der Punkte benötigt. Hierzu wird eine **Schleife** über die Liste X gelegt, die alle Werte aus den Listen X, Y und Z liest, und eine neue Koordinate aus diesen drei Werten erstellt. Diese Koordinate wird dann in einem Array abgelegt. Schliesslich wird mittels der Methode *createLineString(Array der Koordinaten)* die Geometrie dem *FeatureBuilder* hinzugefügt. Die restlichen Attribute werden wie bereits im vorangehenden Abschnitt beschrieben hinzugefügt. Die Werte für die Länge und Fläche des Polygons werden direkt aus der Geometrie heraus gelesen. Hierfür stehen die Methoden *getLength()* und *getArea()* der Klasse *LineString* zur Verfügung.

7.9.3 Methode drawPolygon(Datenbankeintrag, Form-ID)

Diese Methode umfasst dieselben Attribute wie die Methode *drawLine()*. Der Unterschied besteht lediglich darin, dass die Geometrie, die generiert wird, eine andere ist, nämlich ein Polygon. Ein Polygon kann ebenfalls über die *GeometryFactory* generiert werden, jedoch reicht hier ein Array mit Koordinaten nicht aus. Um ein Polygon zu erstellen, muss zuerst ein *LinearRing* generiert werden. Bei einem *LinearRing* handelt es sich um einen geschlossenen Linienzug (*LineString*), d.h. der Anfangs- und der Endpunkt des Linienzugs sind identisch. Die Erstellung des *LinearRing* erfolgt ebenfalls über die *GeometryFactory*, mittels der Methode *createLinearRing(Array der Koordinaten)*. Der *LinearRing* dient dann als Grundlage zur Generierung des Polygons via *createPolygon(LinearRing)*.

7.9.4 Methode drawCircle(Datenbankeintrag, Form-ID)

Die Attribute um einen Kreis zu beschreiben umfassen, neben den oben aufgeführten, 6 weitere Attribute. Dabei handelt es sich um die 3 Koordinaten des Mittelpunktes, den Radius, den Umfang und die Fläche des Kreises. Diese Attributwerte lassen sich direkt aus der Geometrie ableiten. Bei der Geometrie handelt es sich wiederum um einen geschlossenen LinearRing. Um die benötigten Koordinaten für den Ring zu erhalten steht in den Java GeoTools die Klasse *Circle* zur Verfügung. Mit ihr kann die Darstellung des Kreises durch Linien angenähert werden (Abbildung 15).

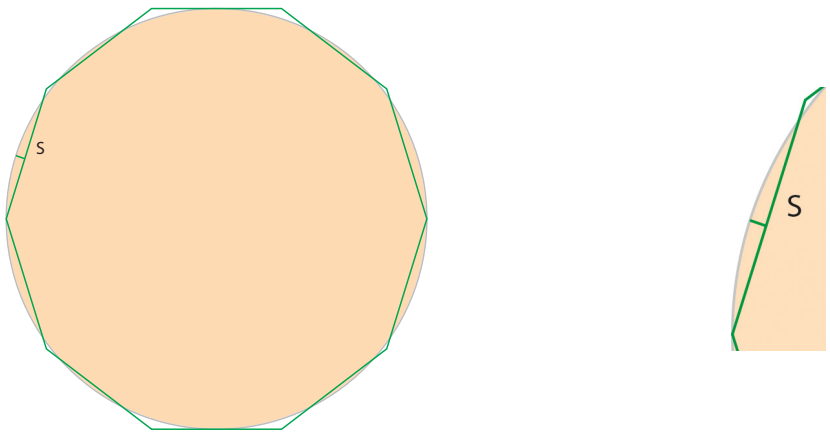


Abb. 15: Grobe Annäherung eines Kreises durch einen Linienzug. Deutlich zu sehen ist auch der Abstand zwischen der Sehne (S) und dem Kreis selbst. Rechts der Ausschnitt mit der Strecke zwischen Kreis und Sehne.

Hierfür steht die Methode *linearizeCircle(P1, P2, P3, Toleranz)* zur Verfügung. P1, P2 und P3 stellen hierbei drei Punkte auf dem Kreis dar. Begonnen wird immer mit dem ersten Koordinatentripple (X, Y, Z). Die Koordinaten der Punkte werden mittels eines einfachen Arrays an die Methode übergeben. Die Toleranz stellt die maximale Distanz zwischen einer Sehne des Kreises (Abbildung) und dem Kreis selbst dar. Desto kleiner die Toleranz gesetzt wird, um so „runder" wirkt der Kreis in der Darstellung. Die Sehne des Kreises wird hierbei zwischen zwei Punkten aufgespannt, die eine Linie der Linearisierung definieren (vgl. hierzu Abbildung 15). Anschliessend werden die Koordinaten der einzelnen linearisierten Punkte dazu verwendet, den *LinearRing* zu erstellen, der wiederum als Grundlage für ein *Polygon* dient. Als Nächstes werden die Geometrie und die Attribute dem *featureBuilder* zugewiesen. Dabei werden die Werte für das Kreiszentrum, die Fläche, den Radius und den Umfang von der Geometrie abgeleitet. Hierfür stehen mehrere get-Methoden zur Verfügung.

7.9.5 Methode drawArc(Datenbankeintrag, Form-ID)

Diese Methode funktioniert ähnlich wie die Methode drawCircle(). Der Unterschied liegt lediglich darin, dass es sich bei einem Kreisbogen nicht um eine geschlossene Form handeln muss. Aus diesem Grund wird die Methode *linearizeArc(P1, P2, P3, Toleranz)* der Java GeoTools-Klasse *Circle* verwendet. Die Punkte P1 bis P3 stellen hierbei wiederum Punkte auf dem Kreisbogen dar, normalerweise handelt es sich bei P1 um den Start- und P3 um den Endpunkt des Bogens. P2 ist ein Punkt zwischen den beiden. Die Toleranz besitzt die gleiche Funktion, wie im vorherigen Abschnitt beschrieben. Der grösste Unterschied in dieser Methode stellt nun die Geometrie dar. Da es sich bei einem Kreisbogen nicht um eine geschlossene Form handelt, kann in dieser Methode nicht mit einem *LinearRing* gearbeitet werden (ein *LinearRing* müsste geschlossen sein). Um dieses Problem zu umgehen, wird statt dem *Polygon*, welches auf einem *LinearRing* basiert, ein simpler *LineString* verwendet. Dieser bietet die Möglichkeit, die einzelnen Linien des linearisierten Kreisbogens darzustellen, ohne den Linienzug schliessen zu müssen.

7.9.6 Methode drawSpline(Datenbankeintrag, Form-ID)

Bei einem Spline handelt es sich mathematisch gesehen um eine Funktion, welche aus einzelnen Polynomen zusammengesetzt wird. Der hier verwendete Begriff Spline beschreibt streng genommen nicht die mathematische Form, sondern bezieht sich auf die abgerundete Form einer Linie, die der Approximation einer Kurve dient. Ein Spline in ElKowmGIS ist somit ein abgerundeter Linienzug. Die Berechnung der Polynome zur Beschreibung der Kurve wird von den Funktionen der JTS-Klassenbibliothek übernommen. Hierzu steht die Methode *smooth(Geometrie, Grad)* zur Verfügung. Als Geometrie wird der *LineString* übergeben, der über die Koordinaten der einzelnen Linienpunkte erstellt wird. Der Grad gibt dabei die Nähe der geglätteten Kurve zur ursprünglichen Linie an. Der Grad liegt hierbei zwischen 0 und 1, wobei 1 die Kurve bildet, die am nächsten zum ursprünglichen Linienzug liegt und der Wert 0 am weitesten entfernt ist. Die ursprüngliche Geometrie wird einfach mit der geglätteten Kurve überschrieben und dem *FeatureBuilder* übergeben. Anschliessend werden alle restlichen Attribute übergeben. Zusätzlich werden die Attribute Länge der Kurve und die bedeckte Fläche der Kurve mit übernommen.

7.10 Zeichnen von Objekten in der Karte

Neben den Methoden, die für das Erstellen der Objekte benötigt werden, existieren in ElKowmGIS weitere Klassen und Methoden, die die verschiedenen Aktionen erfassen, die für das Zeichnen von Geometrien im eigentlichen Sinn verantwortlich sind. Beim Begriff Zeichnen handelt es sich im jetzigen Zusammenhang um den Vorgang der bildlichen Darstellung von Objekten, d.h. der Vorgang des Setzens von Punkten, Erstellen von Linien etc. In diesem Abschnitt soll nun erläutert werden, welche Klassen und Methoden ElKowmGIS verwendet, um die Abbildung der verschiedenen Geometrien während des Zeichenvorgangs anzuzeigen. Hierbei wird der *MouseListener* verwendet. Der *MouseListener*, der der *MapPane* hinzugefügt wird, registriert sämtliche Aktionen, die von der Maus ausgeführt werden. Dies sind die Ereignisse Mausklick (Click), Bewegung der Maus (Move), Maustaste wird gedrückt und gehalten (Press), Maus wird bei gedrückter Maustaste bewegt (Drag), Maustaste wird losgelassen (Release) und das Scrollrad der Maus wird bewegt (MouseWheelMoved). Jedes Ereignis löst eine eigene Action aus und bewegt den *MouseListener* dazu, eine entsprechende Methode auszuführen. Für das Zeichnen der Objekte werden lediglich die Methoden *onMouseClicked(Mausereignis)* und *onMouseMoved(Mausereignis)* direkt benötigt.

Die Methode *onMouseClicked* behandelt alle Ereignisse, bei denen eine Maustaste gedrückt und wieder losgelassen wurde. Dabei wird unterschieden, welche Maustaste (links, rechts oder Mitte) gedrückt wurde. Dies ermöglicht es, beispielsweise mit Hilfe der linken Maustaste Objekte zu zeichnen, die rechte Maustaste bricht den aktuellen Zeichenvorgang ab und mit der mittleren Maustaste kann der Kartenausschnitt verschoben werden. Wird die linke Maustaste geklickt, so wird als Nächstes überprüft, welche Geometrie in der Menuleiste der Karte ausgewählt wurde. Für die einzelnen Geometrien unterscheiden sich die Arbeitsschritte und das Vorgehen, wie das Vorschaubild gezeichnet wird.

Soll ein Punkt gezeichnet werden, so ist es lediglich notwendig zu überprüfen, ob die *Snapfunktion* aktiviert ist und ob bestehende Punkte innerhalb des Snapping Radius liegen. Die Überprüfung ob Punkte innerhalb dieses Kreises liegen erfolgt bereits während der Mausbewegung. In der Methode *onMouseMoved(Mausereignis)* wird überprüft, ob sich im Radius um die Position des Mauszeigers herum Punkte befinden. Falls die Distanz zwischen einem Punkt und dem Mauszeiger kleiner dem Radius ist, wird der Punkt einer Liste hinzugefügt. Gleichzeitig verwandelt sich das Vorschausymbol des Punktes von einem Kreis in ein Kreuz. Hierfür wird zuerst die Weltkoordinate der Mauszeigerposition in Bildschirmkoordinaten verwandelt und anschliessend auf das Fenster zwei Linien mit 10 Pixeln Länge gezeichnet, die sich in den Bildschirmkoordinaten der Mauszeigerposition schneiden (Abb. 16).

```
coord = (Coordinate) snaps.get(i);
AffineTransform world_to_screen = mapPane.getMapContent().getViewport().getWorldToScreen();
pkt_1 = world_to_screen.transform(new Point2D.Double(coord.x, coord.y), null);
graphic.drawLine(pkt_1.getX() - 5, pkt_1.getY() + 5, pkt_1.getX() + 5, pkt_1.getY() - 5);
graphic.drawLine(pkt_1.getX() - 5, pkt_1.getY() - 5, pkt_1.getX() + 5, pkt_1.getY() + 5);
```

Abb. 16: Quelltext zum Zeichnen des Fang-Kreuzes

Abschliessend wird die Oberfläche aktualisiert. Findet sich kein Punkt innerhalb des Radius, so wird ein Kreis mit der Methode *drawOval(Position, Länge, Breite)* der Graphics-Klasse von Java auf den Bildschirm gezeichnet. Länge und Breite sind hierbei identisch, um einen Kreis zu erhalten. Dieses Vorgehen stellt das normale Verhalten dar, falls ein Punkt oder ein Anfangspunkt einer komplexeren Geometrie gezeichnet werden soll. Um auch eine komplexere Geometrie bereits in der Vorschau darzustellen, wurde der Klasse *MapMouseAdapter* eine Variable hinzugefügt, die die Anzahl der getätigten Klicks speichert. Dadurch wird es ermöglicht, zu unterscheiden, ob man gerade erst beginnt, eine Zeichnung zu erstellen (Punktdarstellung) oder ob bereits ein oder mehrere Punkte gesetzt wurden (Liniendarstellung). Ist der Zähler gleich 0, so muss die Vorschau ein Punkt sein. Ist der Zähler grösser 0, so muss ausgehend vom ersten Punkt eine Linie zur Mauszeigerposition gezeichnet werden. Hierfür kann die Funktion *drawLine(Punkt 1.x, Punkt 1.y, Punkt 2.x, Punkt2.y)* der Graphics-Klasse verwendet werden. Die Koordinaten des Punktes 1 werden bereits beim ersten Klick in der Variablen *screenPos* gespeichert, so dass diese wieder abrufbar sind. Diese Variable wird mit jedem Klick überschrieben. Dadurch wird es möglich, die Linien immer vom letzten gewählten Punkt aus zu zeichnen. Für die Kreise und Kreisbögen werden andere Variablen verwendet, die zwei benötigten Punkte im Zwischenspeicher behalten. Diese beiden Geometrie-Typen werden jedoch nicht in der Vorschau angezeigt.
Der Mausklick auf die linke Maustaste wird von der Methode *onMouseClick(Mausereignis)* empfangen. Gleicht die gewählte Geometrie dem Wert “point“, so wird ein neues Objekt der Klasse *DBEntry* generiert und einer *SimpleFeatureCollection* mittels *drawPoint(Eintrag, shape-ID)* hinzugefügt. Anschliessend wird für das neue Objekt ein Dialogfenster geöffnet, indem die Werte der Attribute ausgefüllt werden müssen. Andernfalls wird der Punkt wieder aus der Zeichnung gelöscht.

Falls die Geometrie dem Wert “line“ gleicht, wird überprüft, wie oft vor diesem Klick die linke Maustaste gedrückt wurde. Steht der Zähler auf 0, so wird überprüft, ob ein Punkt innerhalb des Snapping-Radius liegt. Falls dies der Fall ist, so werden die Koordinaten dieses Punktes in den Zwischenspeicher gelegt. Andernfalls werden die Koordinaten der aktuellen Mausposition übernommen und der Zähler um eins erhöht. Ist der Zähler grösser 0, so werden die Koordinaten des zuvor geklickten Punktes in eine Liste gelegt. Anschliessend wird wiederum geprüft, ob sich ein Punkt innerhalb des Fangradius befindet. Findet sich ein Punkt in der Nähe der geklickten Position, so werden die Koordinaten dieses Punktes in eine Liste eingefügt. Findet sich kein Punkt, so werden die Koordinaten der Mauszeigerposition in die Liste eingetragen. Nun wird wiederum ein neuer Datenbankeintrag mittels *DBEntry(Punktinformationen)* angelegt und ein neues Feature einer *SimpleFeatureCollection* mittels *drawLine(DBEntry, shape-ID)* hinzugefügt. Anschliessend öffnet sich das gleiche Dialogfenster zur Eingabe der Attributwerte des Objektes. Schliesslich wird der Klickzähler auf 0 gesetzt und die Variable screenPos geleert.
Für das Zeichnen eines Splines wird prinzipiell die gleiche Technik wie für eine Linie verwendet. Lediglich der Methodenaufruf zum Hinzufügen eines Features unterscheidet sich. In diesem Fall wird die Methode *drawSpline()* verwendet.

Gleicht die Geometrie dem Wert “circle“, so wird das ähnliche Vorgehen wie zuvor beschrieben verwendet. Zuerst wird geprüft, wie oft zuvor die linke Maustaste geklickt wurde und ob sich Punkte

innerhalb des Fangradius befinden. Ist der Klickzähler grösser 0, so wird ein neuer Kreis aus der Klasse *Circle* mittels der Variablen *screenPos* und den Koordinaten des zweiten Punktes erstellt. Auf diesem Kreis werden mittels der Methode *getPoint(Winkel)* drei Punkte ermittelt, deren Koordinaten in die Listen X, Y und Z geschrieben werden. Anschliessend wird ein neues Feature erstellt und einer neuen Collection hinzugefügt. Abschliessend öffnet sich wiederum das Dialogfenster zur Attributeingabe und die Variablen werden wieder auf 0 zurückgesetzt oder geleert.

Die fehlenden beiden Geometrien Arc (Kreisbogen) und Polygon erfordern einen weiteren Zwischenschritt in der Klickabfolge, da Objekte dieser Typen nur mit Hilfe von 3 Punkten erstellt werden können. Hierfür werden zwei weitere Variablen pro Geometrie eingefügt. Zusätzlich wird für das Polygon eine neue Liste angelegt.
Beim Zeichnen des Kreisbogens wird der erste Klick wie in den vorangegangenen Abschnitten in der Variable *screenPos* und der Variable *arcPoint_1* gespeichert. Beim erneuten Klicken der linken Maustaste wird der Punkt in der Variable *arcPoint_2* gespeichert. Dieser Punkt gibt einen beliebigen Punkt auf dem Kreisbogen an. Der dritte Klick stellt den Endpunkt des Bogens dar. Seine Koordinaten müssen nicht separat gespeichert werden. Im nächsten Schritt wird wieder ein Datenbankeintrag erstellt, das Feature mittels *drawArc(Datenbankeintrag, shape-ID)* einer Collection hinzugefügt, das Attributfenster geöffnet und die Variablen zurückgesetzt.

Das Zeichnen eines Polygons erfordert eine andere Arbeitsweise wie bisher, auch in der Methode *onMouseMoved(Mausereignis)*. Zuerst soll nun die Befehlsabfolge innerhalb der Methode *onMouseClicked(Mausereignis)* näher erläutert werden, da sie Bestandteile enthält, die von der Methode *onMouseMoved* benötigt werden. Wird die linke Maustaste geklickt, so wird der Klickzähler geprüft. Besitzt dieser den Wert 0, so werden die Koordinaten wie oben erläutert abgelegt. Zusätzlich werden die Koordinaten der Liste *coord_poly* hinzugefügt. Ist der Wert des Zählers 1, so wird die nächste Koordinate in der Variablen *polyPoint_2* abgelegt und der Liste hinzugefügt. Ist der Wert grösser 1, werden die Koordinaten nur der Liste hinzugefügt und die Variable *screenPos* mit den Koordinaten des neuen Punktes ersetzt. Dies geschieht solange, bis die linke Maustaste doppelt geklickt wird. Der Doppelklick kann über die Methode *getClickCount()* der Klasse *MouseEvent* abgefragt werden. Ist das Ergebnis gleich 2, so werden die Koordinaten der Punkte aus der Liste *coord_poly* in die Listen *X*, *Y*, *Z* gespeichert. Diese Listen dienen der Erstellung eines neuen Datenbankeintrags. Mit Hilfe des Datenbankeintrags wird mit der Methode *drawPolygon(DBEntry, shape-ID)* ein neues Feature einer Collection hinzugefügt. Anschliessend werden alle Variablen und Listen zurückgesetzt.
Die Methode *onMouseMoved(Mausereignis)* umfasst die gleichen Zeicheneigenschaften, wie die bisherigen Methoden, jedoch wird am Ende eine Routine eingefügt, die die bisher gezeichneten Einzellinien auf der Oberfläche abbildet. Hierfür jeweils zwei benachbarte Punkte aus der Liste *coord_poly* ausgelesen und eine Linie zwischen Ihnen gezeichnet. Anschliessend wird der Zähler der Schleife um 1 erhöht und die nächste, anschliessende Linie gezeichnet. Abschliessend wird die Linie zwischen dem letzten Punkt der Liste und dem zuletzt geklickten Punkt gezeichnet.

Werden im Dialogfenster alle Attributwerte eingetragen und das Fenster mit Ok geschlossen, so wird die Klasse *Listener_ButtonAction* wieder aktiv. Sie überprüft alle Einträge, liest sie aus den einzelnen Feldern und setzt ein SQL-Statement zusammen, welches über die Methode *add_context(Statement)* an die Datenbank übermittelt wird. Dieses Statement fügt der Tabelle *context* alle notwendigen Informationen hinzu. Im nächsten Schritt setzt der Listener ein zweites Statement zusammen, welches die Koordinaten der einzelnen Punkte in die Datenbank-Tabelle xyz einfügt. Danach wird das Dialogfenster geschlossen und die Karte mit der neuen Geometrie mittels *createMap()* neu gezeichnet.

7.11 Standpunktbestimmung

Der Menueintrag Standpunkt unter dem Menu GIS, ist mit dem *Listener_MenuAction*, der die Aufrufe der Menus empfängt und verarbeitet, verknüpft. Das *ActionCommand* “stp_from_edm“ löst die Erstellung eines neuen *JInternalFrames* der Klasse *Inframe_gis_stp* aus. In diesem Fenster wird ein neues *JPanel* der Klasse *Panel_stp* generiert. Auf diesem *JPanel* liegen alle benötigten Steuerelemente um die Stationierung eines Tachymeters mit Hilfe des Computers zu bestimmen. Anschliessend wird der Thread *Threads_read_machines* erzeugt und gestartet. Dieser Thread liest aus der Textdatei edms.txt die von ElKowmGIS unterstützten Tachymeter und ihre Konfiguration ein, erstellt für jeden Typ ein eigenes Objekt der Klasse Machine, die die einzelnen Geräte beschreibt, und fügt diese schliesslich in eine Liste ein. Anschliessend wird das Fenster angezeigt.

Das Panel für die Standpunktbestimmung wird bei der Generierung in 5 funktionelle Bereiche untergliedert. Dabei sind zwei Teile interaktiv angelegt. Die restlichen Bereiche werden von der Software selbst verwaltet. Im Bereich links oben finden sich die Elemente für die Bestimmung oder Auswahl des zu messenden Passpunktes, die Schaltfläche zur Auswahl eines Punktes auf der Karte und die Schaltfläche, die den Messdialog öffnet und. Diese Schaltflächen der Klasse *JButton* sind mit einem eigenen Listener, dem *Listener_stp*, versehen, der die beiden *Commands* “from_map“ und “measure“ empfängt. Im rechts angrenzenden Bereich findet sich eine Anzeige, wie genau die Stationierung bestimmt ist. Hierbei werden zwei Varianten angeboten. Zum einen gibt ein Textfeld die Genauigkeit verbal an. Die möglichen Ausgaben sind: Stationierung nicht bestimmt, Stationierung bestimmt oder Stationierung überbestimmt. Die Erklärung dieser Begriffe findet sich im weiteren Verlauf dieses Abschnitts. Zum anderen zeigen zwei *JPanel* farblich den Zustand der Stationierung an. Mögliche Farbkombinationen sind analog zur verbalen Aussage: rot – weiss (Stationierung nicht bestimmt), orange – orange (Stationierung bestimmt) und weiss-grün (Stationierung bestimmt). Für den Benutzer bietet die grafische Darstellung einen schnellen Überblick über den Zustand der Stationierung.
Der nächste Bereich in der Reihe besteht aus drei Textfeldern, die die Koordinaten des berechneten Standpunktes anzeigen. Diese Anzeige wird erst aktualisiert, wenn zwei oder mehr Passpunkte eingemessen wurden. Der letzte Abschnitt in der Reihe beinhaltet den Bestätigungsknopf Ok und einen *JButton* für den Abbruch der Stationierung.

In der unteren Hälfte des Dialogs findet sich eine Tabelle, die beim Öffnen des Dialogs noch leer ist. In ihr werden die angemessenen Punkte aufgelistet. Dabei werden die folgenden Werte angezeigt:

- Punktnummer
- Originale Koordinaten (X, Y und Z)
- Horizontalwinkel
- Vertikalwinkel
- Schrägdistanz
- Horizontaldistanz
- Gerätehöhe
- Prismenhöhe
- Genauigkeit der gerechneten Koordinaten (X und Y).

Im Folgenden soll nun der technische Ablauf einer Stationierung beschrieben werden. Zuerst werden die Punktnummer und die Koordinaten des zu messenden Passpunktes eingegeben. Dies kann entweder manuell durch Eintragen der Werte in die entsprechenden Felder oder automatisch durch Auswahl eines Punktes in der Karte geschehen. Für die automatische Variante wird der *JButton* "Von Karte" geklickt, wodurch sich das Dialogfenster schliesst und die volle Karte wieder sichtbar wird. Gleichzeitig setzt das Programm die Variable *sel_stp* des *MapMouseAdapters* auf true. Dies bedeutet für den *MapMouseAdapter*, dass die Events *onMouseClicked* und *onMouseMoved* wie eine Auswahl behandelt werden sollen. Dies bedeutet, dass mit dem ersten Klick ein Eckpunkt eines Auswahlrahmens gesetzt wird. Mit dem zweiten Klick werden alle Objekte ausgewählt, die innerhalb des Rahmens liegen. Nun wird überprüft, ob sich in der *Collection*, die die ausgewählten Objekte enthält, nur ein Objekt befindet. Ist dies nicht der Fall, so wird ein Fehlerdialog angezeigt und die Auswahl geleert. Findet sich nur ein Objekt in der Auswahlcollection wird überprüft, ob es sich um ein Objekt der Klasse *Point* handelt. Trifft dies zu, so wird der Punkt einer eigenen *Collection*, der Collection *selection_point*, hinzugefügt. Anschliessend wird das *JInternalFrame Inframe_gis_stp* mit dem Konstruktor *Inframe_gis_stp(Titel, selection_point)* erneut generiert. Dieser Konstruktor erstellt das gleiche Fenster wie oben beschrieben, jedoch werden die Koordinaten und die Punktnummer in die Textfelder des ersten Bereichs des Fensters geschrieben. Dies ist durch eine einfache Passage im Quelltext zu erreichen. Hierzu muss lediglich der Klasse eine *Collection* zur Verfügung stehen. Bei der Generierung der Komponenten wird nun überprüft ob die *Collection* leer ist oder nicht. Sobald die *Collection* nicht leer ist, werden die Textfelder mit den Werten des letzten Eintrags der *Collection* gefüllt. Andernfalls werden die Felder leer gelassen.
Der nächste Schritt stellt nun der Klick auf den *JButton* "Messe Punkt" dar, was das *ActionCommand measure* auslöst. Der *Listener_stp* setzt nun die Variable *sel_stp* der Klasse *Listener_MapMouseAdapter* wieder auf false und öffnet ein neues Fenster der Klasse *Inframe_gis_from_edm*. Dieses Fenster beinhaltet lediglich ein *JPanel* vom Typ *Panel_from_edm*, welches die Steuerelemente für die eigentliche Messung beinhaltet. Dieses Fenster gliedert sich in zwei Gruppen. Die erste Gruppe von Elementen dient der Einmessung der

Daten mittels eines, an den Computer angeschlossenen Tachymeters (die Verbindung kann auch kabellos via Bluetooth erfolgen). Die zweite Gruppe dient der manuellen Eingabe der benötigten Informationen. Diese Gruppe besteht aus einfachen Textfeldern für die Werte wie sie in der Auflistung oben bereits aufgeführt wurden. Wird nun die Option "manuelle Eingabe" gewählt, so sorgt *Listener_stp* dafür, dass die Felder unterhalb der Option aktiviert und die Felder der automatischen Eingabe deaktiviert werden. Hierbei kommt eine *ButtonGroup* zum Einsatz, die es ermöglicht die beiden *JRadioButtons* zu steuern. Die *ButtonGroup* sorgt dafür, dass das jeweils nicht geänderte Element verändert wird. Das bedeutet, wird der untere Button aktiviert, so wird der obere deaktiviert und umgekehrt. Um die einzelnen Elemente bei Bedarf deaktivieren und aktivieren zu können, wurden dem *Panel_from_edm* zwei *JPanel* hinzugefügt, auf denen die Elemente liegen. Dadurch reicht es im *Listener_stp* die beiden Jpanel aufzurufen und mittels der Methode *getComponents()* alle Elemente des jeweiligen *JPanels* in ein *Array* zu schreiben. Danach wird mit einer Schleife jedes Element innerhalb des *Arrays* de- oder aktiviert.

Die erste Gruppe hingegen besteht mehrheitlich aus Elementen der Klasse *JComboBox*. Diese Boxen beinhalten die Geräteeinstellungen Gerätename, Baud-Rate, COM-Port, Stop-Bit, Data-Bits und Parität. Hierbei handelt es sich um die Parameter, die für die Kommunikation mit einem Tachymeter notwendig sind. Diese Werte sind in der Datei edms.txt gespeichert. Durch Eintragen weiterer Geräte lässt sich ElKowmGIS auch mit anderen Geräten, wie den bereits Eingetragenen, nutzen. Das Standardverhalten von Java beim Ändern des Inhalts einer *JComboBox* ist relativ simpel. Wird der Wert geändert, so wird kein *ActionCommand* oder *Event* ausgelöst. Somit müsste man alle Boxen auf die richtigen Werte umstellen, wenn man ein anderes Gerät auswählen möchte. Um dies zu verhindern, wurde dem Model der *JComboBox* ger, die die Gerätenamen enthält, ein eigener *DataListener* (Abb. 17) hinzugefügt. Dieser *DataListener* besitzt die Methode *contentsChanged(ListDataEvent),* die es erlaubt Aktionen auszuführen, wenn die Auswahl geändert wird. In diesem Fall liest die Methode die ID des gewählten Eintrages aus und setzt alle selektierten Indizes in den anderen Boxen auf denselben Wert. Dadurch sind die Boxen miteinander synchronisiert. Nachdem alle Werte richtig eingestellt resp. eingetragen sind, wird die Messung über den *JButton* fire ausgelöst. Dieser Button ist mit dem *Listener_stp* verknüpft, der bei Empfang des Signals "Fire" zuerst die Werte aus den einzelnen Feldern ausliest. Dieser Werte übergibt er an den Thread *Threads_read_com*, startet diesen und wartet bis der Thread beendet ist. Der Thread

```
ger.getModel().addListDataListener(new ListDataListener() {
        @Override
        public void contentsChanged(ListDataEvent e) {
                int id = ger.getSelectedIndex();
                com.setSelectedIndex(id);
                baud.setSelectedIndex(id);
                stop.setSelectedIndex(id);
                data.setSelectedIndex(id);
                par.setSelectedIndex(id);}});
```

Abb. 17: Quelltext des DataListeners, der die Änderungen der Geräteauswahl überwacht

übernimmt die Verbindungsparameter und wandelt sie in Konstanten der Klasse *SerialPort* aus dem Paket RXTX um. Hierzu wird der Wert des einzelnen Parameters gelesen. Bei den Werten StopBit und DataBits um ganzzahlige Werte handelt, können zwei *Switch*-Anweisungen verwendet werden. Hierbei wird einer Ganzzahl der Wert der Variablen mittels *Integer.parseInt(Variable)* übergeben. Anschliessend werden mit der *Switch*-Anweisung verschiedene Fälle abgearbeitet. Abbildung 18 zeigt den entsprechenden Quelltext für die Variable databits.

```
int d = Integer.parseInt(data.trim());
      switch (d) {
         case 5:
            databit = SerialPort.DATABITS_5;
         case 6:
            databit = SerialPort.DATABITS_6;
         case 7:
            databit = SerialPort.DATABITS_7;
         case 8:
            databit = SerialPort.DATABITS_8;
      }
```

Abb. 18: Quelltext der Switch-Anweisung zur Überprüfung des Wertes der Variable databits

Hinweis:

Der Anschluss beschränkt sich hier nicht auf den Kabelanschluss. Ein Gerät kann auch mittels einer kabellosen Bluetooth-Verbindung an den Computer angeschlossen werden. Diese Verbindung muss allerdings zuvor im Betriebssystem eingerichtet werden. Dadurch wird ebenfalls ein com-Port vergeben.

Bei der Parität handelt es sich um einen alphanumerischen Wert, der die Werte None, Odd, Even, Mark oder Space annehmen kann. Die Parität oder auch das Paritätsbit dient der Kontrolle ob die Daten richtig übertragen wurden. Im Regelfall kommen lediglich die Werte None, Odd oder Even vor. ElKowmGIS implementiert jedoch auch die beiden restlichen, um allen Eventualitäten vorzubeugen. Ein Paritätsbit kann nach der Übermittlung der Daten vom Sender an den Empfänger gesendet werden. Die Datenübertragung über die serielle Schnittstelle an für sich verläuft binär, d.h. alle Werte werden mit Kombinationen von 0 und 1 dargestellt. Das Paritätsbit gibt nun an, ob die Anzahl der übertragenen Einsen gerade (ODD) oder ungerade (Even) ist. Ist der Wert None eingestellt, so weiss der Empfänger, dass der Sender kein Paritätsbit sendet. Durch die Angabe gerade oder ungerade kann nun der Datenstrom überprüft werden, ob die empfangenen Werte diesem Kriterium entsprechen. Ist dies nicht der Fall, so ist bei der Übertragung

der Daten ein Fehler passiert. Die letzte Variable, der com-Port, ist wiederum eine Ganzzahl und gibt an, an welcher Schnittstelle des Empfängers der Sender angeschlossen ist. Diese Eigenschaft lässt sich unter Windows im Gerätemanager ermitteln und muss in ElKowmGIS von Hand eingestellt werden. Dabei reicht es lediglich die Nummer des Ports in die Datei edms.txt einzutragen. ElKowmGIS ergänzt die Zeichenfolge "COM" selbstständig. Anschliessend finden sich zwei *JTextFields* für die Geräte- und Prismenhöhe. Diese Werte werden für die Berechnung der Höhe des Standpunktes benötigt.

Als Nächstes startet der Thread eine Verbindung mit dem externen Gerät über die serielle Schnittstelle. Hierzu wird zuerst ein *PortIdentifier* (Abb. 19) mit dem Wert des Com-Ports erstellt. Dieser dient dazu, zu überprüfen, ob der Port nicht bereits benutzt wird. Falls der Port nicht benutzt wird, kann der Thread ihn öffnen und testen, ob es sich um einen seriellen Port handelt. Die Übertragung zwischen Tachymeter und Computer läuft prinzipiell immer über den seriellen Port ab, daher ist diese Überprüfung zwingend notwendig. Gleichzeitig kann mit dem Öffnen des Ports auch überprüft werden, ob ein Gerät angeschlossen ist, oder nicht.

Anschliessend werden dem seriellen Port die oben gesetzten Parameter übergeben. Im nächsten Schritt wird ein *InPutStream* generiert, der dem seriellen Port als *EventListener* übergeben wird. Dadurch können

```
CommPortIdentifier portIdentifier = CommPortIdentifier.getPortIdentifier(„COM“+com);
    if (portIdentifier.isCurrentlyOwned()) {
        System.out.println(„Error: Port is currently in use“);
    }
```

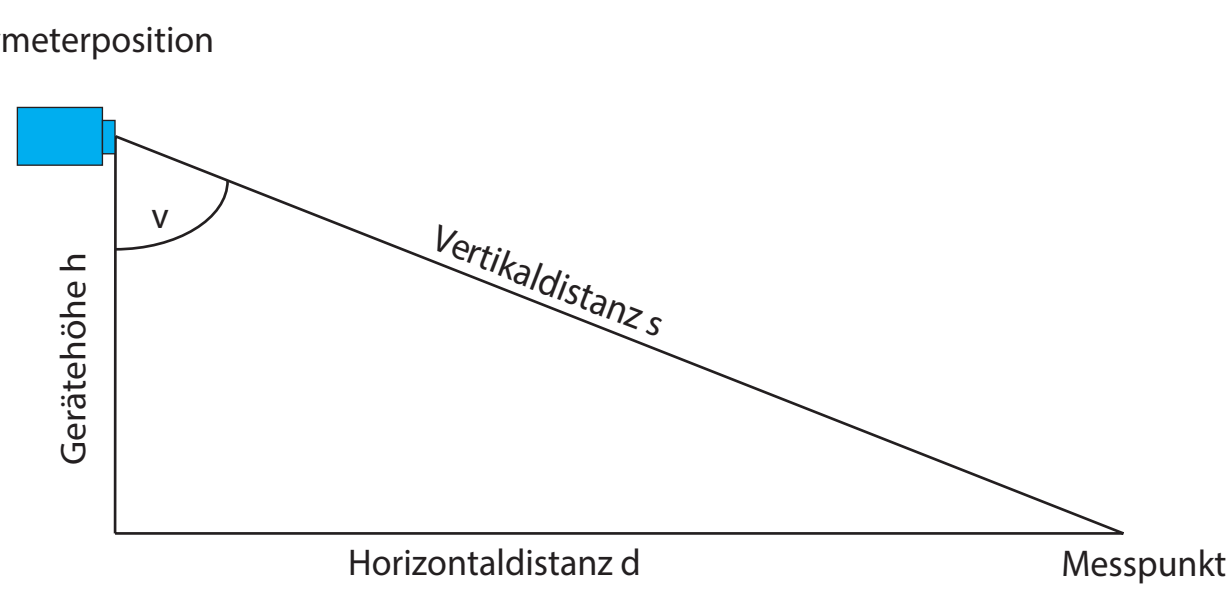
Abb. 19: Erstellung des PortIdentifiers und Überprüfung ob ein Gerät angeschlossen ist

die Werte, die vom Sender ausgesendet werden, empfangen werden. Gleichzeitig wird dem Port ein neuer *OutputStream* übergeben, der es ermöglicht Daten an den Sender zu übermitteln. Im Thread *send* werden dann sämtliche Befehle über einen *SerialWriter* an den Tachymeter gesendet. Diese Befehle fordern vom Tachymeter zuerst das Datenformat an. Hierbei wird zwischen den Datenformaten **GSI-8** und **GSI-16** unterschieden. Anschliessend wird der Tachymeter aufgefordert, eine Messung auszulösen und die Ergebnisse der Messung zurückzuliefern. Bei den Leica-Geräten, welche von ElKowmGIS unterstützt werden, werden die Werte in einem codierten Format geliefert. Über die jeweiligen Schlüsselwerte (Leica Geosystems 2008) lässt sich der Text in seine Einzelteile zerlegen. Der hierzu benötigte StringTokenizer wird in der Methode *serialEvent()* der Klasse *SerialReader* generiert. Er teilt den formatierten Text an Leerzeichen in Einzelteile. Anschliessend wird geprüft, mit welchem Code ein Textstück (sog. Token) beginnt. Über diesen Code lässt sich ermitteln, welcher Wert gerade behandelt wird. Dadurch lassen sich die Werte für den Horizontal- und Vertikalwinkel und die Schräg-, sowie die Horizontaldistanz in das Messfenster übertragen. Abschliessend wird der serielle Port wieder geschlossen.

Nachdem dieser Thread beendet wurde, werden schliesslich noch die Werte der Geräte- und Prismenhöhe der ersten Feldergruppe in die zweite Feldergruppe des Messfensters übertragen. Erst dann kann das Messfenster mit Ok geschlossen werden. Das Schliessen des Messfensters löst im *Listener_stp* eine

Überprüfung der Daten auf die Richtigkeit des Formats aus. Alle Werte in den Feldern müssen numerisch sein. Findet sich ein alphanumerischer oder kein Wert in einem der Felder, so wird dem Benutzer angezeigt, welches Feld fehlerhaft ist und das Messfenster schliesst sich nicht. Sind alle Werte in Ordnung, wird mit der Berechnung der Koordinaten begonnen. Hierzu wird zuerst aus dem Vertikalwinkel und der Schrägdistanz die Horizontaldistanz berechnet (Abb. 20).

a) Skizze



b) Horizontaldistanz $$d = s * \sin(\frac{v * \pi}{200})$$

c) Vertikalwinkel $$v = \arcsin(\frac{d}{s})$$

Abb. 20: Grafik (a) und Formeln zur Berechnung der Horizontaldistanz d aus dem Vertikalwinkel v und der Schrägdistanz s (b) sowie zur Ermittlung des Vertikalwinkels v (c).

Wird der Vertikalwinkel gleich 0 eingegeben, so wird angenommen, dass der Wert für den Vertikalwinkel nicht gemessen wurde, sondern lediglich die Horizontaldistanz bekannt ist. In diesem Fall lässt sich der Vertikalwinkel über die Formel in Abbildung 20c berechnen.

Mit diesen Werten lässt sich nun ein Objekt der Klasse *Hilfspunkt* (Abb. 21) konstruieren, das lediglich dazu dient, die einzelnen Informationen zu einem Punkt zu speichern. Es besitzt keine weiteren geografischen Funktionen. Dieses Objekt wird nun der Punktliste hinzugefügt. Hierbei wird zum einen getestet, ob die Liste leer ist. In diesem Fall kann der Punkt einfach der Liste hinzugefügt werden. Andernfalls muss

```
public Hilfspunkt(String nr, double x,double y, double z,double r, double sd, double vw, double hd,double gh,
double ph){
        this.nr = nr;
        this.x=x;
        this.y=y;
        this.z=z;
        this.r=r;
        this.sd=sd;
        this.vw=vw;
        this.hd=hd;
        this.gh=gh;
        this.ph=ph;    }

    public Hilfspunkt(double x, double y, double z) {
        this.nr = „1“;              //** Die Variable nr wird vorübergehend auf 1 gesetzt**//
        this.x=x;
        this.y=y;
        this.z=z;    }
```

Abb. 21: Die Konstruktoren der Klasse Hilfspunkt im Detail.

geprüft werden, ob der Punkt schon in der Liste vorhanden ist. In diesem Fall würden die Werte des bestehenden Punktes durch die neuen überschrieben werden. Dies ermöglicht es, einen Punkt zu ändern, ohne ihn explizit markieren oder angeben zu müssen. Eine Änderung des Punktes kann dann von Nöten sein, wenn die Stationierung nicht exakt genug ist, oder während der Messung ein Fehler passiert ist, beispielsweise der Punkt vorübergehend verdeckt wurde. Enthält die Punktliste mehr als zwei Einträge, so kann die Berechnung der Standpunktkoordinaten ausgelöst werden. Dies geschieht in ElKowmGIS durch die Erstellung und Ausführung des Threads *Threads_calc_stp*. Ist dieser Thread beendet, so werden die Koordinaten des Standpunktes in die dafür vorgesehenen Felder eingetragen. Im nächsten Schritt wird die Tabelle mit den einzelnen Punkten aktualisiert. Hierzu wird die Tabelle geleert und alle Punkte aus der Punktliste hinzugefügt. Die Koordinaten der Punkte stammen aus den Eingabefeldern des *Inframe_stp*, die

```
for (int i = 0; i < points.size(); i++) {
            Hilfspunkt p = (Hilfspunkt) points.get(i);
            x += p.getX();
            y += p.getY();
            r += p.getR();
            s += p.getHd();
            n += (p.getHd() * Math.sin(((p.getR() * Math.PI) / 200)));
            e += (p.getHd() * Math.cos(((p.getR() * Math.PI) / 200)));
        }

        n = n / points.size();
        e = e / points.size();
        x = x / points.size();
        y = y / points.size();
        nx += (p.getDn() * p.getDx());
        ey += (p.getDe() * p.getDy());
        ex += (p.getDe() * p.getDx());
```

Abb. 22: Einsatz der Klasse Hilfspunkt im Thread Threads_calc_stp.

Winkel und Distanzen aus dem Messfenster und die Angaben über die Genauigkeit wurden im *Thrads_calc_stp* während der Standpunktberechnung ermittelt. Ist die Stationierung bestimmt oder überbestimmt, so kann das Fenster mit Ok geschlossen werden. In diesem Moment werden die Koordinaten des Standpunktes in einer Variablen abgelegt, damit sie für die späteren Messungen zur Verfügung stehen.

7.11.1 Die Berechnung der Standpunktkoordinaten

In diesem Abschnitt sollen nun die mathematischen Grundlagen für die Berechnungen skizziert und erläutert werden. Gleichzeitig soll aufgezeigt werden, wie diese Berechnungen in ElKowmGIS umgesetzt wurden.

Bei der Berechnung von Standpunktkoordinaten handelt es sich prinzipiell um die Berechnung der Koordinaten eines neuen Punktes in einem bestehenden Koordinatensystem mit bekannten Punkten. Die Bestimmung des neuen Punktes erfolgt hierbei über die Messungen vom zu bestimmenden Punkt zu den einzelnen bekannten Punkten. Hierbei werden die Horizontal- und Vertikalwinkel und die Distanzen zwischen den Punkten erfasst. Die Anzahl der notwendigen Parameter, um den Neupunkt zu berechnen, hängt dabei von der gewählten Berechnungsmethode ab. Misst man exakt die notwendigen Parameter und berechnet eine Stationierung, so wird die Stationierung als bestimmt oder eindeutig bezeichnet. Werden mehr Parameter als benötigt gemessen, so erhält man zusätzlich zur Stationierung Korrekturmessungen, die man durch verschiedene Ausgleichsrechnungen in die Stationierung einfliessen lassen kann. In diesem Fall spricht man von einer überbestimmten Stationierung (Kahmen 2006, S.241). ElKowmGIS behandelt die Stationierung eines Tachymeters wie eine freie Stationierung. Dies bedeutet, dass der Tachymeter an einem beliebigen Punkt, dessen Koordinaten nicht bekannt sind, im Koordinatensystem aufgebaut wird und von diesem Punkt aus die verschiedenen Messungen durchgeführt werden. Die Bestimmung der Koordinaten des Standpunktes erfolgt im Normalfall über die Messung von Richtungen und Distanzen zu bekannten Punkten. Die gängigste Berechnung hier stellt die Helmert-Transformation dar, ein Verfahren zur Transformation von Koordinaten, welches nach dem Geodäten Friedrich Robert Helmert (1843-1917) benannt wurde. Bei der Helmert-Transformation handelt es sich um eine Variante der Ähnlichkeitstransformation, die aus einer Verschiebung, Drehung und Massstabsänderung besteht (Kahmen 2006, S. 225). Die Transformation benötigt mindestens 3 bekannte Punkte und die Messungen der Richtungen und Strecken vom zu bestimmenden Punkt aus (Kahmen 2006, S. 226), um eine bestimmte Stationierung zu erhalten. Mit diesen Angaben lässt sich die Aufgabe klar formulieren:

Finde für den Standpunkt S, dessen Koordinaten in einem örtlichen System e und n lauten, die Koordinaten im X-Y-System. Hierzu sind die Punkte P1, P2, …, Pi im X-Y-System, sowie im e-n-System bekannt. Ebenfalls gegeben sind die Richtungswinkel vi zwischen S und Pi sowie die Strecken si zwischen S und Pi.

7

Die Lösung stellt sich wie folgt dar:

Um die Rechnung zu vereinfachen nimmt man an, dass der Punkt S den Ursprung (eS = nS = 0) des örtlichen Systems darstellt. Ferner lässt man den Horizontalwinkel 0 im örtlichen System mit der e-Achse zusammenfallen.

Die Punkte P1, P2, …, Pi sind in beiden Systemen bekannt und identisch. Für die Vereinfachung der Rechnung werden zuerst die Polarkoordinaten in das X-Y-System umgerechnet. Dies geschieht mittels der Formel aus Abbildung 23. Zur Vereinfachung der Rechnung ist es sinnvoll, die Parameter vom Schwerpunkt der bekannten Punktmenge aus zu berechnen.

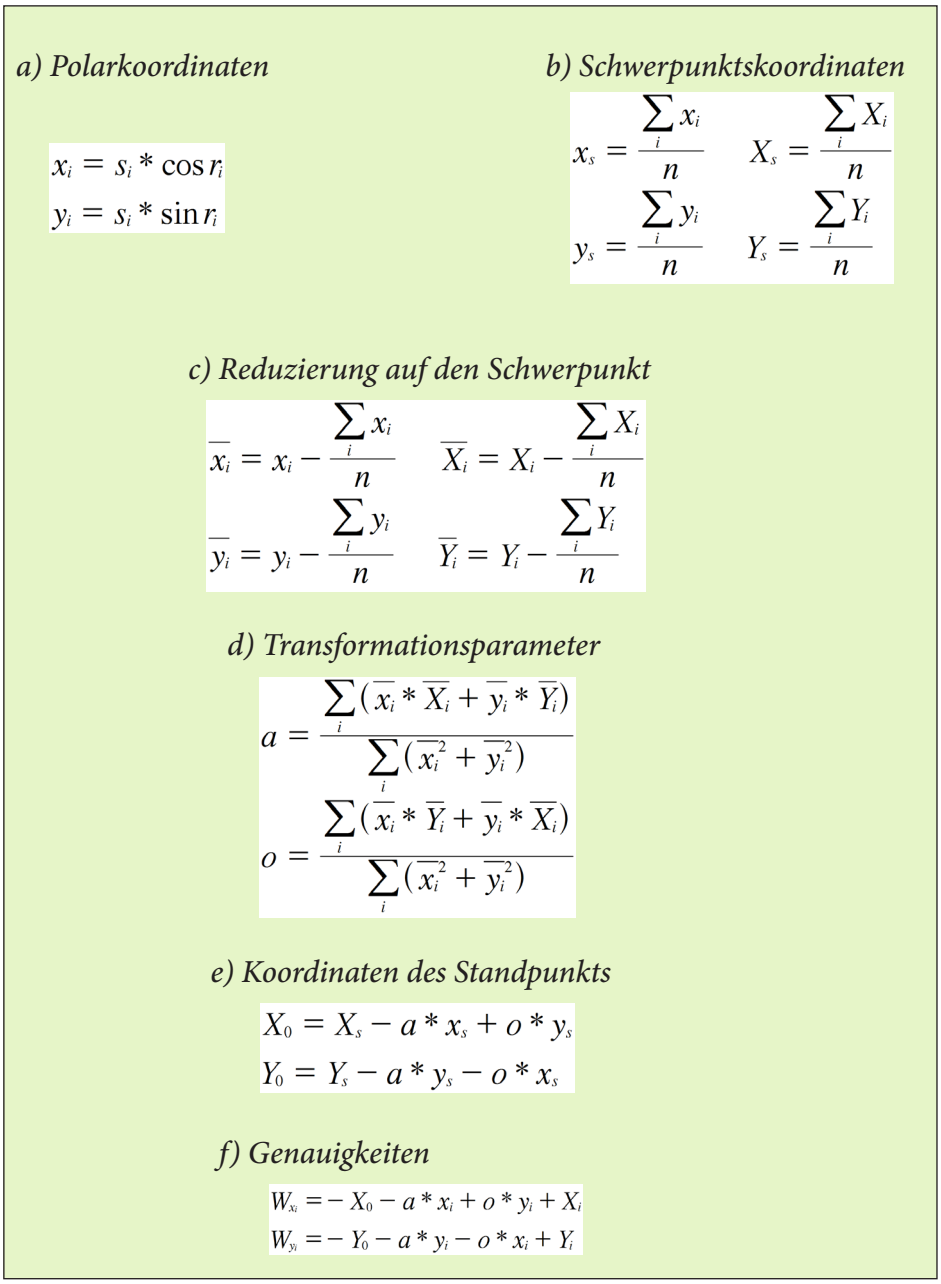


Abb. 23: Formeln zur Berechnung der Polarkoordinaten (a), der Koordinaten des Schwerpunktes (b), der Reduzierung auf den Schwerpunkt (c, nach Gruber und Joeckel 2011), der Transformationsparameter (d), der Standpunktkoordinaten (e) und der Genauigkeiten (f).

Hierbei gilt: Die Koordinaten des Schwerpunkts sind die Summen der Koordinaten geteilt durch die Anzahl der identischen, bekannten Punkte (Formel 23b). Im nächsten Schritt reduziert man die einzelnen Parameter auf den Schwerpunkt (Gruber and Joeckel 2011; Formel 23c). Dabei wird von jedem Parameterwert der Wert des entsprechenden Schwerpunktparameters abgezogen. Im nächsten Schritt lassen sich die Transformationsparameter a und o aus der Formel 23d berechnen. Abschliessend lassen sich die Werte X und Y des Standpunktes mit Formel 23e berechnen.
Im nächsten Schritt lassen sich die Genauigkeiten der Einmessungen der einzelnen Punkte berechnen. Hierzu werden die Einzelwerte in die Formeln 23f eingesetzt. Dabei gilt, dass die Summen der Abweichungen 0 sein müssen.

In ElKowmGIS erfolgt die Berechnung des Standpunktes im *Thread Threads_calc_stp*. Hierzu werden zuerst die benötigten Variablen generiert, um die Parameter abzuspeichern. Im nächsten Schritt werden die Schwerpunktparameter bestimmt. Hierzu wird jeder Hilfspunkt aus der Punktliste, welche bei der Erstellung des Threads übergeben wird, ausgelesen und der jeweilige Wert zu einer Variablen dazu addiert. Nachdem die Liste abgearbeitet wurde, werden die Variablen durch die Grösse der Liste geteilt. Im nächsten Schritt wird die Liste erneut abgearbeitet und jedem Punkt seine Reduktion auf den Schwerpunkt mittels der entsprechenden set-Methode gesetzt. (Beispiel: *p.setDx(p.getX() –x)*, wobei *p.getX()* den X-Wert des Punktes p zurückgibt und x der X-Wert des Schwerpunktes ist. Dx ist der reduzierte X-Wert des Punktes p). Gleichzeitig werden den Variablen für die Summen der Einzelwerte die Werte des aktuellen Punktes hinzugefügt. Diese Summen werden für die weitere Berechnung benötigt. Am Ende dieser Liste lässt sich der Nenner für den Term in Formel 23d aus der Summe der beiden Variablen *sdn* und *sde* bilden. Der Transformationsparameter o berechnet sich dann aus der Differenz der Variablen *ey* und *nx* geteilt durch diesen Nenner. Der Parameter a setzt sich aus der Summe der Variablen *ex* und *ny* geteilt durch den Nenner zusammen. Mit Hilfe dieser Parameter können die Variablen *x_stp* (X-Koordinate des Standpunktes) und *y_stp* (Y-Koordinate des Standpunktes) berechnet werden.

Im letzten Schritt des Threads werden nun für jeden Punkt der Liste die Genauigkeiten nach den Formeln in Abbildung 23f berechnet. Hierzu stehen der Klasse *Hilfspunkt* verschiedene get-Methoden zur Verfügung, die die benötigten Parameter zurückliefern.

7.12 Geländemodell und Funddichte

Definition Geländemodell:

Der Begriff digitales Geländemodell ist nicht einheitlich definiert. Dies liegt darin begründet, dass im nationalen und internationalen Sprachgebrauch mehrere Begriffe verwendet werden, die prinzipiell das Gleiche beschreiben. Unterschieden liegen hierbei meist nur in der Örtlichkeit, globale oder lokale Daten, oder in der Art der Aufnahme, d.h. sind beispielsweise Gebäude mit in den Daten enthalten oder nicht. Bei den Begriffen handelt es sich um die folgenden:

- Digitales Höhenmodell (DHM); (international:) Digital elevation model (DEM)

- Digitales Geländemodell (DGM); (int.:) Digital terrain model (DTM)
- Digitales Oberflächenmodell (DOM); (int.:) Digital Surface model (DSM)

Dabei kann wie folgt zwischen den drei Typen unterschieden werden. Bei einem DHM oder auch DEM handelt es sich um einen Datensatz, der die räumliche Verteilung der Höhe einer Landschaft angibt (Jordan 2007, S. 3). Hingegen handelt es sich bei einem DGM oder auch DTM um die Repräsentation verschiedener Geländewerte in einem definierten Bereich (Moore, Grayson, and Ladson 1991, S. 4). Das digitale Oberflächenmodell (DOM bzw. DSM) hingegen beschreibt die Erdoberfläche inklusive aller Objekte, wie etwa Gebäude und Strassen (Professur für Geodäsie und Geoinformatik (GG) AUF Universität Rostock 2013).

Definition Funddichte

Bei einer Funddichte handelt es sich um eine Kartierung, die angibt, wie viele Funde pro Flächeneinheit vorliegen. Die Flächeneinheit kann hierbei variabel sein. In der Regel arbeitet man jedoch mit Quadratmetern.

Prinzipiell kann ein Geländemodell auch als eine Dichte von Punkten pro Flächeneinheit gesehen werden. Dies ermöglicht es, mit der gleichen mathematischen Vorgehensweise beide Kartierungstypen zu erstellen. In ElKowmGIS wird dabei der Ansatz der inversen Distanzwichtung (engl.: Inverse Distance Weighted (IDW)) nach D. Shepard (Shepard 1968) verfolgt. Dabei wird angenommen, dass nah beieinanderliegende Punkte ähnlicher sind, als weiter voneinander entfernte. Als Beispiel hierfür lässt sich die unbekannte Höhe eines Punktes nennen. Liegt der Punkt nahe bei einem andern, dessen Höhe bekannt ist, so ist es wahrscheinlich, dass der unbekannte Wert ungefähr gleich gross wie der bekannte Wert ist. Die Zuweisung der Höhe wird umso unsicherer, je weiter der bekannte Punkt entfernt liegt. Dieser Zusammenhang wird in der IDW-Methode durch die Multiplikation des Wertes mit einem gewichteten Wert realisiert.

Definition Gewicht/gewichteter Wert:

In der Mathematik versteht man unter einem gewichteten Wert einen Wert, dessen einzelne Faktoren getrennt bewertet werden. Durch diese getrennte Bewertung kann man den Faktoren, die wichtig sind, eine grössere Priorität zuweisen. Dadurch wächst der Einfluss dieser Faktoren auf das Ergebnis.

Bei der IDW-Methode ist das Gewicht proportional zum Inversen der Distanz zwischen dem bekannten Punkt und dem Punkt, an dem ein Wert berechnet werden soll.
Mathematische Beschreibung:

Aufgabe: Der Wert z des Punktes P ist zu berechnen. Bekannt sind die Punkte D1, D2, …, Dn.

Lösung:

Definition: zn: Höhe des Punktes Dn

dn: Distanz zwischen Punkt P und Punkt Dn

Der Wert z ist das Ergebnis aus der Summe des Produkts dn hoch -2 mal zn geteilt durch die Summe der Werte dn hoch -2, falls alle dn ungleich 0. Andernfalls wird z gleich zn gesetzt (Abbildung 24).
Der Term dn hoch 2 stellt hierbei den gewichteten Wert der Höhe dar.

$$f_i(P) = \begin{cases} \dfrac{\sum_i ((d_i)^{-u} * z_i)}{\sum_i (d_i)^{-u}} & falls\ d_i \neq 0\ fuer\ alle\ D_i\ (u > 0) \\ z_i & falls\ d_i = 0\ fuer\ mehrere\ D_i \end{cases}$$

Abb. 24: Formel zur Berechnung des IDW (nach Shepard 1968).

In ElKowmGIS wird diese Berechnung in zwei verschiedenen Threads, einen für die Berechnung des Geländemodells und einen für die Funddichte, durchgeführt. Diese Trennung musste vollzogen werden, da sich die Ausgangsdaten, die verwendet werden, bei den beiden Kartierungen unterscheiden. Für ein Geländemodell wird als Ergebnis die Höhe in Abhängigkeit von den bekannten Höhen benötigt. Für die Modellierung der Funddichte hingegen ist die Anzahl der Funde innerhalb eines bestimmten Abstands notwendig.

Die Schaltfläche Geländemodell öffnet bei bestehender GIS-Sitzung ein neues Fenster der Klasse *Inframe_gis_dgm*. Die Bestandteile dieses Fensters sind auf dem *JPanel* *Panel_DGM* abgelegt und bieten die Möglichkeit die einzelnen Variablen des Modells zu definieren. Dabei handelt es sich um die Anzahl der einzelnen Klassen, in die das Resultat unterteilt wird, die Farbdarstellung und die minimale Anzahl an bekannten Punkten, die innerhalb des Suchradius liegen müssen. Im rechten Abschnitt lässt sich der Kartenausschnitt wählen, in welchem das Geländemodell berechnet werden soll. Die Werte werden automatisch eingetragen, sobald über die Schaltfläche Punkte auswählen, Punkte ausgewählt wurden.

```
for (int i = 0; i < pkt_ber.size(); i++) {
int g = 0;
Point3D pkt_int = (Point3D) pkt_ber.get(i);
BigDecimal zaehl = new BigDecimal(0);
BigDecimal nenn = new BigDecimal(0);
float wert = 0;
int nuller = 0;
    for (int j = 0; j < pkt_gem.size(); j++) {
            BigDecimal distanz = pkt_int.getDist((Point3D) pkt_gem.get(j));
            if (distanz.doubleValue() != new Double(0).doubleValue()) {
                    double zw = Math.pow(distanz.doubleValue(), -2);
                    nenn = nenn.add(new BigDecimal(zw));
                    zw = zw * ((Point3D) pkt_gem.get(j)).get_Z();
                    zaehl = zaehl.add(new BigDecimal(zw));
            } else {
                    if (nuller + 1 < 5) {
                    nuller++;
                    double zw = Math.pow(distanz.doubleValue(), -2);
                    nenn = nenn.add(new BigDecimal(zw));
                    zw = zw * ((Point3D) pkt_gem.get(j)).get_Z();
                    zaehl = zaehl.add(new BigDecimal(zw));
                    } else {
                    nuller++;
                    wert = (float) pkt_int.get_Z();
                    i = pkt_ber.size();
                    }
            }
    }
if (nuller < 5) {
    wert = (float) (zaehl.doubleValue() / nenn.doubleValue());
}
pkt_int.set_Z(wert);
}
```

Abb. 25: Quelltext der Berechnung der Geländemodellierung.

Sind mindestens 4 Punkte ausgewählt, so kann die Berechnung mittels Klick auf die Schaltfläche Ok gestartet werden. In diesem Moment wird der Thread *Threads_Generate_DGM* aufgerufen. Bis zu diesem Schritt verläuft die Auswahl der Punkte für die Berechnung der Funddichte analog. Im folgenden Textteil muss aber zwischen den beiden Threads *Threads_Generate_DGM* und *Threads_Generate_Dens* aus dem oben genannten Grund unterschieden werden. Aus Gründen der Verständlichkeit wird zuerst mit dem Thread *Threads_Generate_DGM* begonnen.

Der *Threads_Generate_DGM* nimmt zuerst die Grenzen des Kartenausschnittes und eine Liste mit den ausgewählten Punkten entgegen und legt diese im Zwischenspeicher ab. Hierfür werden die Koordinaten des Kartenausschnitts in ein *BigDecimal* umgewandelt. Dieser Schritt musste zusätzlich eingebaut werden, da die Berechnung mit den Double-Werten in Java zu ungenau werden würde. Aus den Angaben des Kartenausschnitts wird zusätzlich ein *Envelope* erstellt, der im Prinzip die Umrandung der Fläche, die berechnet wird, darstellt. Dieses Objekt wird später zur Darstellung der Karte benötigt. Anschliessend wird eine Liste generiert, die die Werte enthält, die interpoliert werden sollen. Dabei handelt es sich um ein

Hinweis:

Bei einer Georeferenzierung (siehe unten) handelt es sich streng genommen ebenfalls um eine Koordinatentransformation von einem Koordinatensystem in ein anderes. In diesem Fall wird das Bildkoordinatensystem in ein Weltkoordinatensystem transformiert. Dies bedeutet, dass jeder Punkt des Bildes eindeutig im Raum definiert werden kann. Um die Transformation zu erreichen gibt es mehrere verschiedene Systeme, die teilweise mit, teilweise ohne Passpunkte funktionieren. ElKowmGIS verfolgt die Strategie der Referenzierung mittels Passpunkten. Insgesamt werden 5 Passpunkte benötigt, um ein Bild im Raum zu platzieren. Die Berechnungen finden sich auf Seite 84ff.. ElKowmGIS erwartet jedoch, dass die Korrelation zwischen Bild und Karte manuell durchgeführt wird.

rechtwinkliges Gitter von Punkten, die in einem definierten Abstand, der Schrittweite, zueinander liegen. Diese Schrittweite ist in der Datei *config.properties* unter der Variablen *step* abgespeichert und kann vom Benutzer nach Bedarf geändert werden. Das Gitter wird durch zwei Schleifen generiert, die ineinander verschachtelt sind. Zuerst werden die minimalen X- und Y-Koordinaten als Startkoordinaten verwendet. Nun wird für die Y-Koordinate solange ein X-Wert berechnet und in die Liste gespeichert, bis dieser X-Wert das X-Maximum erreicht. Anschliessend wird der Y-Wert um eine Schrittweite erhöht und der X-Wert wieder auf das Minimum zurückgesetzt. Dies geschieht solange der Y-Wert kleiner als das Y-Maximum ist. Durch dieses Vorgehen wird das Gitternetz zeilenweise aufgebaut.

Nachdem dieses Gitternetz generiert wurde, wird begonnen, für jeden einzelnen Punkt dieses Gitters den Wert der Höhe zu berechnen. Hierzu wird eine neue Schleife angelegt, die mit dem ersten Punkt der Liste beginnt. Die Koordinaten des Punktes werden dazu genutzt, eine neue Variable der Klasse *Point3D* aus den Java GeoTools zu generieren. Diese Klasse dient dazu 3-dimensionale Punkte in ihrer Geometrie zu beschreiben. Als Nächstes werden zwei neue Variablen eingeführt, die die Werte des Zählers und des Nenners speichern. Nun wird in einer Schleife innerhalb der ersten Schleife jeder Punkt aus der Punktliste, die die gemessenen Punkte enthält, ausgelesen. Für jeden Punkt wird die Distanz zum zu berechnenden Punkt bestimmt. Hierfür verfügt die Klasse *Point3D* über die Methode *getDist(Point3D)*, die die Distanz zwischen zwei Objekten der Klasse *Point3D* zurückgibt. Ist die Distanz ungleich 0, so handelt es sich um zwei verschiedene Punkte, die für die Berechnung verwendet werden können. Somit wird zuerst der Term Distanz hoch -2 berechnet. Dieser Wert wird dem Nenner hinzugefügt. Anschliessend wird der Wert mit dem Z-Wert des jeweiligen gemessenen Punktes multipliziert und dem Zähler hinzugefügt.
Ist die Distanz gleich 0, so muss überprüft werden, wie viele Punkte bereits eine Distanz von 0 zum zu berechnenden Punkt besitzen. In ElKowmGIS liegt die kritische Grenze bei 5 Vorkommen. Sollte die Distanz in 5 oder mehr Fällen gleich 0 sein, so wird der Z-Wert des letzten gemessenen Punktes verwendet und die innere Schleife abgebrochen. Andernfalls wird der neue Z-Wert wie oben beschrieben berechnet und der Null-Zähler um eins erhöht. Ist die innere Schleife beendet, so wird dem Punkt, dessen Wert berechnet werden soll, der Quotient aus Zähler und Nennern als Wert zugewiesen und mit dem nächsten Punkt der gleiche Prozess abgearbeitet. Dies geschieht solange, bis alle Punkte des Gitternetzes einen gültigen Z-Wert besitzen.
Ist der Thread beendet, so kehrt das Programm in den *Listener_dgm* zurück und ermittelt aus den minimalen und maximalen X- und Y-Werten die Länge und Breite des abgedeckten Bereichs. Diese beiden Angaben stellen schliesslich die Abmessungen des generierten Bildes dar. Zusätzlich ermittelt ElKowmGIS durch eine Schleife den minimalen und maximalen Z-Wert des Gitternetzes. Hierzu werden den Variablen min_z und max_z zu Beginn die Werte 0 gesetzt. In der Schleife wird nun jeder Z-Wert mit diesen Variablen verglichen. Ist Z kleiner als min_z, so wird min_z durch Z ersetzt. Max_z wird durch Z ersetzt, falls Z grösser ist. Liegt Z zwischen den beiden, wird der nächste Punkt verglichen. Im nächsten Schritt wird ein *GridCoverage2D* generiert, das der Karte hinzugefügt und als Bild auf dem Dateiserver gespeichert wird.

Definition GridCoverage2D:

Bei einem *GridCoverage2D* handelt es sich um ein Objekt aus den Java GeoTools. Es handelt sich hierbei um ein 2-dimensionales Gitternetz, dem Rasterdaten in Form eines Bildes zugrunde liegen. Dieses Objekt ermöglicht es dadurch, Rasterdaten auf eine Karte zu projizieren.

Hierzu wird zuerst eine *GridCoverageFactory*, eine Hilfsklasse zur Erstellung von *GridCoverages*, generiert. Die Factory generiert dann aus den Informationen der Punktliste, die die Punkte des Gitternetzes enthält, und dem *Envelope* ein *GridCoverage2D*. Um dieses schliesslich auf die Karte projizieren zu können, muss für das *Coverage* vorgängig ein *RasterStyle* angelegt werden. Dieser Stil dient der Beschreibung des Aussehens einer Rasterdatei.

Hierzu verfügt ElKowmGIS über die Methode *createGreyScaleStyle(Farbwert, Anzahl Klassen, Minimumwert, Maximumwert)*. Abschliessend wird aus dem *Coverage* und dem *Style* ein neuer *GridCoverageLayer* erzeugt, der dem *MapContent* hinzugefügt werden kann. Ausserdem wird über die Methode *add_cov(Projektnummer, GridCoverage2D, Envelope, Farbwert, Minimum Z, Maximum Z, Anzahl Klassen)* der Klasse *Queries_all_Queries* die Informationen des *Coverage* in die Datenbank gespeichert und ein Bild im Ordner img auf dem FileServer abgelegt.

Für die Berechnung der Funddichte wird auf gleiche Art und Weise ein Gitternetz generiert, welches die zu berechnenden Punkte enthält. Der Unterschied zur Generierung des Geländemodells besteht allerdings darin, dass die Schrittweite step bei der klassischen Darstellung (Abb. 12a) 1 betragen muss. Hierfür wird überprüft ob im Fenster im Feld Interpolation der Wert Anzahl ausgewählt wurde.
Ist dies nicht der Fall, d.h. der Wert Interpoliert ist ausgewählt, so wird mit der Schrittweite aus der config-Datei gearbeitet. Um im nächsten Schritt zu berechnen, wie viele Funde sich innerhalb einer Flächeneinheit befinden, werden wiederum zwei Schleifen ineinander verschachtelt. Die äussere Schleife durchläuft hierbei die Punktliste mit den Gitternetzkoordinaten, während die innere Schleife die Liste der gemessenen Punkte durchläuft. Hierbei werden für jeden Punkt des Gitternetzes die x- und y-Koordinaten ausgelesen und eine Variable für die Anzahl der Objekte erstellt. Nun wird in der zweiten Schleife für jeden gemessenen Punkt überprüft ob seine x-Position zwischen der x-Position des gerechneten Gitterpunktes und der x-Position plus einmal die Schrittweite des gleichen Punktes liegt. Analoges gilt für die y-Position. Liegen beide Werte innerhalb der Intervalle, so wird die Anzahl um eins erhöht.

```
for (int i = 0; i < pkt_ber.size(); i++) {
    Point3D pkt_int = (Point3D) pkt_ber.get(i);
    pos_x = pkt_int.get_X();
    pos_y = pkt_int.get_Y();
    double anz = 0;
    for (int j = 0; j < pkt_gem.size(); j++) {
        if(pos_x<=pkt_gem.get(j).get_X()&&pkt_gem.get(j).get_X()<(pos_x+step)){
            if(pos_y<=pkt_gem.get(j).get_Y()&&pkt_gem.get(j).get_Y()<(pos_y+step)){
                anz++;
            }
        }
    }
    pkt_int.set_Z(anz);
}
```

Abb. 26: Quelltext zur Berechnung der Fundanzahl pro Flächeneinheit “step“*“step“

Ist das Ende der Liste der gemessenen Punkte erreicht, so wird dem berechneten Punkt die Anzahl der Funde als Z-Wert übergeben und mit dem nächsten Punkt weitergerechnet. Sollte nun im Feld Interpolation der Wert Anzahl ausgewählt sein, so wird der Thread beendet und mit denselben Schritten weiterverfahren, wie es beim Geländemodell der Fall war. Andernfalls wird nochmals dieselbe Schleife aufgerufen, wie sie für die Berechnung des Geländemodells verwendet wurde. Diese Schleife weist den Punkten des Gitternetzes einen interpolierten Wert zu, um eine interpolierte Ansicht der Funddichte (Abb. 27) zu erhalten.

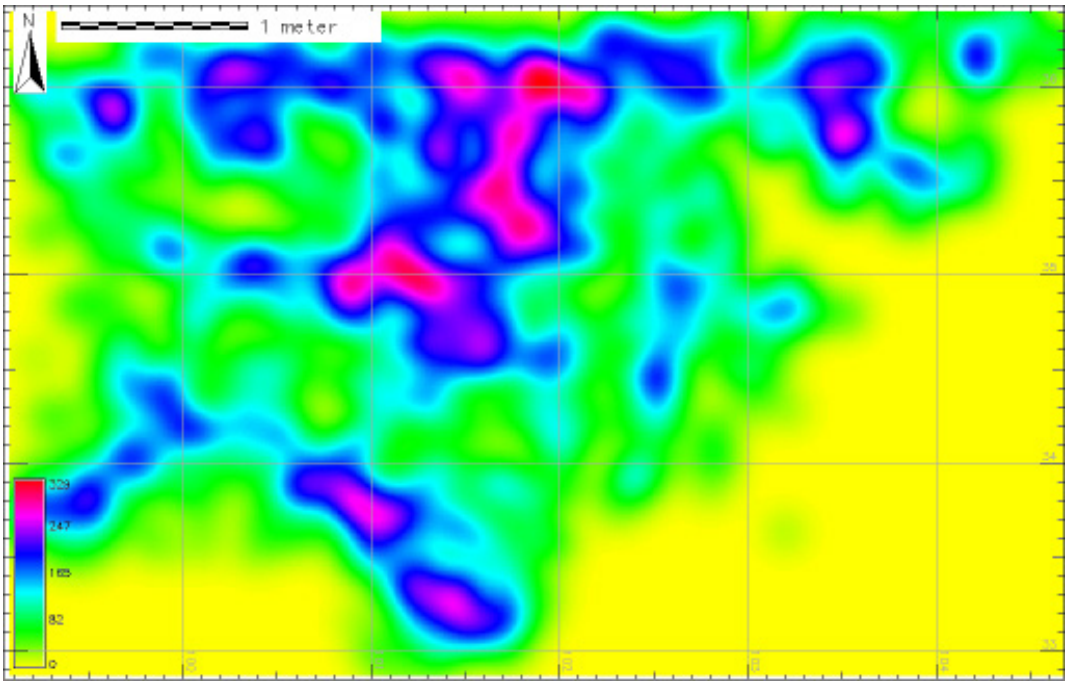


Abb. 27: Interpolierte Ansicht einer Funddichte

7.13 Bildreferenzierung

Mit dem heutigen Stand der Technik sollte eigentlich jedes neu entwickelte GIS- oder CAD-System über die Möglichkeit verfügen, Bilder mit Geodaten verknüpfen zu können. Um diesem Anspruch an eine moderne Softwarelösung gerecht zu werden, wurde in ElKowmGIS eine 2D-Georeferenzierung implementiert. Hierzu wurde die neuen Klassen Inframe_Rect, von der Klasse JInternalFrame abgeleitet, und Listener_ Mouse_Rect, Implementierung eines MouseListeners eingeführt. Der Klick auf Bild entzerren öffnet einen JFileChooser mit dem zuerst das Bild ausgewählt wird. Unterstützt werden unter anderem die Formate raw, tiff, jpeg, png, bmp und gif. Wird ein unterstütztes Bild ausgewählt, so öffnet das Programm das neue InternalFrame welches in 3 Bereiche gegliedert ist. Der erste Bereich enthält eine Vorschau des gewählten Bildes auf einem JScrollPane. Der zweite Bereich enthält die Steuerelemente um die Vorschau zu vergrössern, zu verkleinern und den sichtbaren Ausschnitt zu verschieben. Der dritte Bereich enthält eine Tabelle, welche die Korrelationstabelle zwischen Bild- und Weltkoordinaten darstellt.

Definition Passpunkt:

Ein Passpunkt ist ein Punkt, der in beiden Koordinatensystemen sichtbar ist und dessen Koordinaten in beiden Systemen bekannt sind. (Beispiel für die Transformation eines Bildes: Alter Katasterplan mit Grenzsteinen, deren Koordinaten im heutigen System bekannt sind).

Eine Kopie des Originalbildes wird mittels des Threads *Threads_Resize_Image* aus Kapitel "7.14.4 Bilder Laden und Verkleinerne mittels JAI" auf die anfängliche Grösse 400 Pixel Breite verkleinert. Dieses Vorschaubild wird mit Hilfe eines *ImageIcons* auf ein *JLabel* gelegt, welches mit dem *MouseListener* *Listener_Mouse_Rect* versehen wird. Dadurch lassen sich Mauseingaben auf dem Bilde abfangen. Das *JLabel* wird schliesslich in ein *JScrollPane* verpackt, welches es ermöglicht, den sichtbaren Bildausschnitt zu verschieben. Die Schaltflächen am rechten Rand sind mit einem *Listener_ButtonAction* verknüpft, um bei einem Klick auf eine der Schaltflächen das Ereignis verarbeiten zu können. Das Prozedere zur Referenzierung eines Bildes sieht nun vor, dass der Benutzer einen Punkt auf dem Bild anklickt und den entsprechenden Punkt auf der Karte auswählt. Aus diesem Grund wurde das *JLabel*, auf dem das Bild abgelegt wird, mit dem *MouseListener* versehen. Dieser registriert nun die Position des Mauszeigers beim Klick auf das *JLabel*. Da die Grösse des *JLabels* der Grösse des Bildes entspricht, entsprechen auch die Koordinaten des Mauszeigers den Koordinaten auf dem Bild. Gleichzeitig speichert der *MouseListener* die Bildkoordinaten als *MappedPosition* in eine Liste und setzt die Sichtbarkeit des *Inframes* mittels *setVisible(false)* auf falsch, so dass nur noch das GIS-Fenster sichtbar ist. Bei einer *MappedPosition* handelt es sich um eine Klasse der Java GeoTools, die es ermöglicht die Position eines Punktes in zwei verschiedenen Koordinatensystemen abzuspeichern. In diesem Fall stellt das erste Koordinatensystem das Bildkoordinatensystem dar. Die Koordinaten des zweiten Systems werden im Moment noch mit 0 belegt, da noch nicht bekannt ist, welche Koordinaten der Punkt in der Welt besitzt. Um diese Koordinaten zu erhalten, muss der Benutzer nun den Punkt in der Karte auswählen. Da die Karte bereits mit einem *MouseAdapter* versehen ist, muss dieser lediglich für die Referenzierung angepasst werden. Hierzu wurde dem *MouseAdapter* eine zusätzliche Variable eingefügt, die aussagt, ob sich ElKowmGIS im Auswahlmodus für die Georeferenzierung befindet. Diese Variable wird geprüft, sobald die Maus in der Karte geklickt wird.

```
m = new MappedPosition(new DirectPosition2D(e.getX()/scale, e.getY()/scale), new DirectPositi-
on2D(0, 0));
if (!l.contains(m)) {
    l.add(m);
} else {
MappedPosition m1 = l.get(l.indexOf(m));
m1.setSource(m.getSource());
l.remove(m);
l.add(m1); }
```

Abb. 28: Ausschnitt aus dem Quelltext welcher die MappedPosition aus den Java GeoTools nutzt.

Befindet sich ElKowmGIS im Auswahlmodus, so werden die Weltkoordinaten der *MappedPosition* (Abb. 28) des Punktes hinzugefügt. Hierzu sucht die Software aus der Liste den entsprechenden Punkt aus und fügt ihm die Weltkoordinaten hinzu. Enthält die Liste nun fünf oder mehr Punkte, so startet ElKowmGIS die Transformation des Bildes. Hierbei wird zuerst ein *AdvancedAffineBuilder* generiert. Dieser erlaubt es, die mathematische Transformation mit allen benötigten Variablen anhand der Liste der *MappedPositions* der Punkte automatisch zu generieren. Im nächsten Schritt wird eine *GridCoverageFactory* damit beauftragt, ein *GridCoverage2D* zu generieren. Dies geschieht über die Methode *create(Name, Originalbild, Weltkoordinatensystem, Transformation, Farbdefinitionen, Quellcoverages, Eigenschaften)*. Die Methode erstellt aus den Informationen ein neues *GridCoverage2D*, welches im Raum positioniert ist. Um einen Layer für die Karte erstellen zu können, wird mit Hilfe der Methode *createGrayScaleStyle* ein *RasterStyle* erstellt. Als Nächstes wird dem *MapContent* ein neuer *GridCoverageLayer* hinzugefügt, der das Bild in geografisch richtiger Position enthält.

Abschliessend wird der Tabelle im *Inframe_Rect* eine neue Reihe hinzugefügt, die die Korrelationsdaten des Punktes enthält, und das *Inframe_Rect* angezeigt. Ein Klick auf die Schaltfläche Ok in diesem Fenster ruft die Methode *add_cov(Projektnummer, Coverage, Envelope, Farbwahl, Darstellungsmethode = interpoliert, Minimum = 0, Maximum = 255, Anzahl Klassen=1)* auf, die das *Coverage* als neues Bild auf dem FileServer und seine Informationen in der Datenbank speichert. Danach wird das Fenster geschlossen.

Klickt der Benutzer auf Cancel, so löscht der *Listener_ButtonAction* alle *MappedPositions* aus der Liste, löscht das eventuell bereits eingefügte *Coverage* aus der Karte und schliesst das *Inframe_Rect*.

7.14 Bildverwaltung

Beim Aufruf der Bildverwaltung wird das *ActionCommand* “Bild“ ausgelöst, was den *Listener_MenuAction* dazu bewegt, ein neues *Inframe_img* anzulegen. Hierfür wird der Konstruktor *Inframe_img()* der Klasse *Inframe_img*.java im Paket *inframes* aufgerufen. Dieser erzeugt ein neues Fenster des Typs *JInternalFrame* und lädt alle Bilder, die zum aktuell aktiven Projekt gehören, in den Zwischenspeicher. Dazu wird der Thread *Threads_Load_DB* im Paket *threads* gestartet. Dieser führt die Abfrage “SELECT document,documentid FROM elkowmgis.documents WHERE excavationid = 'projid';“ in der Datenbank aus. Die Datenbank gibt

```
sql_statement = "INSERT INTO elkowmgis.context (excavationid, unit, subunit, id, codeid,
typeid,level, "
+ "state, remarks, owner, history, layer, \"R\", \"G\", \"B\", linetype, linestrength, symbolsize, symbol, "
+ "color_def, shape_id ) "
+ "VALUES("
+ "\'" + projid + "\', \'dokument\', 0," + max_id + ", 1, 2, 1, "
+ "0, 0, 0, 0, 3, 0, 255, 255, 0, 0, 0, 0"
+ ", -3, 13 )";
```

Abb. 29: SQL-Befehl zum Hinzufügen eines GridCoverage in die Datenbank

Warnung:

Die Bilder lassen sich nur löschen, wenn sie nicht gleichzeitig in einer anderen Anwendung geöffnet sind.

als Ergebnis die Dateipfade (document) und die Ids der Dokumente (documentid) der Tabelle documents im Schema elkowmgis aus, deren Grabungsnummer mit der Projektnummer (projid) übereinstimmt. Durch eine Schleife über das Ergebnis wird jedes Bild mittels des Java Advanced Imaging (JAI) geladen, für die Anzeige auf 400x400 Pixel Kantenlänge verkleinert und in der Liste sized abgespeichert. Die Verkleinerung der Bilder erfolgt ebenfalls mittels JAI und berührt das Originalbild nicht. Gleichzeitig wird das Fenster der Bildverwaltung mittels *initComponents* aufgebaut. *initComponents* erzeugt zuerst die Seitenleiste, welche für die Verschlagwortung und die Suche nach Bildern benötigt wird, und tauscht sie gegen die standardmässige Seitenleiste aus. Im nächsten Schritt wird eine *JScrollPane* angelegt, welche später die Bilder enthalten wird. Die *JScrollPane* bewirkt, dass an der rechten Seite des Fensters Scrollbalken angezeigt werden, falls mehr Bilder angezeigt werden müssen, als die Bildschirmfläche in ihrer Grösse zulassen würde.

Nachdem alle Ergebnisse abgearbeitet wurden, ruft der Thread die Methode *redraw(int size)* der Klasse *Inframe_img.java* auf. Diese Methode bewirkt, dass die Bilder in der Liste *sized* im Anzeigefenster mit der Kantenlänge size dargestellt werden. Diese Methode wird ebenfalls aufgerufen, wenn in der Seitenleiste die Grösse geändert wird. Aus diesem Grund wird beim Aufruf der Methode *redraw* der Parameter size verlangt. Für die Darstellung der Bilder wird zuerst das *JPanel*, welches alle Bilder beinhaltet, geleert. Als Nächstes wird berechnet, wie viele Bilder neben- und untereinander angeordnet werden können, bevor die maximale Ausdehnung der Anzeigenfläche erreicht ist. Die Berechnungsschritte hierfür lassen sich wie folgt beschreiben:

- Anzahl der möglichen Bilder pro Reihe = Abrunden(Breite JScrollPane geteilt durch (Breite eines Bildes +10 Pixel))
- Anzahl der benötigten Zeilen = Aufrunden(Anzahl der möglichen Bilder pro Reihe geteilt durch Anzahl aller Bilder)

Die 10 Pixel, die zu der Breite eines Bildes im ersten Berechnungsschritt addiert werden, stellen einen Zwischenraum zwischen den Bildern dar. Da die hier gewählte Darstellung prinzipiell vergleichbar mit einer Tabelle ist, wird in diesem Fall ein Java-eigenes Layout, nämlich das *TableLayout*, für das *JPanel* innerhalb der *JScrollPane* verwendet. Durch das *TableLayout* kann durch Angabe der Reihen- bzw. Spaltennummer bestimmt werden, an welcher Position ein Objekt auf dem *JPanel* platziert wird. Für die Erstellung des *TableLayouts* werden jedoch die Spaltenbreite und die Zeilenhöhe der zukünftigen Tabelle benötigt. Die Angabe dieser beiden Werte wird mittels zweier Arrays durchgeführt. Ein Array enthält die Spaltenbreiten, das andere die Höhen der einzelnen Zeilen. Da zwischen den Bildern jeweils ein Abstand von 10 Pixeln sein soll, muss berücksichtigt werden, dass die Arrays doppelt so gross und ein Eintrag mehr wie die jeweiligen Spalten- resp. Zeilenanzahlen sein müssen. Es gilt:

- Array Spalten: Grösse=2*Anzahl der möglichen Bilder pro Reihe +1

- Array Zeilen: Grösse = 2* Anzahl der benötigten Zeilen +1

Der Wert “+1“ rührt daher, dass am rechten und am oberen Rand jeweils ein Rand von 10 Pixeln sein soll. Daher gibt es insgesamt eine Spalte, bzw. Zeile, mehr, als Anzahl Bilder, resp. Reihen.
Um die beiden Arrays zu füllen sind insgesamt vier Schleifen notwendig:

- Schleife 1: Für alle ungeraden Einträge im Spaltenarray setze den Wert auf 10.
- Schleife 2: Für alle geraden Einträge setze den Wert der Bildgrösse +25 Pixel Rand
- Schleife 3: Für alle ungeraden Einträge im Zeilenarray setze den Wert 10
- Schleife 4: Für alle geraden Einträge setze den Wert der Bildgrösse +25 Pixel Rand.

```
double wide_n = Inframe_img.scrollpane.getWidth() / (val + 10);
wide_n = Math.floor(wide_n);
int anzahl = sized.keySet().toArray().length;
int anzahl_zeilen = new Double(Math.ceil(anzahl / wide_n)).intValue();
double[] col = new double[new Double(wide_n).intValue() + new Double(wide_n).intValue() + 1];
for (int kl = 0; kl < 2 * new Double(wide_n).intValue() + 1; kl += 2) {
    col[kl] = 10;}
for (int kl = 1; kl < 2 * new Double(wide_n).intValue() + 1; kl += 2) {
    col[kl] = val;}
double[] row = new double[2 * anzahl_zeilen + 1];
    row[0] = 10;
for (int kl = 0; kl < 2 * anzahl_zeilen + 1; kl += 2) {
    row[kl] = 10;}
for (int kl = 1; kl < 2 * anzahl_zeilen + 1; kl += 2) {
    row[kl] = val + 25;}
double size[][] = {col, row};
TableLayout tab = new TableLayout(size);
```

Abb. 30: Schleifen zur Berechnung des TableLayouts

Abbildung 30 zeigt den Quelltext der oben angeführten Berechnungsschritte.

Anschliessend wird das *TableLayout* mit den beiden Arrays als Parameter erstellt und dem *JPanel* als Layout übergeben. Die nachfolgende Schleife sorgt dafür, dass alle Bilder der Liste *sized* abgearbeitet werden können. Hierfür wird ein Bild über den Thread *Threads_Resize_Image* auf die vorgegebene Grösse *size* verkleinert. Nachdem der Thread beendet ist, wird das Bild einem eigenen *ImagePanel* übergeben. Das *ImagePanel* erweitert die Klasse *JPanel* und überschreibt die Methode *paintComponent*. Diese Methode

ist für das Zeichnen der Bestandteile des *JPanels* zuständig. Durch das Überschreiben ist es möglich, anstatt einer einfarbigen Fläche ein Bild auf dem *JPanel* darzustellen. Zusätzlich ist diese Methode schneller im Aufbau als beispielsweise die Verwendung eines *ImageIcons*. Das *ImagePanel* wird zusammen mit einer *JCheckBox*, welche für die Markierung des Bildes notwendig ist, auf einem weiteren *JPanel* platziert. Bevor dieses im *TableLayout* platziert wird, wird ihm noch ein *MouseListener* hinzugefügt. Dieser erlaubt es dem Benutzer mittels eines Rechtsklicks auf das Bild, es entweder im Standardbildprogramm des Betriebssystems zu öffnen, oder das Bild in die Zwischenablage zu kopieren, um es in einer anderen Anwendung weiter zu verwenden. Nach dem Platzieren des *ImagePanels* und der *JCheckBox* zur Markierung des Bildes wird der *ViewportView* der *JScrollPane* auf das *JPanel*, welches die Bilder beinhaltet, gesetzt. Dies bedeutet, dass der sichtbare Bereich der *JScrollPane* das *JPanel* darstellt. Danach ist die Schleife beendet und ElKowmGIS verarbeitet das nächste Bild.

Hinweis:

Das Setzen des ViewportView kann auch ausserhalb dieser Schleife erfolgen. Dies hätte lediglich den Effekt, dass während der Verarbeitung der Bilder ElKowmGIS scheinbar nicht reagiert. Wird der ViewportView jedoch innerhalb der Schleife gesetzt, so wird die Ansicht nach jedem Bild aktualisiert und der Benutzer kann somit sehen, dass das Programm noch aktiv ist.

7.14.1 Suche nach Bildern

Das soeben beschriebene Vorgehen lädt alle Bilder, die zu dem aktuell aktiven Projekt gehören, in die Übersicht. Um nun gezielt Bilder zu suchen, bietet die Seitenleiste mehrere Eingabefelder, die die Suchbegriffe entgegennehmen und die Auswahlmöglichkeit, ob es sich um eine AND- oder OR-Suche handeln soll. Die Schaltfläche Suchen ist mit einem *ActionListener* verknüpft, der speziell nur die Ereignisse der Bildverwaltung verarbeitet. Beim *ActionCommand* "suchen" wird zunächst eine Textvariable angelegt, die den Abfragetext beinhaltet, die Eingabefelder der Seitenleiste ausgelesen und deren Inhalt geprüft. Ist der Inhalt eines Eingabefeldes leer, so wird dieses Feld übersprungen. Andernfalls wird der entsprechende Suchbegriff dem Abfragetext entsprechend des Suchmodus angehängt. Sind alle Eingabefelder abgearbeitet, so wird der Abfragetext als Parameter dem Thread *Threads_Search_DB* übergeben. Dieser Thread führt zuerst die Abfrage aus und lädt die resultierenden Bilder anschliessend in den Arbeitsspeicher. Hierfür wird ebenfalls auf die Klassen und Methoden des JAI zurückgegriffen. Danach ruft der Thread die Methode *redraw* der Klasse *Inframe_img* auf und die Bilder werden angezeigt.

7.14.2 Importieren und Verschlagworten von Bildern

Der *ActionListener* der Bildverwaltung verarbeitet auch das Event, welches von der Schaltfläche Import ausgesandt wird. Bei Erhalt des *Commands* "import" öffnet der *ActionListener* einen *JFileChooser*, der es ermöglicht, Dateien und Ordner auszuwählen. Über die Optionen kann der standardmässig angezeigte

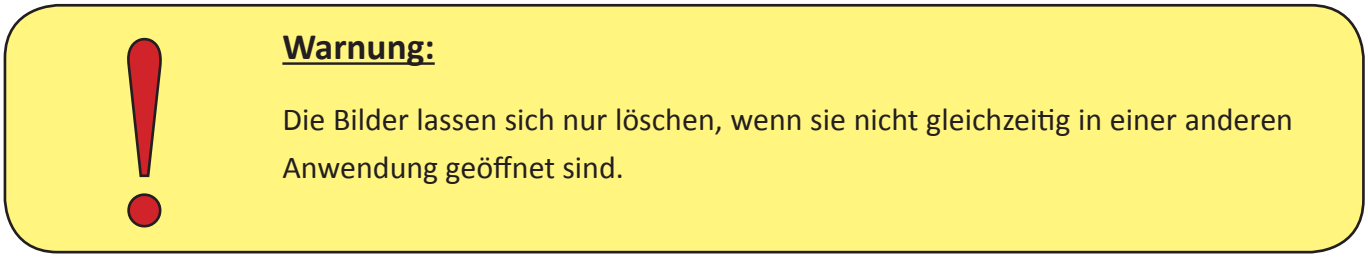
Ordner umgestellt werden. So kann zum Beispiel der Ordner Eigene Bilder der Windows-Umgebung ausgewählt werden. Mittels der Methode *setCurrentDirectory* wird dem Dialog dieser Ordner mitgeteilt. Nachdem die Bilder oder Ordner ausgewählt und der Dialog mittels Öffnen geschlossen wurde, startet der *ActionListener* einen neuen Thread, der die Bilder mittels JAI in die Liste sized der Klasse *Inframe_img*.java speichert. Um die Aktivität des Programmes mitverfolgen zu können, wird ein Statusmonitor, der angibt wie viele Bilder bereits geladen wurden, angezeigt. Abschliessend wird die Methode *redraw* der Klasse *Inframe_img* aufgerufen.

Die Seitenleiste der Bildverwaltung dient neben der Suchfunktion auch zur Verschlagwortung der Bilder. Hierfür werden die Informationen in die Eingabefelder eingetragen und die Bilder in der Übersicht ausgewählt, die mit diesen Schlagworten versehen werden sollen. Ein Klick auf die Schaltfläche Anwenden löst ein *Event* mit dem *Command* "anwenden" aus. Dieses *Event* wird vom *ActionListener* entgegengenommen und speichert die Pfade der markierten Bilder in einer Liste ab. Hierfür wird von jedem *JPanel* der Übersicht der Zustand der *JCheckBox* auf dem *JPanel* abgefragt. Ist diese markiert, so wird das Label der *JCheckBox*, welches den Pfad zum Bild enthält, in die Liste eingetragen. Ist die *JCheckBox* nicht markiert, so wird das nächste *JPanel* geladen. Nachdem alle *JPanel* auf diese Weise abgefragt wurden, wird der Thread *Threads_Upload_Image* mit der Liste und den Einträgen der Seitenleiste als Parameter initialisiert und gestartet. Dieser Thread kopiert nun alle Bilder der Liste in den Ordner, der bei der Anmeldung als Dateiserver bestimmt wurde. Hierfür wird jedes Bild in einen *FileInputStream* umgewandelt. Anschliessend wird eine neue, leere Datei mit dem neuen Dateinamen erstellt. Der neue Dateiname setzt sich aus den Bestandteilen Pfad zum Dateiserver, Grabungsnummer, unit, subunit und Nummer zusammen. Die Dateierweiterung (.jpg, .tif, etc.) wird aus dem alten Dateinamen heraus generiert und dem neuen Dateinamen angehängt. In einem nächsten Schritt wird ein *FileOutputStream* mit der neuen Datei als Ziel erstellt. Schliesslich wird der *InputStream* zum *OutputStream* transferiert, wodurch alle Inhalte der Datei 1:1 kopiert werden.

Abschliessend werden die Informationen der Seitenleiste sowie der neue Dateiname und der Benutzername in der Tabelle documents des Schemas elkowmgis gespeichert. Hierfür wird direkt im Thread ein *PreparedStatement* ausgeführt, welches die Informationen überträgt. Der Vorteil des *PreparedStatements* hier liegt darin, dass das Hochladen der Informationen direkt innerhalb des Threads vollzogen werden kann, ohne eine neue Methode aufzurufen. Diese Variante wurde gewählt, da sie die schnellste Methode darstellt, grosse Mengen an Bildern zu verschlagworten.

7.14.3 Löschen von Bildern

Über die Schaltfläche Löschen ist es möglich, die ausgewählten Bilder aus dem Dateiserver und der Datenbank zu löschen. Hierfür führt der ActionListener der Bildverwaltung den Thread Threads_Delete_Image aus. Hierbei werden die Pfade der markierten Bilder wie bei der Verschlagwortung gespeichert und anschliessend die Liste abgearbeitet. Dabei löscht der Thread jedes Bild der Liste aus dem Dateiserver und der Datenbank. Hierzu lädt das Programm das Bild als File in den Arbeitsspeicher. Mittels der Methode *File.delete()* kann nun das Bild vom Dateisystem gelöscht werden.



Warnung:

Die Bilder lassen sich nur löschen, wenn sie nicht gleichzeitig in einer anderen Anwendung geöffnet sind.

Um den Eintrag in der Datenbank zu löschen, führt ElKowmGIS eine Löschabfrage durch. Hierbei werden alle Einträge gelöscht, bei denen das Feld document dem Dateipfad des zu löschenden Bildes entspricht. Die anderen Einträge sind davon nicht betroffen.

7.14.4 Bilder Laden und Verkleinern mittels JAI

Nachdem in den vorangegangenen Abschnitten mehrfach beschrieben wurde, dass die Bilder mittels der Java-Erweiterung Java Advanced Imaging (JAI) geladen und verkleinert werden, soll hier nun die genaue Beschreibung des Vorgehens folgen.

Das Laden von Bildern erfolgt mittels JAI. Das Programm benötigt einen Namen für die Operation und einen Dateipfad. Mittels der Methode *JAI.create(Name, Dateipfad)* wird das Bild in den Arbeitsspeicher geladen. Dabei erstellt JAI ein sog. *RenderedOp* mit dem Namen und dem Dateipfad. Durch die Konvertierung des *RenderedOps* zu einem *RenderedImage* werden die Eigenschaften des Bildes auf das neue Bild übertragen und es steht zur Weiterverarbeitung zur Verfügung.

Um Bilder zu verkleinern wird prinzipiell ähnlich vorgegangen. Hierbei wird der Methode *JAI.create* neben dem Namen ein sog. *ParamterBlock* übergeben. Dieser beinhaltet sämtliche Informationen über das neue Bild. Ein *ParameterBlock* kann mittels des Konstruktor *ParameterBlock()* erstellt werden. Anschliessend wird mittels *addSource(Quelle)* die Quelle, auf die die Veränderung angewendet werden soll, angegeben. Die weiteren Parameter können mit Hilfe der Methode *add(Parameter)* einzeln hinzugefügt. Als Quelle dient hier das Bild, das verkleinert werden soll. Die Parameter setzen sich dann aus den Skalierfaktoren in x- und y-Richtung, sowie der Verschiebung in x- und y-Richtung zusammen. Die Skalierfaktoren geben dabei an, um wie viel das Bild in der jeweiligen Richtung verkleinert werden soll. Um Verzerrungen zu vermeiden sei hier empfohlen, beide Werte gleich zu belassen. Da hier das Bild lediglich verkleinert werden soll, werden die Werte für die Verschiebung auf 0 gesetzt. Das Ergebnis ist wiederum ein *RenderedOp*, welches für die weitere Nutzung in ElKowmGIS jedoch unbrauchbar ist. ElKowmGIS benötigt für die Darstellung der Bilder *BufferedImages*. Mit Hilfe der Methode *getAsBufferedImage()* der Klasse *RenderedOp* ist es jedoch möglich, das *RenderedOp* als *BufferedImage* zurückzugeben.

```
RenderingHints hints = new RenderingHints(RenderingHints.KEY_RENDERING, RenderingHints.
VALUE_RENDER_QUALITY);
int width = orig.getWidth();
int height = orig.getHeight();
scale = 1;
if (width > height) {
    scale = (float) size / height;
} else {
    scale = (float) (size - 25) / width;
}
ParameterBlock pb = new ParameterBlock();
pb.addSource(orig);                    // Originalbild
pb.add(Float.valueOf(scale));          // Skalierung in x-Richtung
pb.add(Float.valueOf(scale));          // Skalierung in y-Richtung
pb.add(0.0F);                          // Verschiebung in x-Richtung
pb.add(0.0F);                          // Verschiebung in y-Richtung
pb.add(new InterpolationNearest());    // Interpolationsverfahren
System.setProperty("com.sun.media.jai.disableMediaLib", "true");
scaled = resizeOp.getAsBufferedImage();
```

Abb. 31: Die Methode run() des Threads zur Verkleinerung der Bilder

7.15 Die Tabellen und die Suche nach Einträgen

Die Anzeige der Einträge in der Datenbank erfolgt über ein eigenes Fenster der Klasse *Inframe_Tables*. Diese Klasse implementiert ebenfalls das *JInternalFrame* und erzeugt ein Fenster, welches mehrere Karteikarten enthält. Auf den einzelnen Karten werden die Daten der Tabellen des Schemas elkowmgis aufgelistet, die zum aktuell aktiven Projekt gehören. Da die Projekttabelle in diesem Zusammenhang irrelevant ist, wird diese nicht angezeigt. Die Projektinfos zum aktuellen Projekt können über das Menu Projekteigenschaften abgerufen werden.

Insgesamt werden vier neue *JScrollPanes* generiert, die jeweils eine Tabelle enthalten. Dabei werden für jede Tabelle Abfragen in der Datenbank ausgeführt, die den jeweiligen Inhalt zurückgeben. Für die Abfragen wurden die vier Methoden *return_Finds*, *return_Documents*, *return_Samples* und *return_Features* in der Klasse *Queries_all_queries* programmiert. Alle vier Methoden übernehmen dabei die aktuelle Projektnummer als Parameter und führen eine **SELECT-Abfrage** durch. Abbildung 32 zeigt exemplarisch den Quelltext der Abfrage *return_Finds*.

```
Statement stmt = queries.Queries_connect_db.conn.createStatement(ResultSet.TYPE_SCROLL_IN-
SENSITIVE, ResultSet.CONCUR_UPDATABLE);

String sql_statement = "Select * from elkowmgis.context WHERE excavationid = '" + projid + "' AND
codeid=1 ORDER BY id";

set = stmt.executeQuery(sql_statement);
```

Abb. 32: Routine zur Abfrage aller Funde des Projektes mit der Nummer projid

Hierbei ist zu beachten, dass die Abfragen zwei Bedingungen besitzen:

- Zum einen wird die Projektnummer als Bedingung gesetzt (… WHERE excavationid = projid …)
- Zum anderen ist die codeid entscheidend (… AND codeid= …)

Dadurch werden nur die Ergebnisse geliefert, die zum Projekt gehören und gleichzeitig mit der codeid übereinstimmen. Durch die Definition der codeid ist klar, welche codeid welches Objekt zurückgibt. Um beispielsweise alle Funde zu finden, muss die codeid gleich eins sein.

Auch in diesem Fenster stellt die Seitenleiste die Tools zur Suche innerhalb der Datenbank zur Verfügung. Ähnlich der Bildverwaltung werden in der Seitenleiste mehrere Eingabefelder angelegt, in die die Suchbegriffe eingetragen werden können. Ein Klick auf diese Schaltfläche Suchen führt die Suche aus. Das Programm generiert für die Ergebnisse eine neue Karteikarte im Tabellenfenster. Dabei werden die Ergebnisse ebenfalls tabellarisch dargestellt. Da der Vorgang der Suche prinzipiell dem Suchvorgang in der Bildverwaltung entspricht, soll hier nicht näher darauf eingegangen werden.

7.16 Der Verwaltungsbereich

Die Admin-Area wird nur denjenigen Benutzern angezeigt, die in der Datenbank als Administratoren eingetragen sind. Um dies zu überprüfen wird bereits während der Verbindung mit dem Server beim Programmstart ermittelt, ob der Benutzer Administrator ist. Hierfür wird der Eintrag der Tabelle def_user gesucht, bei dem der Benutzername dem eingegebenen Namen entspricht. Im Ergebnis selbst wird dann ermittelt, ob der Eintrag in der Spalte sysrecht ein A enthält. Ist dies der Fall, so wird der Benutzer als Administrator gespeichert. Gleichzeitig werden die Berechtigungen der einzelnen Projekte in einer HashMap gespeichert. Die HashMap bietet die Möglichkeit, viele Daten unsortiert zu speichern und sie mit Hilfe eines Schlüsselwertes wiederzufinden (Ullenboom 2012, S. 992). ElKowmGIS verwendet dabei die Projektnummer als Schlüssel. Somit ist gewährleistet, dass die Berechtigungen schnell wieder abgerufen werden können.

Bei dem eigentlichen Verwaltungsbereich handelt es sich um ein *JInternalFrame* der Klasse *Inframe_admin*. Dieses Fenster enthält zwei Karteikarten über die entweder die Benutzer oder die Projekte verwaltet werden können. Hierfür werden zwei verschiedene *JPanel* der Klassen *Panel_user_admin* und *Panel_project_admin* angelegt.

Die Klasse *Panel_user_admin* erzeugt ein JPanel, welches zwei Tabellen, drei Schaltflächen und eine *JCheckBox* enthält. Die erste Tabelle listet alle Benutzer und Ihre Funktion auf, die in der Tabelle def_user

der Datenbank gespeichert sind. Hierfür wird eine simple SELECT-Abfrage in der Datenbank ausgeführt. Diese Tabelle ist mit einem *MouseListener* der Klasse *Listener_tabsel* versehen, der bei Auswahl eines Benutzers die zweite Tabelle sichtbar macht. Hierfür wird das *MouseEvent*, welches beim Klicken auf einen Benutzer ausgelöst wird, entgegengenommen und in der Tabelle überprüft, welche Zeile ausgewählt wurde. Im nächsten Schritt wird in der Datenbank abgefragt, für welche Projekte der Benutzer welche Berechtigungen besitzt. Hierbei werden zuerst die Projekte aufgelistet, für die Berechtigungen vorhanden sind. Die Projekte, für die der Benutzer (noch) keine Berechtigungen besitzt, werden im unteren Bereich der Tabelle angezeigt. Für die Darstellung der *JCheckBoxen* in der zweiten Tabelle wurde ein eigenes *TableModel* geschrieben. Dies war notwendig, da die *JTables* in Java üblicherweise nur Texteinträge gestatten. Somit wäre die Tabelle mit den Einträgen WAHR oder FALSCH gefüllt gewesen. Zur optischen Hilfe wurde jedoch entschieden, die Zelleneinträge als Checkboxen anzulegen.

```
import javax.swing.table.AbstractTableModel;
import javax.swing.table.DefaultTableModel;

public class UserCellEditor extends AbstractTableModel {
    DefaultTableModel m;
    private String[] columnNames;
    public UserCellEditor(DefaultTableModel mod, String[] cols) {
        this.m = mod;
        this.columnNames = cols;
    }
    @Override
    public Object getValueAt(int rowIndex, int columnIndex) {
        return m.getValueAt(rowIndex, columnIndex);
    }
    public Class getColumnClass(int c) {
        return getValueAt(0, c).getClass();
    }
    public void setValueAt(Object value, int row, int col) {
        m.setValueAt(value, row, col);
        fireTableCellUpdated(row, col);
    }
}
```

Abb. 33: Erstellung des TableModel zur Anzeige der Checkboxen anstatt einer Textrepräsentation des Wertes

Die Klasse *UserCellEditor* erweitert das *AbstractTableModel* und überschreibt dabei sämtliche Methoden. Entscheidend hierbei ist, dass die Methode *getValueAt(int rowIndex, int columnIndex)* überschrieben wird. Durch ein einfaches *return m.getValueAt(rowIndex, columnIndex)* wird die Methode so überschrieben, dass in der Tabelle die CheckBoxen anstelle des Textes erscheinen. In der zweiten Tabelle können nun die Berechtigungen eines Benutzers für die einzelnen Projekte eingestellt werden. Ein Klick auf die Schaltfläche führt die Aktualisierung der Berechtigungen des Benutzers in der Datenbank aus. Hierfür wird eine Update-Abfrage an die Datenbank gesendet.

Die Schaltfläche Neu öffnet ein Dialogfenster, in welchem ein neuer Benutzer angelegt werden kann. Da gleichzeitig ein neuer Benutzer für die komplette Datenbank angelegt wird, ist es notwendig, die Passwortfelder auszufüllen. Die Wiederholung des Passworts dient lediglich dazu Rechtschreibfehler zu verhindern. Gleichzeitig werden alle vorhandenen Projekte aufgelistet. Hier können auch direkt Berechtigungen vergeben werden.

Die Schaltfläche Löschen ermöglicht es, Benutzer aus ElKowmGIS zu löschen. Die Checkbox direkt neben der Schaltfläche bestimmt hierbei, ob der Benutzer nur aus ElKowmGIS oder aus dem gesamten Datenbankserver gelöscht wird.

Wird der Benutzer gelöscht, so wird der entsprechende Eintrag sowohl in der Tabelle der Datenbank wie auch in der Tabelle im Verwaltungsbereich gelöscht.
Wird der Benutzer aus der kompletten Datenbank gelöscht, so werden zuerst alle Berechtigungen des Benutzers entfernt und anschliessend der Benutzer aus der DB gelöscht. Die Einträge in den Spalten History, die den Verlauf eines Objektes beschreiben, bleiben aber weiterhin erhalten. Damit bleibt nachvollziehbar, wer ein Objekt erstellt oder geändert hat.
Das Panel *Panel_project_admin* stellt im Prinzip den zweiten Ansatz zur Benutzersteuerung zur Verfügung. Statt jedem Benutzer seine Projekte zuzuweisen, ist es hier möglich, in einem Projekt mehreren Benutzern gleichzeitig Berechtigungen zu vergeben. Hierfür werden im Fenster eine *JList* für die Projekte, eine Tabelle für die Berechtigungen und drei Schaltflächen angelegt. Die *JList* erhält ihre Einträge direkt aus der Datenbank und ist mit einem *ListSelectionListener* versehen. Dieser Listener bestimmt dabei das Geschehen, wenn ein Eintrag in der Liste ausgewählt wird. Hierbei werden alle Benutzer und ihre Berechtigungen in der Tabelle def_user abgefragt und überprüft, wer Zugriff aus das ausgewählte Projekt besitzt. Falls Berechtigungen vorhanden sind, werde die Benutzer zuerst der Tabelle hinzugefügt. Die restlichen Benutzer erscheinen im unteren Teil der Tabelle. Dieser Tabelle wurde das gleiche *TableModel* wie der Projekttabelle der Benutzerverwaltung zugewiesen. Ein Klick auf die Schaltfläche Update ermöglicht es, Änderungen an die Datenbank zu übermitteln. Hierbei sei aber darauf hingewiesen, dass ein Update für jedes Projekt einzeln durchgeführt werden muss. ElKowmGIS speichert die Änderungen beim Wechsel des Projektes nicht. Die Schaltfläche Neu öffnet den Dialog zum Erstellen eines neuen Projektes, während die Schaltfläche Löschen die Abfrage öffnet, ob das ausgewählte Projekt wirklich gelöscht werden soll. Das Bestätigen des Dialogs mit Ok löst dasselbe Ereignis wie die Schaltfläche Löschen der Projektauswahl im Abschnitt Projektauswahl aus.

7.17 Drucken

Um die Druckfunktionen in ElKowmGIS umzusetzen wurde eine neue Klasse erstellt, die das Interface *Printable* implementiert. Die Klasse *Printer_Map* enthält lediglich die Funktion print, welche die Druckausgabe verarbeitet. Hierfür wird das zu druckende Element in ein *Graphics2D*-Objekt umgewandelt. Anschliessend wird sie mit Hilfe der Funktion *translate* in die Druckränder eingepasst. Im nächsten Schritt wird die Hintergrundfarbe des Bildes bestimmt und mit der Funktion *fill* die Grafik gezeichnet. Abschliessend wird der Wert *Page_Exists* zurückgegeben. Abschliessend wird dem Java-eigenen *PrinterJob* das soeben erstellte Objekt als *Printable*-Objekt übergeben und mit der Funktion *print* gedruckt.

Dieses Verfahren dient vor allem dem Druck des aktuellen Kartenausschnitts. Um die Datentabellen zu drucken, bedient sich ElKowm der Java-eigenen Druckfunktion der *JTable*. Hierfür muss im Quelltext lediglich die Funktion *print()* auf der aktiven Tabelle ausgerufen werden. Dadurch öffnet sich der Druckdialog, indem nochmals die Seitengrösse eingestellt werden kann. Durch die Schaltfläche ok wird der Druck gestartet.

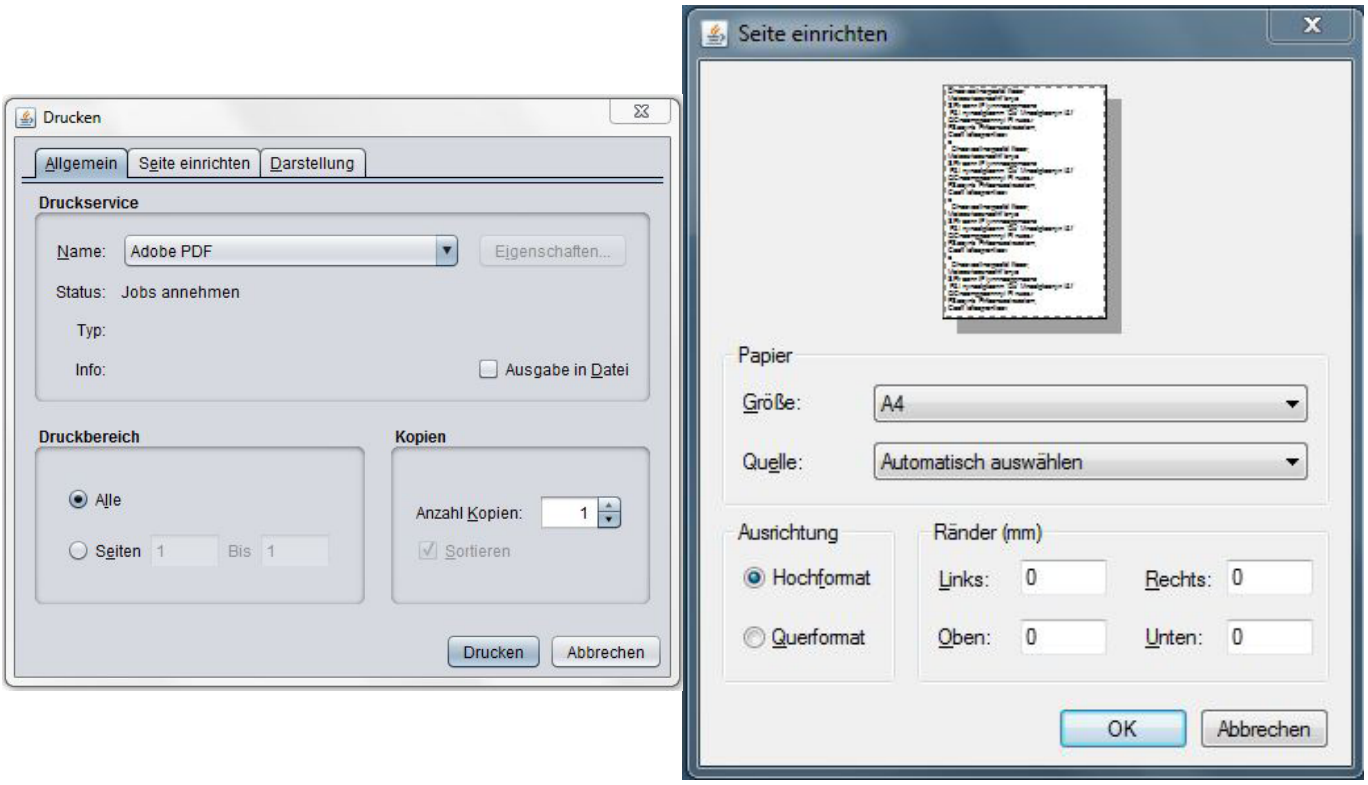


Abb. 34: Druckdialog (links) und die Seiteneinrichtung (rechts) in ElKowmGIS

8 Teststudie Hummal

8.1 Import bestehender Daten

8.1.1 Vorbereitungen

Die Daten der Grabungen in ElKowm liegen in verschiedenen Formaten vor. Die wichtigsten Daten stellen hierbei die Topografie, das Fundjournal, die Flächenfotos und das Probenjournal dar. Die Topografie liegt in Form einer AutoCAD-Zeichnung vor, welche für den Import lediglich in eine **dxf-Datei** umgewandelt werden muss. Das Fundjournal und das Probenjournal liegen als Excel-Liste vor, die als **csv-Datei** umgespeichert werden, um sie mit ElKowmGIS verarbeiten zu können. Dies sind die einzigen Transformationen, die an den Daten vorgenommen werden müssen. Die Flächenfotos liegen als Bilder im tiff-, jpeg- oder bmp-Format vor. Diese Dateitypen werden von den Routinen in ElKowmGIS ohne Weiteres unterstützt.

Um nun die Daten in ElKowmGIS zu nutzen, wird ein neues Projekt erstellt, das die Daten aufnimmt. Im Folgenden soll nun Schritt für Schritt erläutert werden, wie die Daten in ElKowmGIS importiert werden. Zuerst wird über Neues Projekt das Projekt erstellt. Hierbei werden der Projektname, der Grabungsname etc. eingegeben. Um das Projekt von anderen Projekten unterscheiden zu können, wird die Grabungsnummer mit HU_0001 bezeichnet. Als Phasen werden in diesem Projekt die Komplexnummern verwendet, da es nicht ausreichend Datierungen der Fundstellen gibt, um eine feinere Phasenunterteilung vorzunehmen. In erster Linie dient die Phaseneinteilung in diesem Fall nur der Kulturzuweisung der Objekte.

Als Fundkategorien werden die folgenden angelegt:

- Silex Abkürzung: sx Farbe: Rot (255, 0, 0) Layer: Silex Status: verbrannt, nicht in situ, modern beschädigt, Probe, nicht fotografiert
- Knochen Abkürzung os Farbe: Blau (0, 0, 255) Layer: Knochen, Status: ausgeschieden, verbrannt, nicht in situ, modern beschädigt, Probe, nicht fotografiert, Phantom
- Geroell Abkürzung: ger Farbe: Gelb (255, 255, 0) Layer Geroell Status: ausgeschieden, verbrannt, nicht in situ, modern beschädigt, Probe, nicht fotografiert
- Sonstiges Abkürzung: son Farbe Grün (51, 255, 0) Layer Objekt Status keine

Befunde sind in Hummal keine vorhanden, aus diesem Grund werden hier keine Einträge benötigt.

Die Probenbezeichnungen werden wie vom Programm vorgeschlagen übernommen.

Bei den Layern werden die Vorschläge ebenfalls übernommen und um den Eintrag Geroell ergänzt.

- Geroell Abkürzung ger, Farbe:155, 155, 155

Mit diesen Einstellungen ist ElKowmGIS bereit, die Daten aufzunehmen.

8.1.2 Import und Vergleich der CAD- und Tabellendaten

Um die verschiedenen Datensätze zu importieren, sind mehrere Schritte notwendig. Zuerst soll das Fundjournal geladen werden. Hierzu öffnet man unter Extras das Menu Importiere Tabelle und wählt die csv-Datei des Fundjournals aus. Im Dialogfenster werden nun die Spaltennamen der alten Datei mit den Spaltennamen von ElKowmGIS verknüpft:

Excavation-ID:	nv	Für Alle: HU_0001
Unit:	Komplex	
Sub-Unit:	Niveau	
ID:	ID	
Code:	nv	Für Alle: 1 (Funde)
Type:	Fundgattung	
Level:	Niveau	
State:	Status	
Remarks:	Bemerkungen	
Layer:	Layer	
Shape:	nv	Für Alle: 7 (Punkt)
X:	X	
Y:	Y	
Z:	Z	

Mit Ausführung des Imports werden die Daten in die Datenbank geladen. Nach erfolgreichem Import schliesst sich das Fenster. Zur Kontrolle wechselt man nun in das neu erstellte Projekt und betrachtet die Fundtabelle. Insgesamt finden sich nun 15074 Objekte in der Datenbank von ElKowmGIS. Dies entspricht der Anzahl Funde in der Datei Fundjournal.xls.
Die Stichprobe mit den Silices zeigt, dass auch die korrekte Anzahl der einzelnen Kategorien importiert wurden. In der Datei finden sich 8455 Einträge, die als Fundgattung das Kürzel sx tragen. Filtert man nun die Daten in ElKowmGIS nach der Fundgattung Silex, so finden sich 8455 Einträge in der Datenbank. Betrachtet man nun die geografische Verteilung der Objekte im GIS-Fenster von ElKowmGIS, so erhält man die Darstellung aus Abbildung 35.

Im nächsten Schritt soll nun die Topografie von ElKowmGIS importiert werden. Hierzu wählt man den Eintrag Extras Importiere dxf und wählt die Datei HU_topo.dxf aus. Der Import läuft vollautomatisch und fügt ElKowmGIS die Objekte der dxf-Datei hinzu. Die Überprüfung, ob alles importiert wurde, ist

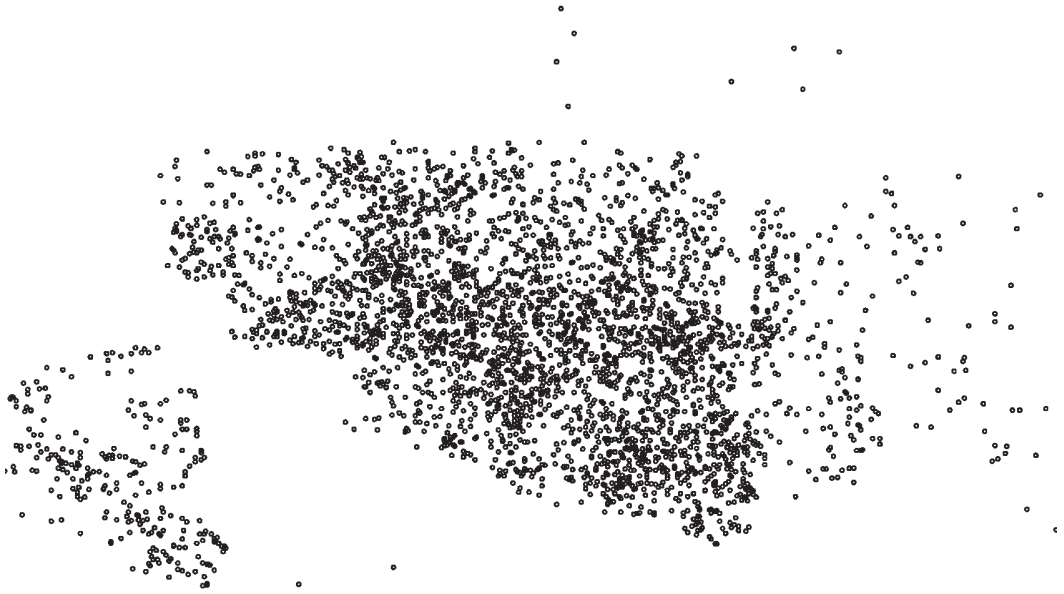


Abb. 35: Ausschnitt aus der Fundverteilung des Sektors West in Hummal.

mit ElKowmGIS nur grafisch möglich. Hierzu öffnet man das GIS-Fenster erneut und schaltet die Layer Knochen, Silex und Geroell durch deaktivieren des ersten Kästchens auf unsichtbar. Dadurch erhält man das Bild wie in Abbildung 36 zu sehen.

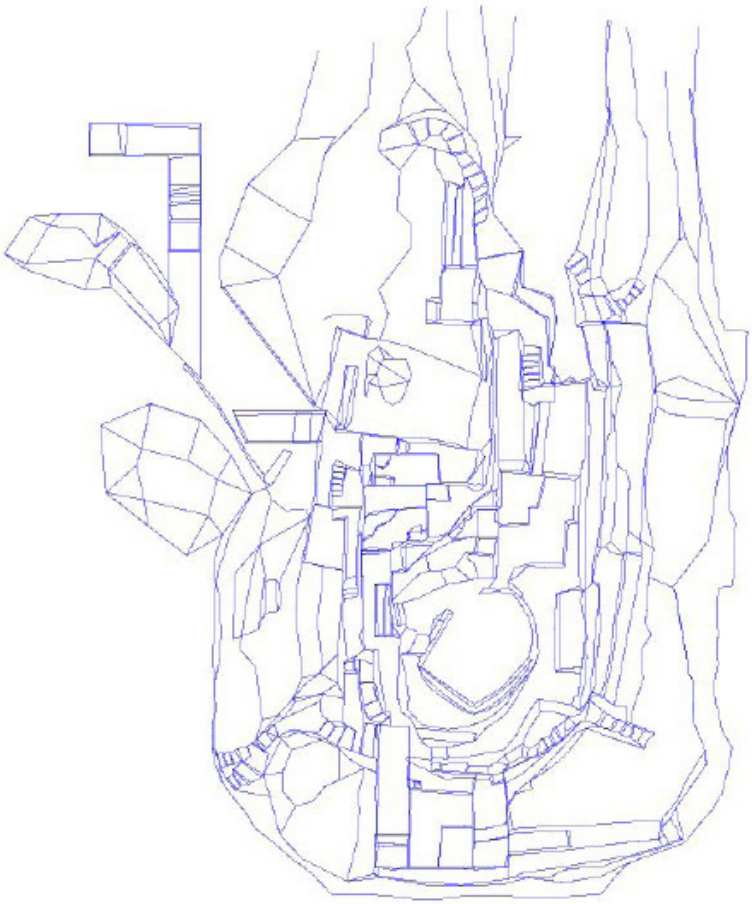


Abb. 36: Topografie von Hummal ohne Funde und Proben.

Um jedoch auch einen rechnerischen Vergleich zu erhalten, wird auf die Möglichkeiten des PostgreSQl-Servers zurückgegriffen. Über seine Kommandozeile kann folgende Abfrage gestartet werden:

"Select count(id) from elkowmgis.context where excavationid like 'HU_0001' AND shape_id = 8".

Diese Abfrage sucht in der Tabelle context alle Objekte, die der Grabung HU_0001 angehören und die shape_id 8 besitzen. Bei Objekten mit einer shape_id von 8 handelt es sich um Linien.
Dieser Trick ist aus zwei Gründen möglich. Zum einen wurde die Topografie von Hummal bisher nur mit Einzellinien aufgenommen. Zum anderen handelt es sich um ein neues, leeres Projekt. Dadurch können keine Linienobjekte ausser den Importierten vorhanden sein. Die Abfrage liefert ein Ergebnis von 2525 Objekten zurück. Betrachtet man die Datei HU_topo.dxf in AutoCAD und markiert alle Objekte, so erhält man insgesamt 2525 Objekte. Dies bedeutet, dass ElKowmGIS alle Linien der Topografie importiert hat. Die Kombination von Linien und Funden ergibt schliesslich das Bild aus Abbildung 37. Für den Import der Proben wird gleich vorgegangen wie bei den Funden. Man wählt Importiere Tabelle aus und gibt die Datei an, die die Informationen über die Proben enthält. Anschliessend weist man den Feldern die entsprechende Spalte in der Tabelle zu.

Die Überprüfung des erfolgreichen Importvorgangs kann nun wieder über die Tabelle vorgenommen

Abb. 37: Ausschnitt aus dem Gesamtplan Hummal. Angezeigt werden die Proben und Gerölle in den Sektoren West und Ost.

werden. Hierzu wählt man in der Tabellenansicht lediglich den Reiter Proben aus und filtert nach den einzelnen Probengattungen. Die Probendatei enthält beispielsweise 424 Proben der Kategorie TL. ElKowmGIS findet in der frischen Datenbank ebenfalls 424 Proben der Kategorie TL.

Den aufwendigsten Teil des Imports stellt der Import der Bilder dar. Hierzu muss erwähnt werden, dass die bereits referenzierten Bilder, welche mit Photoplan und AutoCAD verarbeitet wurden, nicht mit derselben Methodik referenziert wurden, wie es in ElKowmGIS vonstatten geht. Hinzu kommt die Tatsache, dass das Plugin Photoplan mit einer eigenen Entzerrungsebene arbeitet. Das bedeutet, dass durch die Referenzpunkte eine Ebene aufgespannt wird, die ein lokales Koordinatensystem repräsentiert. Die Koordinaten, die in den beiden Zusatzdateien abgelegt werden, stammen aus diesem System. Zwar findet sich in der zugehörigen .prk-Datei eine Korrelationstabelle zwischen den lokalen Koordinaten und den Bildkoordinaten (den Koordinaten der Referenzpunkte im Bild selbst), jedoch lässt sich diese Tabelle erstens nicht einfach auslesen, da das Trennzeichen zwischen den Spalten aus Leerzeichen, deren Anzahl je nach Länge der Spaltenwerte variieren kann und zweitens findet sich in der ersten Spalte keine eindeutige Identifikation, die es erlauben würde, jede Zeile der Tabelle getrennt anzusprechen. Die erste Spalte enthält vielmehr die Namen der Bildpunkte. Dieser kann jedoch so variabel sein, dass es programmiertechnisch nicht möglich ist, alle Varianten abzufangen.

Eine weitere Erschwernis stellt die Tatsache dar, dass die Bilder von Photoplan nicht rechtwinklig abgespeichert werden. Sie werden zwar in der Vorschau und in den Bildverarbeitungsprogrammen parallel zu den Bildschirmrändern angezeigt, jedoch sind die Bilder intern mit einer Drehung versehen. Das bedeutet, dass die Abmessungen und der Ursprung der Bilder immer auf das lokale Koordinatensystem der Referenzierungsebene Bezug nehmen. Dies führt wiederum zu den Problemen, wie sie im oberen Abschnitt beschrieben wurden.
Aus diesen Gründen wurde entschieden, den Import der referenzierten Bilder mit einer Bedingung zu verknüpfen. Um Bilder, welche mit Photoplan referenziert wurden, in ElKowmGIS zu importieren, müssen die Bilder ein sog. World-File besitzen.

Das Menu von Photoplan erlaubt es, für jedes referenziertes Bild ein World-File zu generieren. Hierbei ist

Hinweis:

Bei einem World-File handelt es sich um eine Datei, die angibt, wie gross das Bild ist, wo der Ursprung des Bildes in Weltkoordinaten liegt und mit welcher Drehung das Bild importiert werden muss.

es auch möglich, gleichzeitig für mehrere Bilder die Dateien zu erzeugen. Hierfür werden einfach alle Bilder, für die eine solche Datei generiert werden soll, in eine neue CAD-Datei importiert und über das Menu Photoplan World-File erzeugen alle Bilder ausgewählt. Photoplan generiert anschliessend selbstständig die notwendigen Dateien. Anschliessend lassen sich in ElKowmGIS über das Menu Import Photoplan die Bilder automatisch importieren. Hierbei liest ElKowmGIS die Namen aller Bilder ein und überprüft, ob

ein World-File vorhanden ist. Die Bilder, die ein World-File besitzen, werden importiert, während für die Bilder, die noch kein World-File besitzen, eine Zeile in den Log des Programms eingetragen wird. Somit lässt sich überprüfen, ob alle Bilder importiert wurden, oder nicht.

Um die Daten nun zwischen AutoCAD und ElKowmGIS vergleichen zu können, wird ein Teil der Bilder aus der Grabung Hummal mit einem World-File versehen und in ElKowmGIS importiert. Zusätzlich werden die Referenzpunkte für jedes Bild in ElKowmGIS importiert. Im nächsten Schritt werden die Punkte und die Bilder gleichzeitig angezeigt. Mit dem Messen-Tool aus ElKowmGIS und AutoCAD werden nun manuell die Abstände zwischen den Punkten in Weltkoordinaten und den Punkten auf den Bildern gemessen (Abb. 38). Das Beispiel aus Abbildung 38 zeigt auf, dass ElKowmGIS die Bilder an die korrekte Position setzt.

Abb. 38: Ausschnitt aus einem referenzierten Bild. In Cyan sind die Passpunkte aus ElKowmGIS zu sehen.

8.1.3 Vergleich der GIS-Funktionalität von ElKowmGIS

Nachdem mit diesen einzelnen Tests die Importfunktionalitäten von ElKowmGIS aufgezeigt wurden, sollen nun die GIS-Funktionen mit einer kommerziellen Software (**ArcGIS**) verglichen werden.
Um diese Funktionalitäten aufzeigen zu können, werden verschiedene Auswertungen vorgestellt. Im ersten Beispiel handelt es sich um die Untersuchung der Zusammenhänge zwischen Schicht- und Objektorientierungen, die im Rahmen der Diplomarbeit des Autors dieser Arbeit (Schuhmann 2007) mit Hilfe verschiedener Softwareanwendungen erstellt wurden. Ziel hierbei war es herauszufinden, ob die Neigung und die Hanglage einer Schicht aus dem Komplex des Yabroudiens in der Fundstelle Hummal direkt im Zusammenhang mit der Orientierung der zur Schicht zugehörigen Artefakte steht. Dieser Aspekt ist in Hummal von besonderem Interesse, da die Schichten in Hummal eine Schichtneigung aufweisen, die annehmen lässt, dass die Brunnensituation in der Fundstelle einen direkten Einfluss auf die Schichtverhältnisse hat (Schuhmann 2007, S. 24ff.). Um die Werte für die Hanglage und –neigung der Schichten in Hummal zu berechnen, wurde aus den Profilinformationen für jede Schicht ein Oberflächenmodell berechnet. Anschliessend konnten mit Hilfe von **GRASS GIS** die Durchschnittswerte für die einzelnen Schichten bestimmt werden. Da ElKowmGIS nicht in der Lage ist, aus einer Oberfläche die

Hanglage- und –neigung zu berechnen, sollen hier nur die Oberflächen der Niveaus 8 und 10 exemplarisch mit den Ergebnissen der Diplomarbeit verglichen werden. Zu berücksichtigen ist auch, dass ElKowmGIS mit einer Inverse-Distance-Weighted-Interpolation arbeitet, während die Daten des o.a. Artikels mit Hilfe einer Minimum-Curvature-Interpolation (siehe unten) berechnet. Dies kann dazu führen, dass die Ergebnisse voneinander abweichen.

Für die Erstellung der Schichtoberflächen werden die Punktinformationen in ElKowmGIS geladen, wobei die Informationen nach den Niveaus getrennt werden. Anschliessend wird mit Hilfe der Oberflächeninterpolation die jeweilige Oberfläche berechnet. Als Abstände zwischen den zu berechnenden Punkten werden fünf Zentimeter gewählt. Dieser Wert wurde für die Berechnung mit ElKowmGIS gewählt. Um die Vergleichbarkeit zu gewährleisten, sollten beide Modelle den gleichen Zellabstand besitzen. ElKowmGIS benötigt für die Berechnung der Oberfläche von Schicht 8 2.5 Minuten und von Schicht 10 2.9 Minuten. Um die Dauer der Berechnung zu berechnen, wurde ElKowmGIS temporär mit einem Zeitmesser versehen. Der Zeitmesser wird aktiviert, sobald die Schaltfläche Ok im Fenster der Geländemodellierung gedrückt wird und wird beendet, sobald der Thread Threads_gen_DGM beendet wurde. Dadurch wird gewährleistet, dass die reine Rechenzeit ausgegeben wird. Die Zeiten, die für die Eingabe der Daten, sowie die Sicherung und Anzeige des fertigen Modells benötigt werden, werden nicht berücksichtigt.

Um nun die Modelle miteinander vergleichen zu können, werden alle Modelle in ArcGIS importiert. Die erste Überprüfung der Richtigkeit der Daten aus ElKowmGIS erfolgt über die Positionierung der Modelle. Die beiden Modelle der Schicht 8 müssten übereinanderliegen, da sie über dieselbe Fläche berechnet wurden. Gleiches gilt für die Modelle der Schicht 10. Über diese Ansicht lässt sich die Positionierung am besten überprüfen. Um anschliessend die Unterschiede in den Berechnungen der Oberflächen zu untersuchen, können mit Hilfe von ArcGIS beide Karten voneinander subtrahiert werden. Hierbei werden die Zellenwerte, die in diesem Fall der Höhe entsprechen, der beiden Modelle für jeden Punkt auf der Karte extrahiert. Anschliessend werden die beiden Werte voneinander subtrahiert. Dabei wird das Modell aus ElKowmGIS von dem Modell aus GRASS GIS abgezogen. Dies bedeutet, dass die Karte aus GRASS GIS als Referenz dient. Da beide Modelle identisch sein sollten, von den Unterschieden der Berechnungsalgorithmen abgesehen, müsste ArcGIS eine Karte produzieren, die nur Nullwerte enthält.

Abbildung 39 zeigt die Position der Karten aus GRASS-GIS und ElKowmGIS in unterschiedlichen Farben für

Hinweis:

Der Minimum-Curvatures-Algorithmus (MC) wurde von I. C. Briggs 1974 vorgestellt (Briggs 1974) und stammt aus dem Bereich der geophysikalischen Datenverarbeitung. Briggs ging es in erster Linie darum, die Ergebnisse geophysikalischer Untersuchungen mittels Konturlinien darzustellen. Im Laufe der Zeit wurde die Methodik auch für die Darstellung von Geländemodellen angewandt.

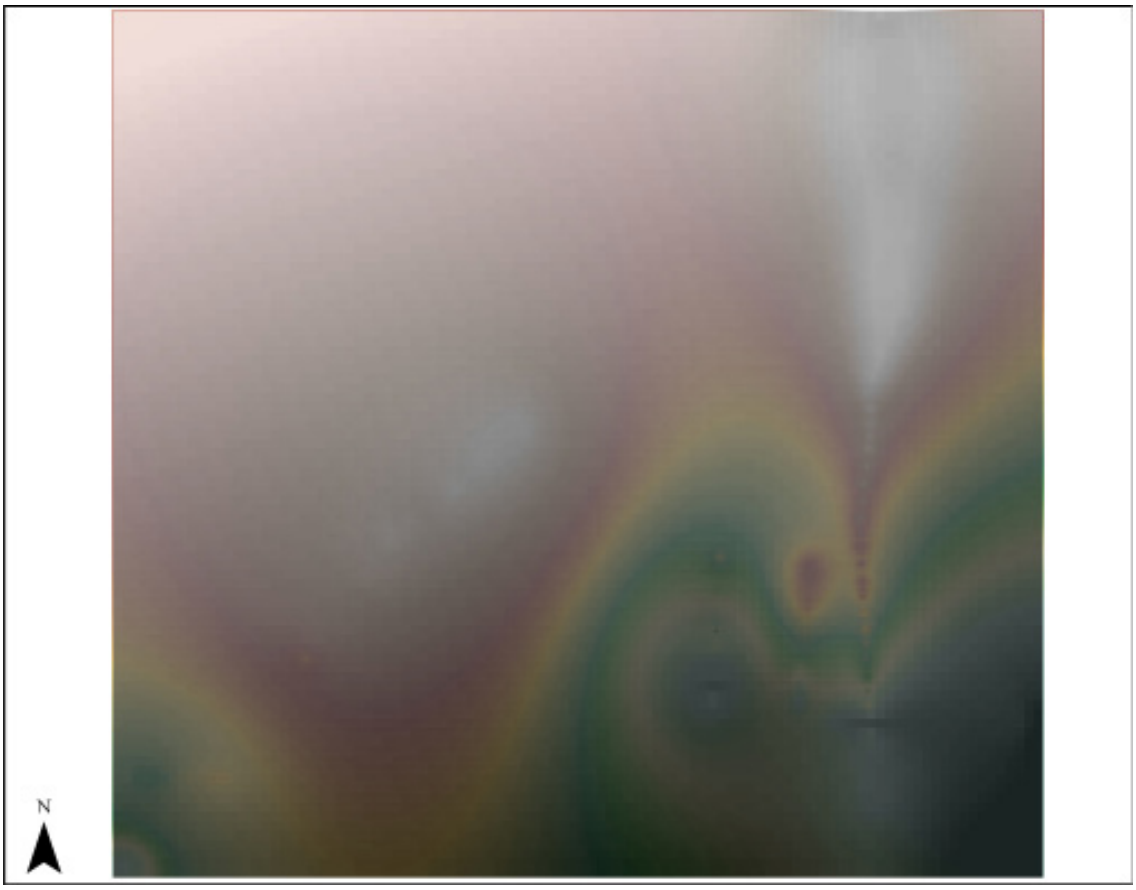


Abb. 39: Positionierung der Geländemodelle der Schicht 8. Farbig das Modell aus ElKowmGIS, in Graustufen und transparent das Modell aus GRASSGIS.

die Modelle der Schicht 8. Man stellt fest, dass beide Karten den gleichen Ausschnitt überdecken. Somit wird die Positionierung des Modells durch ElKowmGIS richtig vorgenommen.

Vergleicht man nun die Karteninhalte, so zeigt die Differenzkarte (Abb. 40) der beiden Modelle der Schicht 8, dass mehr übereinstimmt, als auf den ersten Blick vermutet. In den Abschnitten, in denen Punkte vorhanden sind, zeigen sich keine Unterschiede zwischen den beiden Modellen. Lediglich in den Bereichen, in denen das Modell nur auf berechneten Daten beruht, zeigen sich deutliche Unterschiede. Dabei muss berücksichtigt werden, dass es sich um die Randbereiche und Bereiche, in denen die Punkte linear angeordnet sind.

Um die Herkunft der Unterschiede zu verstehen, muss deutlich gesagt werden, dass die beiden Interpolationsmethoden, die den Modellen zugrunde liegen, sich deutlich unterscheiden. Während die Karten aus GRASS-GIS mit Hilfe eines Minimum-Curvatures-Algorithmus (MC) berechnet wurden, nutzt ElKowmGIS den Inverse-Distance-Weighted-Algorithmus (IDW). Der MC-Algorithmus produziert im Prinzip eine gebogene Ebene. Dabei entspricht der Wert der Ebene an einem Messpunkt dem Messwert. Zwischen den Messpunkten und zu den Rändern hin wird die Ebene interpoliert. Dabei wird die Ebene so berechnet, dass ihre Biegung so gering wie möglich ist. In genau dieser Tatsache liegt nun die Ursache für die Unterschiede in den Oberflächenmodellen. Der MC versucht eine Ebene zu erstellen, die eine möglichst geringe Biegung besitzt. Damit liegen die Werte der Ebenenpunkte immer nahe bei den Messwerten. Der

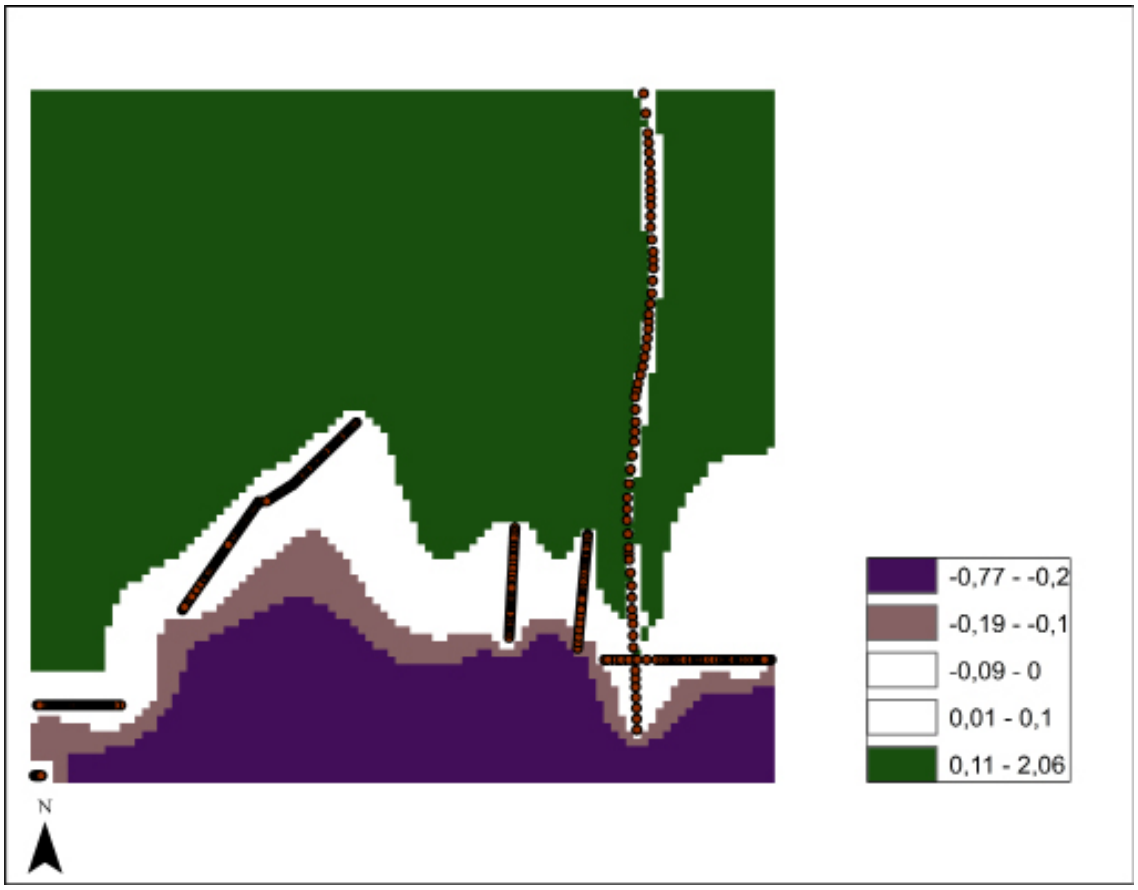


Abb. 40: Differenzkarte der Geländemodelle der Schicht 8. Deutlich zu sehen sind die minimalen Abweichungen im Bereich der Profile.

IDW, wie er in ElKowmGIS genutzt wird, benötigt für die Berechnung eine Mindestanzahl an Messwerten. Liegen nicht genügend Punkte um den zu berechnenden Wert, so wird der berechnete Wert auf den

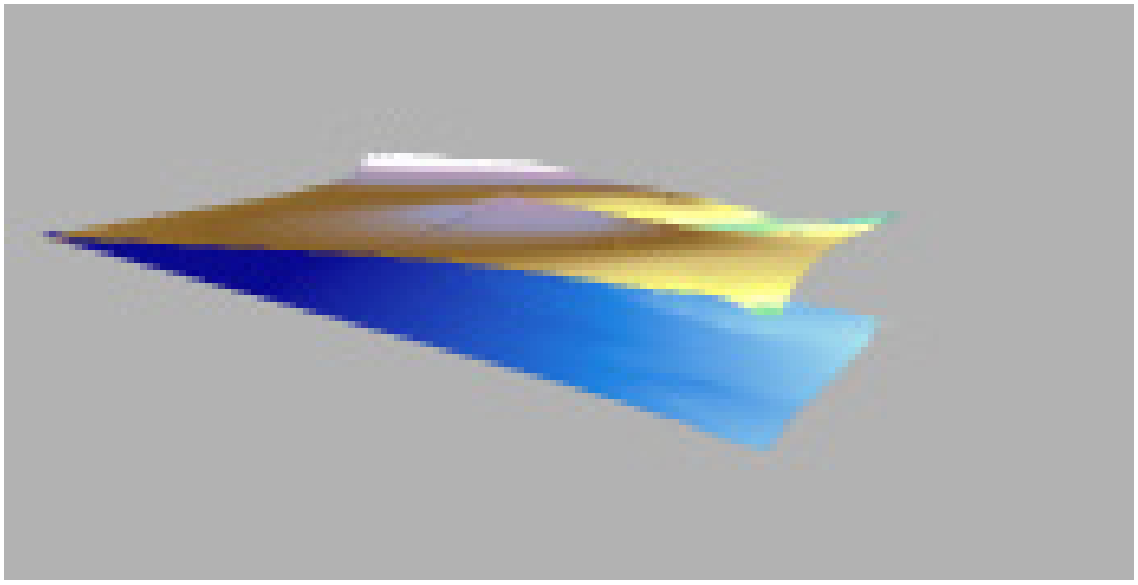


Abb. 41: Die beiden Geländemodelle im Raum. Das Ergebnis des MC ist unten in Blautönen dargestellt. Um zwei Karteneinheiten (Meter) nach oben versetzt findet sich die IDW-Interpolation. Der Versatz dient hier lediglich der Veranschaulichung und hat keinen Einfluss auf das Ergebnis der Berechnung.

nächst-gelegenen Messwert gesetzt. Dies führt zu einer stärkeren Biegung der Ebene. Abbildung 41 zeigt die beiden Ebenen im Raum, wodurch die oben angeführte Problematik veranschaulicht wird.

Für die Interpretation werden die Höhenwerte der Differenzkarte in 29 10 Zentimeter-Klassen zusammengefasst. Die Differenzwerte variieren zwischen -0.77 m und 2.1 m. Insgesamt wurden 9300 Werte berechnet. In den Klassen -0.1 – 0 und 0 – 0.1 liegen 1648 Werte. Dies bedeutet, dass sich ungefähr ein Fünftel der Werte der beiden Modelle um maximal 10 Zentimeter unterscheiden. Jedoch lassen sich bei genauerer Betrachtung der Datenlage (Abb. 40) zwei wichtige Aussagen treffen. Erstens sind die Punkte nicht gleichmässig über die gesamte Fläche verteilt. Dies führt gerade in den Randbereichen zu massiven Störungen, da dort Informationen fehlen. Zum Zweiten sind die vorhandenen Punkte meist linear angeordnet. Bedenkt man jedoch, dass es sich lediglich um Profilinformationen handelt, so ist dies nur logisch. Diese Tatsache führt aber auch dazu, dass die Interpolationen nicht richtig funktionieren können, da in vielen Fällen lediglich eine Richtung (x- oder y-Richtung) mit mehreren Werten abgedeckt ist, während die andere Richtung lediglich durch einen Wert repräsentiert wird. Für die Berechnung der Schicht 8 standen 7 Profile zur Verfügung, wovon zwei in Richtung Ost-West, drei in Richtung Nord-Süd und eines schräg dazu verlaufen. Das siebte Profil ist so klein (Länge: 15 cm), dass es im Vergleich zu den

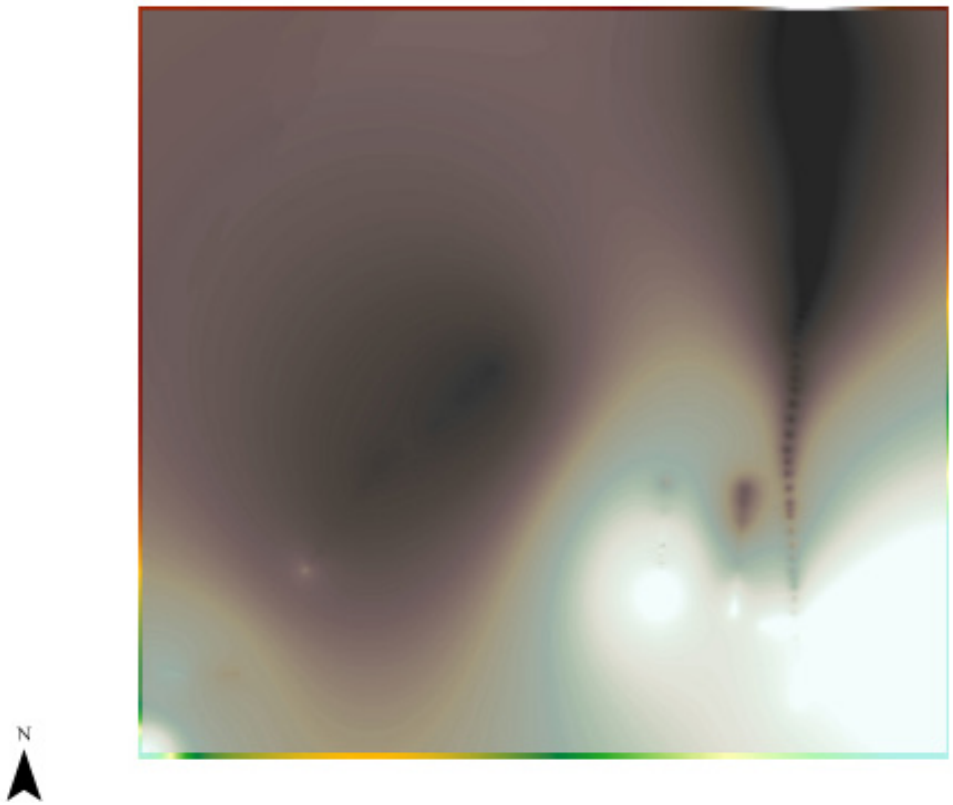


Abb. 42: Geländemodelle der Schicht 8 mittels IDW-Interpolation. Farbig das Modell aus ElKowmGIS, in Graustufen und transparent das Modell aus ArcGIS.

anderen Profilen keine Aussagekraft besitzt und eher als Einzelpunkt gewertet werden muss. Dadurch ist die Datenlage eher ungeeignet um einen Vergleich zu ziehen.

Um dennoch einen aussagekräftigen Vergleich zu erhalten, wird mit Hilfe von ArcGIS das Modell der Oberfläche von Schicht 8 mittels des IDW und den gleichen Einstellungen wie bereits zu Beginn des

Abschnitts aufgeführt, neu berechnet. Das Resultat für diese Berechnung wird in Abbildung 42 mit dem Ergebnis aus ElKowmGIS verglichen. Die beiden Bilder zeigen eine grosse Ähnlichkeit.

Auch hier wird die Differenzkarte generiert. Die Werte der neuen Karte variieren von -0.21 bis 0. 13 Meter. Dies bedeutet, dass die Punkte zwischen -21 und 13 Zentimeter Differenz aufweisen können. Abbildung 42 zeigt jedoch deutlich, dass es dabei extreme Ausreisser sind und über 92% der Punkte einen Differenzwert von 2 Zentimetern aufweisen. Diese Differenz sollte bei einer Oberflächenmodellierung akzeptabel sein

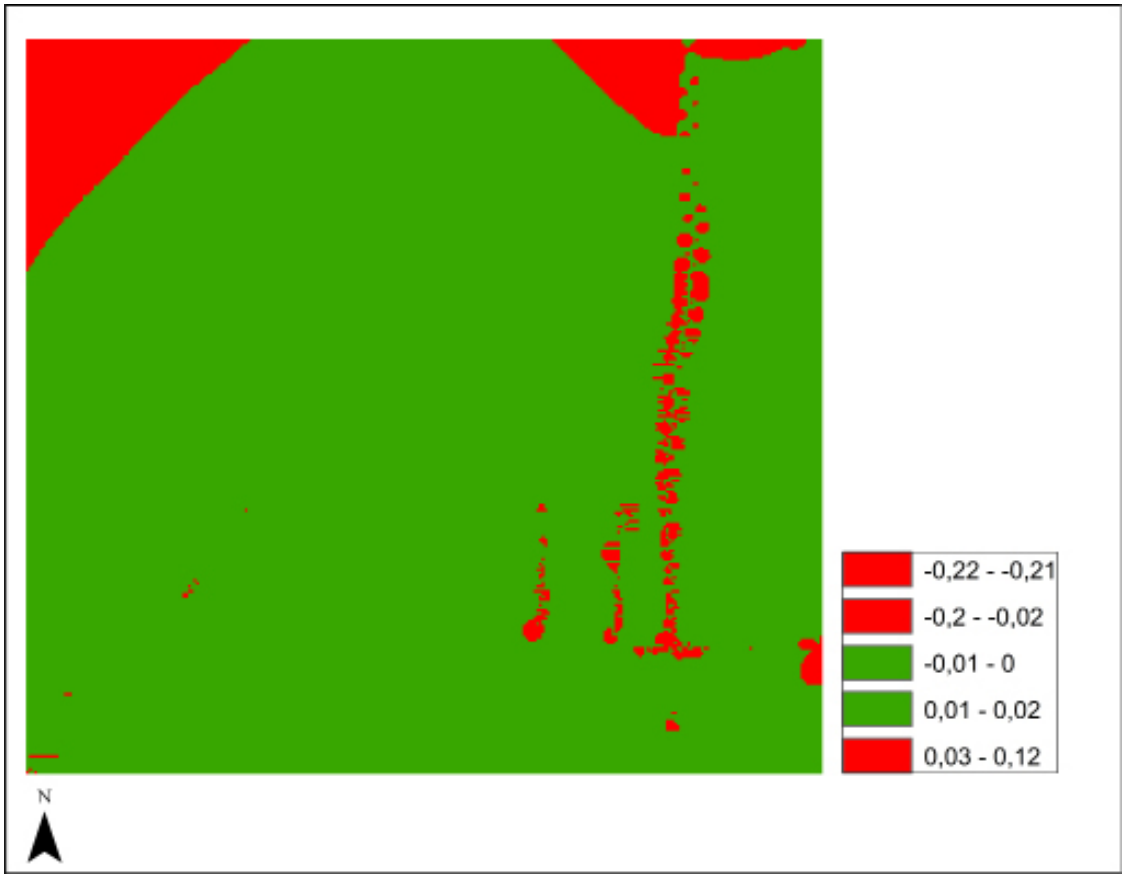


Abb. 43: Differenzkarte der Schicht 8 mittels IDW-Interpolation. In Grün zu sehen sind die Bereiche die sich 2 cm oder weniger unterscheiden.

und kann auf eine andere Implementierung des Algorithmus innerhalb von ArcGIS erklärt werden.

Für die Schicht 10 sehen die Werte ähnlich beschaffen aus (Abb. 44). Die Positionierungen stimmen miteinander übereinander ein, jedoch weisen über 37 Prozent der Werte eine Abweichung von maximal 10 Zentimetern auf. Auch in diesem Fall spielen die bereits aufgeführten Faktoren der Datenlage und der Interpolationsmethode eine grosse Rolle.

Dadurch ergibt sich die Erkenntnis, dass die beiden Interpolationsmethoden aber auch die gewählten Datensätze sich nicht für einen Vergleich eignen. Beide Interpolationsalgorithmen sind zu verschieden, um sie miteinander vergleichen zu können. Ferner hat sich gezeigt, dass die Grundlagendaten, die zur Berechnung der Modelle dienten, nicht gleichmässig verteilt sind. Es handelt sich hierbei um Einzelpunkte,

Von	Bis	Anzahl	Prozent
-0.80	-0.70	16	0.33
-0.70	-0.60	65	1.35
-0.60	-0.50	166	3.46
-0.50	-0.40	217	4.52
-0.40	-0.30	254	5.29
-0.30	-0.20	271	5.65
-0.20	-0.10	300	6.25
-0.10	0,00	683	14.23
0.00	0.10	1118	23.29
0.10	0.20	516	10.75
0.20	0.30	349	7.27
0.30	0.40	153	3.19
0.40	0.50	142	2.96
0.50	0.60	132	2.75
0.60	0.70	120	2.50
0.70	0.80	102	2.13
0.80	0.90	83	1.73
0.90	1.00	64	1.33
1.00	1.10	37	0.77
1.10	1.20	12	0.25

Abb. 44: Verteilung der Differenzen in 10 Zentimeter-Schritten des Modells der Schicht 10.

die aus den Profilen ausgelesen wurden. Dadurch ergeben sich lineare Strukturen, von denen auf eine ganze Fläche geschlossen werden soll. Solange sich weitere Punkte finden, die in einem bestimmten Winkel zu diesen Linearstrukturen liegen, lässt sich die Flächeninformation problemlos berechnen. Ohne die Punkte in den Flächen werden die Modelle relativ ungenau.

Aus diesem Grund soll abschliessend ein weiteres Beispiel für die Oberflächenmodellierung aufgeführt werden. Während der Grabungskampagne 2009 der Grabung Mutzig – Rain (Detrey et al. 2010b), an der die urgeschichtliche Abteilung der Integrativen Prähistorischen und Naturwissenschaftlichen Archäologie der Universität Basel beteiligt war, wurden über 400 Punkte eingemessen, die die Topografie der Grabung

und des umliegenden Geländes repräsentieren. Mit Hilfe dieser Daten wurde mit der Software Surfer 8 ein Oberflächenmodell der Fundstelle berechnet (Detrey et al. 2010a, S. 21). Dieses Modell wurde zwar wiederum mit dem Minimum-Curvature-Algorithmus berechnet, jedoch stehen die Grunddaten, die gleichmässig über die einzelnen Flächen verteilt sind, für diese Arbeit zur Verfügung. Somit ist es möglich, das Modell mit Hilfe der IDW-Methode in ArcGIS neu zu berechnen. Anschliessend werden die Daten in ein neues Projekt in ElKowmGIS importiert und auch dort ein Modell berechnet. Zum Vergleich der beiden Modelle (Abb. 45) wird die Oberfläche aus ElKowmGIS in ArcGIS importiert und von der Karte aus ArcGIS subtrahiert.

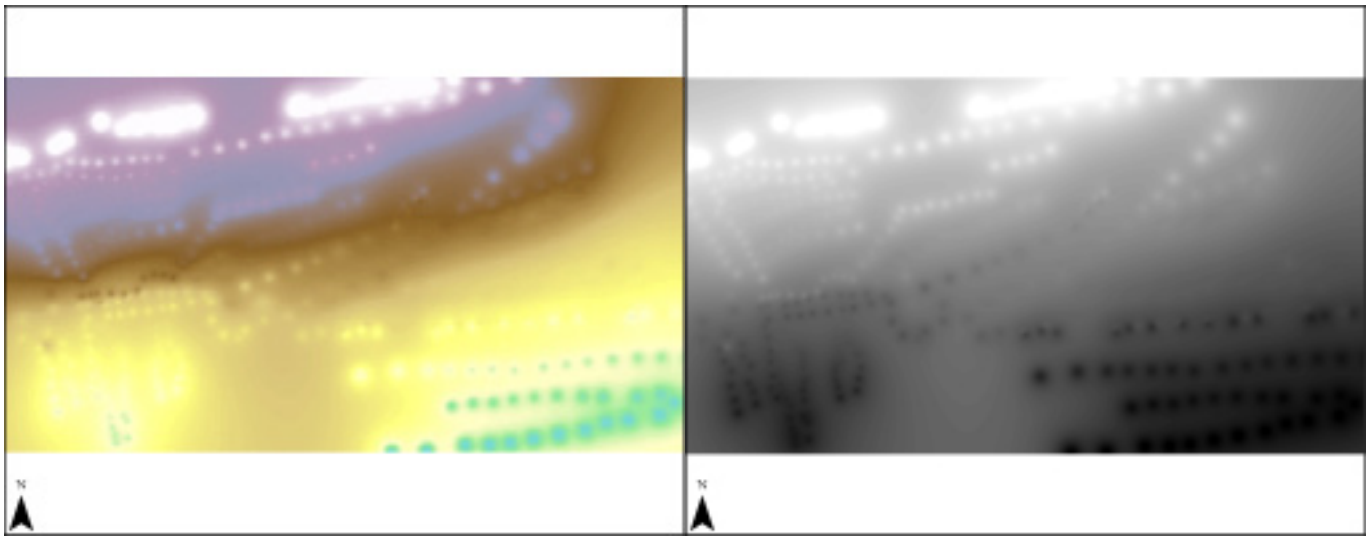


Abb. 45: Geländemodelle der Grabungen in Mutzig. Links mit ElKowmGIS berechnet, rechts das Ergebnis aus ArcGIS.

Die so entstandene Differenzkarte ist in Abbildung 46 dargestellt und zeigt deutlich, dass über 99% der Punkte maximal 10 Zentimeter voneinander abweichen. Insgesamt wurden 49770 Punkte berechnet, die eine Fläche von 3100 m² abdecken. Die durchschnittliche Abweichung liegt bei 6 Millimetern. Diese Werte zeigen, dass die Berechnung von ElKowmGIS durchaus richtig ist. Die Abweichung zu ArcGIS rührt daher, dass die Entwickler von ArcGIS einen anderen Ansatz verwenden als ElKowmGIS.

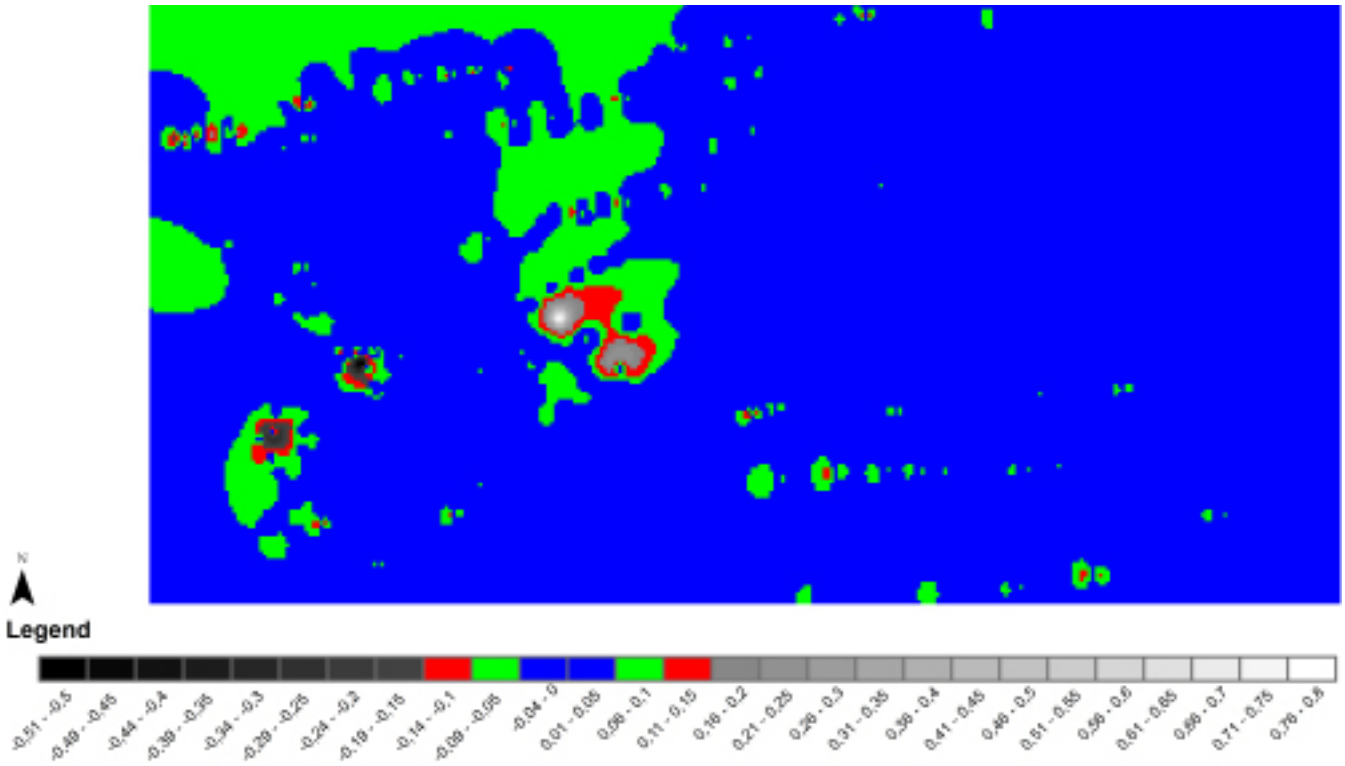


Abb. 46: Differenzkarte der Grabungen in Mutzig. Unterteilung in 5 Zentimeter-Schritten.

9 Fazit

Um diese Arbeit abzuschliessen, soll an dieser Stelle das Zitat von Ian Tattersall nochmals aufgegriffen werden:

> „Da jede Ausgrabung die Fundstelle unwiederbringlich zerstört, muss man jede nur mögliche Information vor der Beendigung der Arbeit festhalten."
>
> (Tattersall 1999, S. 95)

Dieser Fakt ist in Archäologenkreisen durchaus bekannt, auch wenn er vielleicht anders formuliert wird. Das Bekanntsein dieser Tatsache bedeutet jedoch nicht, dass in allen Fällen jegliche verfügbare Information einer Grabung festgehalten wird. Hierfür gibt es durchaus plausible Gründe (Zeitmangel, Finanzknappheit oder auch nicht pragmatische Dokumentationsmittel), die erklären, warum nicht jede Information gesichert wurde.

Die hier vorliegende Arbeit hatte nun als Ziel, einen der oben genannten Gründe aus der Liste zu entfernen respektive nicht weiter als entscheidenden Grund für eine Nichtdokumentation aufzuführen. ElKowmGIS, wie es in dieser Arbeit vorgestellt wurde, soll eine Arbeitserleichterung bei der Dokumentation und Auswertung archäologischer Fundstellen darstellen. Der erste Auslöser für die Entwicklung der eigenen Softwarelösung war die unbefriedigende Situation auf dem kommerziellen Softwaremarkt. Viele verfügbare Programme sind auf die Bedürfnisse einzelner Institutionen und/oder Ausgrabungen angepasst und lassen sich, wenn überhaupt nur schwerlich auf eine spezielle Situation anpassen. Der zweite Auslöser stellte der Trend dar, dass archäologische Informationen immer öfter vereinheitlicht oder zentral zugänglich gemacht werden. Eindrückliche Beispiele hierfür finden sich in den Projekten NESPOS (Neanderthal Studies Professional Online Service, NESPOS Society e.V. 2013), CIDOC Conceptual Reference Model (CRM, CIDOC 2013) oder der Archaeology Data Service (ads, Archaeological Data Service 2013) aus Grossbritannien. Diese Projekte haben zum Ziel, archäologische Daten zu vereinheitlichen und/oder sie auf einfache Art und Weise zur Verfügung zu stellen. Auch in der Schweiz finden sich mittlerweile Ansätze zur Standardisierung der Grabungsdokumentation (Arbeitsgruppe Standards 2013). Die Beispiele zeigen deutlich, dass in der Archäologie realisiert wurde, wie wichtig es ist, die einzelnen Aktivitäten sorgfältig zu dokumentieren. Zusätzlich erkannt wurde auch, dass die Daten einheitlich aufgenommen werden müssen, sollen sie später mit anderen Vorgängen verglichen werden.

ElKowmGIS schlägt die gleiche Richtung ein, indem es Archäologen eine Softwarelösung bietet, die zeit- und ortsunabhängig funktioniert und für jedes Projekt die Daten in der gleichen Art und Weise speichert. Dennoch ist es möglich, die einzelnen Elemente für einen Vorgang zu personalisieren und sie auf die speziellen Bedürfnisse anzupassen. Hierfür wurden bei der Entwicklung des Programms insgesamt sieben Ziele verfolgt, die in diesem Abschnitt nochmals aufgenommen und kommentiert werden sollen.

Wie die Vorstellung der einzelnen Module gezeigt hat, ist ElKowmGIS in verschiedene Bereiche untergliedert. Diese Bereiche decken unterschiedliche Anwendungsgebiete ab, die bei der Dokumentation und Auswertung einer Grabung in Anspruch genommen werden. Momentan handelt es sich neben der reinen Projektverwaltung um die Bereiche GIS/CAD, Bildverwaltung und Dokumentenverwaltung. Somit ist gewährleistet, dass für die notwendigen Arbeiten auf einer Grabung lediglich eine Softwarelösung

eingesetzt werden muss. Dies bietet den Vorteil, dass sich die Ausgräberin, resp. der Ausgräber, sich nicht auf das Erlernen mehrerer Anwendungen konzentrieren muss, sondern sich vollumfänglich der Dokumentation widmen kann. Ferner bietet der modulare Aufbau die Möglichkeit, das Programm beliebig zu erweitern. Zusätzlich lassen sich durch die modulare Bauweise die einzelnen Module beliebig austauschen, um beispielsweise neue Funktionen im GIS-Modul einzuführen. Zum modularen Aufbau lässt sich auch die Integration einer externen Datenbank zählen.

Mit Hilfe der PostgreSQL-Datenbank ist es möglich die Daten unabhängig vom Programm zu nutzen. Dieses Vorgehen bot zwei entschiedene Vorteile gegenüber einer Speicherung der Daten innerhalb der Software oder gar in einem eigenen Format. Mit Hilfe des Datenbankservers können die Daten zum einen auch über andere Programme abgerufen und verarbeitet werden. Zusätzlich erlaubt es der Server auf eine Sicherung der Daten in einem bestimmten Format zu verzichten. Da es sich beim PostgreSQL-Server um einen SQL-basierten Server handelt, muss man sich um die Lesbarkeit der Daten in der Zukunft keine zusätzlichen Gedanken machen. Gerade in Hinsicht auf eine Langzeitsicherung der Daten ist dieser Punkt äusserst wichtig. Ein weiterer Vorteil in der Nutzung des Datenbankservers liegt in der Möglichkeit, die Dokumentenstruktur und ihre Verwaltung von der Software durchführen zu lassen. ElKowmGIS erwartet lediglich eine Angabe wohin die einzelnen Dokumente gespeichert werden sollen. Die restlichen Verwaltungsaufgaben übernimmt es selbstständig. Wie in den Beispielen gezeigt, werden die fertigen Bilder und Grids in den Ordner img gesichert und der relative Pfad in der Datenbank gespeichert. Die restlichen Dokumente werden im Ordner doc abgelegt. Dennoch sind durch dieses Vorgehen und die Beschriftung der Dateien diese eindeutig identifizierbar und zugreifbar ohne die Software einzusetzen. Dadurch kann man sagen, dass ElKowmGIS die künstliche Intelligenz besitzt, wie sie zu Beginn des Abschnitts 2.2.2 definiert wurde.
Der wichtigste Punkt in der Arbeit lag in der Anpassbarkeit der Software an die Grabungssituation. Auch wenn es nicht möglich war, während der Programmierung und Entwicklung der Software jede Dokumentationsmöglichkeit einer Grabung zu erfassen und zu berücksichtigen, so ist ElKowmGIS doch sehr neutral gestaltet. Die vordefinierten Strukturen stellen die Strukturen dar, die auf jeder Grabung erfasst werden sollten. Der Aufbau der Definitions- und Datentabellen wurde zwar für ElKowmGIS fest gewählt, jedoch ist es möglich jede Art von Dokumentationssystem unterzubringen. Es ist somit nicht notwendig, komplette Systeme umzustellen, nur damit sie in die Software importiert werden können. Es wurde ebenfalls versucht, möglichst neutrale Begriffe für die Benennung der einzelnen Spalten zu verwenden. Gerade im Bereich der Mehrsprachigkeit der Software war dies ein Balanceakt. Als Beispiel sei hier nur der Begriff Layer im englischen erwähnt. Viele englischsprachige Ausgrabungen arbeiten mit dem Begriff Layer im Sinne von Schicht oder Niveau. ElKowmGIS hält sich hierbei an die gängigen Softwarelösungen, für die der Begriff Layer eine einzelne Ebene, die Daten enthält, darstellt. Um nun eine Schicht (oder auch ein Niveau) bietet ElKowmGIS zwei Möglichkeiten. Zum einen stehen die Spalten units und subunits zur Verfügung (oftmals sind diese Begriffe gleichbedeutend mit einer Schicht). Die zweite Möglichkeit besteht darin, statt dem Begriff Layer, den Begriff Level zu verwenden, wie er in ElKowmGIS ebenfalls zur Verfügung steht. Zu guter Letzt steht auch nichts dem Vorgehen im Wege, die Definition der Ebenen (=Layer) gleichzeitig als Schichtdefinition zu übernehmen. Einziger Nachteil in diesem Vorgehen

würde die Trennung der einzelnen Fundkategorien darstellen. Um nun Silices und Knochen aus dem Niveau 8 getrennt darzustellen, müssten zwei Layer für eine Schicht angelegt werden. Ein Layer enthält die Knochen, der andere die Silices der Schicht 8. Aus diesem Grund wird eher vorgeschlagen die Trennung über die Level-Spalte oder über die Spalten units und subunits vorzunehmen. Dadurch ist es möglich, alle Fundgattungen auf einem Layer darzustellen und dennoch die Trennung nach Schichten beizubehalten.

Der Punkt Plattformunabhängigkeit wurde bereits in Kapitel 2.2.5 ausführlich diskutiert und Bedarf hier eigentlich keiner weiteren Erklärung. Ein Test der Software in einer virtuellen Maschine unter Linux zeigte, dass die Software unter verschiedenen Betriebssystemen einsetzbar ist.

Als letzter Punkt bleibt lediglich die Anwenderfreundlichkeit zu diskutieren. Bei der Umsetzung der grafischen Teile von ElKowmGIS wurde darauf geachtet, dass alle Elemente gut sichtbar und eindeutig beschriftet sind. Des Weiteren wurde versucht, keine unnötigen Wege zu produzieren. Dies bedeutet, dass Wert darauf gelegt wurde, die Anzahl der zu betätigenden Schaltflächen, Menus und Fenster auf ein Minimum zu reduzieren. Hinzu kommt ein weiterer Aspekt, der ElKowmGIS anwenderfreundlicher machen soll. Hierbei handelt es sich um die Möglichkeit, eigene Sprachdateien mit den Übersetzungen der Schaltflächen, Menus und Textausgaben zu erstellen. Diese Sprachpakete können anschliessend in ElKowmGIS ausgewählt werden, wodurch ElKowmGIS auch in dieser Sprache verfügbar ist. Mitgeliefert werden ein deutsches und ein englisches Sprachpaket. Diese Pakete können mit einem normalen Texteditor geöffnet und bearbeitet werden. Zusätzlich enthält ElKowmGIS ein leeres Sprachpaket, welches als Vorlage für weitere Pakete dient.

Zusammenfassend lässt sich sagen, dass ElKowmGIS ein Werkzeug für die archäologische Dokumentation und Auswertung einer Fundstelle darstellt. Nach der Definition der Ziele und der Strukturierung der Datenbank wurde das Konzept der Arbeit auf zwei Tagungen dem Fachpublikum vorgestellt und diskutiert. Dabei hat es sich als positiv erwiesen, dass das Publikum auf den beiden Tagungen verschiedene Hintergründe hatte. Während der Graduiertentagung des IPNA und der Ur- und Frühgeschichtlichen und Provinzialrömischen Archäologie der Universität Basel es sich mehrheitlich um Archäologen handelte, waren an der Tagung der **CAA** 2010 (Schuhmann Forthcoming) mehrheitlich Vertreter der Fachrichtung Archäoinformatik vertreten. Die Diskussionen und Gespräche auf beiden Tagungen lieferten somit Inputs sowohl aus rein archäologischer wie auch aus eher informatischer Sicht. Viele dieser Inputs waren sehr hilfreich und wurden bei der Entwicklung von ElKowmGIS berücksichtigt. Beide Tagungen haben auch deutlich gezeigt, dass ein reges Interesse an einer solchen Softwarelösung, wie ElKowmGIS es ist, besteht.

10 Literatur

Apache Software Foundation. 2013. Log4j 2. http://logging.apache.org/log4j/2.x/ (accessed March 28, 2013).

Arbeitsgruppe Standards. 2013. Richtlinien für Archäologische Untersuchungen. http://horizont2015.ch/media/f8a7dff2518e1753ffff8621ffffffe7.pdf (accessed March 28, 2013).

Archaeological Data Service. 2013. Archsearch. http://archaeologydataservice.ac.uk/archsearch/ (accessed March 28, 2013).

Briggs, I. C. 1974. Machine contouring using minimum curvature. Geophysics 39(1):39-48. http://www.fcaglp.unlp.edu.ar/referenciacion/images/Machine_contouring_using_minimum_curvature.pdf.

Brönnimann, D., and D. Schuhmann. 2006. Vermessung mittels Tachymeter und TachyCAD. In Travaux de la Mission Archéologique. Jean-Marie Le Tensorer, Reto Jagher, Daniel Schuhmann, and Ruth Mienert, eds. Pp. 39–44 11. Basel.

Byous, J. 1998. JAVA TECHNOLOGY: THE EARLY YEARS. http://web.archive.org/web/20050420081440/http://java.sun.com/features/1998/05/birthday.html (accessed March 28, 2013).

CIDOC. 2013. CIDOC Conceptual Reference Model. http://www.cidoc-crm.org/ (accessed March 28, 2013).

Detrey, J., M. Guélat, T. Hauck, O. Putelat, P. Rentzel, D. Schuhmann, and T. Vigreux. 2010a. Mutzig (67) : "Rain", Boulevard Clémenceau. Sélestat.

Detrey, J., M. Guélat, T. Hauck, P. Rentzel, and T. Vigreux. 2010b. L'abri-sous-roche Paléolithique moyen de Mutzig, Rain (Bas-Rhin, F): Reprise des travaux. Bulletin de la Société préhistorique française(3):581–84.

Drosdowski, G., and A. Klosa. 1996. Duden Rechtschreibung der deutschen Sprache. 21., völlig neu bearb. u. erw. Aufl. /. Der Duden : das Standardwerk zur deutschen Sprache / hrsg. vom Wiss. Rat der Dudenred. Günther Drosdowski … ; Bd. 1. Mannheim: Dudenverl.

Eisentraut, P. 2003. PostgreSQL: Das offizielle Handbuch. 1st ed. Datenbanken. Bonn: mitp-Verl.

Fischer, P., and P. Hofer. 2008. Lexikon der Informatik. 14th ed. s.l: Springer-Verlag.

Gosling, J., and H. McGilton. 1996. The Java Language Environment. http://www.oracle.com/technetwork/java/langenv-140151.html (accessed March 28, 2013).

Gruber, F. J., and R. Joeckel. 2011. Formelsammlung für das Vermessungswesen. 15th ed. Vieweg Studium. Wiesbaden: Vieweg+Teubner Verlag / Springer Fachmedien Wiesbaden GmbH, Wiesbaden.

M. Guggisberg. 2012. mündliche Mitteilung, 27. Juni 2012.

Gülcü, C. 2002. Short introduction to log4j. http://logging.apache.org/log4j/1.2/manual.html (accessed March 28, 2013).

Huber, M. 2012. Trophäensammler. Linux Magazin 2012(05). http://www.linux-magazin.de/Ausgaben/2012/05/LNM-Awards.

Jordan, G. 2007. Digital Terrain Analysis in a GIS Environment. Concepts and Development. In Lecture Notes in Geoinformation and Cartography. Robert J. Peckham and Gyozo Jordan, eds. Pp. 1–43. Berlin, Heidelberg: Springer Berlin Heidelberg.

Kahmen, H. 2006. Angewandte Geodäsie: Vermessungskunde. 20th ed. de Gruyter Lehrbuch. Berlin: de Gruyter.

Krüger, G., and T. Stark. 2011. Handbuch der Java-Programmierung: Standard Edition Version 6. 6th ed. Programmer's choice. München: Addison-Wesley.

Leica Geosystems. 2008. GSI ONLINE for Leica TPS and DNA.

Le Tensorer, J.-M., V. von Falkenstein, H. Le Tensorer, and S. Muhesen. 2011. Hummal: a very long Palaeolithic sequence in the steppe of central Syria - considerations on Lower Palaeolithic and the beginning of Middle Palaeolithic. In The Lower and Middle Palaeolithic in the Middle East and neighbouring regions: Basel Symposium, mai [i.e. may] 8 - 10 2008. Jean-Marie Le Tensorer, ed. Pp. 235–48. Études et recherches archéologiques de l'Université de Liège 126. Liège: Univ.

McPherron, S. J. 2005. Artifact orientations and site formation processes from total station proveniences. Journal of Archaeological Science 32(7):1003–14.

Moore, I. D., R. B. Grayson, and A. R. Ladson. 1991. Digital terrain modelling: A review of hydrological, geomorphological, and biological applications. Hydrol. Process. 5(1):3–30.

NESPOS Society e.V. 2013. Neanderthal Studies Professional Online Service. www.nespos.org (accessed March 28, 2013).

Open Source Geospatial Foundation. 2013. GeoTools. http://geotools.org/ (accessed March 28, 2013).

Oracle. Java. 2013. http://www.java.com/de/download/ (accessed March 28, 2013).

———. 2013a. Java Advanced Imaging (JAI) API. http://www.oracle.com/technetwork/java/javase/tech/jai-142803.html (accessed March 28, 2013).

———. 2013b. JRootPane. In Java Platform SE7 API. Oracle, ed. http://docs.oracle.com/javase/7/docs/api/javax/swing/JRootPane.html#setDefaultButton%28javax.swing.JButton%29 (accessed March 28, 2013).

Professur für Geodäsie und Geoinformatik (GG) AUF Universität Rostock. 2013. Geoinformatik-Service GI-Lexikon. http://www.geoinformatik.uni-rostock.de/einzel.asp?ID=1267 (accessed March 28, 2013).

RXTX. 2013. RXTX wiki. http://rxtx.qbang.org (accessed March 28, 2013).

Schuhmann, D. 2007. Digitale Modellierungen und Schichtrekonstruktionen der paläolithischen Fundstelle

Hummal, Syrien. Diplomarbeit, Universität Basel, Basel.

———. 2010. Das Dokumentationssystem des Projektes in El Kowm. In Travaux de la Mission Archéologique. Jean-Marie Le Tensorer, Reto Jagher, Daniel Schuhmann, and Ruth Mienert, eds. Pp. 65–67. Basel.

———. 2011. A three-dimensional model of the Palaeolithic site of Hummal (Central Syria). In The Lower and Middle Palaeolithic in the Middle East and neighbouring regions: Basel Symposium, mai [i.e. may] 8 - 10 2008. Jean-Marie Le Tensorer, ed. Pp. 279–88. Études et recherches archéologiques de l'Université de Liège 126. Liège: Univ.

———. Forthcoming. ElKowmGIS: A New Program for the Documentation of Archaeological Sites. In Proceedings of the 38th Conference on Computer Applications and Quantitative Methods in Archaeology. F. J. Melero, Pedro Cano, and Jorge Revelles, eds. Granada, Spain.

Shepard, D. 1968. A two-dimensional interpolation function for irregularly-spaced data. In Proceedings of the 1968 23rd ACM national conference. ACM, ed. Pp. 517–24. New York, NY: ACM.

Tattersall, I. 1999. Neandertaler: Der Streit um unsere Ahnen. Ein Peter N. Nevraumont Buch. Basel: Birkhäuser.

The PostgreSQL Global Development Group. 2011. PostgreSQL JDBC Driver. http://jdbc.postgresql.org/ (accessed March 28, 2013).

———. 2013a. Mac OS X packages. http://www.postgresql.org/download/macosx/ (accessed March 28, 2013).

———. 2013b. PostgreSQL. http://www.postgresql.org/ (accessed March 28, 2013).

———. 2013c. PostgreSQL 9.0.12 Documentation. http://www.postgresql.org/docs/9.0/static/index.html (accessed March 28, 2013).

The University of Chicago Press. 2013. Current Anthropology Style Guide. http://www.press.uchicago.edu/journals/ca/style.html?journal=ca (accessed March 28, 2013).

Ullenboom, C. 2012. Java ist auch eine Insel: Das umfassende Handbuch ; [aktuell zu Java 7 ; Programmieren mit der Java Platform, Standard Edition 7 ; Java von A bis Z: Einführung, Praxis, Referenz ; von Klassen und Objekten zu Datenstrukturen und Algorithmen]. 10., aktualisierte und überarb. Aufl., 1. Nachdr. Galileo computing. Bonn: Galileo Press.

11 Abbildungen

12 Glossar

ArcGIS

Gängigste GIS-Software der Firma Esri

AutoCAD

Die Anwendung AutoCAD ist ein Programm der Firma Autodesk und dient zur Erstellung von Plänen und Konstruktionszeichnungen vor Allem in den Bereichen Bauwesen, Architektur und Maschinenbau. Es wird immer häufiger in archäologischen Ausgrabungen eingesetzt.

Bildschirmkoordinaten

Koordinaten eines Punktes auf dem Bildschirm eines Computers.

Bytecode

Hardwareunabhängige Sammlung von Befehlen in der Informatik. Dadurch wird es möglich, dass Programme auf verschiedenen Rechnern lauffähig sind.

CAA

Computer Applications and quantitative methods in Archaeology: Vereinigung der Forschenden im Bereich der Archäoinformatik

CAD

Computer Aided Design: computergestütze Konstruktion. Software zur Erstellung von Plänen und Konstruktion komplexer Gebilde

csv-Datei

comma separated value: Textdatei in tabellarischer Form. Die Spalten werden dabei durch ein definiertes Zeichen, meist ein “;“ oder “,“, getrennt.

Dump

Englischer Begriff für den Speicherauszug aus einer Datenbank.

12

dxf-Datei

drawing interchange file format: Datei zum Austausch von CAD-Daten

ElKowmGIS

Der Begriff setzt sich aus zwei Teilen zusammen zum einen handelt es sich hierbei um die Region El Kowm in Syrien. In dieser Region liegt unter anderem die paläolithische Fundstelle Hummal, in deren Rahmen diese Arbeit entstanden ist. Zum anderen wird mit dem Begriff GIS ein geografisches Informationssystem abgekürzt.

GIS

Geografisches Informationssystem. Eine Anwendung zur Verarbeitung, Analyse und Darstellung geografischer Datensätze. Bekannteste Vertretet: ArcGIS und GRASS GIS.

GRASS GIS

Ein OpenSource-GIS

GSI-8 / GSI-16

Datenformate für die Übertragung von Daten über das RS232-Protokoll.

Identifikator

Wert oder Wort zur Bezeichnung (Identifikation) eines Objektes, Punktes o. ä..

Java-API

Englisch: application programming interface. Dient in der Programmiertechnik als Schnittstelle um neue Teile in eine bestehende Programmierung einzufügen.

Java Virtual Machine (JVM)

Ein Programm zur Interpretation von ByteCode.

Java Runtime Environment (JRE)

Programm, das zur Ausführung von Java-Programmen benötigt wird. Diese Software ist dabei unabhängig vom Betriebssystem.

Java Developement Kit (JDK)

Entwicklungsumgebung für die Erstellung von Java-Programmen.

Kompilierung

Vorgang der Übersetzung von Programmcode in Code, der für einen Computer lesbar ist.

Konstruktor

Methode zur Erstellung neuer Objekte in einer objektorientierten Programmiersprache.

Log

Bei einem Log handelt es sich um ein Protokoll, das vordefinierte Ereignisse festhält. Dient in der Regel zur Behebung von Fehlern in einer Anwendung.

Metadaten

Weiterführende Informationen zu einem Objekt, Dokument oder Foto.

Photoplan

Die Software Photoplan der Firma kubit ist eine Erweiterung für AutoCAD, die es ermöglicht mit Rasterdaten in AutoCAD zu arbeiten.

RS232-Schnittstelle

Standard für die Schnittstelle in einem Computer, die die Kommunikation mit externen Geräten erlaubt. Der Standard dient speziell der seriellen Schnittstelle.

Schema

Eigentlich Datenbankschema. Aufbau der Tabellen, Beziehungen und Benutzer einer Datenbank.

12

Select-Abfrage

Bei einer Select-Abfrage handelt es sich um eine Abfrage einer Datenbank, in der bestimmte Einträge aus der Datenbank ausgewählt werden.

SQL-Syntax

Einheitliche Befehlsabfolge zur Steuerung und Abfrage von Datenbanken.

TachyCAD

Das Programm TachyCAD der Firma kubit stellt eine Erweiterung von AutoCAD dar und bietet Routinen zur Verbindung eines Tachymeters mit einem Computer.

Weltkoordinaten

Koordinaten eines Punktes im Raum

Anhänge

A Programmspezifische Anhänge

A.1 Java Dokumentation

A.2 Softwaremetrik

A.3 Quelltext

B Datenbankspezifische Anhänge

B.1 Schema def

B.2 Schema elkowmgis

B.3 Vorgeschlagene Einträge

A.1 Java-API- Dokumentation

Bei der Java-API-Dokumentation handelt es sich um die Beschreibung der Klassen eines Java-Programms. Da ElKowmGIS insgesamt über 107 Klassen mit mehr als 420 Methoden besitzt, wird auf den Abdruck der API-Dokumentation verzichtet. Sie kann beim Autor eingesehen werden.

A.2 Softwaremetrik

Die Softwaremetrik enthält einige Zahlen und Fakten über das Programm. Darin enthalten sind die Anzahl der Pakete, Klassen und Zeilen Programmcode. Ferner befinden sich darin die Angaben über die zyklomatische Komplexität und den Zusammenhalt (LCOM 4). Die letzten beiden Begriffe sind im Glossar ausführlich beschrieben.

Klassen

Total:	107

Pakete mit den meisten Klassen

listener:	19
panel:	19
classes:	18
iframes:	14
threads:	11

Methoden

Total:	423

Klassen mit den meisten Methoden

Queries_all_queries:	47
Hilfspunkt:	31
DBEntry:	20
Definitions_feature:	19
DistanzListe:	19

Programmcodezeilen

Insgesamt:	23‘744

Die 5 grössten Klassen

Listener_ButtonAction:	2195
Listener_MapMouseAdapter:	1531
Queries_all_queries:	1404
DXF_Color:	1334
Panel_map:	712

Die 5 grössten Packages

listener:	6354
classes:	4516
panel:	3854
inframes:	2217
threads:	1609

Importe aus anderen Klassen

Total:	1159

Klassen mit den meisten Importen

Listener_MapMouseAdapter:	63
Listener_ButtonAction:	44
Panel_map:	41
Listener_MenuAction:	34
Listener_dgm:	30

Pakete:

listener:	354
panel:	269
inframes:	133
threads:	108
classes:	107

Zyklomatische Komplexität

Durchschnitt:	3.94

Methoden mit hoher Komplexität:

DXF_Color.getCol:	257
Listener_ButtonAction.actionPerformed:	208
Listener_MapMouseAdapter.onMouseMoved:	93
Listener_MapMouseAdapter.onMouseClicked:	64
Listener_MenuAction.actionPerformed:	54

LCOM 4

Durchschnitt:	1

Klassen mit höchstem LCOM:

DBEntry:	16
Hilfspunkt:	13
Listener_Inframe_Close:	8
Listener_GISDraw:	7
Machine:	7

Pakete mit hohem LCOM 4:

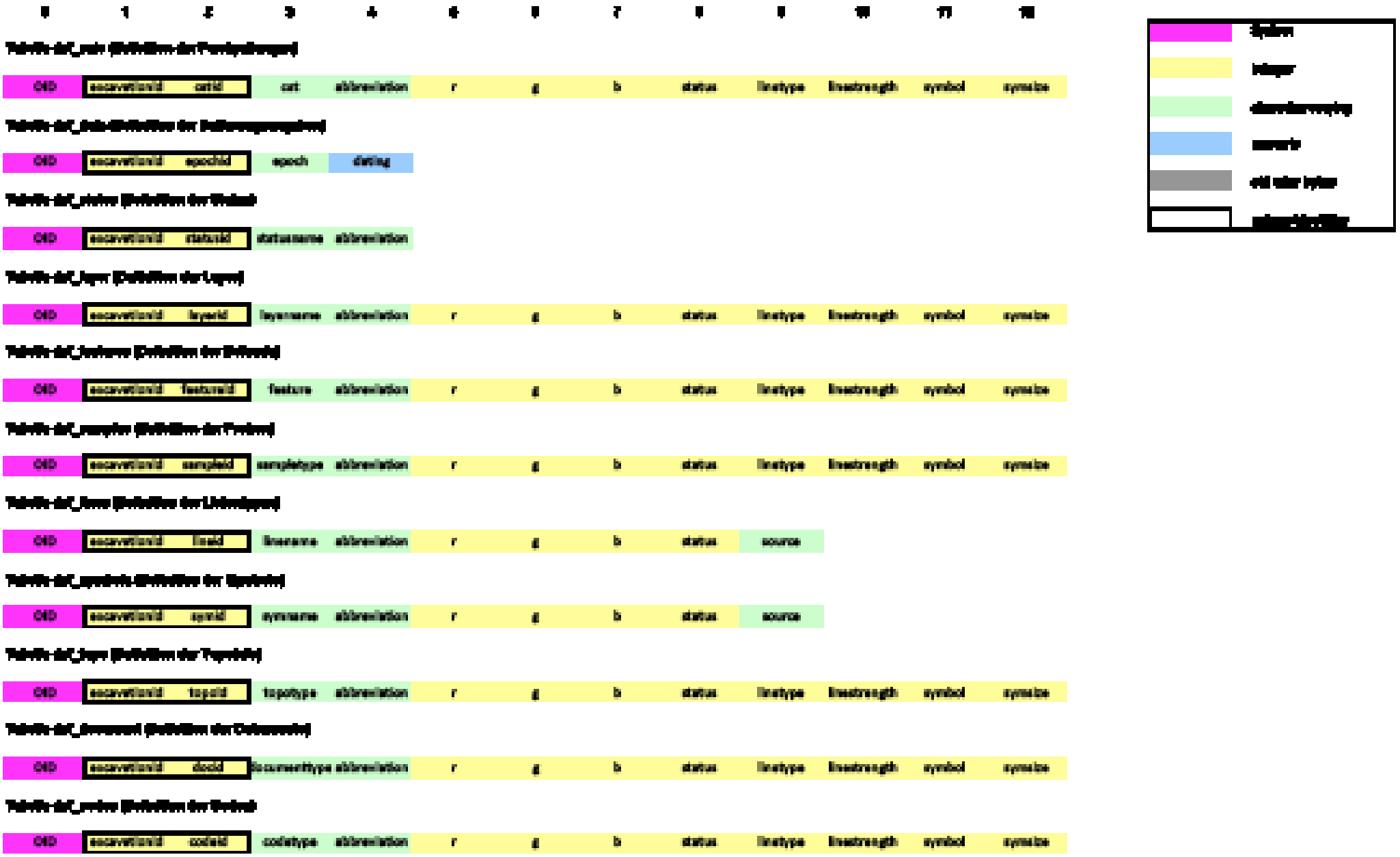
classes:	3
listener:	3
bildarchiv:	2
definitions:	1
dialogs:	1

A.3 Quelltext von ElKowmGIS

Der Quelltext von ElKowmGIS umfasst ca. 24‘000 Zeilen Text. Aus diesem Grund wird in Absprache mit Herrn Prof. Le Tensorer und Herrn Prof. Conard darauf verzichtet, den kompletten Quelltext abzudrucken. Einzelne Ausschnitte finden sich in Kapitel 7 der Arbeit wieder. Der komplette Quelltext kann beim Autor eingesehen werden.

B.1 Schema def

Auflistung der Tabellen im Datenbankschema def mit den Spaltenbezeichnungen und den Spaltenformaten


0 1 2 3 4 6 8 7 8 9 10 11 12
OID
excavationid
catid
cat
abbreviation
r
g
b
status
linetype
linestrength
symbol
symsize
epochid
epoch
dating
statusid
statusname
layerid
layername
featureid
feature
sampleid
sampletype
lineid
linename
source
symid
symname
topoid
topotype
docid
documenttype
codeid
codetype

B.2 Schema elkowmgis

Auflistung der Tabellen im Datenbankschema elkowmgis mit den Spaltenbezeichnungen und den Spaltenformaten

0	1	2	3	4	5	6	7	8	9	10	11	12

Tabelle context (Basistabelle)

OID	excavationid	unit	subunit	id	codeid	typeid	level	layer	state	remarks	owner	history

Tabelle xyz (Koordinatentabelle)

OID	excavationid	unit	subunit	id	suffix	x	y	z	remarks	history

Tabelle documents (Dokumententabelle)

OID	excavationid	unit	subunit	id	documentid	typeid	document	remarks	history

Tabelle excavations (Grabungsinformationen)

OID	excavationid	excavation	place	campaign	format	orientation	owner	state	remarks	history

Tabelle datings (Datierungen)

OID	excavationid	unit	subunit	id	datingid	epoche	datierung	owner	remarks	history

	System		numeric
	integer		oid oder bytea
	character varying		unique identifier

B.3 Vorgeschlagene Eiträge

Vorlagen für die Funde, Befunde, Layer, Proben etc., wie sie in ElKowmGIS verwendet werden.

Funde

Name Silex ***Abkürzung*** sx

Farbe 153,51,0

Layer Silex

Status

- Ausgeschieden
- Verbrannt
- Nicht in situ
- Modern beschädigt
- Probe
- Nicht photographiert

Name Keramik ***Abkürzung*** ker

Farbe 255,0,255

Layer Objekt

Status

- Ausgeschieden
- Verbrannt
- Nicht in situ
- Modern beschädigt
- Probe
- Nicht photographiert

Name Knochen ***Abkürzung*** os

Farbe 255,0,0

Layer Knochen

Status

- Ausgeschieden
- Verbrannt
- Nicht in situ
- Modern beschädigt
- Probe
- Nicht photographiert
- Phantom

Name Eisen ***Abkürzung*** fer

Farbe 0,0,255

Layer Objekt

Status:

- Ausgeschieden
- Verbrannt
- Nicht in situ
- Modern beschädigt
- Probe
- Nicht photographiert

Name Geroell ***Abkürzung*** ger

Farbe 255,255,0

Layer Objekt

Status

- Ausgeschieden
- Verbrannt
- Nicht in situ
- Modern beschädigt
- Probe
- Nicht photographiert

Name Glas ***Abkürzung*** gl

Farbe 153,51,0

Layer Objekt

Status:

- Ausgeschieden
- Verbrannt
- Nicht in situ
- Modern beschädigt
- Probe
- Nicht photographiert

Befunde

Name Mauer ***Abkürzung*** M

Farbe 0,0,255

Layer Befunde

Name Grube ***Abkürzung*** G

Farbe 255,0,0

Layer Befunde

Proben

Name Mikromorphologie ***Abkürzung*** MM

Farbe 255,0,0

Layer Proben

Name OSL ***Abkürzung*** OSL

Farbe 255,0,0

Layer Proben

Name TL ***Abkürzung*** TL

Farbe 0,0,0

Layer Proben

Name Stein ***Abkürzung*** ST

Farbe 255,0,0

Layer Proben

Name Schlaemmen ***Abkürzung*** SCH

Farbe 255,0,0

Layer Proben

Name Mollusken ***Abkürzung*** MOL

Farbe 255,0,0

Layer Proben

Name Paläomagnetismus ***Abkürzung*** PM

Farbe 0,0,0

Layer Proben

Name Botanik ***Abkürzung*** BOT

Farbe 0,255,0

Layer Proben

Name ESR ***Abkürzung*** ESR

Farbe 0,0,0

Layer Proben

Layer

Name Fixpunkte ***Abkürzung*** FP

Farbe 255,0,0

Name Layout ***Abkürzung*** LAY

Farbe 0,0,0

Name Silex ***Abkürzung*** SX

Farbe 0,0,255

Name Knochen ***Abkürzung*** OS

Farbe 255,0,0

Name Objekt ***Abkürzung*** OBJ

Farbe 0,0,0

Name Bildpunkte ***Abkürzung*** BP

Farbe 255,0,255

Name Bildlayer ***Abkürzung*** BL

Farbe 0,0,0

Name Befunde ***Abkürzung*** BEF

Farbe 0,0,255

Name Niveaus ***Abkürzung*** NIV

Farbe 0,255,0

Name Topo ***Abkürzung*** TOP

Farbe 0,0,0

Name Proben ***Abkürzung*** SAM

Farbe 0,0,0

C Beispielprotokoll des Log4j

Zur Veranschaulichung des HTMLLayout und der Darstellung eines Logs aus ElKowmGIS.

Curriculum vitae

Titel	Dr. des. phil. II
Vorname	Daniel
Nachname	Schuhmann
Nationalität	Deutsch
Geburtsdatum	3. März 1982
Adresse	Feierabendstrasse 53, 4051 Basel, Schweiz
Kontakt	email: D.Schuhmann@unibas.ch
	phone: ++41 (0)77 428 48 98

Ausbildung

Okt. 2007 – Nov. 2013	Integrative Prähistorische und Naturwissenschaftliche Archäologie Universität Basel, Doktorarbeit (Prüfung abgelegt und bestanden: 25.11.2013) Titel: " ElKowmGIS: Ein neues System zur Dokumentation archäologischer Fundstellen" Referent: Prof. Dr. J.-M. Le Tensorer
Nov. 2009	ESRI-Zertifikate: Learning ArcGIS Desktop und Learning ArcGIS 3D Analyst
Apr. 2003 – Jun. 2007	Institut für Prähistorische und Naturwissenschaftliche Archäologie Universität Basel, Diplomarbeit Titel: "Digitale Modellierungen und Schichtrekonstruktionen der paläolithischen Fundstelle Hummal, Syrien."
Okt. 2001 – Apr. 2003	Institut für Physik, Universität Basel
Aug. 1992 – Jun. 2001	Hans-Thoma-Gymnasium Lörrach - Abitur
Mai 1988 – Jun. 1992	Grundschule Tumringen (Lörrach)

Ausgrabungen und Praktika

Sep. 2013	Prospektionskampagne im Kanton Obwalden (CH), ohne Schwerpunkt Leitung: Prof. Dr. P.-A. Schawrz, Dr. P. Nagy
Sep. 2012	Prospektionskampagne im Kanton Obwalden (CH), ohne Schwerpunkt Leitung: Prof. Dr. P.-A. Schawrz, Dr. P. Nagy
Jul. 2010 – Aug. 2010	Ausgrabungen in Mutzig (F), paläolithisch Leitung: Jean Detrey (Section d'archéologie et de paléontologie Jura)
Mai 2010 – Jun. 2010	Ausgrabungen in El Kowm (SYR), paläolithisch Leitung: Prof. Dr. J.-M. Le Tensorer
Aug. 2009	Ausgrabungen in Mutzig (F), paläolithisch Leitung: Jean Detrey (Section d'archéologie et de paléontologie Jura)
Mai 2009 – Jul. 2009	Ausgrabungen in El Kowm (SYR), paläolithisch Leitung: Prof. Dr. J.-M. Le Tensorer
Aug. 2008 – Sep. 2008	Ausgrabungen in El Kowm (SYR), paläolithisch Leitung: Prof. Dr. J.-M. Le Tensorer
Jul. 2007 – Sep. 2007	Ausgrabungen in El Kowm (SYR), paläolithisch Leitung: Prof. Dr. J.-M. Le Tensorer
Aug.2006 – Oct. 2006	Ausgrabungen in El Kowm (SYR), paläolithisch Leitung: Prof. Dr. J.-M. Le Tensorer
Aug.2005 – Oct. 2005	Ausgrabungen in El Kowm (SYR), paläolithisch Leitung: Prof. Dr. J.-M. Le Tensorer
Jun. 2005 – Jul. 2005	Ausgrabungen in Biesheim (F), römisch Leitung: PD Dr. P.-A. Schwarz, C. Schucany
Sep. 2004 – Oct. 2004	Anthropologisches Praktikum im Naturhistorischen Museum Basel Leitung: Dr. G. Hotz
Jul. 2004 – Aug. 2004	Ausgrabungen in Biesheim (F), römisch Leitung: PD Dr. P.-A. Schwarz, C. Schucany
Jun.2003 – Aug. 2003	Ausgrabungen in Biesheim (F), römisch Leitung: PD Dr. P.-A. Schwarz, C. Schucany

Lehre

HS 2012	Übung: “Geomagnetische Prospektion“ im Rahmen der Übung Einführung in die Archäometrie, zusammen mit D. Brönnimann
FS 2012	Seminar: Neuste Forschungsergebnisse in der Ur- und Frühgeschichte und naturwissenschaftlichen Archäologie, zusammen mit J. Schibler, B. Röder, S. Straumann, F. Wegmüller, Ph. Wiemann
HS 2011 Übung:	GIS-Anwendungen in der Archäologie, zusammen mit R. Ebersbach und U. Brombach
HS 2011 Übung:	Workshop zu ausgewählten digitalen Techniken in der archäologischen Dokumentation, zusammen mit J. Schibler, B. Röder, S. Straumann, F. Wegmüller, Ph. Wiemann
FS 2007 – heute	Übung: “Digitale Vermessung mittels AutoCAD und TachyCAD“ im Rahmen der Übung Computational Archaeology (jeweils im Frühjahrssemester)

Während der Grabungen und Prospektionen: Ausbildung des Grabungspersonals in digitaler Grabungsvermessung mittels Tachymeter und AutoCAD (inkl. TachyCAD und PhoToPlan)

Sprachen

Deutsch	Muttersprache
Englisch	Gut
Französisch	Durchschnittlich

Anstellungen

Sep. 2012 – heute

GIS-Mitarbeiter, Archiv der Römerstadt Augusta Raurica ,
Poststrasse 1, 4302 Augst
Tätigkeitsmerkmale:

- GIS-Eingaben diverser Altgrabungen
- Erstellen von Vorlagen für die Planerstellung
- Unterstützung der Grabungstechniker beim Datenmanagement der Tachymeter

Mar. 2012 – heute

Wissenschaftlicher Mitarbeiter im Projekt “ Mediterraner Import und seine Rezeption nördlich von Etrurien: Transfer von Ideen und Ideologien in einer Kontaktzone der Mittelmeerwelt, 7. – 5. Jahrhundert v. Chr.“, Seminar für Klassische Archäologie, Universität Basel, Petergraben 51, 4051 Basel
Tätigkeitsmerkmale:

- GIS-gestützte Auswertung der Datenbank
- Erstellen einer online-Datenbank (inkl. Eingabe- und Suchmasken)
- Einrichten einer WebGIS-Anwendung zur Darstellung der Ergebnisse

Jan. 2010 – Feb. 2012

Koordinator des Digitalisierungsprojektes des Departements Altertumswissenschaften, Universität Basel, Petersgraben 51, 4051 Basel
Tätigkeitsmerkmale:

- Koordination der beteiligten Institutionen und Fachstellen
- Organisation und Führung der Hilfsassistierenden
- Unterstützung bei der Einrichtung der Datenbank und des Servers

Nov. 2004 – Apr. 2009

System- und Netzwerkadministrator am Institut für Prähistorische und Naturwissenschaftliche Archäologie, Universität Basel, Spalenring 145, CH-4055 Basel
Tätigkeitsmerkmale:

- Betreuung der Mitarbeitenden bei EDV-Problemen und Fragen
- Wartung und Betreuung des Servers, sowie der Hard- und Software des Instituts
- Planung von Neuanschaffungen

1999 – 2001

Ferienjob in den Schulferien: Technischer Zeichner (CAD) und Programmierer im Büro für Baurealisierung Weil am Rhein GmbH, Hauptstrasse 404, 79576 Weil am Rhein (D)
Tätigkeitsmerkmale:

- Konstruieren von Bauwerken mittels CAD (AutoCAD und CasCADe)
- Ingenieurvermessung mittels Tachymeter und Nivelliergerät
- Programmierung von Controllingsoftware (Arbeitszeiterfassung) für das Büro

Weitere Projektbeteiligungen (unentgeltlich)

Sep. 2011 – heute	Geomagnetismus-Projekt des IPNA und der Ur- und Frühgeschichtlichen und Provinzialrömischen Archäologien der Universität Basel (zusammen mit D. Brönnimann
Apr. 2010 – heute	Webmaster der EXPO¦ARCH¦DISS (http://expoarchdiss.archeo.unibas.ch/)
2009 – 2011	Mitarbeit in der PR-Gruppe des IPNA, Universität Basel

Publikationen

Schuhmann, D. (2014, in Vorbereitung), ElKowmGIS - Ein neues System zur Dokumentation archäologischer Fundstellen. Publikation der Dissertation.

Schuhmann, D. (2011), ElKowmGIS: A New Program for the Documentation of Archaeological Sites. In: Melero, F. Javier; Cano, Pedro; Revelles, Jorge, Proceedings of the 38th Conference on Computer Applications and Quantitative Methods in Archaeology Granada, Spain, April 2010, 1-10.

Schuhmann, D. (2011), A three-dimensional model of the Palaeolithic site of Hummal (Central Syria). In: Le Tensorer, Jean-Marie ; Jagher, Reto ; Otte, Marcel (Hrsg.), The Lower and Middle Palaeolithic in the Middle East and Neighbouring Regions. Etudes et Recherches Archéologiques de l'Université de Liège (ERAUL), Band 126, 2011, 279-288.

Schuhmann, D. (2010), ElKowmGIS: A New Program for the Documentation of Archaeological Sites. In: Melero, F. Javier; Cano, Pedro; Revelles, Jorge (Hrsg.), Fusion of Cultures - Abstracts Of The XXXVIII Annual Conference On Computer Applications And Quantitative Methods In Archaeology. Granada, Spain 2010, 511-514.
Schuhmann, D. (2006), Der Rhein in der Neuzeit. In: Hüster Plogmann, H. (Hrsg.), Fisch und Fischer aus zwei Jahrtausenden. Eine fischereiwirtschaftliche Zeitreise durch die Nordwestschweiz. Forschungen in Augst, Band 39, 2006, 63-69.

In Vorbereitung:
- Publikation der Ergebnisse des Anthropologischen Praktikums 2004:
  Die frühmittelalterlichen und mittelalterlichen Bestattungen des Theodorskirchplatz in Basel.

Zusätzlich:
- Diverse Grabungsberichte zu Händen des Schweizerischen Nationalfonds
- Grabungsbericht zu Händen des Pôle d'Archéologie Interdépartemental Rhénan (F)