Inhaltsverzeichnis

1 Einführung

"Ein Geo-Informationssystem ist ein rechnergestütztes System, das aus Hardware, Software, Daten und Anwendungen besteht. Mit ihm können raumbezogene Daten digital erfaßt und redigiert, gespeichert und reorganisiert, modelliert und analysiert sowie alphanumerisch und graphisch präsentiert werden." [BILL/FRITSCH 1991]

Somit ist die zu erstellende Software als Teil eines Geo-Informationssystems zu verstehen, dem Teil, der die Präsentation der Daten übernimmt. Die Vorteile der ALK als Geo-Informationssystem, ihre Aktualität durch tägliche Fortführung, ihre Zuverlässigkeit durch die Berücksichtigung geprüfter Katasterberechnungen und ihre strenge Standardisierung mit dem Objektschlüsselkatalog bieten ideale Voraussetzungen für eine breite Anwendung als Basis für einen Großteil raumbezogener Datenbanken.

"Die ALK als Teil des automatisierten Liegenschaftskatasters soll die Basis flurstücksbezogener Informationssysteme bilden. Dies setzt voraus, daß alle Anwender und Nutzer den digitalen Kartennachweis problemlos verarbeiten können. Für den Datenaustausch wird deshalb die für den Datenübergang im ALK-System zwischen ALK-Datenbankteil und ALK-Verarbeitungsteil konzipierte Einheitliche Datenbankschnitstelle (EDBS) verbindlich festgelegt." [ALK-RICHTLINIEN BRB]

Das Beharren der Landesregierung und der Kataster- und Vermessungsämter auf der aufwendig gestalteten Schnittstelle EDBS und die zurückhaltende Informationspolitik haben bisher eine breite Anwendung der ALK-Daten verhindert. Eine Veröffentlichung von Auszügen der ALK soll nun einen breiten Anwenderkreis für die ALK-Problematk interessieren.

Die Präsentation der ALK-Daten soll über ein Medium geschehen, welches sich erst in jüngster Zeit entwickelt hat, dem World-Wide-Web. Dabei sind Probleme zu bewältigen, die die rasante Entwicklung des WWW mit sich gebracht hat. Die geringen Datendurchsätze und die hohen Kosten der in Deutschland erhältlichen Internet-Anschlüsse behindern eine Übertragung größerer Datenmengen. Die fehlenden Standards für viele Anwendungsbereiche machen oft Lösungen notwendig, die einen großen Teil der Internet-Nutzer von vornherein ausschließen.

Die im Frühjahr 1995 von *Sun* erstmalig vorgestellte Programmiersprache JAVA macht die Entwicklung von systemunabhängiger Software möglich. Die mit JAVA entwickelten Programme können über das Internet verteilt und in Internet-Browsern ausgeführt werden. Damit wird es möglich, Daten weltweit zur Verfügung zu stellen,

die nicht den Standardformaten entsprechen. Diese Technik soll in der zu erstellenden Software angewendet werden.

2 Grundlagen

2.1 ALK, EDBS

2.1.1 Die ALK in Brandenburg

Gemäß Vermessungs- und Liegenschaftsgesetz Brandenburg stellen die Ergebnisse der Landesvermessung und die Nachweise des Liegenschaftskatasters ein öffentliches raumbezogenes Basisinformationssystem dar [VERMLIEGG]. Die Einrichtung und Fortführung des Liegenschaftskatasters sowie seiner Weiterentwicklung sind so zu gestalten, daß es den Anforderungen des Rechtsverkehrs, der Verwaltung und der Wirtschaft an ein öffentliches raumbezogenes Basisinformationssystem gerecht wird. Die Verantwortung liegt dabei beim Landesvermessungsamt, es ist laut §6 VermLiegGZW zuständig für die *„Einführung, Beschaffung, Ersatzbeschaffung und laufende Unterhaltung einheitlicher Verfahrenslösungen, Meß-, Auswerte- und Informationssysteme des Landesvermessungsamtes und der Katasterbehörden, die zur landeseinheitlichen Wahrnehmung der Aufgaben gemäß Vermessungs- und Liegenschaftsgesetz erforderlich sind.“* [VermLiegGZW]

Die ALK-Brandenburg mit den durch den Runderlaß III Nr.25/1995 des Ministerium des Innern eingeführten ALK-Richtlinien basiert auf der Verfahrenslösung Automatisierte Liegenschaftskarte der Arbeitsgemeinschaft der Vermessungsverwaltungen der Bundesrepublik Deutschland (AdV). Mit dieser Verfahrenslösung haben die Bundesländer ein entsprechendes Rahmenkonzept für ein fachübergreifend nutzbares, liegenschaftsbezogenes Geoinformationssystem entwickelt.

Die Voraussetzungen für die Errichtung der Automatisierten Liegenschaftskarte im Land Brandenburg waren denkbar schlecht. Koordinaten für Grenz- bzw. Gebäudepunkte lagen kaum vor und noch heute werden ein Teil der Fortführungsvermessungen durch die fehlenden Verdichtung des Trigonometrischen Netzes in lokalen Koordinatensystemen realisiert. Die graphische Genauigkeit und der Inhalt der vorliegenden Kartenwerke entsprechen im Regelfall, insbesondere bei Inselkarten im ländlichen Raum, nicht den Erfordernissen, die für eine Digitalisierung anzusetzen sind. Ein flächendeckender Aufbau der ALK-Brandenburg mit hoher Genauigkeit und vollständigen Inhalt ist aufgrund dieser Voraussetzungen

mittelfristig nicht zu realisieren. Um aber potentiellen Sekundäranwendern (Kommunen, Leitungsnetzbetreiber, Energieversorgungsunternehmen) die Nutzung der ALK zu ermöglichen, war es notwendig, frühzeitig raumbezogene Basisdaten bereitzustellen.

Um ein Mindestmaß an Informationsgehalt und geometrischer Genauigkeit zu gewährleisten, wurde in den ALK-Richtlinien [ALK-RICHTLINIEN BRB] eine ALK-Vorstufe definiert. Als ihre Ziele werden dort genannt:

- den Sekundäranwendern frühzeitig und bedarfsorientiert raumbezogene Basisdaten mit einer landesweit einheitlichen Datenstruktur und mit landesweit einheitlichen Mindestgehalt bereitzustellen,
- die analoge Liegenschaftskarte bedarfsorientiert durch den Datenbestand der ALK zu ersetzen,
- sie zur Endstufe der ALK weiterzuentwickeln.

Die Vorstufe der ALK umfaßt als Grunddatenbestand (Mindestinhalt)

- die Grenzen der BR Deutschland und des Landes Brandenburg,
- die Grenzen der Kreise, Gemeinden, Gemarkungen, Fluren und Flurstücke,
- die in der herkömmlichen analogen Liegenschaftskarte dargestellten Gebäude,
- die Flurstücksnummern, Hausnummern, Straßennamen sowie weitere erläuternde Beschriftungen,
- die Aufnahme-, Grenz- und Gebäudepunkte des amtlichen Nachweises (Punktdatei) und
- die Topographie der Liegenschaftskarte.

Es wird angestrebt, den Grunddatenbestand vorrangig um

- den vollständigen Gebäudebestand,
- die weitere Topographie, soweit Bedarf besteht, und
- die tatsächliche Nutzung

zu erweitern.

Darüber hinaus sollen weitere Daten für die ALK-Vorstufe nur erfaßt werden, wenn

- dies ohne erheblichen Mehraufwand möglich ist,
- die Angaben nach der ALK-Systematik zu verschlüsseln sind,
- die Fortführung der Datenbestände der Liegenschaftskarte gesichert ist.

Die ALK-Richtlinien definieren die geometrische Genauigkeit der ALK-Vorstufe in Abhängigkeit zur analogen Liegenschaftskarte. „Die geometrische Genauigkeit der dargestellten Liegenschaften muß mindestens der Genauigkeit der herkömmlichen

analogen Liegenschaftskarte entsprechen. Die geometrische Genauigkeit ist bei jeder sich bietenden Gelegenheit zu verbessern."

Um die Vorstufe der ALK bedarfsgerecht zu produzieren, sollen von den Kataster- und Vermessungsämtern Prioritätslisten in Abstimmung mit Kommunen und Sekundäranwendern aufgestellt werden. Dabei werden die Gebiete, in denen besonderer Bedarf an Daten der ALK-Vorstufe besteht, in die Prioritätsstufen 1 bis 3 eingeteilt. Für die Prioritätengebiete soll eine Bestandsaufnahme durchgeführt werden, bei der

- die wirtschaftlichste Methode zur Errichtung der Vorstufe der ALK,
- die voraussichtliche Bearbeitungsdauer und
- die notwendige Vorarbeit

ermittelt werden.

Auf der Grundlage der Prioritätenanalyse und der Bestandsaufnahme soll in Abstimmung mit den Kommunen und Sekundäranwendern zum 1. Juli jeden Jahres ein aktueller Arbeitsplan aufgestellt bzw. fortgeschrieben werden.

Der Arbeitsplan beinhaltet folgende Angaben:

- die geplante zeitliche Reihenfolge der Einrichtung in den Prioritätengebieten,
- den voraussichtlichen Beginn und die geplante Fertigstellung der Prioritätengebiete,
- die Einrichtungsmethode,
- notwendige Vorarbeiten.

Als Standardmethode zur Errichtung der ALK-Vorstufe soll unter Verwendung von geeigneten vorliegenden Punktdaten eine 1:1 – Digitalisierung der Liegenschaftskarte oder anderer geeigneter Karten durchgeführt werden.

Es hat sich gezeigt, daß durch die Einrichtung der ALK-Vorstufe die kalkulierten Kosten zu Erstellung der ALK um über ein Drittel gesenkt und die geplante Bearbeitungsdauer wesentlich reduziert werden konnte [MASUR 1997].

2.1.2 Die EDBS-Schnittstelle

Die Einheitliche Datenbankschnittstelle (EDBS) ist im engen Zusammenhang mit der ALK zu sehen. Die ALK besteht im wesentlichen aus drei Datenbanken:

- der Punktdatei
- der Grundrißdatei
- und der Datei der Messungselemente.

Dabei bildet das Punktkennzeichen den Hauptschlüssel in allen drei Datenbanken.

Um eine Kommunikation der ALK-Datenbank mit Anwendungsprogrammen zu ermöglichen, war es notwendig, eine Schnittstelle zu entwickeln, die:

- einen einheitlichen Aufbau hat
- so flexibel ist, daß es unterschiedliche Informationen aufnehmen kann
- die Art der Bearbeitung enthält
- und die Datenbankabfragen aufnehmen kann.

Dieses Datenformat stellt die Einheitliche Datenbankschnittstelle dar.

Für die Arbeit mit der Datenbank sind innerhalb der EDBS-Schnittstelle folgende Funktionen vorgesehen:

- Auftragsverwaltung und Auftragsorganisation,
- Einrichten von Primärdaten
- Fortführen von Primärdaten
- Definition von Benutzeranforderungen
- Übergabe von Ergebnisdaten
- Übergabe von Verarbeitungsprotokollen.

Für Daten, die in die ALK übernommen werden sollen, ist die EDBS-Schnittstelle zwingend vorgeschrieben. In den ALK-Datenerfassungsrichtlinien wird gefordert: *„Vom Auftragnehmer ist zu verlangen, daß landesrechtliche Vorschriften zur Datenstrukturierung und zum Datenaustausch Anwendung finden. Dazu gehören insbesondere OSKA- und OBAK-LIKA-Brandenburg und die Einheitliche Datenbankschnittstelle (EDBS) als Datenaustauschformat.“* [ALK-RICHTLINIEN BRB]

Für den Austausch von Basisdaten innerhalb der brandenburgischen Landesverwaltung ist die EDBS-Schnittstelle gemäß dem Kabinettsbeschluß "Digitale Karte" vom 28. Juni 1994 zwingend vorgeschrieben, für den Austausch von allgemeinen geographischen Daten wird sie empfohlen.

Die Abgabe von Daten der ALK an Sekundäranwender wird von den Katasterämtern des Landes Brandenburg nur über die EDBS-Schnittstelle angeboten. Andere Länder gestalten die Datenabgabe kundenfreundlicher, so werden zum Beispiel vom Landesvermessungsamt Baden-Würtenberg auch Daten im BGRUND-, DXF- und im SICAD-GDB-Format angeboten. Eine flexiblere Handhabung der Schnittstellenproblematik würde die Akzeptanz der ALK als Basis für alle raumbezogenen Daten sicherlich erhöhen.

2.2 Internet

2.2.1 Geschichte

Die Grundlagen für das erst im jüngster Zeit in die Öffentlichkeit getretene Internet wurden bereits Ende der 60er Jahre gelegt. Die Firma Xerox entwickelte das Ethernet-Konzept und das TCP/IP-Protokoll, das heute den de-facto-Standard der Internetkommunikation darstellt. Eine Eigenschaft der TCP/IP-Protokoll ist es, daß Nachrichten in einzelne Pakete aufgeteilt werden, die unabhängig voneinander zum Ziel gelangen. Am Zielort werden sie wieder zu der kompletten Nachricht zusammengesetzt, verlorene Pakete können nachgefordert werden.

Auf dieser Basis entstand das Netz der ARPA (Advanced Research Projekt Agency), einer Projektgruppe des amerikanischen Verteidigungsministeriums. Mit diesem Netz wurden zunächst vier Universitäten und Forschungseinrichtungen verbunden, um Großrechnerkapazitäten besser ausnutzen zu können.

Parallel zur Entwicklung des ARPANET wurde Anfang der 80er Jahre das CSNET(Computer Science Research Network) aufgebaut. Mit diesem Netz wollten die Universitäten, die keinen Zugang zum ARPANET hatten, den damit verbundenen Nachteil bezüglich der Forschung wettmachen. Das 1970 entwickelte UUCP (Unix-to-Unix-Copy-Protokoll) ermöglichte dabei eine einfache Datenübertragung über das bestehende Telefonnetz mit Hilfe von Modems.

Die über das TCP/IP-Protokoll realisierte Verbindung zum ARPANET markierte dann den ersten Meilenstein auf dem Weg zum heutigen Internet. Das kostenlos zur Verfügung gestellte Protokoll fand rasche Verbreitung und wurde Anfang der 80er Jahre vom Department of Defense zum nationalen Standard erklärt.

Mit der Entstehung des Internet-Dienstes WWW (World Wide Web), welches 1989 von Physiker des Kernforschungszentrums CERN in Genf entwickelt wurde, begann der Aufstieg des Internets zum Massenmedium. Diese Netzwerkarchitektur baut auf drei Komponenten auf:

- dem Hypertext-Transfer-Protokoll (HTTP),
- der Textauszeichnungssprache HTML,
- dem WWW-Browser.

Die Tatsache, daß die Benutzung keine umfangreichen Kenntnisse erfordert und auch die Erstellung eigener WWW-Dokumente in kurzer Zeit erlernt werden kann, hat zu einer rasanten Verbreitung des World Wide Web geführt.[TOLKSDORF 1996]

2.2.2 HTML - die Sprache des WWW

HTML ist eine Dokumentbeschreibungssprache, die für das WWW entwickelt wurde. Mit ihr wird der logische Aufbau der Struktur eines Dokumentes beschrieben.
Die Strukturen des Dokumentes werden mit sogenannten Anfangs- und Ende-Tags markiert, so können Absätze, Überschriften, Listen und Tabellen gekennzeichnet werden, aber auch Grafiken und Multimedia-Dateien in das Dokument integriert werden. In HTML-Dokumente eingebundene Links können auf andere Dokumente im Internet verweisen, damit ist die Navigation von Seite zu Seite möglich. HTML-Dateien können systemunabhängig mit einer Präsentations-Software dargestellt werden. Aus diesem Grund wird HTML auch immer mehr in Hilfe- oder Navigationssystemen benutzt, die auf unterschiedlichen Plattformen lauffähig sein sollen.

2.2.3 Grafik im Internet

Im WWW haben sich zwei Grafikformate, das GIF und das JPEG-Format, durchgesetzt. Sie werden inzwischen von allen gängigen Internet-Browsern dargestellt. Bei beiden handelt es sich um pixelorientierte Formate, die aber für unterschiedliche Zwecke eingesetzt werden sollten:

- Das GIF-Format wurde speziell für den Online-Einsatz entwickelt, es zeichnet sich durch eine hohe Komprimierungsdichte aus. Das GIF-Format kann maximal eine Farbtiefe von 256 Farben abspeichern, es eignet sich besonders für Grafiken, die größere Flächen mit gleichen Farbwerten besitzen, also für Buttons, Symbole und Cliparts.
- Das JPEG-Format kann 16,7 Millionen Farben bei sehr guten Komprimierungsraten speichern, allerdings werden diese durch eine verlustbehaftete Speicherung erreicht. JPEG eignet sich besonders zum Speichern von Fotos oder Bildern mit feinen Farbverläufen.

Vektorgrafikformate, die eine Beeinflussung des dargebotenen Inhalts durch den Nutzer möglich machen, haben sich im Internet noch nicht durchgesetzt. [www.teamone.de/selfhtml]

2.3 Java

Eine der neuesten Entwicklungen auf dem Softwaremarkt ist *Suns* neue Programmiersprache JAVA. Diese hat sich in den letzten zwei Jahren zu einem

beliebten „Spielzeug“ im WWW entwickelt. Doch in JAVA steckt ein wesentlich größeres Potential, als seine derzeitige Marktposition vermuten läßt.

2.3.1 Geschichte

Die Entstehungsgeschichte von JAVA begann im Jahre 1990; ein Team von Ingenieuren bei *Sun Microsystems* stellte Konzepte für eine neue Richtung bei High-Tech- und Endkonsumenten- Technologien auf. Schon 1990 war klar, daß Computer in sämtlichen Lebensbereichen zu finden sein werden und sie die Steuerung von vielen Home-Produkten übernehmen werden. Dazu mußten die Computer aber kleiner, billiger und vor allem einfacher zu bedienen sein werden.
Im Jahre 1991 stellte Sun ein Team von Entwicklern auf, die unter der Führung von James Gosling und Billy Joy Hard- und Software für interaktives Fernsehen und andere Geräte der Konsumelektonik entwickeln sollten. Bestandteile dieses Projekts, daß später den Namen GREEN bekam, waren ein Betriebsystem, ein Interpreter, ein Grafiksubsystem und diverse Hardwarekomponenten.
Im Herbst 1992 konnte das Team ein Gerät mit dem Namen „*7“ (Star Seven) firmenintern vorstellen, die Entwicklung beeindruckte einige Sun-Manager und es wurde im November 1992 die Firma *First Person, Inc.* gegründet. Die Versuche von *First Person*, Verträge zur Vermarktung von STAR SEVEN abzuschließen, scheiterten allesamt, so daß man sich 1994 entschloß, die Firma aufzulösen.
Inzwischen begann die rasante Entwicklung des World Wide Web. Der erste graphische Web-Browser NCSA MOSAIC war im April 1993 verfügbar, mit ihm konnte jedermann graphisch aufbereitete Informationen im WWW abrufen. *Sun* erkannte das darin liegende Potential und stellte die Entwicklung des GREEN-Projektes auf das World Wide Web um. Im Mai 1995 konnte *Sun* den HOT-JAVA-Browser auf der SunWorld ´95 vorstellen, mit ihm konnten kleine Programme, Applets genannt, über das WWW geladen und innerhalb des Browsers ausgeführt werden.
Der Durchbruch der JAVA-Technologie gelang, als sich die Firma *Netscape* entschied, JAVA von *Sun* zu lizensieren und in der Version 2.0 ihres NAVIGATORS einzubauen. Damit wurde JAVA einem breiten Publikum zur Verfügung gestellt, binnen kurzer Zeit wurden hunderte Applets geschrieben, die schon früh einen Eindruck von den Möglichkeiten der Sprache vermittelten. Ein weiterer Meilenstein in der Entwicklung von Java war die Implementierung in *Microsofts* INTERNET-EXPLORER.

Mit der Weiterentwicklung von JAVA wurde die aus den verbliebenen Mitgliedern des Green-Teams gegründete Firma *JavaSoft* betraut. Sie brachte im Januar 1996 das Java-Development-Kit (JDK) 1.0 auf den Markt. In der folgenden Zeit wurden wichtige strategische Allianzen mit Firmen wie *Borland*, *Lotus*, *Oracle*, *IBM*, *Netscape* und *Symantec* gebildet. Bis Ende 1996 wurden zahlreiche Neuerungen vorgestellt, die ersten integrierten Entwicklungsumgebungen wurden vom *Microsoft* und *Symantec* vorgestellt. Im Frühjahr 1997 wurde das JDK 1.1 vorgestellt, bei dem einige Änderungen am Sprachkonzept vorgenommen wurden. [KRÜGER 1997]

2.3.2 Eigenschaften von Java

Die Entwickler von Java faßten ihre Bemühungen wie folgt zusammen:

„Java soll eine einfache, objektorientierte, verteilte, interpretierte, robuste, sichere, architekturneutrale, portable, performante, nebenläufige, dynamische Programmiersprache sein.“ [KRÜGER 1997]

- Java ist einfach
  Da Java eine Programmiersprache mit fortschrittlichen Eigenschaften ist muß sie wie eine solche erlernt werden. Die starke Anlehnung an den C++-Syntax soll vielen Programmierern den Umstieg erleichtern, dabei soll das Fehlen C++-typischer Features, wie Pointer, Mehrfachvererbung und Templates, Fehler bei der Programmierung vermeiden. Die Klassenbibliothek wurde neu entworfen und kommt damit ohne die Altlasten gewachsener Bibliotheken aus.
- Java ist objektorientiert
  Java ist konzeptionell verwandt mit verschiedenen anderen Hochsprachen wie C++, Eiffel, Pascal und Modula. Es arbeitet wie C++ in logischen Einheiten, Klassen genannt, die aus Daten und Methoden bestehen. Aus Klassen können Objekte erzeugt werden, die wie Variablen angelegt und verwendet werden können.
- Java ist verteilt
  Java verfügt über eine umfangreiche Bibliothek für den Zugriff auf TCP/IP-Protokolle wie etwa http und ftp. Der Zugriff auf Objekte im Netz erfolgt so einfach wie der Zugriff auf ein lokales Objekt.

- Java ist interpretiert

  Der Javaquellcode wird zunächst in Java-Byte-Code kompiliert. Der Java-Byte-Code ist der Maschinensprache strukturell verwandt. Der kompilierte Quellcode wird von einem Interpreter, der Java Virtual Machine, ausgeführt.
- Java ist robust

  Durch die Vereinfachung der Sprache Java gegenüber komplexeren Sprachen ergibt sich der Vorteil, daß einige Fehlerquellen von vornherein ausgeschaltet werden. Die Speicherverwaltung wird vollständig durch die Java-Laufzeitumgebung erledigt.
- Java ist sicher

  Durch die Möglichkeit, Java-Anwendungen über das Netz zu verteilen, ist ein erhöhtes Maß an Sicherheit erforderlich. Gerade im Internet, wo der Datenverkehr theoretisch für jedermann zugänglich ist, kommt diese Anforderung zum tragen. Java besitzt zu diesem Zweck ein Byte-Code-Verifier, der Änderungen am Code erkennen und gewisse Laufzeitregeln überprüfen soll.
- Java ist architekturneutral

  Die besondere Eigenschaft von Java ist die Architekturunabhängigkeit des Byte-Codes. Java ist auf jedem System lauffähig, für das ein Java-Interpreter verfügbar ist.
- Java ist portabel

  Durch die Architekturunabhängigkeit von Java muß nicht mehr von einer Plattform auf die andere portiert werden. Mit dem Abstract Windowing Toolkit (AWT) steht eine GUI-Bibliothek zur Verfügung, mit der der Programmierer portable Oberflächen entwerfen kann.
- Java ist performant

  Die Geschwindigkeit von interpretiertem Code ist ausreichend für die meisten Anwendungen. Rechenintensive Programme können allerdings um den Faktor 10 bis 20 langsamer als vergleichbarer C-Code sein. Durch die Entwicklung von Just-In-Time-Compilern, Native-Code-Compilern oder Java-Prozessoren wird sich in Zukunft die Performance der von C-Programmen annähern.
- Java ist nebenläufig

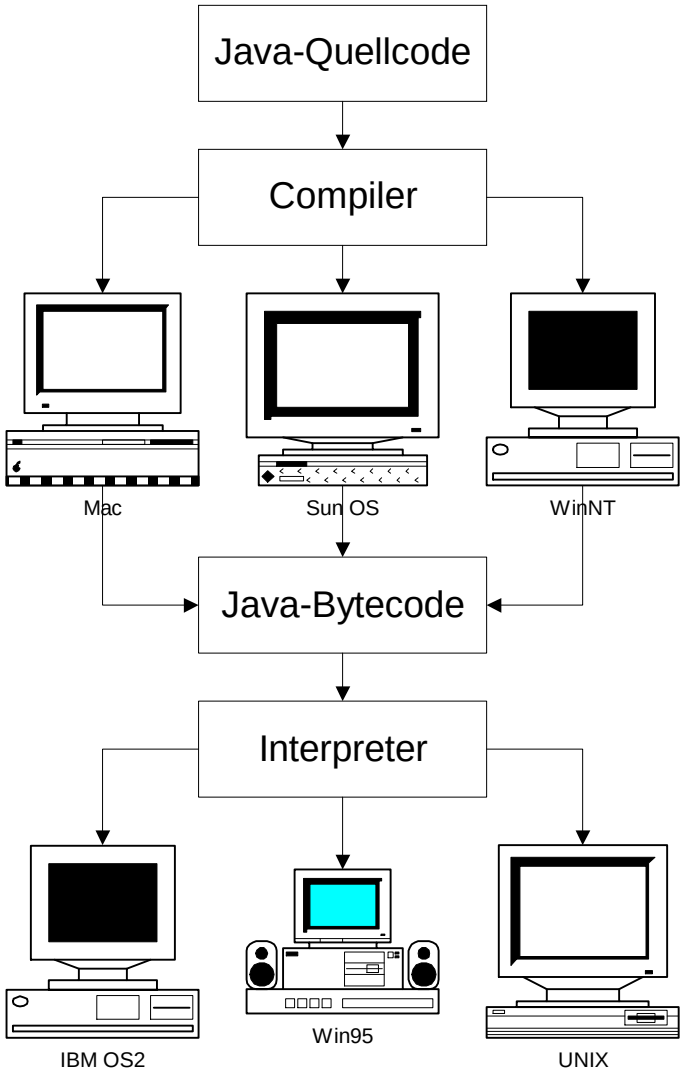
  Java unterstützt Multithreading. Dadurch verbessert sich die interaktive Antwortfähigkeit und das Laufzeitverhalten von Programmen. Dieses Verhalten ist allerdings stark von dem zugrundeliegenden Betriebssystem abhängig.

- Java ist dynamisch

  In Java werden die Klassen erst zur Laufzeit gebunden. Modifikationen an den Klassenbibliotheken können deshalb jederzeit nachträglich vorgenommen werden, ohne daß der gesamte Quellcode neu kompiliert werden muß. [KNOBLOCH 1997]

2.3.3 Arbeiten mit Java

Damit Javaprogramme unabhängig vom jeweiligen Betriebssystem sind wird ein systemunabhängiger Byte-Code erzeugt. Dieser Byte-Code kann von beliebigen Stellen des Netzwerkes geladen werden.



Java Virtual Machines sind inzwischen für alle gängigen Betriebssysteme verfügbar. Die von *Sun* frei zur Verfügung gestellte Entwicklungsumgebung Java-Development-

Kit (JDK) ist für die Plattformen x86\Solaris, SPARC-Solaris, Linux, MacOS, WindowsNT und Windows 95 erhältlich.

2.3.4 Java-Applets und Applikationen

Im wesentlichen unterscheiden sich die beiden möglichen Programmtypen Applet und Applikation nicht. Java-Applikationen sind eigenständige Programme, die zur Ausführung einen Stand-Alone-Interpreter benötigen, während Java-Applets Programme sind, die zur Ausführung einen Web-Browser benötigen. Dabei unterliegen Applets einigen Einschränkungen:

- Sie dürfen keine Dateien auf der lokalen Maschine lesen oder schreiben.
- Sie dürfen keine Netzwerkverbindungen eröffnen, außer zu der Maschine, von der der Appletcode geladen wurde.
- Es dürfen keine Programme gestartet werden.
- Sie dürfen keine nativen Methoden aufrufen.

Dagegen bieten Applets einige zusätzliche Möglichkeiten; die Ausgabe von Sounds beispielsweise ist standardmäßig auf Applets beschränkt. [SCHMIDT 1997]

3 Konzeption

Im Rahmen dieser Arbeit soll ein Programmsystem entwickelt werden, welches es möglich macht, Vektordaten, hier speziell ALK-Daten, über das Medium Internet zu veröffentlichen.

3.1 Vorbetrachtungen

Derzeit gibt es wenig Möglichkeiten, im Internet Vektordaten darzustellen. Die meisten Lösungen basieren auf Plug-Ins, Zusatzprogramme, die die speziellen Vektorformate in dem jeweiligen Internet-Browser darstellen. Diese Lösung ist nicht sehr nutzerfreundlich, die Plug-Ins sind in der Regel nicht kostenfrei erhältlich oder müssen zumindest zeit- und damit kostenintensiv downgeladen werden. Würde das geplante Programmsystem nur für den Datenaustausch mit Kunden des Landesvermessungsamtes Brandenburg gedacht sein, wären diese Zusatzprogramme durchaus noch einsetzbar. Sollen allerdings die ALK-Daten einem breiteren Publikum zugänglich gemacht werden, sind Techniken notwendig, die für den Durchschnittsnutzer des Internets erreichbar sind. Hierfür bietet Java ideale Voraussetzungen.

Es ist durchaus denkbar, das auch Vektordaten, die nicht aus ALK-Systemen stammen, im Internet veröffentlicht werden sollen. Es ist also bei der Erstellung des Programmsystems darauf zu achten, daß die interne Datenbank auch offen für andere Vektorformate gestaltet wird. Das Programmsystem könnte in späteren Versionen auch Grundlage für die Veröffentlichung von Bauprojekten oder technischen Zeichnungen sein. Speziellen Augenmerk soll auf die spätere Lesbarkeit von DXF-Daten gelegt werden, können doch die meisten Vektorgrafikprogramme dieses Format in guter Qualität erzeugen.

Das zu erstellende Programmsystem soll über das sehr schnellebige Medium Internet arbeiten. Die Techniken im Umfeld des Internets sind einer ständigen Dynamik unterworfen, so daß Programme in diesem Bereich kurzen Entwicklungs- und Updatezyklen unterworfen sind. Ebenso muß man mit sich ändernden Anforderungen an die EDBS-Schnittstelle und den Objektschlüsselkatalog rechnen. Je größer der Nutzerkreis der ALK-Daten wird, desto umfangreicher werden die Strukturen, die die Schnittstelle übertragbar machen muß. Besonderen Augenmerk ist also bei der Erstellung des Programmsystems auf die Erweiterbarkeit des Programmcodes zu legen.

Die Datenübertragungsquoten über das Internet sind immer noch so gering, daß das Laden von größeren Datenmengen mit hohem Zeit- und Kostenaufwand verbunden sind. Eine kurzfristige Änderung der Situation ist auch nicht in Sicht, weil die ständig steigenden Nutzerzahlen den Ausbau der Netzinfrastruktur relativieren. Die zu übertragenden Datenmengen sollten für den Normalnutzer 100kb nicht überschreiten. Daten für interessiertes Fachpublikum können durchaus auch den doppelten Umfang erreichen. Es muß also ein eigenes Dateiformat erstellt werden, da EDBS-Dateien für die Übertragung zu umfangreich sind. Dieses Dateiformat sollte möglichst kompakt sein. Durch Umstellung auf das Binärformat und durch sinnvolle Datenreduktion kann dies erreicht werden.

Bei der Konzeption eines Java-Programmsystems sind weiterhin einige Dinge zu beachten, die man von anderen Programmiersprachen her nicht kennt. So wird doch der erzeugte Byte-Code von einem Interpreter ausgeführt, der nicht oder noch nicht in jeder Version für jedes Betriebssystem erhältlich ist. Als Entwicklungsumgebung wurde *Suns* Java-Development-Kit in der Version 1.1 gewählt. Die Java-Version 1.1 unterstützen zwar nur die neuesten Internet-Browser, aber mit der Version 1.1 wurden Änderungen am Sprachkonzept, besonders bei der Eventbehandlung, eingeführt die sehr aufwendige Programmumgestaltungen notwendig machen würden, wenn der Quelltext im nachhinein umgestellt werden müßte. An Serverstatistiken kann man feststellen, daß sich der Großteil der Internetnutzer relativ schnell die neuesten Programme zulegt, so daß nicht allzuviel Internetteilnehmer von der Nutzung des geplanten Programmsystems ausgeschlossen werden. So waren in der Statistik der Firma Webhits Internet Design GmbH am 3.Dezember 1998 54% aller Browser fähig, Java 1.1 zu interpretieren. [www.webhits.de]

Auch bei der Gestaltung der Oberfläche ist auf das Ziel zu achten, das das Programm auf unterschiedlichen Betriebssystemen genutzt wird. Es sollten also möglichst nur die Dialogelemente genutzt werden, die das Abstract Windowing Toolkit (AWT) bietet. Diese Dialogelemente sind in der Regel gut an die jeweiligen Plattformen angepaßt, so daß der Anwender nicht mit einer ungewohnten Benutzerschnittstelle konfrontiert wird [STEPPAN 1998].

3.2 EDBS- Datenstruktur

3.2.1 Aufbau eines EDBS-Datensatzes

Eine EDBS-Datei besteht aus einer Abfolge von Datensätzen, die in der Regel eine Zeile einnehmen. Die Datensätze können jedoch auch bei Überlänge über mehrere, hintereinanderliegende Zeilen verteilt sein.

Ein EDBS-Datensatz ist immer wie folgt aufgebaut:

von Byte	bis Byte	Kürzel	Länge (Byte)	Bedeutung
0	3	SA	4	Satzart
4	11	SL	8	Satzlänge
12	15	OP	4	Operationsschlüssel
16	27	QU	12	Quittierungs- und Editierschlüssel
28	35	IN	8	Informationsname
36	...	II	variabel	Informationsinhalt
...	...	SK	variabel	Suchkriterium

- Satzart SA

  Die Satzart stellt die Kennung eines EDBS-Satzes dar. Sie besitzt immer den Wert „EDBS“. Die Satzart sollte durch das Einleseprogramm geprüft werden, bei Widerspruch muß die gesamte Datei abgewiesen werden.

- Satzlänge SL

  Die Satzlänge enthält zwei Werte. Die ersten vier Byte stellen die Länge des Datensatzes ab dem Operationsschlüssel dar, die folgenden vier Byte enthalten die Entfernung zum Beginn des Suchkriteriums beginnend ab dem Operationsschlüssel. Das Einleseprogramm kann die Satzlänge mitführen. Da die folgenden Aggregate alle feste Längen besitzen, ist dies nicht zwingend notwendig.

- Operationsschlüssel OP

  Der vier Byte lange Operationsschlüssel gibt an, wie mit den folgenden Daten verfahren werden soll. Folgende Schlüssel sind gebräuchlich:

OP-Schlüssel	Bedeutung
Gruppe A...	Operationen zur allgemeinen Auftragsverwaltung
AKND	Auftragskennsatz
AEND	Endekennsatz
Gruppe F...	Operationen der Fortführung der Primärdaten
FEIN	Eintragen von Daten
FAEN	Verändern von Daten
FLOE	Löschen von Daten
FLAG	Fortführung, allgemein
Gruppe B...	Operationen der Benutzung von Primärdaten
BSPE	Ausgeben von Daten
BOBI	Ausgeben von Objekten
BALL	Ausgeben von Daten und Objekten

Gruppe P...	Operationen des Verarbeitungsprotokolls
PROT	Ausgeben von Verarbeitungsprotokollen

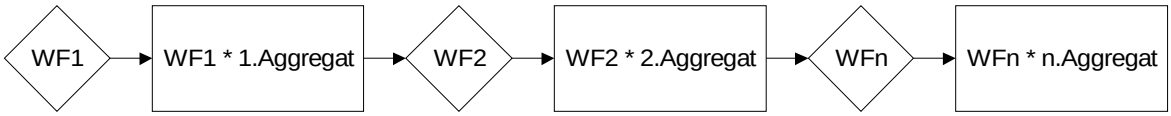
Bei der Abgabe von Daten an Sekundäranwender wird der Operationsschlüssel BSPE vergeben. Diese Operation der Gruppe B sollte das Einleseprogramm in jedem Fall annehmen. Weiterhin kann die Operation FEIN angenommen werden, alle anderen Operationen aus der Gruppe F sind für die Übernahme nicht sinnvoll, da sonst die Grundrißdatenbank in ihrer gesamten Informationstiefe laufendgehalten werden müßte. Weiterhin können Daten aus dem Auftragskennsatz AKND entnommen werden.

- Quittierungs- und Editierschlüssel QU
  In den ersten 6 Byte wird die fortlaufende EDBS-Satznummer eingetragen. Diese kann vom Programm gelesen und zur Ablaufkontrolle genutzt werden. In den folgenden 2 Byte ist der Editier- bzw. Zugehörigkeitsschlüssel vermerkt. Damit werden Dateizeilen gekennzeichnet, die zu einem EDBS-Datensatz gehören. Ein „A“ markiert die erste Dateizeile, ein „F“ die Folgezeilen und ein „E“ die letzte zum Datensatz gehörige Zeile. Die Beachtung dieses Schlüssels durch das Einleseprogramm ist zwingend erforderlich. Schließlich ist in Byte 9-12 der Quittierungsschlüssel vermerkt.
- Informationsname IN
  In den zur Verfügung stehenden 8 Byte ist der Name des Datenaggregates verzeichnet, auf das die Operation angewandt werden soll. Der Name des Datenaggregates entscheidet über der grundsätzlichen Aufbau des folgenden Informationsinhaltes. Für die Übernahme durch das Einleseprogramm sind folgende Datenaggregate zu beachten:
  ULQA000 Auftragskennzeichen
  ULO8ALK Grundrißdaten ALK-GIAP
  ULOBNN Grundrißdaten ALK/ATKIS
  Das Landesvermessungsamt Brandenburg und die Katasterämter übergeben Grundrißdaten über ULO8ALK-Aggregate, in anderen Bundesländern(z.B. Niedersachsen) werden jedoch auch ULOBNN-Aggregate verwendet. Das Programm sollte mit beiden Aggregaten umgehen können.
- Informationsinhalt II
  In diesem Bereich stehen die eigentlichen Daten. Der Aufbau wird in 3.2.2 und 3.2.3 näher beschrieben.

- Suchkriterium SK

  Das Suchkriterium steht im Auftragskennsatz, es ist für die Übernahme nicht relevant.

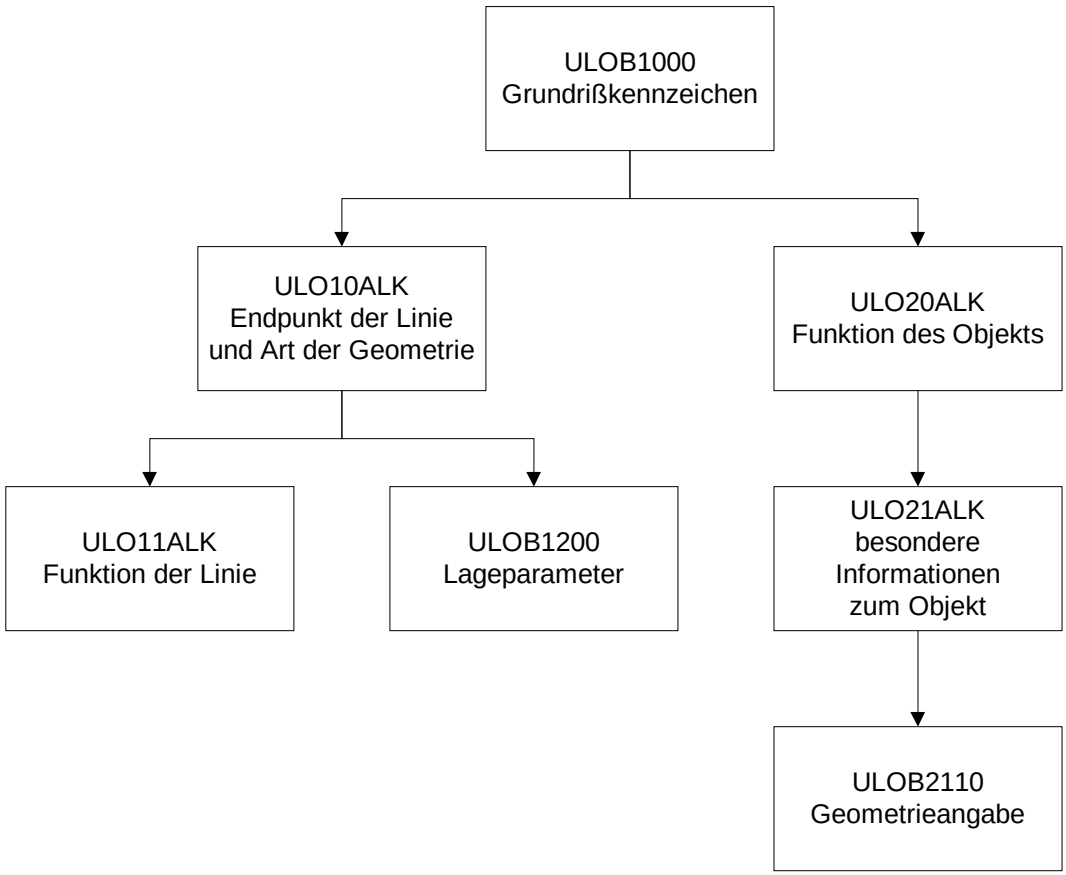
Die Informationen sind in sogenannten Standardaggregaten gespeichert. Dabei ist eine gewisse Abfolge zwingend, die Anzahl der Aggregate kann jedoch variieren. Um die Anzahl zu kennzeichnen sind zwischen die Aggregate 4 Byte lange Wiederholungsfaktoren eingefügt. [RUNDSCHREIBEN LVA AUG.1995]



Diese Wiederholungsfaktoren sind von dem Programm auszuwerten und für den Ablauf des Einlesevorgangs zu verwenden.

3.2.2 Aufbau des Datenaggregates ULO8ALK

Das Standardaggregat ULO8ALK besteht aus weiteren Aggregaten und besitzt folgenden grundsätzlichen Aufbau:



- Standardaggregat ULOB0000

  Dieses Aggregat dient der Ordnung und der Kennzeichnung der Grundrißeinheiten. Es besteht aus den Datenelementen DLOB0001–DBLOB0003. In ihnen ist die Objektkoordinate bzw. Der Linienanfangspunkt verzeichnet.

  Nach dem Lesen des Aggregates ULOB0000 wird entweder in den zum Endpunkt der Linie oder zur Funktion des Objekts verzweigt.

- Standardaggregat ULO10ALK

  In diesem Aggregat wird der Endpunkt der Linie sowie die Art der Geometrie nachgewiesen. Die Datenelemente DLOB1001 und DLOB1002, die die Koordinate des Endpunktes nachweisen, entsprechen im Aufbau den Datenelementen im Aggregat ULOB0000. Die Art der Geometrie wird im Datenelement DLOB1003 mit einem zweistelligen numerischen Schlüssel nachgewiesen. Ist dieser Schlüssel ungleich „11“ (Gerade), muß das Standardaggregat ULOB1200 im Datensatz enthalten sein.

- Standardaggregat ULO11ALK

  In diesem Aggregat wird die Funktion der Linie beschrieben, und die Linie wird dem Objekt zugeordnet. In den Datenelementen DLOB1101 und DLOB1102 ist die Folie sowie die Objektart (OSKA-Schlüssel) verzeichnet. In den Datenelementen DLOB1103 und DLOB1104 sind die Objektnummer rechts und links der Linie gespeichert. Durch diese Objektnummern kann die Funktion der Linie dem jeweiligen Objekt zugeordnet werden. Eines beider Datenelemente muß belegt sein. In den Datenelementen DLOB1107 und DLOB1108 ist die Linienteilung verzeichnet.

- Standardaggregat ULOB1200

  In diesem Aggregat werden Festlegungen zum Linienverlauf nachgewiesen. Die Art der hier gespeicherten Werte hängt von der im Standardaggregat ULO10ALK gespeicherten Art der Geometrie ab.

- Standardaggregat ULOB2000

  Dieses Aggregat ordnet dem Bezugspunkt aus dem Aggregat ULOB0000 die Funktion eines Objektes zu. In den Datenelementen DLOB2101 und DLOB2002 sind Folie und Art des Objektes gespeichert. In dem Datenelement DLOB2004 ist der Objekttyp verzeichnet. Er kann folgende Werte annehmen:

  P punktförmiges Elementarobjekt

  L linienförmiges Elementarobjekt

F flächenförmiges Elementarobjekt

R Rahmenobjekt

Die Objektnummer ist im Datenelement DLOB2005 gespeichert. Über diese Nummer können die zum Objekt gehörigen Linien zugeordnet werden.

- Standardaggregat ULOB2100

  In diesem Aggregat werden zum Objekt gehörige besondere Informationen verschiedenster Art gespeichert. Die im Datenelement DLOB2101 registrierte Art der besonderen Information beschreibt die folgenden Datenelemente. Die besonderen Informationen können folgende Werte annehmen:

  11-19 Objektname bzw. weiterer Objektname

  21-29 Schriftzusatz

  31-39 Objektausgestaltung

  61-69 Kartenrahmenausgestaltung

  71-79 Informationen, die nicht dargestellt werden

  80 Fachattribut

  99 Geometrie untergegangener Objekte

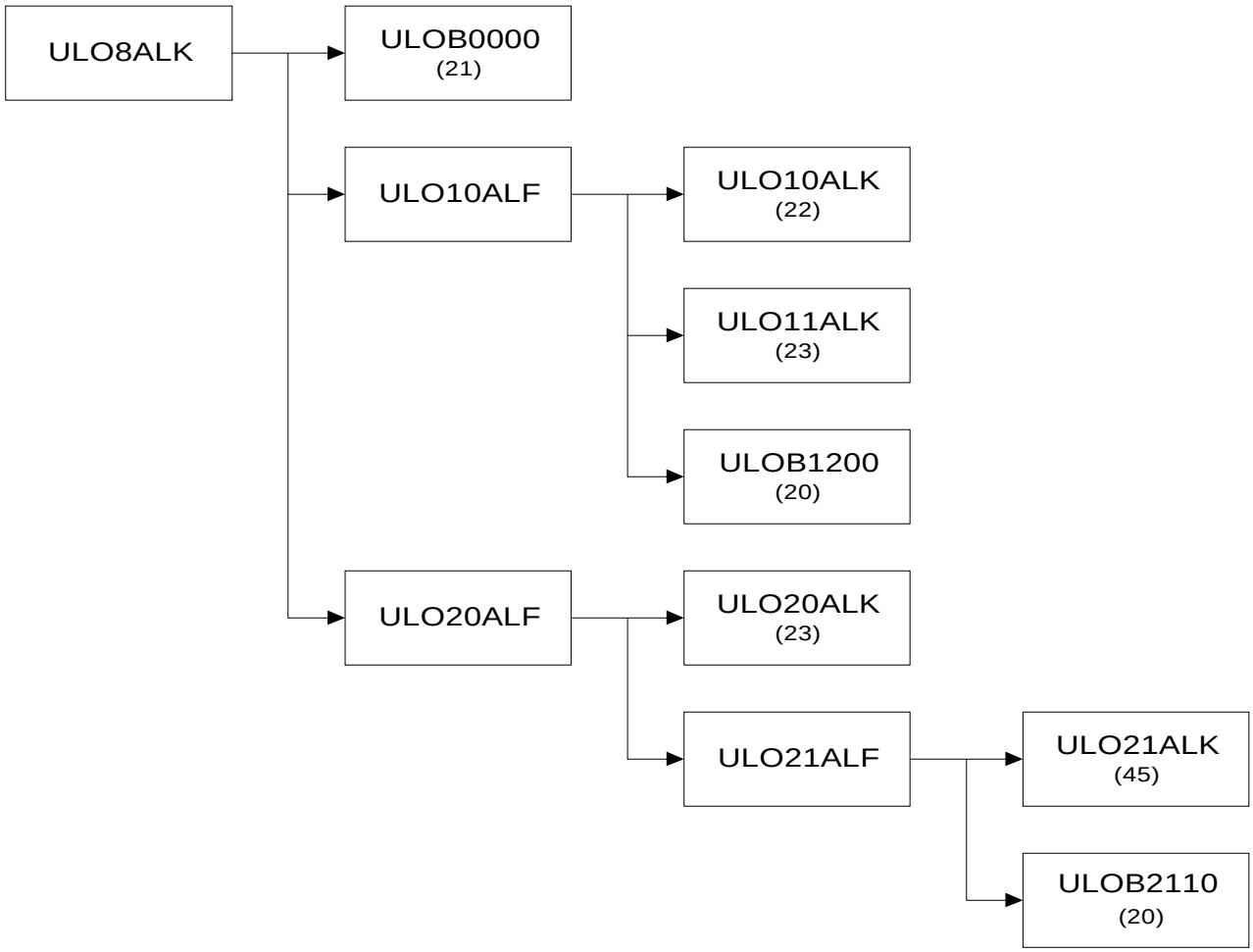
  Die Objektart der besonderen Information wird in dem Datenelement DLOB2103 gespeichert. Es folgt im Datenelement DLOB2104 der Text der Information, also der Objektname oder ein Schriftzusatz. Die in Element DLOB2105 verzeichnete Art der Geometrie entscheidet über die Interpretation der Werte, die in dem eventuell folgenden Aggregat ULOB2110 zu finden sind.

- Standardaggregat ULOB2110

  In diesem Aggregat ist die Geometrie der besonderen Information zum Objekt gespeichert. Bei der Auswertung der Geometrieangabe ist die Art der Geometrie aus Datenelement DLOB2105 entscheidend.

Die Aggregate ULO10ALK, ULO11ALK und ULOB1200 werden nochmals zu einem komplexen Aggregat ULO10ALF zusammengefaßt. Die Aggregate ULO21ALK und ULOB2110 werden ebenfalls zu einem komplexen Aggregat ULO21ALF zusammengeführt, welches mit dem Aggregat ULO20ALK zum Aggregat ULO20ALF vereinigt wird. Diese komplexen Aggregate werden ebenfalls über Wiederholungsfaktoren vervielfacht.

Für das Einlesemodul wird folgender Funktionsaufbau geplant:

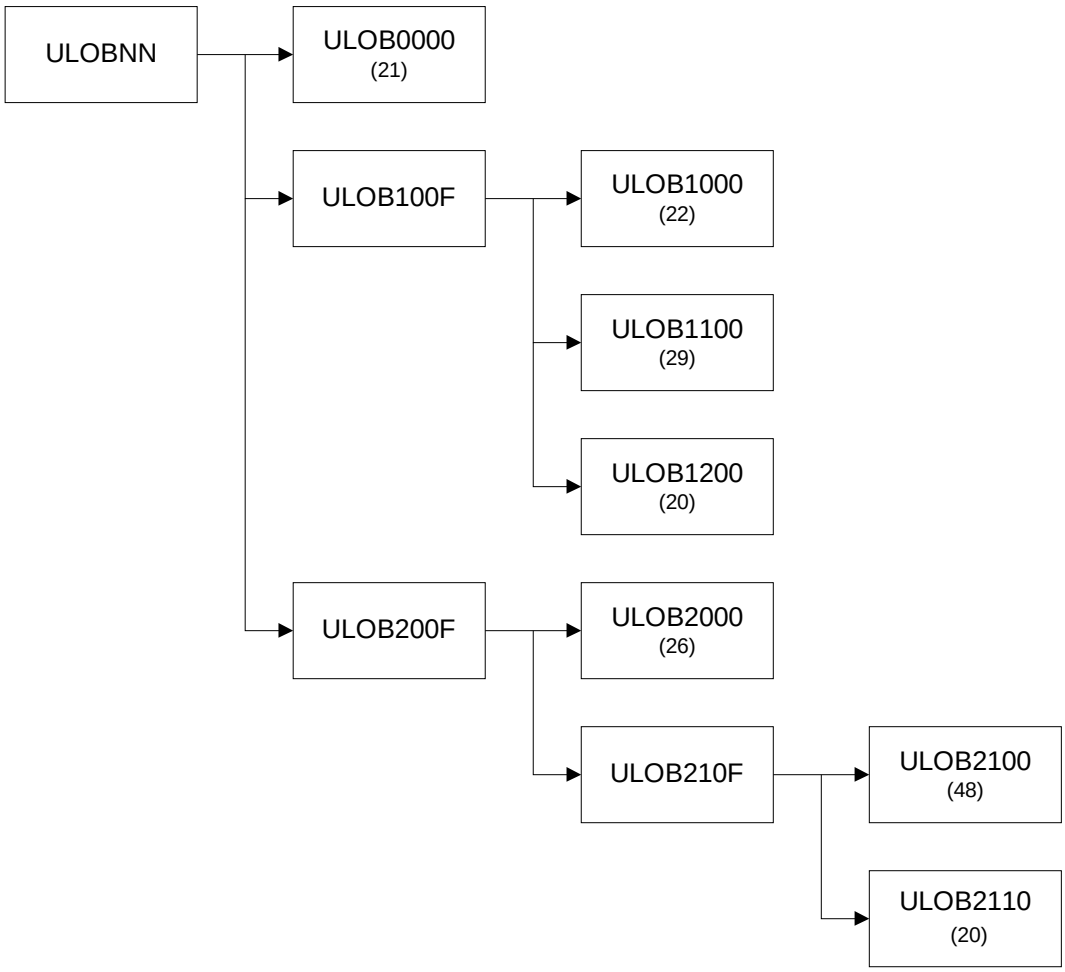


(21) Länge des Datenelements in Byte

3.2.3 Aufbau des Datenaggregates ULOBNN

Das Datenaggregat ULOBNN unterscheidet sich nicht wesentlich vom Aggregat ULO8ALK. Einige ULOBNN-Aggregate enthalten zusätzliche Datenelemente, die aber für die Übernahme in das zu erstellende Programm nicht relevant sind, da sie nur bei der Übertragung von und zum ATKIS belegt sind.

Trotz der geringen Unterschiede sollen eigene ULOBNN-Lesefunktionen geschrieben werden, um den Programmablauf durchsichtig zu gestalten.



(21) Länge des Datenelements in Bytes

3.2.4 Zusammenstellung der Datenelemente

Name des DE	Bezeichnung des DE	Länge (Byte)	Belegung	Zulässige Zeichen
ULOB0000				
DLOB0001	Numerierungsbezirk	8	m	0-9
DLOB0002	Koordinate im NB	12	m	0-9
DLOB0003	Prüfzeichen	1	m	0-9
		21		
ULOB1000 ULO10ALK				
DLOB1001	Numerierungsbezirk	8	m	0-9
DLOB1002	Koordinate im NB	12	m	0-9
DLOB1003	Art der Geometrie	2	m	0-9,A,B,C,D,I
		22		

ULOB1100 ULO11ALK				
DLOB1101	Folie	3	m	0-9
DLOB1102	Objektart	4	m	0-9,A,L,N,R,Z
DLOB1103	Objektnummer (rechts)	7	k	0-9, A-Z, Space
DLOB1104	Objektnummer (links)	7	k	0-9, A-Z, Space
DLOB1105 [1)]	Objektteilnummer 1(Rechts)	3	k	0-9, A-Z, Space
DLOB1106 [1)]	Objektteilnummer 2 (Links)	3	k	0-9, A-Z, Space
DLOB1107	Linienteilung (Rechts)	1	b	0-3
DLOB1108	Linienteilung (Links)	1	b	0-3
		29		
ULOB1200				
DLOB1201	Lageparameter	20	m	0-9, Space
		20		
ULOB2000 ULO20ALK				
DLOB2001	Folie	3	m	0-9
DLOB2002	Objektart	4	m	0-9
DLOB2003	Aktualität des Objekts	2	m	0-9
DLOB2004	Objekttyp	1	m	F,K,L,P,R,V,1-8
DLOB2005	Objektnummer	7	b	0-9, A-Z, Space
DLOB2006 [1)]	Modelltyp	2	b	1-9,K,M
DLOB2007	Entstehungsdatum	6	b	0-9, Space
DLOB2008 [1)]	Veränderungskennung	1	k	A,B,N,V,L
		26		
ULOB2100 ULO21ALK				
DLOB2101	Art der besonderen Information	2	m	0-9
DLOB2102	Kartentyp	2	b	0-9,A-Z
DLOB2103	Objektart bzw. Untergangsdatum	6	b	0-9,A,L,N,R,Z Space
DLOB2104	Text der Information	33	b	sämtl. Zeichen
DLOB2105	Art der Geometrie	2	b	0-9
DLOB2106 [1)]	Objektteilnummer	3	k	0-9,A-Z, Space
		48		
ULOB2110				
DLOB2111	Geometrieangabe	20	m	0-9,T, Space
		20		

1) Für die ALK nicht relevant. Die Datenelemente kommen in den ULO__ALK- Sätzen nicht vor. In den ULOB____-Sätzen werden sie mit Leerzeichen belegt

3.3 Interne Datenstruktur

Zwei Kriterien sind für den Aufbau der internen Datenstruktur entscheidend:

- Die Datenmengen sollten so gering wie möglich gehalten werden, da sie über das Netz übertragen werden sollen.
- Der Rechenaufwand zur Darstellung der Informationen sollte nicht allzu hoch sein, da das den Umfang der Programmdateien anwachsen lassen würde. Auch diese müssen beim Datenabruf übertragen werden.

Es steht also außer Frage, daß die Datenmengen, die beim Einlesen der EDBS-Dateien entstehen, reduziert werden müssen. Da das zu erstellende Programmsystem eher informativen Charakter besitzen soll, kann eine Reduzierung der Teilmenge, die der Information dient, nur durch sinnvolle Klassifizierung erreicht werden. Mehr Abstriche können bei der Ausgestaltung der Informationen gemacht werden, da von einer Karte, die über das Internet übertragen wird, sicher nicht die Qualität eines Herausgabeoriginals erwartet wird.

3.3.1 Koordinaten

Ein Großteil der erforderlichen Reduktion der Daten kann schon durch die Wahl eines geeigneten Koordinatensystems erreicht werden. Die Wahl viel auf ein ganzzahliges Bildkoordinatensystem, welches in einem Zahlenbereich von 0 bis +65535 definiert ist. Dieser Zahlenbereich kann mit 2 Byte dargestellt werden. Der Koordinatenbereich $\max(x_{max} - x_{min}, y_{max} - y_{min})$ der Quelldatei wird mit einer 3-Parametertransformation in dieses Bildsystem transformiert. Die Genauigkeit der erzeugten Koordinaten ist damit abhängig von der maximalen Ausdehnung der Koordinaten im Quellsystem.

$$\sigma_x=(x_{max} - x_{min}) / k_{max}$$

$$\sigma_y=(y_{max} - y_{min}) / k_{max}$$

$$\text{mit } k_{max} = 65535$$

Damit liegt die Lagegenauigkeit eines Punktes aus einem Koordinatenbereich mit 5 km Ausdehnung bei etwa 10cm, diese sollte für die Bildschirmdarstellung, und mehr will das Programmsystem nicht leisten, ausreichen.

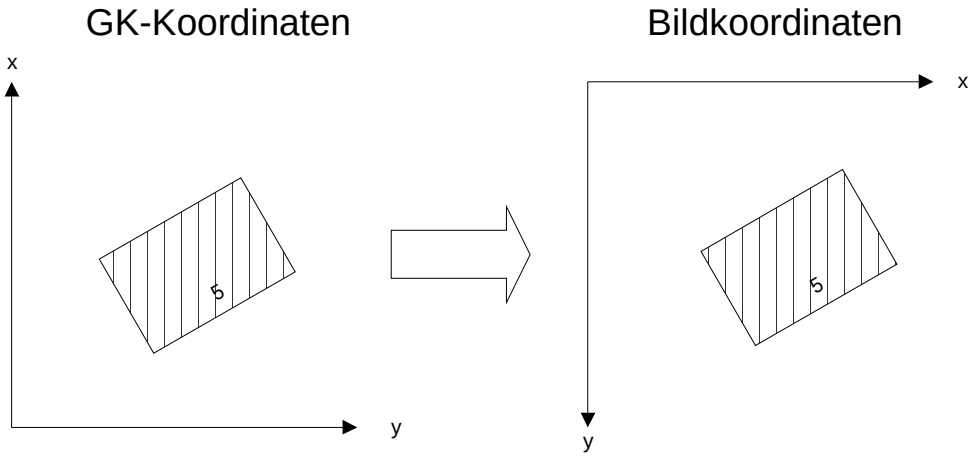
Die Transformationsformeln lauten:

$$X = m \cdot (y - y_{min})$$

$$Y = m \cdot (-x - x_{min})$$

$$\text{mit } m = k_{max} / \max(x_{max} - x_{min}, y_{max} - y_{min})$$

An dieser Stelle wird auch die Umnummerierung der Koordinatenachsen vorgenommen.



3.3.2 Datenmodell

3.3.2.1 Grundelemente

Die Wahl viel auf eine relationales Datenmodell, es ist einfach zu verwirklichen und leicht in Dateiform abzubilden. Zur Abbildung der Grundelemente werden drei Relationen gewählt:

- Punkte

x (int) x-Koordinate	y (int) y-Koordinate	w (short) Symbolwinkel	eb (byte) Ebene	art (byte) Art (Symbol)
10223	20158	0	1	10
10586	25348	10	1	11
...	...	...	...	...

Die Werte in [eb] und [art] sind Zeiger auf die entsprechenden Relationen.

- Linien

pa (int) Anfangspunkt	pe (int) Endpunkt	h (int) Pfeilhöhe bzw.Spline	eb (byte) Ebene	art (byte) Art	typ (byte) Linientyp
10	11	0	1	5	0

25	39	5023	8	2	1
38	40	7	8	3	2
...	...	...	...	...	...

Die Werte in [pa] und [pe] sind Zeiger auf die Punktrelation. Der Wert in [h] hat abhängig vom Linientyp unterschiedliche Bedeutungen.

[typ] = 0	Gerade	[h] = 0
[typ] = 1	Bogen	[h] = Pfeilhöhe
[typ] = 2	Kreis	[h] = Pfeilhöhe = Radius
[typ] = 3	Spline	[h] = Zeiger auf Splineliste

Die Werte in [eb] und [art] sind Zeiger auf die entsprechenden Relationen.

Es wurde die Pfeilhöhe als Definition eines Bogen gewählt, weil dieser Wert kontrollierbarer ist. Bei Speicherung des Radius oder des Kreismittelpunktes könnten die Werte leicht den maximalen Koordinatenbereich k_{max} verlassen.

- Texte

x (int) x-Koordinate	y (int) y-Koordinate	w (short) Textwinkel	eb (byte) Ebene	art (byte) Art	text (String) Text
25356	35968	0	1	3	Weg
26234	45789	10	2	7	Whs.
…	…	…	…	…	…

Die Werte in [eb] und [art] sind Zeiger auf die entsprechenden Relationen.

Für die Speicherung der Objekte wird ebenfalls eine Relation gebildet:

- Objekte

x (int) x-Koordinate	y (int) y-Koordinate	tw (short) Textwinkel	eb (byte) Ebene	art (byte) Art
25356	35968	98	1	3
26234	45789	10	2	7
…	…		…	…

typ (byte) Objekttyp	oska (short) OSKA	name (String) Objektname	idx (int) Linien
1	233	125/1	125
2	1211	25	678
…	…	…	…

In den Felder [x] und [y] wird der Referenzpunkt des Objektes gespeichert. Der Wert in [w] gibt den Textwinkel an, mit der der Objektname dargestellt wird. Die Werte in [eb] und [art] sind Zeiger auf die entsprechenden Relationen. Der Objekttyp gibt an, ob das Objekt linien- oder flächenförmig abgebildet werden soll. Im Feld [oska] wird der Objektschlüssel abgelegt. Der Wert in [idx] ist ein Zeiger auf die Objektlinienliste. Diese Relationen sollen bis zu 65535 (2 Byte) Elemente aufnehmen können.

3.3.2.2 Relationen zur Ordnung und Darstellung der Grundelemente

Um die Grundelemente ordnen und um ihre Darstellung und graphische Ausprägung beeinflussen zu können, werden fünf weitere Tabellen angelegt.

- Ebenen

color (byte) Farbe	name (String) Name der Ebene	isVisible (boolean) Sichtbarkeit	isUsed (boolean) ist belegt
1	Flurstücke	true	true
5	Gebäude	false	true
…	…	…	…

Durch die Einordnung in Ebenen sollen die Grafikelemente in eine gewisse Grundordnung gezwungen werden. Über die Ebene wird gesteuert, ob die auf ihr liegenden Grundelemente dargestellt werden und mit welcher Farbe. Die Zuordnung wird durch das Feld [eb] realisiert, welche jede Relation der Grundelemente besitzt.

- Punktarten

pae (Objekt[ ]) Punktartelement	width (int) Größe der Punktart	isUsed (boolean) ist belegt
Array aus Punktartelementen	10	true
Array aus Punktartelementen	15	true
…	…	…

Über das Feld [art] in der Punkte-Relation wird dem Punkt eine Punktart zugeordnet. Im Feld [pae] wird ein Array aus Punktartelementen gespeichert, dieser wird in 0 näher beschrieben. Die Größe der Punktart wird in den Größeneinheiten des Koordinatensystems angegeben, sie wird bei der Erstellung des Koordinatensystems mit erzeugt.

- Linienarten

lf (byte[]) Linienform	width (int) Größe der Linienart	isUsed (boolean) ist belegt
1	10	true
1,2,2,1	15	true
…	…	…

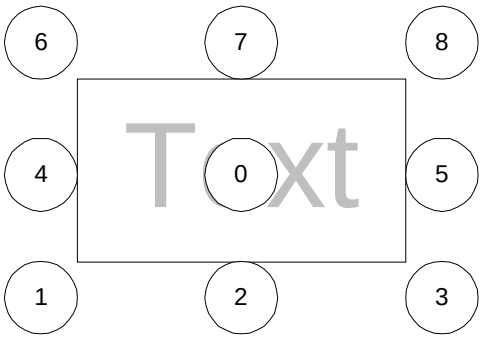
Die Linienart wird der Linie über den Wert [art] in der Linien-Tabelle zugeordnet. Über die Folge der Byte-Werte im [lf]-Array kann die Darstellungsform der Linie gesteuert werden. Die Werte mit geradem Index zeichnen ein Linienstück, die ungeraden Indexwerte lassen eine Lücke. Eine Strich-Punkt-Linie hätte z.B. den Array: {5,1,0,1}.

Der Wert im Feld [width] gibt die Länge des Wertes 10 im [lf]-Array in Größeneinheiten des Koordinatensystems an.

- Textarten

height (int) Textgröße	n (byte) Textneigung	b (byte) Zeichenabstand	z (byte) Textzentrum	isUsed (boolean) ist belegt
35	0	0	1	true
25	3	1	0	true
…	…	…	…	…

Die Textgröße wird wiederum in Einheiten des Koordinatensystems angegeben. Die Belegung der Felder [n] und [b] wird unter 0 näher beschrieben. Der Wert des Textzentrums, mit dem die Textkoordinate berechnet wird ist wie folgt definiert:



- Objektarten

color (int) Farbe	la (byte) Linienart	ta (byte) Textart	name (String) Objektname	isAgVisible (boolean)
1	1	2	Flurstück	true
5	1	5	Wohngebäude	false
…	…	…	…	…

isFlVisible (boolean)	isTxVisible (boolean)	isTxAbsolut (boolean)	isUsed (boolean)
false	true	true	true
true	true	false	true
…	…	…	…

Mit der Farbe [color] wird die Fläche des Objektes gefüllt. Die Werte in den Feldern [la] und [ta] sind Zeiger auf die Linien- und Textartenrelation. Mit der Linienart [la] wird der Objektumring gezeichnet und mit der Textart [ta] der Objektname dargestellt. Die Boolean-Schalter haben folgende Bedeutung:

isAgVisible Ausgestaltungsgeometrie wird dargestellt

isFlVisible Objektfläche wird dargestellt

isTxVisible Objektname wird dargestellt

isTxAbsolut Text wird blattrandorientiert oder
mit dem Textwinkel aus der Objektrelation dargestellt

Der Schalter [isUsed] gibt in allen Tabellen an, ob die Art von einem Grundelement belegt ist.

Die Ebenen- bzw. Artenrelationen sollen eine feste Anzahl von jeweils 100 Elemente aufnehmen können.

3.3.2.3 Relationen zur Darstellung der Arten

Um den Punkten eine Symbolik zuzuordnen, wurde die Relation Punktart erzeugt. Diese enthält ein Array aus Punktartelementen. In diesen soll die Geometrie der Punktart gespeichert werden.

- Punktartelement

nr (byte) Geometrie	e1 (byte)	e2 (byte)	e3 (byte)	fm (byte) Füllmodus
1	55	56	0	0
2	55	13	3	1
3	91	99	15	0
4	11	99	0	1

Das Feld [nr] gibt die Art der Geometrie des Punktartelements an. In den Feldern e1-e3 sind Koordinaten bzw. Maße gespeichert. Ist das Feld [fm] mit einer Zahl größer Null belegt, wird die Geometrie ausgefüllt.

Folgende Belegungen sind vorgesehen:

Gerade

nr = 1

e1 = Koordinate Anfangspunkt

e2 = Koordinate Endpunkt

e3 = 0

fm = 0

Kreisbogen

nr = 2

e1 = Koordinate Mittelpunkt

e2 = Winkel des Bogens, ist der Wert mit 0 belegt wird ein Vollkreis gezeichnet

e3 = Radius

fm = 0 oder 1

Dreieck

nr = 3

e1 = Koordinate 1. Punkt

e2 = Koordinate 2. Punkt

e3 = Koordinate 3. Punkt

fm = 0 oder 1

Rechteck

nr = 4

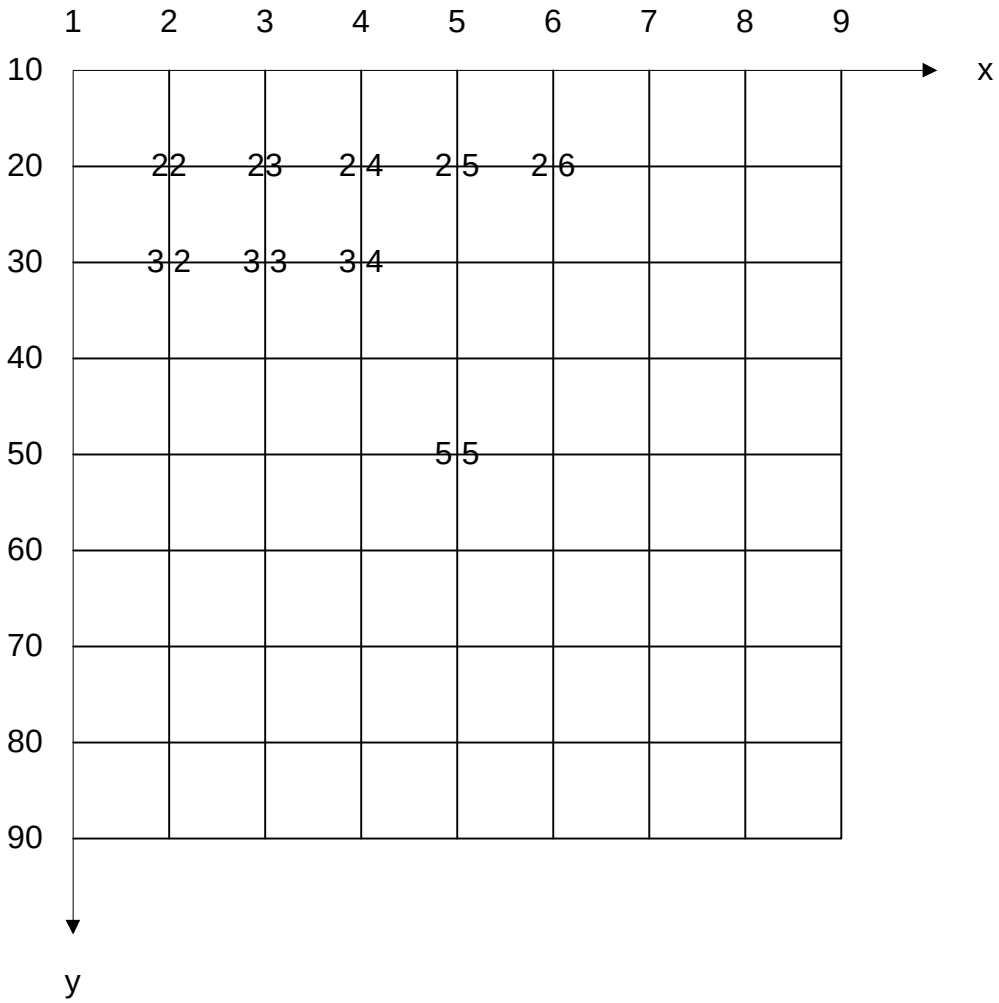
e1 = Koordinate linksoben

e2 = Koordinate rechtsunten

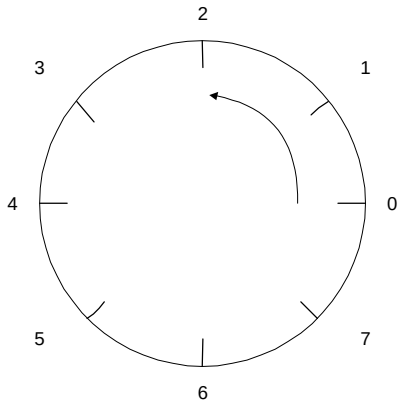
e3 = 0

fm = 0 oder 1

Die Koordinaten werden im folgenden Koordinatensystem erzeugt:


1 2 3 4 5 6 7 8 9
10
20
30
40
50
60
70
80
90
x
y
22 23 24 25 26
32 33 34
55

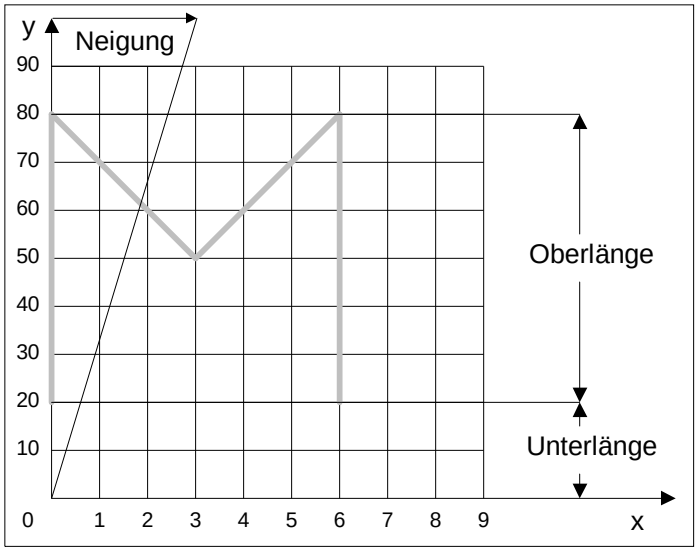
Die Koordinate, die in den Feldern e1-e3 gespeichert wird, ergibt sich aus der Summe der x- und y-Komponente. Die Punktartelemente sollen mit dem Mittelpunkt (55) des Systems und dem Winkel, der dem Feld [w] aus der Punktrelation zu entnehmen ist, über die Koordinate des jeweiligen Punktes transformiert werden.
Der Wert für den laufenden Winkel im Punktartelement Kreis ergibt sich aus folgendem Prinzip:



Der Wert e2 ergibt sich aus: e2 = (Anfangsrichtung · 10) + Endrichtung
Nach diesem Prinzip können zwar nur Winkel mit den Werten n · 45° gespeichert werden, das sollte aber für die Erzeugung von Symbolen ausreichen.

- Zeichen

Ein gesondertes Problem ist die Darstellung von Zeichenketten. Die Grafikfunktionen aus der Klasse java.awt.Graphics erlauben nur eine horizontale Schriftausrichtung. Da aber der Verzicht auf objektorientierte Texte eine zu große Qualitätseinbuße der Grafik bedeuten würde, soll ein eigener Zeichensatz erzeugt werden. Dazu ist wiederum die Definition eines Koordinatensystems notwendig, in dem die Zeichen dargestellt werden.



Das Zeichen wird mit einem Array aus Polylines gespeichert. Im Fall des Buchstabens “M” wird eine Polyline {20,80,53,86,26} im Array abgelegt. Die Koordinaten ergeben sich wieder wie bei der Definition des Punktartelementes aus der Summe der x- und y-Komponeneten. Zusätzlich wird pro Zeichen ein Wert [width] angelegt. Dieser bestimmt die Breite des Zeichens. Dabei wird der äußere rechte Rand des Zeichens plus Eins angegeben. Somit kann eine Proportionalschrift erzeugt werden. Im Fall des Zeichens “M” würde der Wert “7” angegeben werden. Bei der Zeichendarstellung sollen auch die Felder [n]=Neigung und [b] = Zeichenabstand aus der Textartenrelation verarbeitet werden. Als Neigung wird der aus der Grafik ersichtliche Abstand zur y-Achse angegeben. Der Zeichenabstand [b] soll zur Zeichenbreite [width] addiert werden.

Das Zeichenkoordinatensystem wird mit dem Einfügepunkt {20} und der jeweiligen Textrichtung in die Grafik transformiert.

- Zeichensatz

Im Zeichensatz werden die 256 ASCII-Zeichen als Zeichen-Array angelegt. Zusätzlich sollen folgende Methoden analog zur Klasse java.awt.FontMetrics bereitgestellt werden:

getAscent()	gibt die Oberlänge an
getDescent()	gibt die Unterlänge an
getLeading()	gibt den Zeilenabstand an
getHeight()	gibt die Texthöhe an

Zusätzlich soll eine Methode getStringWidth(String s) angeboten werden. Diese Methoden sind zur Berechnung der Position des Strings notwendig.

- Farben

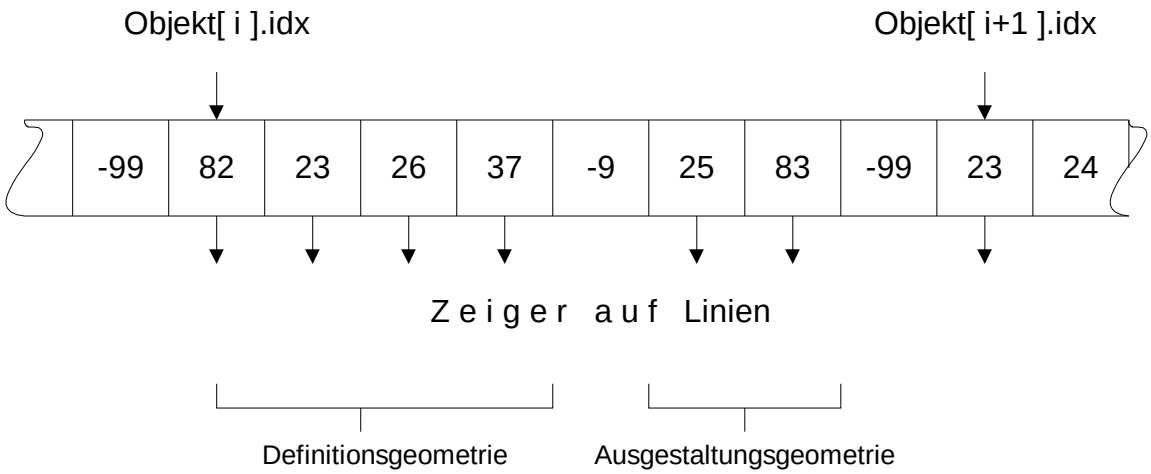
Das Java-Farbmodell basiert auf dem RGB-Farbmodell. Dieses stellt Farben mit 24 Bit Tiefe dar.

Den Ebenen und Objektarten sollen Farben zugeordnet werden, die auch vom Benutzer geändert werden können. Es soll eine Auswahl von 100 gut unterscheidbaren Farbtönen zur Verfügung gestellt werden. Dazu wird ein Array mit java.awt.Color- Objekten angelegt.

3.3.2.4 Listen zur Verknüpfung der Grundelemente

- Objektlinienliste

Um den Objekten ihre Definitions- und Ausgestaltungsgeometrie zuordnen zu können wird eine Zeigerliste angelegt. Diese Liste enthält Zeiger auf die Linienrelation. Sie soll folgende Struktur besitzen:



Der Wert [–9] kennzeichnet das Ende der Definitionsgeometrie, der Wert [-99] das Ende der zum Objekt gehörigen Linien.

- Splinepunktliste

  Die Punkte, die den Verlauf der Splines definieren, werden in einer gesonderten Liste gespeichert. Sie werden als java.awt.Point angelegt, da sie keine weiteren Darstellungsattribute benötigen. Die Liste ist ähnlich angelegt wie die Objektlinienliste. Die Zeiger Linie[i].h zeigen auf den ersten Splinepunkt, hinter dem letzten Punkt des Splines wird ein Punkt mit den Koordinaten (<0, <0) gespeichert.

3.3.3 Datenorganisation

Alle unter 3.3.2 angelegten Relationen und Listen werden in einer gemeinsamen Klasse angelegt. Zusätzlich werden dort sämtliche Darstellungsoptionen und Datenbankattribute abgelegt, damit beim Ein- bzw. Auslesen von Daten diese Werte abgerufen oder geändert werden können.

Weiterhin soll die Datenbank-Klasse Methoden zur Verfügung stellen, mit der die Daten abgerufen oder geändert werden können.

3.4 Datenübertragung mit IVF

Wie schon unter 3.1, besprochen muß ein kompaktes Dateiformat erzeugt werden, welches eine schnelle Übertragung der Grafikdaten über das Internet gewährleistet. Dieses Dateiformat soll IVF (Internet-Vektor-Format) genannt werden.

Konzeptionell soll die Dateistruktur an das AutoCad-DXF- Format angelehnt werden [RUDOLPH 1993]. Dieses Format überträgt im Gegensatz zum EDBS- oder anderen geläufigen Formaten fast vollständig eine Zeichnung mit sämtlichen Symbolen und Ausgestaltungen. Dies hat den Vorteil, daß Konverter ohne Symbol- oder Artenzuordnungstabellen, die oft mit großem Aufwand laufendgehalten werden müssen, Daten in andere Formate wandeln können. Der Datenbestand soll auf einfache Art und Weise von anderen Programmen interpretiert werden können.

Das IVF-Format soll also folgende Daten aufnehmen:

- die eigentlichen Daten
- die Tabellen zur Datenorganisation
- Programmeinstellungen und Darstellungsattribute

Die Farbtabelle und die Zeichentabelle sollen nicht gespeichert werden, ihre Darstellung hängt also von dem Zielsystem ab.

Damit eine innere Ablaufkontrolle beim Einlesen der Daten vorgenommen werden kann, wird folgende Struktur geplant:

```
IVFKENNUNG
BLOCK HEADER {
      LIST HEADER1 {
            KENNUNG1   Einstellung bzw. Attribut
            …
      } ENDLIST
      LIST HEADER2
            KENNUNG1   Einstellung bzw. Attribut
            …
      } ENDLIST
} ENDBLOCK
BLOCK TABELLEN {
      LIST EBENEN {
            Ebenen
            …
      } ENDLIST
      LIST PUNKTARTEN
            Punktarten
            …
      } ENDLIST
      LIST LINIENARTEN
            Linienarten
```

```
            …
      } ENDLIST
      LIST TEXTARTEN
            Textarten
            …
      } ENDLIST
      LIST OBJEKTARTEN
            Objektarten
            …
      } ENDLIST
} ENDBLOCK
BLOCK PUNKTE {
      LIST KOO
            Koordinaten
            …
      } ENDLIST
      LIST SYM {
            Punktattribute
            …
      } ENDLIST
} ENDBLOCK
BLOCK LINIEN {
      LIST LINIEN {
            Linien
            …
      } ENDLIST
} ENDBLOCK
BLOCK TEXTE {
      LIST TEXTE
            Texte
            …
      } ENDLIST
} ENDBLOCK
BLOCK OBJEKTE {
      LIST OBJEKTE
            Objekte
            …
      } ENDLIST
} ENDBLOCK
EOF
```

Die Daten sollen binär gespeichert werden, dabei erhalten die Datensätze feste Längen. So wird z.B. die Struktur eines Punkt-Datensatzes wie folgt geplant:

```
Punkt {
      Punkt.eb        1 Byte          //Ebene
      Punkt.art       1 Byte          //Art
      Punkt.x         2 Byte          //x-Koordinate
      Punkt.y         2 Byte          //y-Koordinate
}
```

Attribute, die nur in seltenen Fällen belegt sind, sollen nicht in jedem Datensatz belegt werden. Da der Wertebereich einiger Attribute nicht ausgeschöpft ist, kann darüber die Belegung der nicht in jedem Fall belegten Attribute gesteuert werden. So sind die Attribute Punkt.eb und Punkt.art nur von 0-99 definiert, mit einem Byte kann aber ein Zahlenbereich von 0-255 dargestellt werden. Ein Punktdatensatz kann also auch wie folgt aussehen:

```
Punkt {
      Punkt.eb         1 Byte          //Ebene
      Punkt.art+100    1 Byte          //Art
      Punkt.x          2 Byte          //x-Koordinate
      Punkt.y          2 Byte          //y-Koordinate
      Punkt.w          2 Byte          //Symbolwinkel
}
```

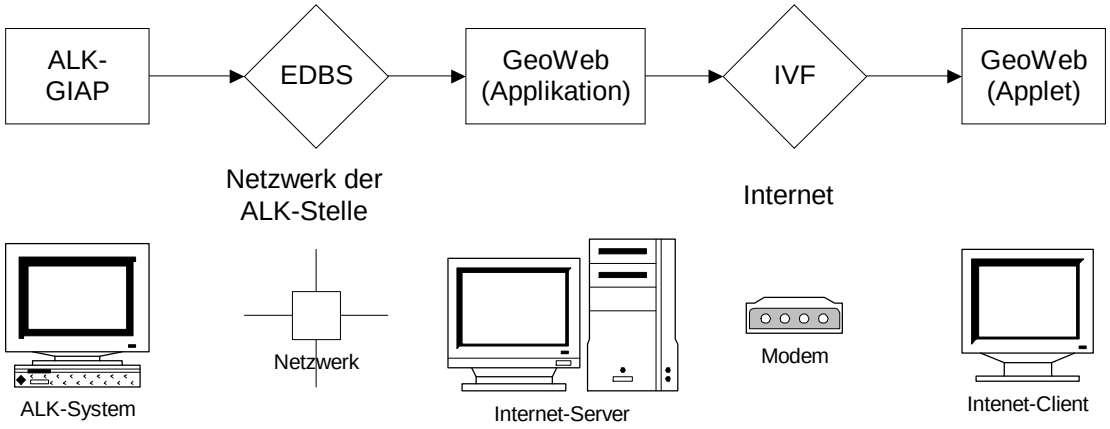
Erkennt das Einleseprogramm für das Attribut Punkt.art einen Wert > 100, muß der Wert um 100 minimiert werden und nach dem Punkt.y-Wert noch eine 2 Byte-Zahl in das Attribut Punkt.w gelesen werden.

Damit das Einleseprogramm die gelesenen Daten ohne Zwischenspeichern in einen Array ablegen kann – die Anzahl der Array-Elemente muß bei der Initialisierung bekannt sein –, wird nach der Listenkennung die Anzahl der folgenden Elemente als 2 bzw. 4-Byte-Zahl angegeben.

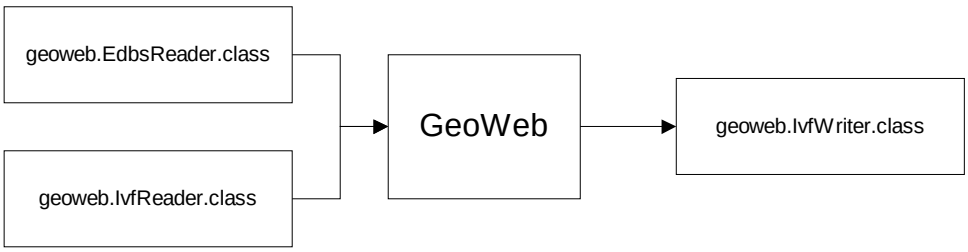
Eine Besonderheit stellt noch das Speichern von Strings dar. Um nicht unnötig viel Leerzeichen in die Datei zu schreiben, wird nach dem letzten Zeichen eine Stringendekennung geschrieben. Diese kann dann vom Ausleseprogramm ausgewertet werden.

3.5 Datenfluß

Für den Datenfluß ergibt sich folgender Ablauf:


ALK-
GIAP
EDBS
GeoWeb
(Applikation)
IVF
GeoWeb
(Applet)
Netzwerk der
ALK-Stelle
Internet
ALK-System
Netzwerk
Internet-Server
Modem
Intenet-Client

Die beiden zu erstellenden Programme sollen GeoWeb genannt werden. Die Applikation (siehe 2.3.4), die auf dem Server installiert wird, und das Applet, welches im Download-Verzeichnis abgelegt wird, sollen weitestgehend auf den selben Klassen basieren. Für den Datenaustausch sind drei Klassen zuständig:



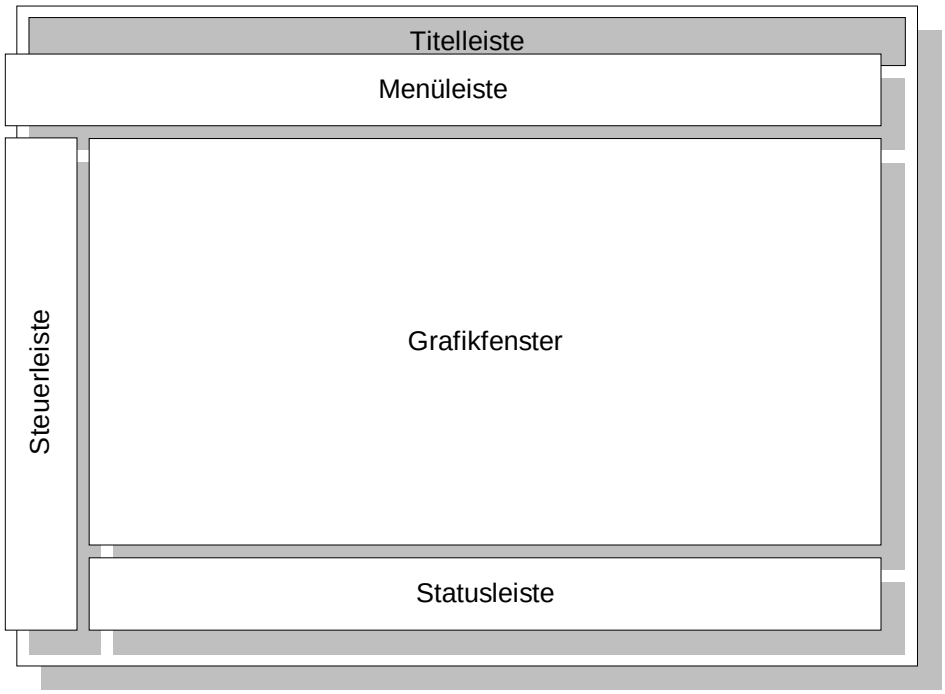
Die Klasse “EdbsReader” soll nur in der Server-Applikation enthalten sein.

4 Die Applikation GeoWeb

Im folgenden Kapitel soll die GeoWeb-Server-Applikation betrachtet werden. Mit ihr werden die EDBS-Daten eingelesen, bearbeitet und als IVF-Datei ausgelesen.

4.1 Benutzerinterface

Um Überraschungen bei der Portierung auf andere Plattformen zu vermeiden, soll die Benutzerschnittstelle so einfach wie möglich gehalten werden. Im wesentlichen besteht sie aus vier Grundelementen:



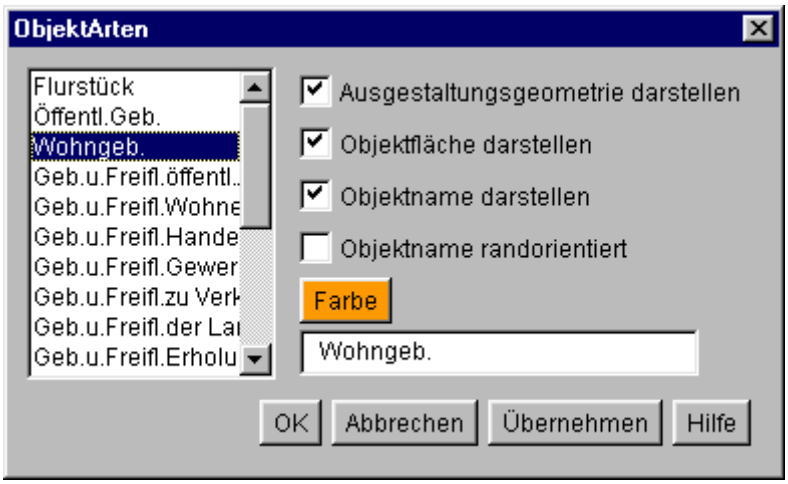
Folgende Klassen repräsentieren diese Grundelemente:

- geoweb.MainMenu.class
  Über die Menüleiste werden sämtlich Klassen geladen, die zuständig sind für:
  Daten- Ein- und Ausgabe
  Änderungen am Datenbestand
  Einstellungen für das Grafikfenster
  Hilfe und Informationen
- geoweb.SymbolPanel.class
  Die Steuerleiste nimmt Buttons auf, mit denen die Antworten auf Mausaktionen im Grafikfenster gesteuert werden.

- geoweb.InfoPanel.class
  Die Statusleiste gibt Informationen über Aktionen oder angewählte Elemente in der Grafik wieder.
- geoweb.GraphicPanel.class
  Im Grafikfenster werden die eingelesenen Daten gezeichnet. Das Fenster soll auf Mausaktionen je nach Einstellung der Buttons in der Steuerleiste reagieren.

Alle Benutzereingaben sollen über Dialogboxen entgegengenommen werden. So sind folgende Dialoge über das Menü ladbar:

- geoweb.FileInfoDialog.class
  Hier können Informationen über den aktuellen Datenbestand abgerufen und eingestellt werden.
- geoweb.ZoomDialog.class
  Diese Dialogbox nimmt die gewünschte Grafik-Zoomstufe entgegen.
- geoweb.EbenenDialog.class
  Einstellungen der Ebenen können in diesem Dialog vorgenommen werden.
- geoweb.ObjektArtenDialog.class
  Hier kann die Darstellung der Objekte gesteuert werden.



- geoweb.OptionenDialog.class
  Einstellungsmöglichkeiten, die die Grafikdarstellung beeinflussen, werden in diesem Dialogfenster angeboten.
- geoweb.HilfeDialog.class
  Dieser Dialog gibt Hilfestellungen zu einzelnen Programmpunkten.

- geoweb.InfoDialog.class
  Diese Dialogbox gibt eine kurze Versionsinformation wieder. Sie wird aber auch benutzt um Fehlermeldungen anzuzeigen, die der Benutzer mit einem “OK” beantworten soll.

4.2 Datenschnittstellen

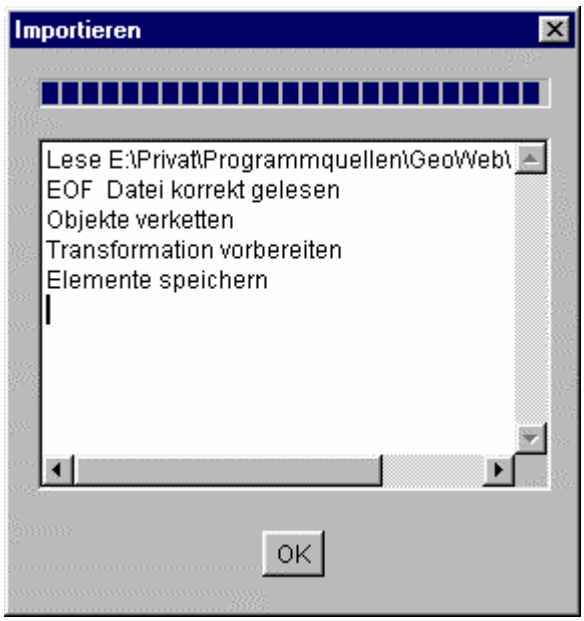
Die Datei- Ein- bzw. Ausgabe wird im wesentlichen durch drei Klassen realisiert, die Klasse geoweb.EdbsReader.class ist zuständig für den EDBS-Import und die Klassen geoweb.IvfReader.class und geoweb.IvfWriter.class für das Lesen und Schreiben von IVF-Dateien.
Zusätzlich wird noch eine Klasse geoweb.IvfTextWriter.class angeboten, die die Grafikdaten in einem lesbaren Textformat ausgibt.

4.2.1 EDBS-Import

Der Import von EDBS-Daten läuft im wesentlichen in vier Schritten ab:

- Daten lesen
- Objekte verketten
- Transformation vorbereiten
- Transformieren und Speichern

Die Schritte werden in der Dialogbox dokumentiert.



4.2.1.1 Daten lesen

Für die innere Ablaufkontrolle und die korrekte Verarbeitung und Zuordnung der Elemente müssen mehr als letztendlich zum Aufbau der GeoWerb-Datenbank benotigte Elementattribute gelesen und zwischengespeichert werden. Dazu wurden weitere Klassen erzeugt:

- geoweb.EdbsPunkt.class
  In Instanzen dieser Klasse werden die Koordinaten der Linien und punktförmige Objekte gespeichert.
- geoweb.EdbsLinie.class
  Objekte der Klasse EdbsLinie nehmen Linien und Funktionen von Linien auf.
- geoweb.EdbsText.class
  Objektbeschriftungen (Art der Geometrie 20-29) werden vom Objekt abgetrennt in Instanzen dieser Klasse abgelegt.
- geoweb.EdbsObjekt.class
  Flächen- und linienförmige Objekte werden in Instanzen dieser Klasse gespeichert.

Die Instanzen dieser Klassen werden in Vector-Objekten, die Java-Repräsentation einer linearen Liste, abgelegt. Die Größe der Vector-Objekte kann zur Laufzeit geändert werden, sie eignen sich damit besonders zur Speicherung von Objekten, deren Anzahl unbekannt ist. Folgende Vector-Objekte wurden angelegt:

- Vector P, L, T, O zur Speicherung der oben genannten Objekte
- Vector OB zur Speicherung von Linien, die zur Ausgestaltungsgeometrie vom Objekten gehören
- Vector S zur Speicherung von java.awt.Point-Objekten, die der Definition eines Splines dienen.

Der Lesevorgang beginnt, indem jeweils ein EDBS-Datensatz in ein String-Objekt umgewandelt wird. Daraufhin wird eine Methode zur Prüfung der EDBS-Kennung, der Operation und des Aggregates gestartet. Das Ergebnis der Prüfung entscheidet über den weiteren Programmablauf, entweder wird der Datensatz abgelehnt, oder die dem Aggregat entsprechende Methode gestartet. Diese ruft weitere read-Methoden auf bis sämtliche anstehenden Aggregate gelesen sind.

Da erst zur Laufzeit bekannt wird, an welcher Spaltenposition welches Aggregat zu finden ist, wird eine Variable “Spalte“ mitgeführt. Die read-Methoden entnehmen aus ihrem Wert die Startposition des Aggregates, für dessen Lesen sie zuständig sind. Damit können die Datenelemente adressiert werden. Nach dem abgeschlossenen Lesevorgang addiert die jeweilige Methode die Länge ihres Aggregates auf “Spalte“ auf.

Der Lesevorgang wird beendet, nachdem der Operationsschlüssel “AEND” gelesen wurde. Die Daten stehen zur weiteren Verarbeitung zur Verfügung.

4.2.1.2 Objekte verketten

Über die Datenelemente DLOB1103 und DLOB1104 (siehe 3.2.4) können die Funktionen der Linien den Objekten zugeordnet werden. Aus dem Anhang 3 des OBAKs NRW [OBAK-LIEGKAT NRW] ergibt sich für die Zusammenstellung flächenhafte Objekte folgender Programmablauf:

```
buildObjekt() {
      Linien = suche alle Linien, die zum Objekt gehören;
      PolygonAnfang = suche Punkt(Linien), von dem nur zwei
      Linien abgehen
      suche FolgeLinie(Linien), bis PolygonAnfang wieder
erreicht
}
```

Dabei sind die Prioritäten, die beim Vorfinden von mehr als zwei Linien pro Verknüpfungspunkt zu beachten sind, in den Programmteil eingearbeitet.

Das gefundene Polygon wird als Liste von Zeigern in einem Vector OL abgespeichert.

Diesem wird dann, wenn noch weitere Geometrieelemente vorhanden sind, die Polygonendekennung [-9] angefügt. Weiterhin werden dem Vector OL Zeiger auf eventuell vorhandenen Linien der Ausgestaltungsgeometrie und schließlich die Objektendekennung [-99] hinzugefügt.

Bei linienförmigen Objekten werden die Linienzeiger unsortiert in den Vector OL geschrieben. Dies erfolgt ebenfalls, wenn der buildObjekt()-Algoritmus nicht zum gewünschtem Ergebnis führt.

4.2.1.3 Transformation vorbereiten

Zur Bestimmung der unter 3.3.1 erläuterten Transformationsparameter müssen von sämtlichen eingelesenen Koordinaten die Minima und Maxima gesucht werden. Dieser Untersuchung werden die EdbsPunkt-, EdbsText-, EdbsObjekt- und Spline-Vector-Objekte unterzogen.

4.2.1.4 Transformieren und speichern

In diesem Schritt wird die Umwandlung von EDBS-Objekten in die GeoWeb-Objektklassen vorgenommen. Dazu finden folgende Abbildungen statt:

geoweb.EdbsPunkt.class > geoweb.Punkt.class

geoweb.EdbsLinie.class > geoweb.Linie.class

geoweb.EdbsText.class > geoweb.Text.class

geoweb.EdbsObjekt.class > geoweb.Objekt.class

Zur Transformation der Koordinaten werden die Formeln aus 3.3.1 herangezogen.

Aufwendig gestaltet sich die Umwandlung der Folien und Objektschlüssel in die GeoWeb eigenen Ebenen und Arten. Die 1000 mögliche Folien und 10 000 Objektschlüssel müssen auf einen Bereich von jeweils 100 Ebenen und Objektarten abgebildet werden. Dazu sind Klassifizierungstabellen aufgestellt worden die folgende Form besitzen:

von Folie	bis Folie	von OSKA	bis OSKA	Objektart	Bedeutung
11	19	1201	1399	1	Wohngebäude
11	19	1401	1699	2	Geb. Handel und Dienstl.
21	29	1200	1399	3	Geb. und Freifläche Wohnen
…	…	…	…	…	…

Solche Tabellen sind für Ebenen, Punktarten, Linienarten, Textarten und Objektarten aufgestellt worden.

Es wurde versucht diese Klassifizierung so allgemeingültig wie möglich zu halten. Da aber die einzelnen Bundesländer nicht unerheblich voneinander abweichende Objektschlüsselkataloge aufgestellt haben, wird eine Anpassung dieser Tabellen auf eventuell vorhandene weitere Interessenten notwendig sein. Um darauf flexibel reagieren zu können wurden die gesamte Umsetzung der Folien und Objektschlüssel in eine extra Klasse eingeordnet. Nur diese muß dann ausgetauscht werden. Die Klasse geoweb.EdbsOskaKatalog.class besitzt folgende Methoden:

- Methoden zur Übersetzung von Folien in Ebenen und Objektschlüssel in Arten
- Methoden zur Erstellung von Ebenen und Arten sowie derer Unterelemente
- Methoden zur Umwandlung von Objektnamen in lesbare Texte
- Methoden zur Stringumwandlung

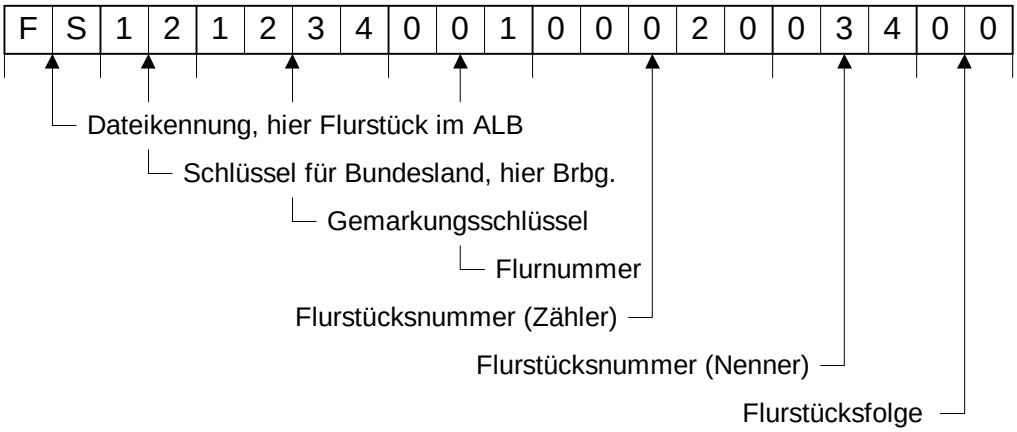
Kann für eine Folie oder eine Objektart kein entsprechender Pendant gefunden werden, wird ein neues Objekt angelegt. Die umgesetzten Ebenen erhalten dann den Namen “Folie ###”, die Objektarten den Namen “OSKA ####”. Der Anwender kann im nachhinein einen Klassenbegriff eintragen und die Standardfarbwerte und –einstellungen ändern.

Den Punkt-, Linien- und Textobjektschlüsseln, die nicht in der Umsetztabelle eingetragen sind , wird eine Standardart zugeordnet.

Es werden nur diejenigen Ebenen und Arten erstellt, die bei der Übersetzung der Folien und Objektschlüssel abgefragt wurden. Bei ihnen wird das Flag [isUsed] gesetzt. Nur diese Arten und Ebenen werden in eine IVF-Datei geschrieben, so daß sie nicht mit ungenutzten Elementen belastet wird.

Die Objektnamen bei eingerichteter Fachdatei (Art der Information 11 und 13) sind verschlüsselte Zeichenketten. Da nur Teilbereiche dieser Zeichenketten für die Darstellung in der Grafik relevant sind, sollen sie bereits bei der Übernahme in Instanzen der Klasse geoweb.Objekt.class umgewandelt werden. Dafür ist die Methode getObjektName() zuständig. Diese wertet folgende Fachdateikennungen aus:

- FS = Flurstück im ALB
- FB = Flurstück im BEDV
- FL = Flur im ALB
- GM = Gemarkung im ALB
- HA = Gebäude



Aus dem in der Abbildung gezeigten Flurstückskennzeichen würde das Programm den Text “20/34” herausfiltern. Aus dem Objektnamen mit der Fachdateikennung “FL” wird die Flurnummer entnommen und ihr der Text “Flur” vorangestellt. Dasselbe Prinzip wird bei der Gemarkungsnummer angewendet. Aus den Gebäudekennzeichen wird die Hausnummer, der Adressierungszusatz und die Nummer des laufenden Gebäudes entnommen. Für Nebengebäude wird die laufende Nummer in runde Klammern gesetzt und gespeichert. Bei Hauptgebäuden wird die Hausnummer mit Adressierungszusatz gespeichert. Sollte es sich dabei um eine Pseudohausnummer handeln, wird sie in mit “<>” geklammert.

Umlaute werden von den ALK-Systemen beim Schreiben der EDBS-Dateien durch ASCII-Zeichen ersetzt, die in dem Bereich von 0 bis 126 liegen. Die Methode getText() in der Klasse geoweb.EdbsOskaKatalog.class übernimmt die Rückübersetzung. Dabei werden auch eventuell vorhandene Leerzeichen am Ende des Strings abgeschnitten.

Nach diesen Transformationen und Umsetzungen wird das jeweilige Element in den vorbereiteten Array in der GeoWeb-Datenbank gespeichert. Dabei tritt beim Speichern der Linien eine Besonderheit auf. In den EDBS-Daten können Linien mehrere Funktionen besitzen; so wird eine Flurstücksgrenze, die zugleich auch Gebäudekante und Nutzungsartengrenze ist, mit den drei Objektschlüsseln 233, 1013 und 241 gespeichert. Diese Informationen werden auch vom Einleseprogramm in mehreren Instanzen der Klasse geoweb.EdbsLinie.class abgelegt. Nach der Verknüpfung der Objekte sind diese Informationen nicht mehr notwendig, da alle Linien, die zur Definitionsgeometrie der Objekte gehören, mit der Linienart des Objektes dargestellt werden. In die GeoWeb-Datenbank wird nur die erste Linienfunktion geschrieben, dadurch kann erheblich Speicherplatz gespart werden.

Ein weiteres Problem ergab sich bei der Speicherung der Objekte. Sie sollen in der Reihenfolge in der Objekt-Array geschrieben werden, in der sie von der Grafikklasse dargestellt werden sollen, damit dieser umfangreiche Umsortierungen erspart bleiben. Versuche, die Objekte nach Flächengröße oder aufsteigend nach Objektschlüssel zu sortieren, erbrachten nicht das gewünschte Ergebnis; die Flächen überdeckten sich in einer für den Betrachter ungewohnten Weise. Realisiert wurde die Sortierung, indem zuerst die Objekte von Folien mit flächendeckenden Nachweis in den Objektarray geschrieben werden. Die restlichen Objekte werden aufsteigend nach Folienschlüssel sortiert.

4.2.1.5 Fehlerbehandlung

Der Programmbereich, der die EDBS-Daten importiert, ist sicher der fehleranfälligste im gesamten Projekt. Es wurde eine umfangreiche Fehlerbehandlungsroutine implementiert, sie gibt die Fehlermeldung mit genauer Adressierung der letzten begonnenen Leseoperation an. Es bleibt die Hoffnung, daß diese Methode selten angesprochen wird.

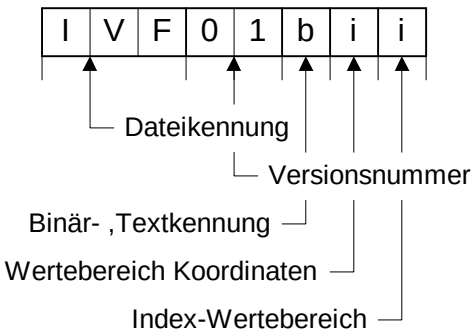
4.2.2 Lesen und Schreiben von IVF-Dateien

Für das Lesen und Schreiben der IVF-Dateien sind die beiden Klassen geoweb.IvfReader.class und geoweb.IvfWriter.class zuständig. Sie wurden von der Klasse java.awt.Dialog abgeleitet, es erscheint also beim Aufruf dieser Klassen ein eigenes Fenster. Dieses wurde nicht so umfangreich gestaltet, wie die EdbsReader-Klasse, geht doch der Lese- bzw. Schreibvorgang wesentlich schneller von statten, die Fehleranfälligkeit ist ebenfalls geringer.

Der Konstruktor der IvfReader-Klasse erwartet entweder einen Dateinamen und den Dateipfad oder eine URL. Mit beiden Argumenten wird ein InputStream erzeugt, der der Klasse DataInputStream übergeben wird. Mit dieser Konstruktion wird gewährleistet, daß die Klasse geoweb.IvfReader.class auf einer lokalen Station oder im Internet Daten lesen kann. In der Klasse IvfWriter ist diese Funktionalität nicht implementiert, sie soll nur auf einer lokalen Station Dateien schreiben können.

Der Aufbau einer IVF-Datei entspricht dem unter 3.4 erstellten Konzept. Als Steuerzeichen sind Byte-Werte im Bereich von 201-255 vorgesehen. Es kann zwar nicht garantiert werden, daß diese Werte nicht auch an anderen Stellen in der IVF-Datei auftreten, können doch die Koordinaten jeden beliebigen Wert annehmen, sie sollten jedoch eine genügende Ablaufkontrolle bieten, die zum Abbruch des Einlesevorgangs führt, wenn eine fehlerhafte Datei vorliegt.

Die ersten acht Byte der Datei sind mit einer Dateikennung belegt.



Die IVF-Kennung ist immer mit der Zeichenkette “IVF” belegt. Eine anderweitige Belegung soll zur sofortigen Zurückweisung der Datei führen. Die zweistellige Versionsnummer soll eine Erkennung von späteren Formaterweiterungen möglich machen. Die Binär- ,Textkennung gibt an, ob die vorliegende Datei im Binär- bzw. Textformat gespeichert ist. Die letzten beiden Zeichen geben die Wertebereiche für die Koordinaten und die Anzahl der Grundelemente an. Vorerst ist nur ein “i” für den Wertebereich 0-65535 vorgesehen, d.h. die Koordinaten und die Elementindizes

werden mit 2 Byte gespeichert. Eine Erweiterung des Formates auf 4 Byte-Werte ist vorgesehen, um eine Speicherung größerer bzw. genauerer Koordinatenfelder und eine größere Elementanzahl zu gewährleisten. Die Kennung soll dann mit einem “I” belegt sein.

Die Daten werden in Blöcken mit folgenden Blockkennungen gespeichert:

- Header (201),
- Tabellen (202),
- Punkte (203),
- Linien (204),
- Texte (205),
- Objekte (206).

Jeder Block wird mit einer Blockendekennung (254) versehen. Die Blöcke sind in Listen unterteilt, die Listenkennungen beginnen wiederum mit dem Wert größer 200, das Ende der jeweiligen Liste wird mit dem Wert 253 markiert.

- Block Header

Im Block Header ist vorerst nur eine Liste vorgesehen. Sie enthält die Grafikeinstellungen und die Werte, die die in der Datei enthaltenen Daten beschreiben.

Kennung	Name	Typ	Länge [byte]	Wertebereich
201	Projektname	String	beliebig	ASCII 0-127
202	Lage	String	beliebig	ASCII 0-127
203	Datenquelle	String	beliebig	ASCII 0-127
204	Konverter	String	beliebig	ASCII 0-127
205	Erzeuger	String	beliebig	ASCII 0-127
206	Eigentümer	String	beliebig	ASCII 0-127
207	isObjPunkt	boolean	1	0, 1
208	isObjColor	boolean	1	0, 1
209	isObjName	boolean	1	0, 1
210	isObjOska	boolean	1	0, 1
211	Version	byte	1	0-99

Die Strings werden mit der Stringendekennung (250) versehen.

- Block Tabellen

Der Tabellen-Block enthält mehrere Listen, sie bilden die Elemente zur Ordnung der Geometrie ab. Nach der jeweiligen Listenkennung folgt 1 Byte, daß die Anzahl der folgenden Elemente angibt. Die Elemente werden hintereinander in die Datei geschrieben.

Ebenen (201)

Name	Typ	Länge [byte]	Wertebereich
Nummer	byte	1	0-99
Farbnummer	byte	1	0-99
isVisible	boolean	1	0, 1
Name	String	beliebig	ASCII 0-127

Punktarten(202)

Name	Typ	Länge [byte]	Wertebereich
Nummer	byte	1	0-99
Größe	int	2	0-kMax
Anzahl Punktartelemente	byte	1	0-99
Punktartelement	byte[]	je 5	0-99

Linienarten(203)

Name	Typ	Länge [byte]	Wertebereich
Nummer	byte	1	0-99

Größe	int	2	0-kMax
Anzahl Linienartelemente	byte	1	0-99
Linienartelement	byte	je 1	0-99

Textarten (204)

Name	Typ	Länge [byte]	Wertebereich
Nummer	byte	1	0-99
Größe	int	2	0-kMax
Textneigung	byte	1	0-199
Textabstand	byte	1	0-99
Textzentrum	byte	1	0-8

Da die Textneigung ist in der GeoWeb-Datenbank von –99 bis +99 definiert. Sollte ein negativer Wert auftreten, wird (-neigung+100) in die Datei geschrieben.

Objektarten (205)

Name	Typ	Länge [byte]	Wertebereich
Nummer	byte	1	0-99
Farbnummer	byte	1	0-99
Linienart	byte	1	0-99
Textart	byte	1	0-99
Visible	byte	1	0-15
Name	String	beliebig	ASCII 0-127

Der Wert Visible setzt sich aus den Boolean-Schaltern isAgVisible, isFlVisible, isTxVisible und isTxAbsolut zusammen, sie belegen die ersten 4 Bit.

Hinter den Listenkennungen der folgenden Blöcke wird jeweils ein 2-Byte-Wert gespeichert, der die Anzahl der daraufhin folgenden Elemente angibt. Die Nummer des jeweiligen Elementes ergibt sich aus der Stellung in seiner Liste. Die Reihenfolge muß beim Lesen eingehalten werden, da die Listen mit Zeigern verkettet sind.

- Block Punkte

Für die Speicherung von Punkten sind zwei Varianten in Abhängigkeit der Belegung mit Art und Ebene möglich. Sind der Großteil der Punkte mit Art und Ebene belegt werden alle Punkte in der Liste “Punkte” gespeichert.

Punkte (201)

Name	Typ	Länge [byte]	Wertebereich
Ebene	byte	1	0-199
Art	byte	1	0-199
x	int	2	0-kMax

y	int	2	0-kMax
Symbolwinkel	int	2	0-360

Die Punktart wird nur gespeichert, wenn sie mit einem Wert größer 0 belegt ist. In diesem Fall wird der Wert Ebene mit Ebene+100 belegt. Das Feld Symbolwinkel wird ebenfalls nur bei Belegung gespeichert. In diesem Fall wird die Punktart mit einem Wert größer 100 belegt.

Die zweite Variante zur Speicherung der Punkte wird durch zwei aufeinander folgende Listen realisiert. Diese Variante wird der Regelfall sein, da selten mehr als 10% der Punkte mit Art und Ebene belegt sind.

Koordinaten (202)

Name	Typ	Länge [byte]	Wertebereich
x	int	2	0-kMax
y	int	2	0-kMax

Symbole (203)

Name	Typ	Länge [byte]	Wertebereich
Punktindex	int	2	0-iMax
Ebene	byte	1	0-99
Art	byte	1	0-199
Symbolwinkel	int	2	0-360

In der Symbolliste werden nur die Punkte gespeichert, die mit Art und Ebene belegt sind. Der Punktindex ist ein Zeiger auf das jeweilige Element der Koordinatenliste.

Der Symbolwinkel wird analog zur Punktliste nur bei Belegung gespeichert. Der Wert der Punktart wird in diesem Fall um 100 erhöht.

- Block Linien

Für den Block Linien, wie auch die folgenden Blöcke, ist vorerst nur eine Liste vorgesehen.

Linien (201)

Name	Typ	Länge [byte]	Wertebereich
Ebene	byte	1	0-199
Art	byte	1	0-199
Linienanfang	int	2	0-iMax
Linienende	int	2	0-iMax
Typ	byte	1	0-99
h	int	2	0-kMax

Im Feld “Ebene” wird der Wert Ebene+100 gespeichert, wenn der Linientyp von 0 abweicht. In diesem Fall ist die Linie keine Gerade, und es werden weitere Parameter zur Beschreibung der Geometrie benötigt, die Felder “Typ“ und “h” sind dann belegt. Ist die Linie ein Kreisbogen, wird im Feld “Typ“ der Wert 10 gespeichert, im Feld “h” wird die positive Pfeilhöhe gespeichert. Ein negativer Wert wird markiert, indem zu dem Wert “Art” 100 addiert werden.
Ist die Linie ein Spline wird das Feld “Typ” mit dem Wert 20 belegt. Das Feld “h“ enthält die Anzahl der Splinepunkte. Diese werden nachfolgend geschrieben.

Name	Typ	Länge [byte]	Wertebereich
x	int	2	0-kMax
y	int	2	0-kMax

- Block Texte

Texte (201)

Name	Typ	Länge [byte]	Wertebereich
Ebene	byte	1	0-99
Art	byte	1	0-199
x	int	2	0-kMax
y	int	2	0-kMax
Textwinkel	int	2	0-360
Text	String	beliebig	ASCII 0-127

Das Feld Textwinkel wird nur belegt, wenn der Wert von 0 abweicht. Dieses wird markiert, indem zur Textart 100 addiert werden.

- Block Objekte

Objekte (201)

Name	Typ	Länge [byte]	Wertebereich
Ebene	byte	1	0-99
Art	byte	1	0-99
Oska	int	2	0-9999
Typ	byte	1	1,2
x	int	2	0-kMax
y	int	2	0-kMax
Objektname	String	beliebig	ASCII 0-127
Anz. DefLinien	int	2	0-iMax
Anz. AusgLinien	int	2	0-iMax

Die beiden letzten Felder sind mit der jeweiligen Anzahl der Definitions- und der Ausgestaltungsgeometrielinien belegt. Anschließend wird die Liste von Zeigern auf diese Linien als 2-Byte-Werte gespeichert.

Nach der Speicherung aller Elemente wird die Blockkennung EOF (255) geschrieben. An dieser Stelle soll der Lesevorgang beendet werden.

4.3 Grafikausgabe

Die Klasse geoweb.GraphicPanel.class, die für die Grafikausgabe zuständig ist, wurde von der Klasse java.awt.Panel abgeleitet. Dieses Panel wurde fest in die Oberfläche des Programmsystems integriert.

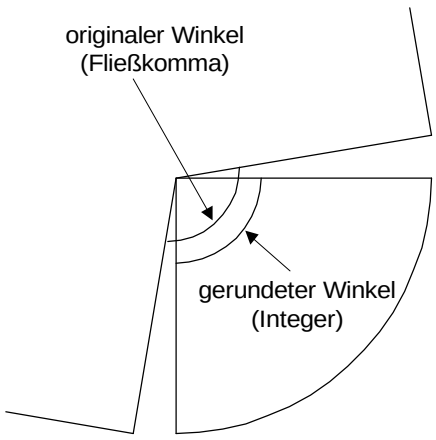
Der Aufbau der Grafik läuft in folgenden Schritten ab:

- Objektflächen zeichnen,
- Objektumring und Ausgestaltungsgeometrie zeichnen,
- Objekttexte zeichnen,
- Linien zeichnen,
- Texte zeichnen,
- Punkte zeichnen.

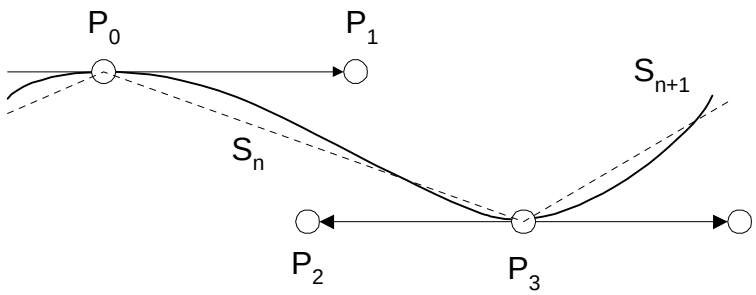
Damit wird gewährleistet, daß die Objektflächen nicht andere Grafikelemente überdecken.

Die eigentlichen Methoden, die das Zeichnen der Grundelemente realisieren, sind in den Klassen PunktArt, LinienArt und TextArt und deren untergeordneten Klassen integriert. Dabei wird diesen Methoden der transformierte Einfügepunkt übergeben, die Transformation der Einzelelemente wird dann von den Grafikmethoden der Arten-Klassen übernommen.

Für die Darstellung der Kreisbögen wurde nicht auf die Methoden des Paketes java.awt zurückgegriffen. Diese erwarten als Start- bzw. Endwinkel einen Integer-Wert. Da aber bei größeren Radien sich die Bogenlänge eines gerundeten Winkels erheblich ändert, erbrachten die Methoden nicht die gewünschten Ergebnisse.



Es wurde eine Methode arcToPoly() implementiert, die einen Kreisbogen in ein Sehnenpolygon umwandelt. Dieses Polygon wird dann von den Grafikmethoden gezeichnet. Es wird auch verwendet, um die Objektflächen mit zu definieren.
Ein ähnliches Vorgehen bietet sich bei der Darstellung von Splines an, werden doch auch sie als Definitionsgeometrie der Objekte verwendet. Da eine feste Anzahl von Zwischenpunkten vorgesehen ist, kann die Berechnung dieser beim Start der Grafikroutine erfolgen. Die Koordinaten der Zwischenpunkte brauchen beim Aufruf der repaint-Methode nur noch in den aktuellen Ausschnitt transformiert werden. Zur Speicherung der Punkte wird ein Array innerhalb der Grafikklasse angelegt.
Zur Darstellung der Splines wurde die Bézierkurve gewählt. Zu ihrer Berechnung eines Kurvenabschnittes sind außer den beiden Stützpunkten zwei weitere Punkte zu berechnen, die die Tangentenvektoren im Anfangs- und Endpunkt beschreiben.



Die Tangenten stehen senkrecht auf den Winkelhalbierenden in den Punkten P_0 und P_3. Für den Abstand der Punkte P_0 und P_1 wird die Strecke $\min(s_n, s_{n-1})/2$ angesetzt, für den Abstand P_3 und P_2 analog die Strecke $\min(s_n, s_{n+1})/2$. So haben nah beieinander liegende Stützpunkte kaum Einfluß auf die Kurvenausbiegung. Die Berechnung der Kurvenzwischenpunkte ist ausführlich in [SEDGEWICK 1992] und auf [www-cg-hci.informatik.uni-oldenburg.de/~pgse96/] dargestellt.
Die Transformation der Koordinaten der Datenbank in das Grafikfenster übernimmt die Methode getPunkt(). Für die Transformation sind drei Parameter notwendig. Der Maßstab wird aus der Größe des Koordinatenbereiches kMax der Datenbank, der Größe des Ausgabebereiches und des Zoomfaktors entwickelt.

```
m = min(width, height)) / kMax * zoomFaktor;
```

Die Verschiebungsvektoren ergeben sich aus der Differenz der Mittelpunkte des Datenbank-(xm, ym) und des Grafikkoordinatensystems (Xm, Ym). Damit ergibt sich die entgültige Transformationsformel:

$$X = Xm + m \cdot (x - xm)$$

$$Y = Ym + m \cdot (y - ym)$$

Das Grafikfenster soll in Abhängigkeit von dem in der Symbolleiste eingestellten Wert auf Benutzeraktionen reagieren. Dazu wurden die Interfaces java.awt.event.MouseListener und java.awt.event.MouseMotionListener implementiert.

Als Aktionen sind vorgesehen:

- Zentrieren, dabei wird der Mittelpunkt des Datenbank-Koordinatensystems verschoben, der Anwender kann sich durch die Grafik bewegen,
- Zoomen, dabei wird der Wert der Variablen zoomFaktor verdoppelt oder halbiert,
- Objekt anwählen.

Bei letzterer Aktion wird der Objekt-Array sequentiell durchlaufen, und die Objektkoordinate mit der in das Datenbank-Koordinatensystem transformierten Mauszeiger-Koordinate verglichen. Das Objekt mit dem kürzesten Abstand zum Mauszeiger wird daraufhin mit einer komplementären Farbe nachgezeichnet, so daß es für den Benutzer anwählbar wird. Wird daraufhin die Maustaste gedrückt, werden Informationen zum Objekt in Abhängigkeit von eingestellten Programmparametern in der Statusleiste angezeigt.

Das Verfahren des sequentiellen Durchsuchens des Objekt-Arrays ist für die zu erwartenden kleineren Datenmengen noch genügend performant, bei größeren Punktmengen müßten Gitterverfahren angewandt werden, die ausführlich in [SEDGEWICK 1992] beschrieben sind.

4.4 Kommunikation

Um auf Benutzeraktionen reagieren zu können, muß eine Kommunikation zwischen den Programmkomponenten möglich sein. Dazu wurde im Hauptprogramm GeoWeb.class ein Action-Listener implementiert, der die von den Komponenten erzeugten Events registriert und verarbeitet. Folgende Events werden verarbeitet:

Erzeuger	ActionEvent	Aktion
MainMenu.class	Menübefehl	Dateioperationen, Dialoge
GraphicPanel.class	Text	Text wird in der Statusleiste angezeigt
SymbolPanel.class	Symbolnummer	Grafikkommando wird gesetzt
EbenenDialog.class	“repaint”	Grafikfenster neu zeichnen
ObjektArtenDialog.class	“repaint”	Grafikfenster neu zeichnen
ZoomDialog.class	“repaint”	Grafikfenster neu zeichnen

Durch dieses Kommunikationsmodell wird gewährleistet, daß die Klassen weitestgehend autark agieren können. So brauchen für die Komponenten, sollten sie in weiteren Programmen genutzt werden, nur ein Action-Listener registriert und die Events behandelt werden.

5 Die GeoWeb-Applets

Bei der Erstellung der Applets zeigten sich die Vorteile der objektorientierten Programmierung. Da die einzelnen Klassen weitestgehend unabhängig voneinander existieren können, mußte relativ wenig Code neu erstellt bzw. geändert werden.
Es sollen zwei Programme erstellt werden. Das Applet GeoWebClient soll mit dem selben Erscheinungsbild auftreten wie die Applikation, also als eigenes Frame; das Applet GeoWebInsideClient soll als Multimedia-Komponente in eine HTML-Seite integriert werden können.
Weiterhin soll ein Hilfesystem für die Applets erzeugt werden.

5.1 GeoWebClient

Das Programm GeoWebClient wurde in zwei Klassen zerlegt, die von java.awt.Applet abgeleitete Klasse GeoWebClient und die von java.awt.Frame abgeleitete Klasse GeoWebClientFrame. Die Klasse GeoWebClient ist nur dafür zuständig, die Parameter aus dem HTML-Code zu übernehmen und das Fenster GeoWebClientFrame zu starten.
Zur Erstellung der Klasse GeowebClientFrame.class wurde der Quelltext der Klasse GeoWeb kopiert. Die Parameterauswertung wurde neu erzeugt und es wurden aus dem ActionListener die Events entfernt, die in dem Applet nicht mehr auftreten. Diese Events sind Dateioperationen, die nur in der Applikation möglich sind, das Lesen und Schreiben von Dateien. An die Dialogboxen DateiInfoDialog.class, EbenenDialog.class und ObjektArtenDialog.class wird im Gegensatz zur Applikation der Parameter isEditable=false übergeben, damit sind einige Werte nicht mehr durch den Benutzer änderbar.
Neu erzeugt wurde ebenfalls die Klasse MainMenuClient.class. Aus dem Haupmenü des Applets wurden die Menüpunkte “Öffnen“, “Speichern“, “Importieren“ und “Exportieren“ entfernt. Hinzugefügt wurden Dateinamen, die dem Applet beim Aufruf übergeben werden können.

Die restlichen Klassen, die von dem Hauptprogramm in die Oberfläche integriert werden, können ohne Neukompilation aus der Applikation übernommen werden.

Zum Aufruf des Applets muß in die jeweilige HTML-Seite folgender Code eingefügt werden:

```
<applet
align=middle
code=GeoWebClient.class
width=50
height=50
alt="Aktivieren Sie Java!">
<param name=datei0 value=edbs.ivf>
<param name=datei1 value=bspbbg.ivf>
<param name=datei2 value=testBbg.ivf>
<param name=datei3 value=test01.ivf>
</applet>
```

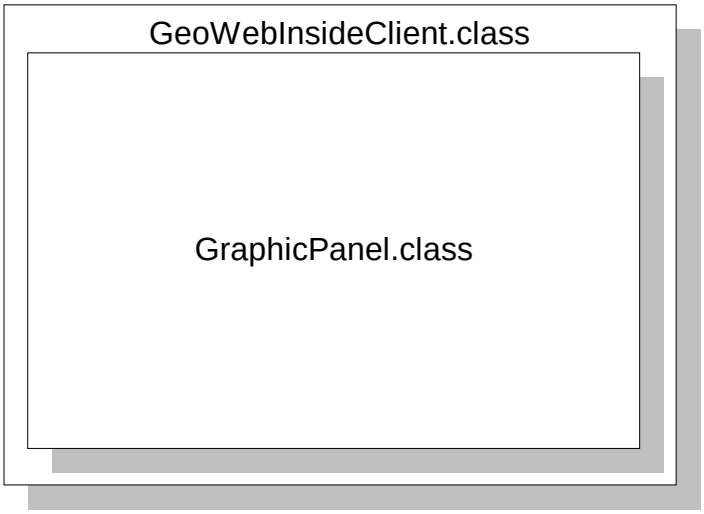
Dem Applet können die Parameter “datei0” bis “datei9” übergeben werden. Die in den Parametern angegebenen Dateien erscheinen im Hauptmenü und sind vom Benutzer anwählbar. Die Datei, die in dem Parameter “datei0” angegeben ist, wird vom Applet nach dem Start geladen. Dieser Parameter ist nicht optional.

Die dem Applet zu übergebenden Parameter “width” und “height” bestimmen die Größe des eigentlichen Applets, der Klasse, die das Frame startet. Die Größe sollte nicht auf zu geringe Werte eingestellt werden, damit das Applet mit der Maus anwählbar bleibt. Eventuelle Fehlermeldungen des Java-Interpreters werden nur in der Statuszeile des Browsers angezeigt, wenn mit der Maus die Appletfläche betreten wird.

Die Größe des Frames erzeugt sich das Programm selbst aus der eingestellten Bildschirmauflösung.

5.2 GeoWebInsideClient

Das Applet GeoWebInsideClient.class ist für die Integration in eine HTML-Seite ausgelegt. Das ist besonders sinnvoll, wenn nicht die Daten im Vordergrund stehen, sondern sie erläuternd zum Text wirken. Aus dem Applet wurden außer der Klasse GraphicPanel sämtliche Komponenten entfernt, somit stehen dem Benutzer nur noch die Aktionen “Objekt anwählen”, “Zentrieren” und “Zoomen” zur Verfügung. Diese Aktionen werden in dem Grafikfenster mit der Maus ausgelöst.



Die erläuternden Texte, die von der Methode setInfoText() in der Applikation zum InfoPanel gesendet wurden, werden in diesem Applet in die Statuszeile des Browsers umgeleitet.

Das Applet wird wie folgt in die HTML-Seite integriert:

```
<applet
align=middle
code=GeoWebInsideClient.class
width=500
height=400
alt="Aktivieren Sie Java!">
<param name=datei value=bspbbg.ivf>
</applet>
```

Die dem Applet zu übergebenden Parameter “width” und “height” bestimmen die Größe des Applets auf der HTML-Seite. Mit dem Parameter “align” kann das Applet analog zur Einbindung von Grafiken auf der Seite positioniert werden. Mögliche Werte sind: `left`, `right`, `top`, `texttop`, `middle`, `absmiddle`, `baseline`, `bottom` oder `absbottom`.

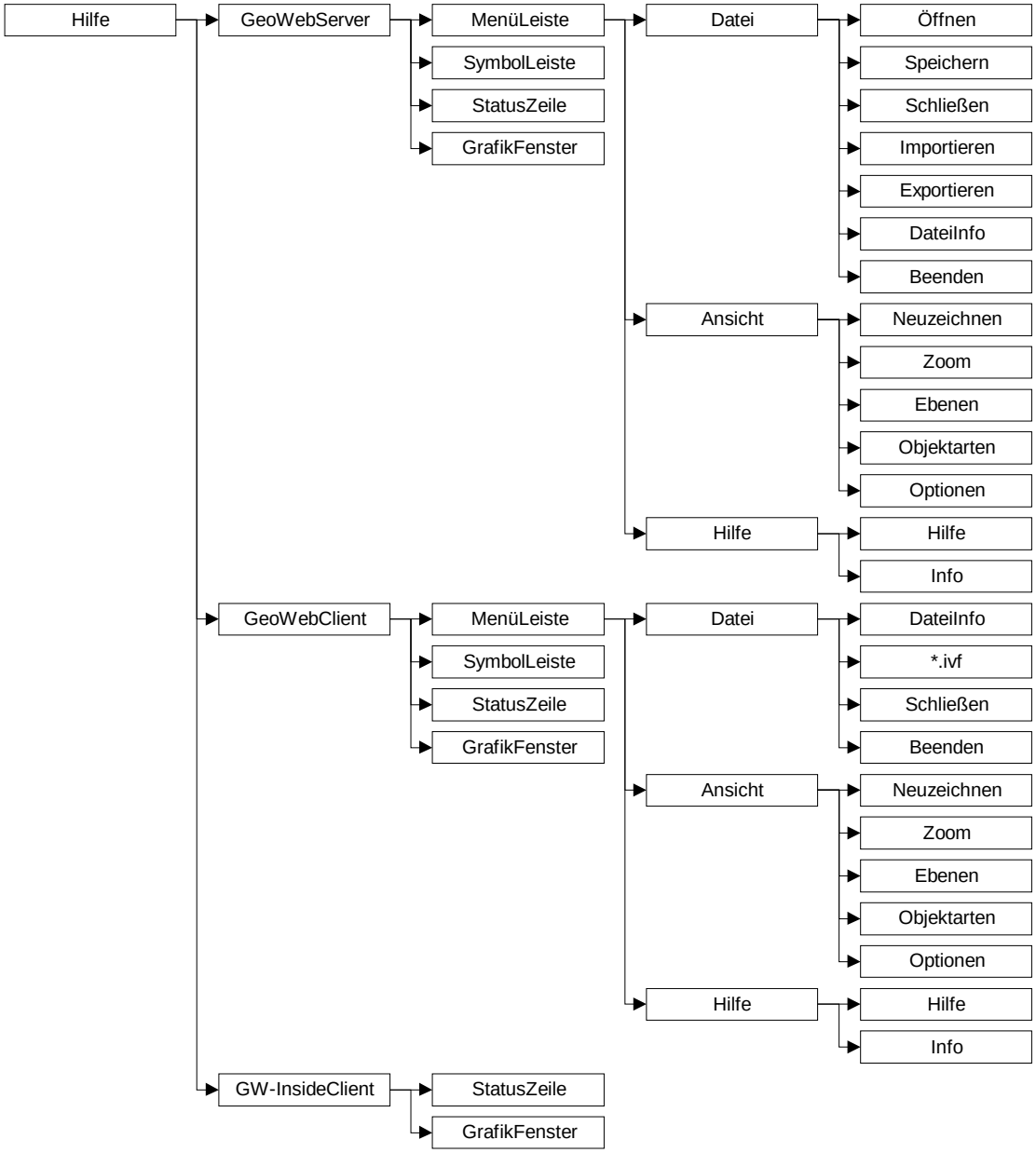
Weitehin können dem Applet die Parameter `vspace` und `hspace` übergeben werden. Die Pixelangaben bestimmen den Abstand des Applets zum umgebenden Text.

Dem Programm kann nur der Parameter “datei” übergeben werden. Die in diesem Parameter angegebene Datei wird nach dem Start des Applets geladen. Es können durch das Applet keine weiteren Dateien nachgeladen werden, das Applet kann aber mehrmals mit unterschiedlichen Parametern in eine HTML-Seite integriert werden.

5.3 Hilfesystem

Für die Programme GeoWeb und GeoWebClient wurde zwar ein Hilfe-Dialog, erreichbar über den Menüpunkt “Hilfe”, erzeugt, seine Nutzbarkeit ist jedoch sehr eingeschränkt. Hilfesysteme werden auf den verschiedenen Betriebssystemen unterschiedlich erzeugt. Sie greifen in der Regel auf systemspezifische Komponenten zurück. Eine systemübergreifende Lösung innerhalb des Programmsystems ist also nur eingeschränkt möglich.

Da HTML-Seiten auf jedem System visualisiert werden können und der Umgang mit diesen den meisten Nutzern gewohnt ist, wurde beschlossen, das Hilfesystem in HTML zu realisieren. Folgende Dokumentstruktur wurde erzeugt:



Die Dokumentstruktur und das Navigationssystem wurde der WWW-Suite von Stefan Münz entnommen, welches ein überzeugendes Beispiel bietet.
[www.teamone.de/selfhtml]
Für das Hilfesystem sollte eine feste URL eingerichtet werden, damit es bei der Einbindung der Applets in beliebige Seiten zur Verfügung gestellt werden kann. Dazu wurde bei T-Online ein Account eingerichtet, die URL lautet: http://home.t-online.de/geoweb/gw_h.htm.

Wird ein Link auf diese Seite in ein HTML-Dokument eingebunden, verläßt der Internet-Browser die vorliegende Seite und lädt das neue Dokument. Damit wird auch das GeoWeb-Applet ungültig. Um diesen Vorgang zu umgehen, sind zwei Varianten möglich:

- Verzweigung zu den Hilfeseiten mit Hilfe des Attributs TARGET,
- Verzweigung mit Hilfe einer JavaScript-Funktion.

Im ersten Fall wird dem Link zu den Hilfeseiten das Attribut TARGET="_top" übergeben, damit erscheinen die Hilfeseiten in einem neuen Browserfenster. Der Code, der in eine HTML-Seite einzubinden ist, könnte wie folgt lauten:

```
<A HREF="http://home.t-online.de/geoweb/gw_h.htm"
TARGET="_top">Hilfe zu GeoWeb</A>. [TOLKSDORF 1996]
```

Im zweiten Fall wird eine JavaScript-Funktion in das HTML-Dokument eingebunden, die beim Verweis auf das Hilfe-Dokument angesprochen wird.

```
<HTML>
<HEAD>
      …
</HEAD>
<SCRIPT LANGUAGE="JavaScript">
function link(URL) {
      var fh = window.open(URL, "FH", "width=300, height=400,
      toolbar=no, location=no, directories=no, status=no,
      menubar=no, scrollbars=yes");
  }
</SCRIPT>
<BODY>
      …
      <A HREF="javascript:link('http://home.t-online.de/
      geoweb/gw_h.htm')"> Hilfe zu GeoWeb</A>
      …
</BODY>
</HTML>
```

Die Funktion link erzeugt mit der Methode open ein neues window-Objekt. In dieses Fenster wird die mit dem Parameter URL übergebene Dokument geladen. Das Fenster erhält den Namen "FH". Die im letzten Parameter übergebenden Attribute bestimmen das Aussehen des Fensters. Die Attribute width und height

bestimmen die Dimension des Hilfe-Fensters. Die folgenden Attribute schalten die Funktionsleiste, die URL-Eingabezeile, die Verzeichnisschaltflächen, die Statusleiste und die Menüleiste aus. Die Anzeige der Scrollbalken werden mit dem Attribut `scrollbars=yes` eingeschalten. [EISENMENGER 1997]

Mit dieser Konstruktion wird ein Hilfe-Fenster erzeugt, welches durch seine geringe Dimensionierung während des Arbeitens mit dem Programmsystem GeoWeb geöffnet bleiben kann.

6 Test mit ALK- Daten

Vom Landesvermessungsamt Brandenburg wurden zwei EDBS-Dateien zur Verfügung gestellt, die Dateien bspbbg.edb und testbbg.edb. Die Datei bspbbg.edb enthält Daten, die auf Grundlage der Richtlinien zur Datenerfassung der Liegenschaftskarte als Vorstufe der ALK erzeugt wurden. [ALK-RICHTLINIEN BRB]
Die Datei testbbg.edb enthält auch Elemente der Topographie. In beide Dateien sind die Daten in ULO8ALK-Aggregaten (3.2.2) gespeichert.
Zum Testen der EDBS-Schnittstelle wurde weiterhin eine von der Firma HHK Datentechnik Braunschweig zur Verfügung gestellte Datei edbs.edb benutzt. Grundlage für diese Datei war eine Testdigitalisierung für das Land Niedersachsen. Die Datensätze in dieser Datei sind in ULOBNN-Aggregaten (3.2.3) gespeichert.
Getestet werden soll die Performance der EDBS-Umsetzung und die Netzwerktauglichkeit der Clients. Beides sind wichtige Parameter, die über die Praxistauglichkeit entscheiden. Ein korrektes Funktionieren des Programmsystems wird natürlich vorausgesetzt.

6.1 Performance der EDBS-Umsetzung

Die zu schwache Performance der Java-Interpreter ist eines der Hauptprobleme bei der Entwicklung größerer Java-Applikationen. Bei der Entwicklung der vorliegenden Applikation wurde festgestellt, daß besonders der Dateizugriff Performanceschwierigkeiten bereitet. Daher soll das Lesen von EDBS-Dateien im folgenden betrachtet werden.
Für den Test stand folgendes System zur Verfügung:

Prozessor: Pentium 100Mhz
Hauptspeicher: 24Mb
Betriebssystem: Microsoft Windows 95
Java-Interpreter: java (Sun JDK 1.1) und
jView (MS Internet-Explorer 4.0)

Dieses System kann sicher als Minimal-Voraussetzung angesehen werden.
Folgende Laufzeiten wurden gemessen:

Datei	Größe [kb]	Laufzeit [s] java	Laufzeit [s] jView
bspbbg.edb	137	32	6
testbbg.edb	572	325	38

edbs.edb	648	160	20

Der Test mit Suns Java-Interpreter ergibt eine wesentlich zu lange Laufzeit. Für eine 2 Mb große EDBS-Datei ergibt sich hochgerechnet eine Laufzeit von etwa einer halben Stunde. Dies ist für einen Praxiseinsatz nicht akzeptabel.

Die wesentlich kürzere Laufzeit mit Microsofts jView läßt sich auf den integrierten JIT-Compiler zurückführen.

Der in der zum Dezember 1998 von Sun angekündigten JDK-Version1.2 soll der Java-Interpreter einen integrierten JIT-Compiler enthalten. Damit ist diese Technologie systemübergreifend verfügbar, die Performance dürfte der von Microsofts jView entsprechen.

6.2 Performance der Internet-Übertragung

Der Erfolg einer Internet-Veröffentlichung hängt auch wesentlich von der Dauer der Datenübertragung ab. Eines der wichtigen Ziele bei der Entwicklung des Programmsystems war die Erzeugung von einem kompakten Datenformat, welches die Darstellung von raumbezogenen Daten im Internet erst möglich macht. Bei den ersten Online-Versuchen stellte sich heraus, daß die Übertragung der Java-Klassen einen nicht unwesentlichen Anteil an der Gesamtübertragungszeit besitzt.

Zur Ausführung des GeoWebInsideClients sind 21 Klassen mit insgesamt 54,5 kb notwendig. Zum Start des GeoWebClients ist das Laden von 25 Dateien mit 61,8 kb erforderlich, weitere 15 Klassen mit insgesamt 32,6 kb werden beim Aufrufen der Dialoge nachgeladen. Die Dateien wurden auf den eingerichteten Speicherplatz auf dem T-Online-Server mittels FTP übertragen. Die Übertragung wurde dann mit Microsofts Internet-Explorer 4.0 und Netscapes Communicator 4.06 bei abgeschalteten Cache getestet. Dabei wurden bei einer Verbindung mit 33600 bps folgende Werte erreicht:

	Internet-Explorer	Netscape Communicator
GeoWebClient	46 s	43 s
GeoWebInsideClient	30 s	23 s

Um die Ladezeiten weiter zu reduzieren, wurden die Klassen in JAR-Archive gepackt, das Programm zur Komprimierung liegt dem JDK 1.1 bei. Die Dateien GeoWebClient.jar und GeoWebInsideClient.jar sind 62,2 kb und 33,0 kb groß. Bei der Übertragung ergaben sich folgende Ladezeiten:

	Internet-Explorer	Netscape Communicator
GeoWebClient	25 s	23 s

GeoWebInsideClient	19 s	15 s

Die geringeren Ladezeiten ergeben sich aus den geringeren Umfang der zu übertragenden Daten und dadurch, daß nur noch ein GET-Befehl vom Internet-Browser abgesetzt werden muß. Dadurch entfallen die Zeiten, die der Browser auf Antwort vom Server warten muß. Dadurch, daß der Browser den Ladezustand anzeigt, ist es auch besser für den Benutzer abschätzbar, mit welchen Ladezeiten er noch rechnen muß.

Nach der Initialisierung der Applets beginnt die Klasse IvfReader mit dem Laden der eigentlichen Daten.

Die Umsetzung von EDBS-Daten in das IVF-Format ergab folgende Dateigrößen:

Datei	Größe EDBS-Datei [kb]	Größe IVF-Datei [kb]	Übertragungszeit bei 2,0 kb/s [s]
bspbbg	137	19,3	10
testbbg	572	62,7	32
edbs	648	30,3	15

Die IVF-Dateien besitzen 1/7 bis 1/20 der Größe ihrer originalen EDBS-Dateien. Die Größe entspricht im Internet üblichen Bilddateien im GIF- bzw. JPEG-Format. Damit wurde das gestellte Ziel erreicht, Daten aus Geoinformationssystemen mit relativ geringen Mitteln im Internet veröffentlichen zu können. Bei der Erzeugung größerer Dateien, die beim Umsetzen von Kartenblättern von Gemeinden entstehen würden, sollte der Internetnutzer auf die zu übertragenden Daten im HTML-Text vorbereitet werden.

7 Zusammenfassung

Bei der Erzeugung raumbezogener Daten werden in den letzten Jahren weitestgehend rechnergestützte Technologien angewendet. Zunehmend werden die Daten in komplexen Geo-Informationssystemen gespeichert. Die einfache und schnelle Nutzung solcher Daten ist aber noch vielen Einschränkungen unterworfen, bereiten doch die vielen unterschiedlichen Datenbank- und Schnittstellenkonzepte oft Probleme bei der Datenübertragung. Durch die geringe Unterstützung der GIS-Schnittstellen durch Standardapplikationen kann man von einer allgemeinen Zugänglichkeit von Geo-Daten nicht sprechen.

Die ALK als Nachweis des Liegenschaftskatasters bietet ideale Voraussetzungen als Basis für raumbezogene Datenbanken. Der Nutzerkreis der ALK könnte wesentlich breiter sein als bisher. Ein Grund für die noch zu geringe Verbreitung der ALK ist sicher die nicht allgemein zugängliche Information über die Verfügbarkeit der Daten.

Eine Veröffentlichung im WWW kann die Verbreitung der ALK wesentlich unterstützen. Mit dem in dieser Arbeit erstellte Programmsystem können dem potentiellen Kunden Daten, die erwerbbar sind, im Internet vorgeführt werden. Ebenfalls kann es bei der Lieferung von ALK-Daten den EDBS-Dateien beigelegt werden, um den Nutzer bei eventuellen Problemen mit der Datenumsetzung zu unterstützen.

Nach Abschluß der ersten Testphase wurde das Programmsystem GeoWeb dem Leiter der Projektgruppe “Internet” und dem Leiter der Arbeitsgruppe “ALK” beim Landesvermessungsamt Brandenburg vorgestellt. Dabei sollte auch getestet werden, ob die Funktionalität des Programmsystems für den Anwender ohne Hilfestellung erfaßbar ist. Es zeigte sich, daß das Anwendungsprogramm für die Testpersonen sofort beherrschbar war.

Es wurden weitere EDBS-Daten aus dem ALK-System ausgelesen, um die Stabilität des Programms zu testen. Hierbei zeigte sich, daß das Programmsystem auch ohne weiteres größere Datenmengen aufnehmen kann. Eine 6,5 Mb große Beispieldatei konnte ohne Probleme in das IVF-Format umgesetzt und mit den Applets dargestellt werden. Überraschend für die Begutachter war die relativ originalgetreue Abbildung des ALK-Datenbestandes.

Die Funktionalität des Systems wurde für ausreichend befunden. Eine Ausweitung des Funktionsumfangs ist vorstellbar, für die Veröffentlichung von ALK-Daten vorerst aber noch nicht erwünscht. Der Umfang und die Strategie der Veröffentlichung von

ALK-Daten soll in der Projektgruppe “Internet” beim Landesvermessungsamt Brandenburg kurzfristig geklärt werden.

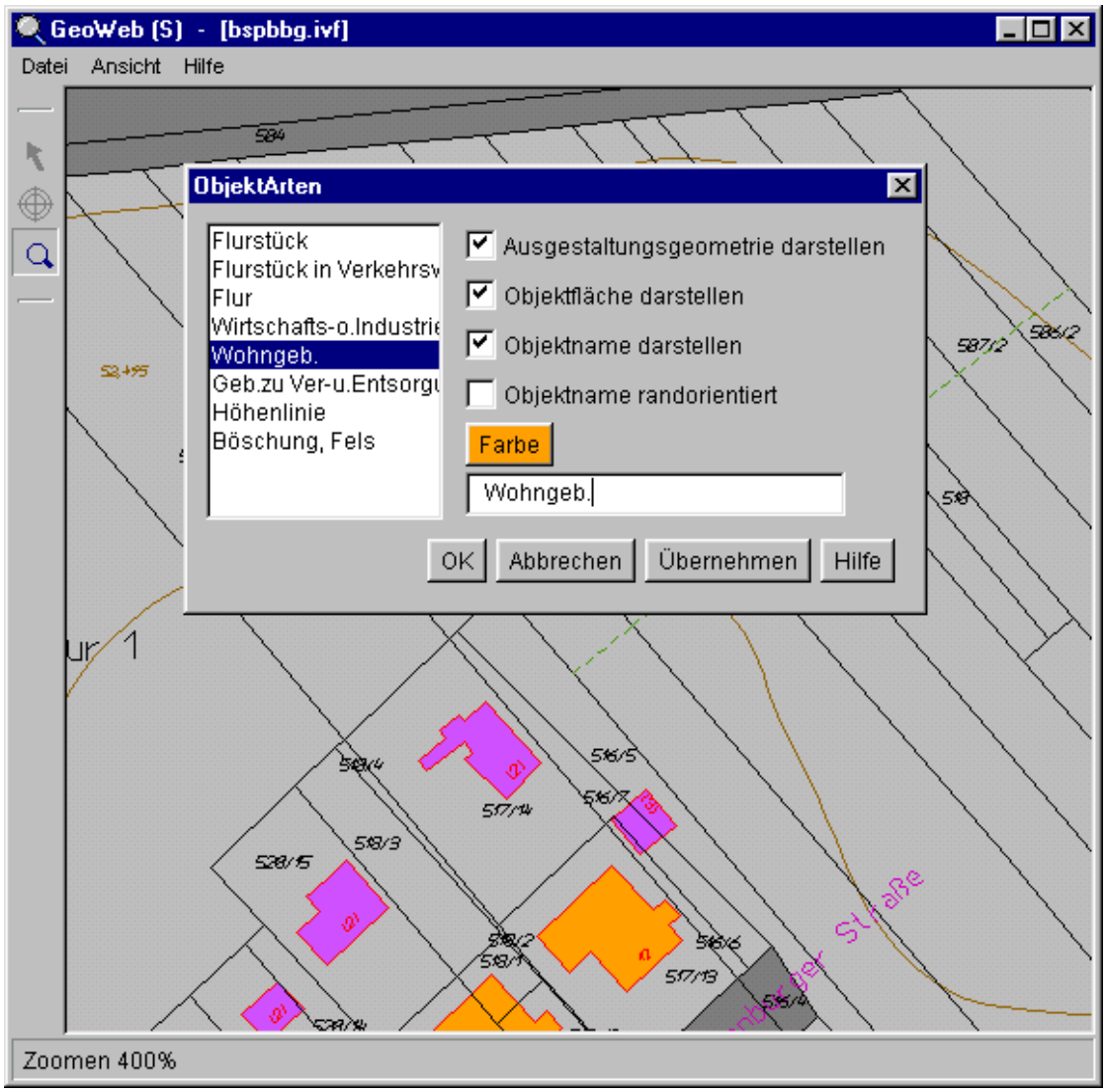
Das Programmsystem kann weiterhin genutzt werden, um, wenn dies politisch gewollt ist, der Öffentlichkeit aktuelle Ausschnitte aus der Liegenschaftskarte zur Verfügung zu stellen. Auch könnte es Unterstützung bei der Veröffentlichung von Bebauungs- oder Flächennutzungsplänen leisten.

Weitere Anwendungsfälle sind denkbar, wird doch für viele Firmen durch die rasant steigenden Nutzerzahlen das Medium Internet immer interessanter. Dazu ist aber eine Erweiterung dieses Programmsystems notwendig. So sollte das Programm unbedingt eine Möglichkeit bieten, die Daten auszudrucken. Ebenfalls sollte es das Kopieren in die Zwischenablage anbieten, beides ist ab der Java 1.1-Version möglich.

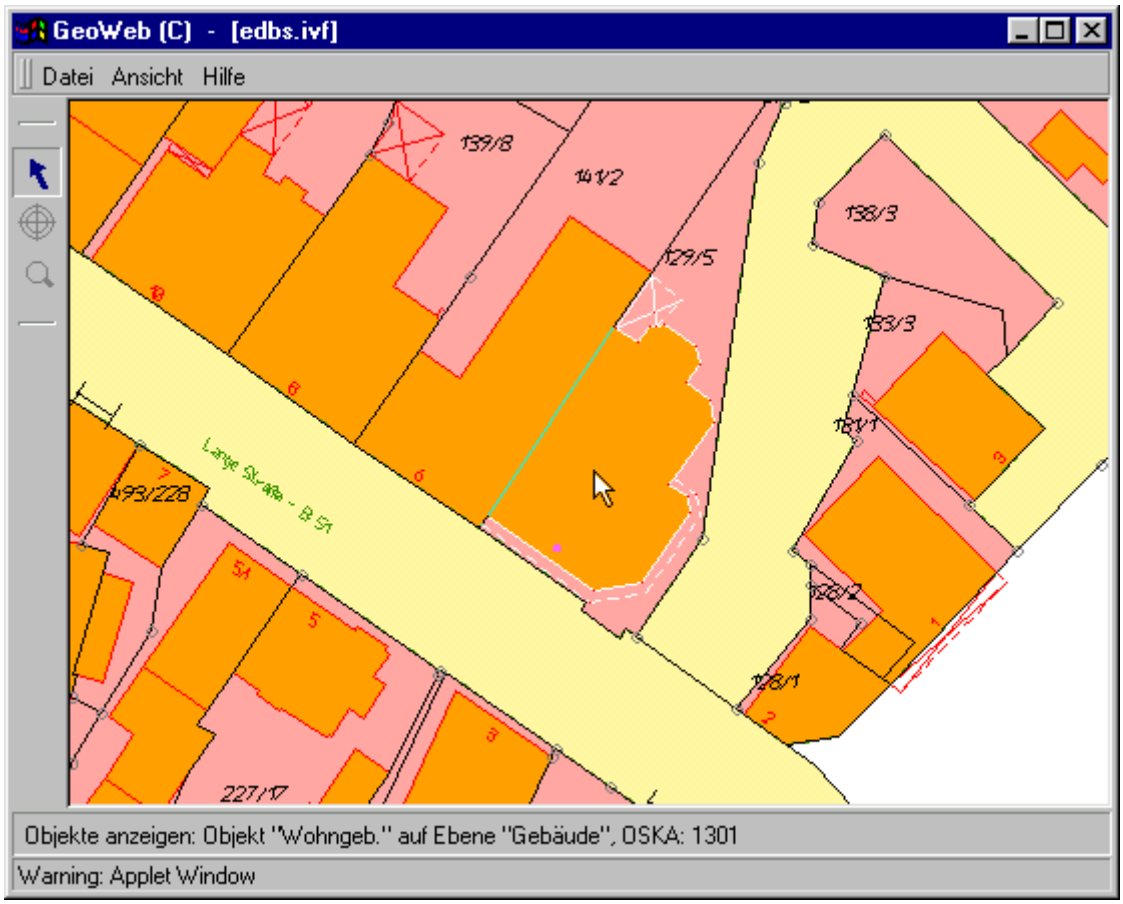
Um die Anwendungsmöglichkeiten wesentlich zu erweitern, ist eine Unterstützung weiterer Vektor-Datenformate unumgänglich. Bei der Vorführung des Programmsystems beim Landesvermessungsamt Brandenburg wurde bereits über eine Unterstützung von ATKIS-Daten gesprochen. Diese Erweiterung des Programmsystems sollte ohne weiteres möglich sein, unterscheiden sich doch die ATKIS-Daten nicht wesentlich von denen der ALK.

Weit verbreitete Dateiformate sind das DXF- [RUDOLPH 1993] und das HPGL-Format [BORN 1997]. Die Unterstützung dieser Formate würde eine Veröffentlichung von technischen Darstellungen und Bauzeichnungen ermöglichen.

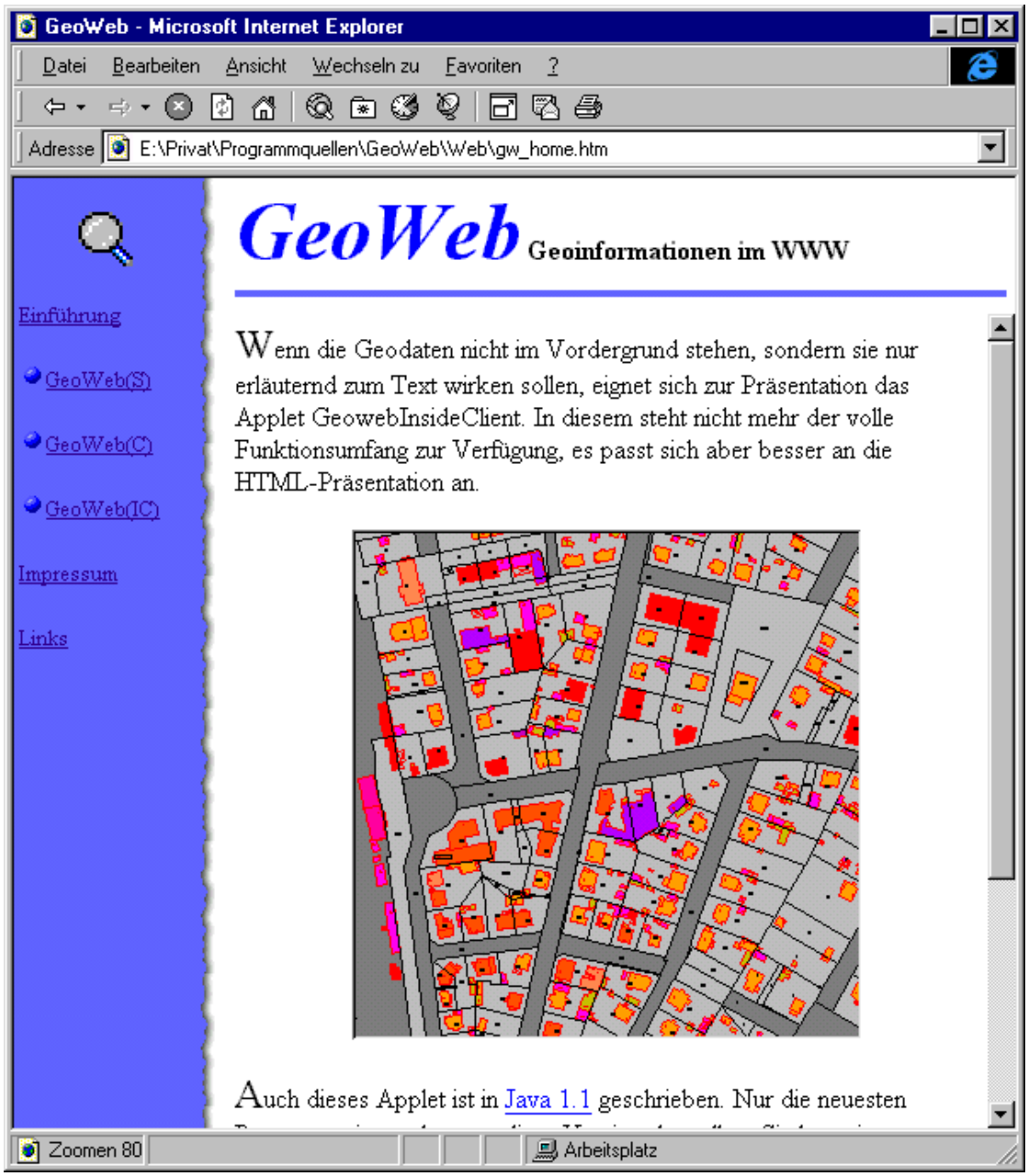
Bei entsprechender Resonanz, die die Vorstellung im Internet hoffentlich mit sich bringt, steht einer Weiterentwicklung des Programmsystems GeoWeb nichts im Wege.



Die Applikation GeoWeb beim Einstellen der Objektarten



Das Applet GeoWebClient bei der Abfrage der Objekte



Das Applet GeoWebInsideClient im Microsoft Internet-Browser

8 Glossar

ALK	**A**utomatisierte **L**iegenschafts**k**arte
EDBS	**E**inheitliche **D**aten**b**ank**s**chnittstelle, Datenbankschnittstelle zur Übertragung von raumbezogenen Daten.
GIF	**G**raphics **I**nterchange **F**ormat, von Compuserve eingeführtes Grafikformat.
HTML	**H**yper **T**ext **M**arkup **L**anguage, Textauszeichnungssprache, die für die Erstellung von WWW-Seiten benutzt wird.
JPEG	Grafikformat, daß von der **J**oint **P**hotographic **E**xperts **G**roup entwickelt wurde.
TCP/IP	**T**ransmission **C**ontrol **P**rotocol/**I**nternet **P**rotocol, Protokollfamilie, die die Übertragung im Internet steuert.
URL	**U**niform **R**esource **L**ocator, standardisierte Adresse im WWW
WWW	**W**orld **W**ide **W**eb, Internetdienst, der aus mit Links verbundenen HTML-Dokumenten besteht.

9 Quellenverzeichnis

ALK-RICHTLINIEN BRB

Richtlinien für die Einrichtung der Automatisierten Liegenschaftskarte in Brandenburg; Hrsg. Ministerium des Innern des Landes Brandenburg; 1995

BILL/FRITSCH 1991

Bill, R., Fritsch, D.: Grundlagen der Geoinformationssysteme.- Bd.1, Hardware, Software Daten.-Karlsruhe: Wichmann Verlag, 1991

BORN 1997

Born, Günter: Referenzhandbuch Dateiformate: Datenbanken, Tabellenkalkulationen, Text, Grafik, Multimedia, Sound und Internet.- Bonn; Reading, Massachusetts [u.a.]: Addison-Wesley-Longman, 1997

EISENMENGER 1997

Eisenmenger, Richard: JavaScript.- Haar bei München: Markt und Technik, Buch und Software-Verl., 1997

KNOBLOCH 1997

Knobloch, Frank: Java.- Kaarst: bhv Verlags GmbH,1997

KRÜGER 1997

Krüger, Guido: Java™ 1.1 lernen.- Bonn; Reading Mass.[u.a.]: Addison-Wesley-Longman, 1997

MASUR 1997

Masur, Klemens: Die Automatisierte Liegenschaftskarte (ALK); Beitrag in “Vermessung Brandenburg”, Nr. 1/1997,- Hrsg.: Ministerium des Innern des Landes Brandenburg

OBAK-LIEGKAT NRW

Vorschriften für die Bildung und Abbildung von Objekten der Automatisierten Liegenschaftskarte in Nordrhein-Wetsfalen (OBAK-LiegKat NRW), Entwurf ; Hrsg.: Innenministerium des Landes Nordrhein-Westfalen, 1994

RUNDSCHREIBEN LVA AUG.1995

Rundschreiben des Landesvermessungsamtes Brandenburg: Datenformate für die Abgabe von Punkt- und Grundrißdaten an die KVÄ des Landes Brandenburg; Aug. 1995

RUDOLPH 1993

Rudolph, Dietmar: Der DXF-Standard.- München: Rossipaul, 1993

SEDGEWICK 1992

Sedgewick, Robert: Algorithmen in C.- Bonn; München; Paris [u.a.]: Addison-Wesley, 1992

SCHMIDT 1997

Schmidt, Volker: Java 1.1.-Haar bei München: Markt und Technik, Buch- und Software-Verl., 1997

STEPPAN 1998

Steppan, Bernhard: Write once, debug anywhere, Portierung in der Praxis: Grenzen der Java-Systemunabhängigkeit.- Beitrag im Java-Magazin, Ausgabe 5.98

TOLKSDORF 1996

Tolksdorf, Robert: Die Sprache des Web: HTML 3.-Heidelberg: Verlag für digitalee Technologie Gmbh, 1996

VERMLIEGG

Gesetz über die Landesvermessung und das Liegenschaftskataster im Land Brandenburg – VermLiegG.- Gesetzblatt für das Land Brandenburg Teil I, Nr.1 vom 16. Januar 1998

VermLiegGZW

Vermessungs- und Liegenschaftsgesetz – Zuständigkeitsverordnung VermLiegGZW.- Gesetz- und Verordnungsblatt für das Land Brandenburg Teil II, Nr.6 vom 12. Januar 1996

www.teamone.de/selfhtml

Münz, Stefan: HTML-Dateien selbst erstellen.-http://www.teamone.de/selfhtml/.- Version 6.1 vom 15.06.1997

www.webhits.de

Webhits-Internetstatistiken.- webhits internet design gmbh, Frankfurt am Main 1998

www-cg-hci.informatik.uni-oldenburg.de/~pgse96/

Rnesv`qddqf nmml hrbgd @rodjsd adh cdq Fdrs`kstmf unm VVV,a`rhdqsdq Kdqmrnesv`qd `l Adhrohdk undqdrtmfradfkdlsdmcdq L`sdq`kdm yt !Bnl otsdq Fq`oghbr!-, Projektgruppe 96/97 Computer Grafik und Software-Ergonomie an der Carl von Ossietzky Universität Oldenburg, Nov. 1998