TECHNISCHE UNIVERSITÄT WIEN

VIENNA UNIVERSITY OF TECHNOLOGY

M A S T E R A R B E I T

Implementierung eines historischen 3D-Weltatlasses auf der Basis von X3D

ausgeführt am Institut für

Softwaretechnik und Interaktive Systeme
Arbeitsgruppe für Interaktive Multimediale Systeme
der Technischen Universität Wien

unter Anleitung von
DI Dr. Christian Breiteneder, Univ. Prof. und
Dr. Horst Eidenberger, ao. Univ. Prof.

durch

Angelika Preißler Bakk.techn.

Leystraße 26/42
A-1200 Wien

Ort, Datum

Unterschrift

Die vorliegende Arbeit beschäftigt sich mit der Umsetzung eines historischen 3D-Weltatlasses. Ein wichtiges Prinzip von Web 2.0 ist die aktive Beteiligung von Benutzern als Produzenten von Inhalten. Wikis sind ein Vertreter des Web 2.0 und stellen genau diese Funktionalität zur Verfügung. Die grafische Visualisierung erfolgt mittels X3D. Die Datenbasis wird über ein Wiki verwaltet und kann durch Benutzer erweitert und bearbeitet werden, wobei die geografischen Daten direkt in den Wikitext als XML eingebettet werden. Anhand von Links, dem Grundelement von Wikis, werden die geografischen und historischen Strukturierungen vorgenommen. Die Transformation von XML zu X3D erfolgt durch XSL. Durch die Offenheit des Systems und der Verwendung von bekannten Umgebungen wird die Grundlage für eine umfangreiche Erweiterung der historischen Datenbank geschaffen.

This diploma thesis deals with the realization of a historical 3D-Worldatlas. An important principle of Web 2.0 is the active attendance of users as producers of contents. Wikis are a part of Web 2.0 and offer exactly this functionality for users. Graphical visualization is done by X3D. The database will be administrate by a wiki and can be extended and arranged whereas the geographical data is directly embedded in the wiki text. Geographical and the historical structuring will be resolved by links, the basic elements of wikis. The transformation from XML to X3D take place by XSL. Because of the openness of the system and the usage from well known environments, a large expansion of the historical database is given.

Inhaltsverzeichnis

1 Einleitung

Der Grundgedanke vom World Wide Web war die einfache Austauschmöglichkeit von wissenschaftlichen Daten. Das WWW entwickelte sich immer mehr zur Konsumierung von Inhalten und nicht zur Produzierung dieser. Web 2.0 zeigt jedoch neue Methoden und Ideen, in denen der Inhalt von Benutzern für Benutzer verfasst wird. Auch Wikis folgen diesem Prinzip, sie bieten das technische Umfeld für die Erstellung von Artikeln. Durch die Verlinkung dieser Beiträge entsteht ein umfassendes Nachschlagewerk.

Diese Arbeit beschäftigt sich mit der Erstellung einer Applikation, mit der es möglich ist geografische Daten mit textuellen Annotationen auf einer dreidimensionalen Weltkugel zu visualisieren. Durch das Hinzufügen der zeitlichen Komponente (von Ereignissen) kann durch Navigationselemente die historische Entwicklung der Welt verfolgt werden. Die Datenbasis des Systems bildet eine Wiki-Engine, in der die Eingabe der geografischen Informationen seitens der Benutzer getätigt wird. Mittels XSL-Transformation werden diese Daten in die 3D-Modellierungssprache X3D umgewandelt. Die Visualisierung basiert auf den Verlinkungen der einzelnen Artikel. Es besteht die Möglichkeit verschiedene geografische Objekte (kurz: Geo-Objekte) wie Punkt, Linie, Fläche zu erstellen und zusätzliches Kartenmaterial zu integrieren.

Ziel dieser Arbeit ist es, ein technisches System zu schaffen, in das Wiki-Benutzer die Daten zu diesem historischen 3D-Weltatlas eingeben können. Kapitel 2 gibt eine detaillierte Übersicht über die Entwicklung und die Spezifikation von X3D. Des Weiteren wird ein kurzer Überblick über Wikis, XSL-Transformationen und geografische Daten gegeben. Das Konzept zur Erstellung der Applikation wird in Kapitel 3 vorgestellt, wobei auf die Anforderungen an das System und deren Architektur näher eingegangen wird. Die technische Umsetzung des Entwurfs wird in Kapitel 4 näher betrachtet. Die verschiedenen Seitentypen, Geo-Objekte und Na-

vigationselemente werden aufgelistet und deren Realisierung dargestellt. Kapitel 5 beinhaltet eine ausführliche Beschreibung für die Benutzung der Applikation. Es werden dabei verschiedene Anwendungsmöglichkeiten gezeigt.

Auf eine geschlechtergerechte Sprache wurde zu Gunsten der besseren Lesbarkeit verzichtet. Bei allen Formulierungen sind sowohl Frauen als auch Männer gemeint.

2 Hintergrund

Zur Einführung in die Thematik werden in diesem Kapitel die Grundlagen der verwendeten Techniken näher beschrieben. Am Beginn werden die Entwicklung von X3D und ein Vergleich mit dessen Vorläufer VRML dargestellt. Weiters wird die Architektur und X3D anhand von praktischen Beispielen betrachtet, ergänzt durch die Auflistung hilfreicher Tools und vorhandener Browser bzw. Plug-ins. Die Grundlagen einer XSL Transformation, eines Wikis und die Visualisierung von geografischen Daten sind weitere Schwerpunkte.

2.1 Extensible 3D - X3D

Für die Visualisierung der historischen Daten wird X3D verwendet. In diesem Kapitel wird näher auf die Entwicklung, Architektur und Spezifikation von X3D eingegangen. Weiters werden auch verschiedene Werkzeuge vorgestellt, die zur Erstellung von X3D-Inhalten dienen.

2.1.1 Virtual Reality Modeling Language - VRML

Bei VRML handelt es sich um ein einfaches Austauschformat für 3D-Objekte und Welten, in dem grundlegende Elemente wie Geometrien, Materialeigenschaften, Transformationen, Lichtquellen, Animation, Interaktion und Viewpoints definiert sind. Weiters kann VRML als 3D-Gegenstück zu HTML betrachtet werden, durch VRML ist es möglich dreidimensionale Webseiten zu veröffentlichen. [7] In der VRML-Spezifikation [45] wird am Beginn der Einleitung folgender Verwendungszweck definiert:

> The Virtual Reality Modeling Language (VRML) is a file format for describing interactive 3D objects and worlds. VRML is designed to

> be used on the Internet, intranets, and local client systems. VRML is also intended to be a universal interchange format for integrated 3D graphics and multimedia.

Der Anfang von VRML liegt bei der ersten World Wide Web Konferenz in Genf im Mai 1994. Dort präsentierten Marc Pesce und Toni Parisi ihren ersten Prototypen eines 3D-Browsers (*Labyrinth*). Daraufhin starteten Diskussionen auf einer Mailing List über eine mögliche Spezifikation für 3D-Darstellung im WWW. Open Inventor, entwickelt von Silicon Graphics Inc., diente als Vorlage für die erste VRML-Spezifikation. Grundlegende Elemente von OpenInventor wurden mit zusätzlichen Features erweitert, die für das World Wide Web notwendig waren. Bereits im Oktober 1994 wurden Entwürfe von VRML 1.0 auf der zweiten World Wide Web Konferenz in Chicago vorgestellt und nach weiteren 6 Monaten im April 1995 als Spezifikation verabschiedet. [7] [9]

Nach der Veröffentlichung des VRML 1.0-Standards entstanden neue Diskussionen über die Weiterentwicklung. Die Meinungen teilten sich, einige forderten eine komplette Überarbeitung, während andere nur zusätzliche Funktionalitäten verlangten. Nur in einem Punkt waren sich Anwender wie auch Entwickler einig: es fehlten wichtige Features wie Animation und Interaktion. Daraufhin wurde die *VRML Architecture Group (VAG)* gegründet, die die Koordination der Weiterentwicklung übernahm. Der Durchbruch erfolgte, als im Herbst 1995 einige Firmen erste Vorschläge für eine neue Version vorstellten. Es beteiligten sich verschiedene Unternehmen (Silicon Graphics und Sony, Microsoft, SUN, Apple, IBM) an einer vom VAG organisierten Abstimmung im WWW, welcher Vorschlag als zukünftiger Standard verwendet werden sollte. Mit großer Mehrheit (über 70%) wurde *Moving Worlds* von Silicon Graphics und Sony gewählt, gefolgt von Microsofts *ActiveVRML*. VRML 2.0 wurde im August 1996 als neue Spezifikation veröffentlicht. Das *VRML-Konsortium* (*VRMLC*) wurde kurz nach der Veröffentlichung von VRML 2.0 gegründet. SUN, SGI, Microsoft sind einige von vielen Mitgliedern. Im September 1997 wurde VRML 2.0 nach kleinen Änderungen unter dem Namen *VRML97* als ISO-Standard anerkannt. Die wichtigsten Erneuerungen zu VRML 1.0 sind die Integration von Ton, Animation, Interaktion, Scripts und dem EAI (External-Authoring Interface). [7] [9] [17]

2.1.2 Entwicklung von X3D

Der Erfolg von VRML97 war sehr gering. Die Gründe dafür lagen in den damaligen Hardware-Limitationen, der teilweisen unterschiedlichen Darstellungen der gleichen 3D-Szene durch mehrere Browser und der Syntax, die den Benutzer nicht so geläufig war, wie zum Beispiel HTML. Mit X3D sollten die Schwachstellen von VRML97 behoben werden und den Anforderungen von 3D Darstellungen im World Wide Web gerecht werden. [12] [24] Als Basis für X3D wurde *XML* (*Extensible Markup Language*) festgelegt. Durch die Integration von XML erhoffte sich das Konsortium eine bessere Interaktion mit dem WWW. [5] Das Hauptaugenmerk bei XML liegt in der Trennung von Inhalt und Form, weiters können Elemente durch eine entsprechende *DTD* (*Document Type Definition*) beschrieben werden. Diese DTD kann je nach Erweiterung ausgebaut werden. [11] Auf der Webseite des Web3D-Konsortium wird X3D wie folgt definiert: [47]

> X3D is a royalty-free open standards file format and run-time architecture to represent and communicate 3D scenes and objects using XML. It is an ISO ratified standard that provides a system for the storage, retrieval and playback of real time graphics content embedded in applications, all within an open architecture to support a wide array of domains and user scenarios.

Der Stand der Entwicklung von X3D kann unter [48] eingesehen werden, der aktuelle ISO-Standard ist. *ISO/IEC 19776:2005* X3D unterstützt derzeit 2D- und 3D-Graphik, Animation, Audio, Video, User Interaktion, Navigation, Scripting, von User definierte Objekte, Netzwerke, CAD-Daten, Physikalische Simulationen und Echtzeitkommunikation.

Aus dem VRML-Konsortium entstand das Web3D-Konsortium. [11] Dieses besteht aus verschiedenen Arbeitsgruppen, die unterschiedliche Themen bearbeiten: [46]

X3D Earth ist ein System für dreidimensionale Visualisierungen von Informationen und Objekten in Bezug auf geografische Daten.

X3D Networking befasst sich mit Netzwerkfähigkeiten.

User Interface beschäftigt sich mit der Entwicklung von Funktionalitäten im User Interface Bereich.

CAD entwickelt ein Dateiformat für die Konvertierung von CAD-Daten in ein offenes Format.

Medical befasst sich mit der Visualisierung der menschlichen Anatomie anhand verschiedener Eingabedaten.

VizSim beschäftigt sich mit dem Austausch und der Wiederverwendbarkeit von dreidimensionalen Daten für Simulation und Modellierung.

X3D Conformance Program kümmert sich um die Vergabe des Handelszeichens von X3D.

Programmable Shaders beschäftigt sich mit der Integrierung von high-level Shaders.

GeoSpatial war für die Darstellung von geografischen Daten zuständig. Diese Arbeitsgruppe wurde mittlerweile im X3D Earth integriert.

DIS-XML befasst sich mit der Entwicklung von Distributed Interactive Simulation.

H-Anim entwickelt eine Schnittstelle zwischen Software für die Modellierung und Tools, die für die Animierung zuständig sind.

X3D Source beschäftigt sich mit der Weiterentwicklung der X3D-Spezifikation.

Durch den Einsatz von XML, anstelle des Formats von VRML, besteht ein größerer Leistungsumfang durch die Verwendung der vorhandenen Standards und Tools von XML. Weiters verfügen die meisten Benutzer über günstige und leistungsstarke Grafikkarten, die eine optimale Visualisierung ermöglichen. Daraus ergeben sich gute Chancen, dass X3D erfolgreicher wird als VRML.

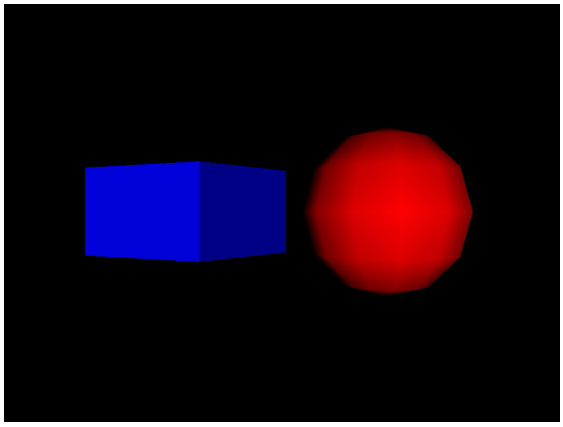

Abbildung 2.1: Zeigt die entsprechende Visualisierung zu den Codebeispielen in VRML und X3D.

2.1.3 VRML und X3D im Vergleich

Die in VRML beschriebenen 3D-Objekte werden in einer hierarchischen Struktur (Szenegraph) organisiert. Dieser Szenegraph kann sich aus verschiedenen Knoten zusammensetzen, zum Beispiel *Shapes*, *Geometry*, *Appearance*, *Material*, *Animation*, ... Die Eigenschaften eines Knoten werden in Datenfeldern definiert. Jedes dieser Felder ist von einem bestimmten Datentyp, zum Beispiel *SFFloat* oder *SFString*. [6] [18]

Quelltext 2.1 zeigt ein Beispiel einer einfachen VRML-Szene. [6] Der *Group*-Knoten ist der Wurzelknoten und beinhaltet zwei *Transform*-Knoten (*Codezeile 4 und 17*). Im 1. Knoten erfolgt eine Verschiebung der definierten *Sphere* (*Codezeile 8*) durch das Feld *translation* an die Position (4 0 0). Diese Translation hat jedoch keine Auswirkung auf den 2. Knoten, in dem eine *Box* (*Codezeile 22*) definiert ist. In diesem Knoten erfolgen eine Verschiebung (durch *translation*) und eine Rotation (durch *rotation*). Jedem dieser *Transform*-Knoten ist ein *Shape*-Knoten untergeordnet. Die Felder *geometry* und *appearance* sind den Knoten *Sphere* bzw. *Appearance* zugeteilt, letzterer beinhaltet einen weiteren Knoten *Material*. Durch die Definition der Felder *radius* (*Codezeile 9*), *size* (*Codezeile 23*) und *diffuseColor* (*Codezeile 13 und 27*) werden der Radius der Kugel, die Größe der Box und die Farben der Objekte bestimmt. In Abbildung 2.1 wird das zuvor beschriebene Beispiel dargestellt.

```
#VRML V2.0 utf8
Group {
  children [
    Transform {
      translation 4 0 0
      children [
        Shape {
          geometry Sphere {
            radius 3.0
          }
          appearance Appearance {
            material Material {
              diffuseColor 1 0 0
        }}}
      ]
    }
    Transform {
      rotation 0 1 0 45
      translation -3 0 0
      children [
        Shape {
          geometry Box {
            size 5 3 5
          }
          appearance Appearance {
            material Material {
              diffuseColor 0 0 1
        }}}
      ]
    }
  ]
}
```

Quelltext 2.1: VRML-Codebeispiel.

Quelltext 2.2 zeigt dasselbe Beispiel (Quelltext 2.1 bzw. Abbildung 2.1) in X3D. Im Gegensatz zu VRML wird jeder Knoten als Element dargestellt und beinahe jedes Datenfeld als Attribut übernommen. [18] Am Beginn des X3D-Dokuments stehen die XML-Deklaration (*Codezeile 1*) und eine optionale Document Type Definition (*Codezeile 2*). In *Codezeile 3* wird das Element *X3D* mit dessen Attributen beschrieben, wobei ein entsprechendes XML-Schema und im Attribut *profile* eines der Profile angegeben wird. Anschließend wird das Element *head* definiert, indem verschiedene Metadaten (*Codezeile 5-7*) angeführt werden können. Wie zum Beispiel der Dateiname, Beschreibung der Datei und Autor. [13] Im Element *Scene* (*Codezeile 9-28*) erfolgt die eigentliche Definition der Darstellung. Wie im Beispiel zuvor sind dem Element *Group* zwei *Transform*-Elemente (*Codezeile 11 und 19*) untergeordnet, mit den Attributen *translation* und *rotation* wird die Translation und die Rotation definiert. Jedem *Transform*-Element sind je ein *Shape*-Element (*Codezeile 12 und 20*) untergeordnet, in denen sind weiters *Appearance*-Elemente (*Codezeile 14 und 22*) und einmal ein *Sphere*-Elemente (*Codezeile 13*) und ein *Box*-Elemente (*Codezeile 21*) zugeordnet. Den *Appearance*-Elementen sind dann je ein *Material*-Element (*Codezeile 15 und 23*) zugeteilt.

In der Spezifikation [51] werden Elemente aber weiterhin als Knoten bezeichnet und Attribute als Datenfelder. [18] Sowohl VRML als auch X3D basieren auf einem Szenegraph, siehe Abbildung 2.2, und beide benötigen einen eigenen Browser oder ein Browser-Plug-in für die Darstellung der 3D-Szene. Aufgrund der Syntax von *XML* bietet X3D eine kompaktere Schreibweise als VRML, das wegen der teilweisen tiefen Verschachtelungen nicht sehr gut lesbar ist. Ein weiterer Vorteil von X3D ist die sofortige Überprüfung der Gültigkeit des Dokuments durch die Validierung mit einer DTD. [18]

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE X3D PUBLIC "http://www.web3d.org/specifications/x3d-3.1.dtd" "file:///www.web3d.org/TaskGroups/x3d/translation/x3d-3.1.dtd">
<X3D profile="Immersive" version="3.0" xmlns:xsd="http://www.w3.org/2001/XMLSchema-instance" xsd:noNamespaceSchemaLocation="http://www.web3d.org/specifications/x3d-3.1.xsd">
  <head>
    <meta content="blueBoxRedSphere.x3d" name="filename"/>
    <meta content="simple X3D example" name="description"/>
    <meta content="Angelika Preissler" name="creator"/>
  </head>
  <Scene>
    <Group>
      <Transform translation="4 0 0">
        <Shape>
          <Sphere radius="3.0"/>
          <Appearance>
            <Material diffuseColor="1 0 0"/>
          </Appearance>
        </Shape>
      </Transform>
      <Transform rotation="0 1 0 45" translation="-3 0 0">
        <Shape>
          <Box size="5 3 5"/>
          <Appearance>
            <Material diffuseColor="0 0 1"/>
          </Appearance>
        </Shape>
      </Transform>
    </Group>
  </Scene>
</X3D>
```

Quelltext 2.2: X3D-Codebeispiel.

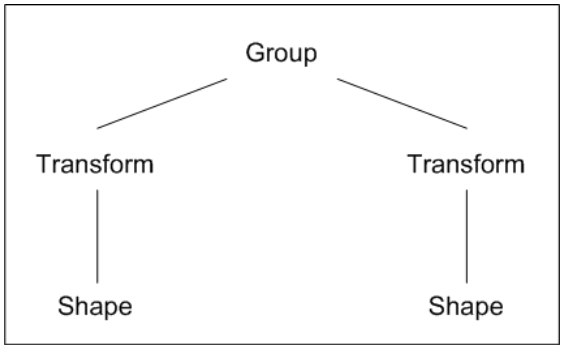

Abbildung 2.2: Zeigt den dazugehörigen Szenegraph zu den Codebeispielen in VRML und X3D.

2.1.4 Architektur und Komponenten von X3D

Jede X3D-Applikation beinhaltet grafische und akustische Objekte. Diese werden über ein Netzwerk geladen und können dynamisch modifiziert werden. Die X3D Semantik beschreibt eine abstrakte funktionale Verhaltensweise von interaktiven, auf Zeit basierenden 3D-Informationen. In Abbildung 2.3 wird die X3D-Architektur dargestellt. X3D definiert keine physikalischen Geräte oder andere Konzepte, die von Implementationen abhängig wären, wie Eingabegeräte und Bildschirmauflösung. Das heißt, es wird zum Beispiel die Existenz einer Maus nicht vorausgesetzt. Durch X3D kann die Verhaltensweise von Objekten definiert werden und es bietet die Möglichkeit, eine Verbindung zu externen Applikationen mittels Programmier- und Scriptsprachen zu erstellen, siehe Spezifikation. [51] [13]

Im Gegensatz zu VRML ist der Aufbau von X3D modular. Die Spezifikation von X3D besteht aus 24 *Komponenten*, die verwandte Funktionalitäten zusammenfassen. Jede Komponente ist weiters durch *Ebenen* unterteilt. Mit Hilfe der Ebenen wird definiert, inwieweit eine Komponente unterstützt wird. Bestehende Komponenten können durch Hinzufügen von neuen Ebenen modifiziert oder erweitert werden bzw. können durch die Erweiterung von neuen Fähigkeiten neue Komponenten generiert werden. [13] [47] [18]

Tabelle 2.1 stellt die Komponente Geometry3D und ihre 4 Ebenen dar. In der ersten Ebene sind die Grundelemente spezifiziert, wie *Box*, *Kegel*, *Zylinder* und *Kugel*. Ebene 2 definiert alle Knoten der Ebene 1 und zusätzlich einen neuen Knoten *IndexedFaceSet*. Ebene 3 beinhaltet neben den Knoten von Ebene 1 und 2

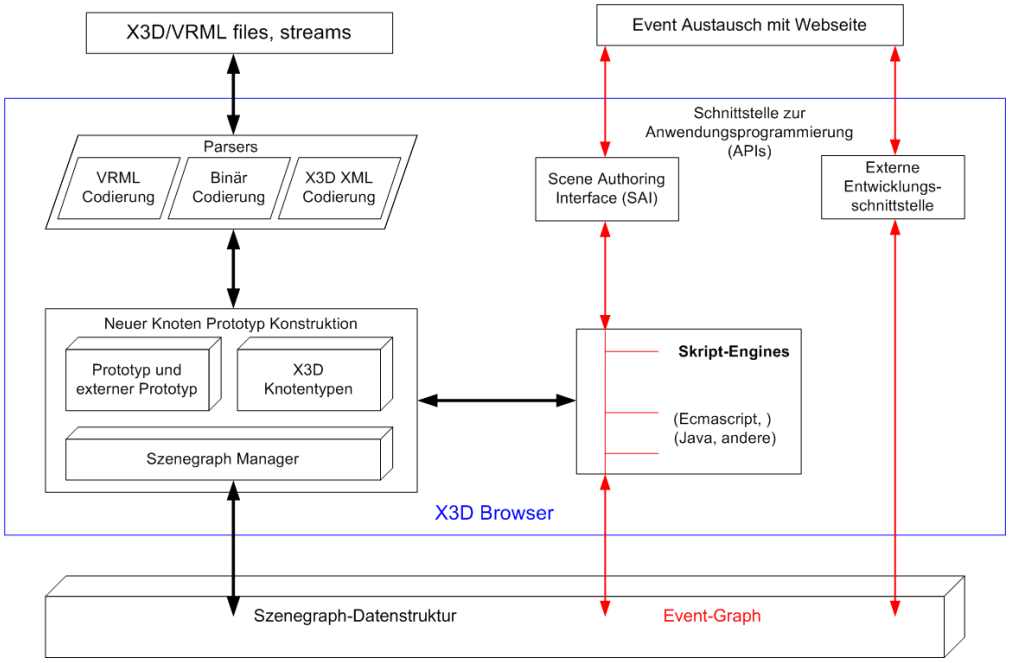

Abbildung 2.3: X3D-Architektur. [51]

Ebene	Knoten/Features
1	Box, Cone, Cylinder, Sphere
2	Alle Geometrie-Knoten von Ebene 1 + IndexedFaceSet
3	Alle Geometrie-Knoten von Ebene 2 + ElevationGrid
4	Alle Geometrie-Knoten von Ebene 3 + Extrusion

Tabelle 2.1: Geometry3D Komponente und die dazugehörigen Ebenen.

den Knoten *ElevationGrid* und in Ebene 4 wird der Knoten *Extrusion* hinzugefügt. [51] [18]

Aus den bestehenden Komponenten werden *Profile* zusammengestellt. Es wird zwischen sechs Profilen unterschieden. Diese sind unterteilt in vier Basis-Profile, siehe Abbildung 2.4: [47]

Interchange definiert die Kommunikation zwischen den Applikationen. Geometrie, Texturierung, grundlegende Beleuchtung und Animation werden in diesem Profil unterstützt.

Interactive ermöglicht ein Navigieren des Benutzers und somit eine Interaktion

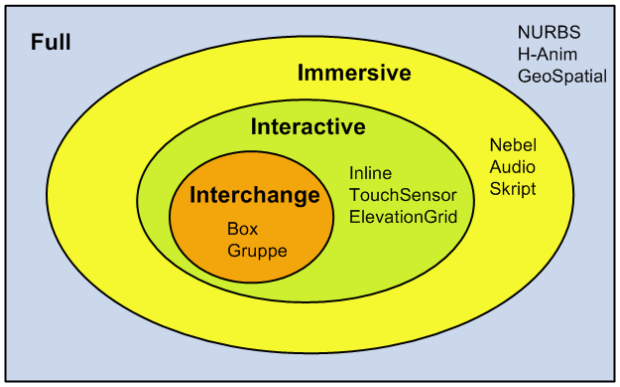

Abbildung 2.4: Die vier Basis-Profile.

mit einer 3D-Umgebung durch Hinzufügen von verschiedenen Sensoren (z.B. TouchSensor). Weiters bietet dieses Profil zusätzliche Beleuchtungsmöglichkeiten und eine erweiterte Zeitsteuerung.

Immersive bietet sowohl Unterstützung für Audio, Nebel, Kollision, Interaktion als auch eine vollständige 3D Graphik.

Full beinhaltet die Implementierung aller Knoten, inklusive GeoSpatial, NURBS, H-Anim.

und zwei zusätzlichen Profilen:

MPEG-4 Interactive unterstützt Broadcasting und Mobilgeräte.

CDF (CAD Distillation Format) befindet sich noch in der Entwicklung. Gedacht ist dieses Profil für die Übersetzung von CAD-Daten in ein offenes Format.

Der Vorteil dieser modularen Architektur ist die Möglichkeit neue Komponenten zu integrieren, ohne damit die gesamte Spezifikation zu beeinträchtigen. [47] X3D ist dadurch wesentlich einfacher zu erweitern als VRML97. Bei der Anwendung müssen nicht alle vorhandenen Funktionalitäten verwendet werden. Durch den Einsatz von Profilen, Komponenten und Ebenen lassen sich, je nach Bedarf, jeweils die erforderlichen Features einbinden.

2.1.5 X3D-Werkzeuge

Die Erstellung von dreidimensionalen Inhalten in X3D wird durch eine Vielzahl von Tools unterstützt. Diese reichen von speziellen Editoren, Browsern, Plug-ins bis zu Export-Plug-ins für Modellierungs-Software. [52]

Editor

X3D-Edit [49] ist ein Editor mit grafischer Benutzeroberfläche. Durch diesen X3D-Editor ist es möglich, ein X3D-Dokument sehr einfach zu erstellen und gleichzeitig auch das Dokument zu validieren. Zusätzlich werden Tooltips angeboten, die für jeden Knoten eine kompakte Zusammenfassung bieten und somit sowohl Anfänger als auch fortgeschrittene Anwender unterstützen. X3D-Edit basiert auf Java und Xeena, einem XML-Editor von IMB. [13]

Im Hauptfenster wird das X3D-Dokument in Form eines Szenegraphs dargestellt. Links davon werden sämtliche Knoten aufgelistet, die X3D zu Verfügung stellt. Je nachdem welches Profil selektiert wurde, sind die entsprechenden Knoten aktiviert oder deaktiviert. Nach Auswahl eines Knotens werden links unten die jeweiligen Attribute des Knoten eingeblendet. Zusätzliche Funktionalitäten bietet X3D-Edit mit der Konvertierung von X3D in ein anderes Format mittels XSLT, z.B.: VRML, HTML und die Möglichkeit ein X3D-Dokument in der XML-Syntax zu betrachten. Es wird jedoch keine direkte Modifizierung der XML Elemente angeboten. Vorteile von X3D-Edit sind die einfache Erstellung von Szenen, sowie die Plattformunabhängigkeit durch die Verwendung von Java.

Browser bzw. Plug-ins

Um eine 3D-Welt in X3D darstellen zu können, ist entweder ein Browser wie *Xj3D* oder ein Plug-in für einen Webbrowser erforderlich. Eine Auflistung von möglichen Browsern und Plug-ins ist unter [52] zu finden.

Die Firma *Bitmanagement Software GmbH* bietet die Plug-ins *BS Contact VRML/X3D 7.0* und *BS Contact Geo* an. Im BS Contact Geo wird neben den Funktionalitäten von BS Contact VRML/X3D 7.0 noch zusätzlich die *Geospatial*-Komponente von X3D teilweise unterstützt. Bei beiden Plug-ins handelt es sich um kommerzielle Produkte, es werden aber kostenlose Testversionen angeboten. [4]

Software	Type	OS			Browser		X3D
		Win	Linux	Mac	IE	Firefox	
Cortona	P	X		X	X	X	
Octaga Player	P,S	X	X		X	X	X
Flux Player	P,S,T	X			X	X	X
BS Contact	P	X			X	X	X
blaxxun Contact	P	X			X		
Cosmo Player	P	X				X	
FreeWRL	P,S,T		X	X		X	X
OpenVRML	P,S,T	X	X	X		X	X
Xj3D	S,T	X	X	X			X
Carina	S	X	X	X			X
Demotride	S	X					X
blaxxun3D	A	X	X	X	X	X	
Typen: P=Plug-in, S=standalone Programm, T=Toolkit, A=Applet							

Tabelle 2.2: Übersicht von X3D-Plug-ins, Applets, Toolkits und Browser. [35]

Der *Flux Player 2.0* wird von der Firma *Media Machines* entwickelt. Es handelt sich hier ebenfalls um ein Plug-in für Webbrowser. Im *Flux Player* werden die X3D-Szenen durch ActiveX control geladen, die Verbindung zu Elementen anderer Webseiten erfolgt mittels XML/DOM. [25] [52]

Xj3D ist eine Referenzimplementierung des Web3D-Konsortiums. Die Idee war ein Toolkit für X3D und VRML97 in Java zu implementieren. In diesem Toolkit ist auch ein eigener Viewer enthalten. [53] [52] Die Hauptziele für die Entwicklung von *Xj3D* waren: [23]

- die Entdeckung und Identifizierung von Problemen bei der Entwicklung der X3D-Spezifikation
- eine Plattform für die Weiterentwicklung zu bieten
- zu zeigen, dass alle Komponenten der Spezifikation in einem Browser implementiert werden können.

Tabelle 2.2 zeigt eine Übersicht verschiedener X3D-Plug-ins, Applets, Toolkits und Browser.

2.1.6 X3D-Spezifikation mit Beispielen

X3D verwendet ein kartesisches, rechtshändiges, dreidimensionales Koordinatensystem. Der Betrachter blickt von der z-Achse in Richtung Ursprung, der negativen z-Achse entgegen. Die y-Achse zeigt nach oben und die x-Achse nach rechts. Das Längenmaß des Weltkoordinatensystems ist Meter, Winkel werden im Bogenmaß und die Zeit in Sekunden angegeben. Als Farbraum wird das RGB-Farbsystem (Rot, Grün, Blau) verwendet. [51]

In den nachfolgenden Punkten werden einige X3D-Knoten näher betrachtet. Die verwendete Spezifikation kann unter [51] nach gelesen werden.

3D-Objekte

Ein Objekt, oder eine Welt, wird zumeist im Knoten *Shape* beschrieben. Dieser besteht aus einer Verknüpfung eines *Geometrie*- und *Appearance*-Knotens, welche die Form und das Aussehen beschreiben. Mittels des Knotens *Geometrie* lassen sich zweidimensionale Objekte, grundlegende dreidimensionale Objekte (*Box*, *Cone*, *Cylinder*, *Sphere*) und komplexere 3D-Objekte (*ElevationGrid*, *IndexedFaceSet*, *NURBS*) definieren. Alle dreidimensionalen Geometrie-Knoten beinhalten das Datenfeld *solid*. Durch dieses Feld wird bestimmt ob eine Geometrie sichtbar ist, wenn sich die Position des Betrachters innerhalb der Geometrie befindet.

Der Knoten *Box*, siehe Quelltext 2.3, definiert einen Quader. Mit dem Feld *size* wird die Größe bestimmt. Das Beispiel wird in Abbildung 2.5 dargestellt.

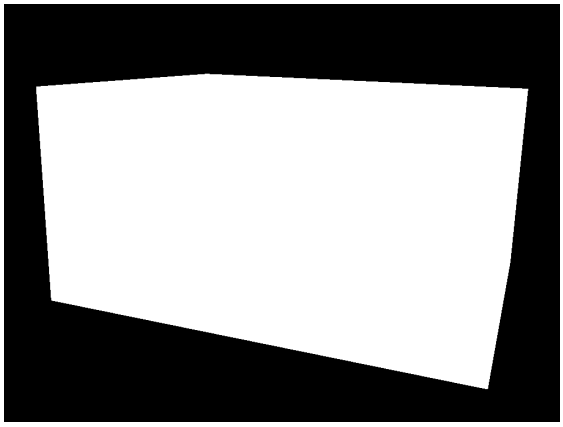

Abbildung 2.5: Zeigt die entsprechende Visualisierung der Box.

```
Box : X3DGeometryNode {
  SFNode   [in,out] metadata NULL   [X3DMetadataObject]
  SFVec3f  []       size     2 2 2  (0,infinite)
  SFBool   []       solid    TRUE
}

Beispiel:
<Scene>
  <Shape>
    <Box size="4 2 3"/>
  </Shape>
</Scene>
```

Quelltext 2.3: Spezifikation des Knotens *Box* mit Beispiel.

Der Knoten *Cone*, siehe Quelltext 2.4, definiert einen Kegel. Mit dem Feld *bottomRadius* wird der Radius der Grundfläche, mit *height* die Höhe des Kegels bestimmt. Über *side* und *bottom* kann definiert werden, ob der Mantel bzw. die Grundfläche sichtbar sein sollen. In Abbildung 2.6 wird die Visualisierung des Beispiels gezeigt.

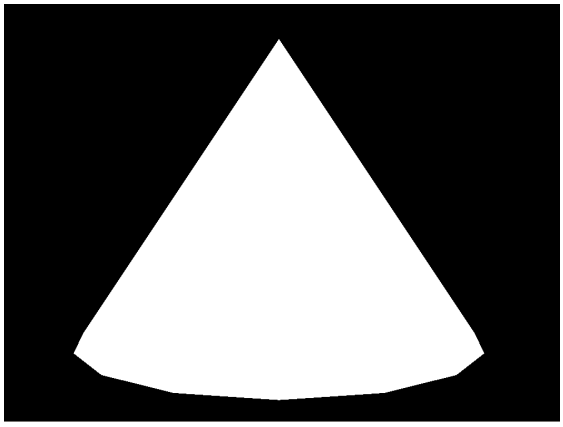

Abbildung 2.6: Zeigt die entsprechende Visualisierung des Kegels.

```
Cone : X3DGeometryNode {
  SFNode   [in,out] metadata     NULL [X3DMetadataObject]
  SFBool   []       bottom       TRUE
  SFFloat  []       bottomRadius 1    (0,infinite)
  SFFloat  []       height       2    (0,infinite)
  SFBool   []       side         TRUE
  SFBool   []       solid        TRUE
}

Beispiel:
<Scene>
  <Shape>
    <Cone bottomRadius="2" height="3"/>
  </Shape>
</Scene>
```

Quelltext 2.4: Spezifikation des Knotens *Cone* mit Beispiel.

Der Knoten *Cylinder*, siehe Quelltext 2.5, definiert einen Zylinder. Mit dem Feld *radius* wird der Radius und mit *height* die Höhe des Zylinders bestimmt. Der Zylinder besteht aus drei Teilen: *top*, *side* und *bottom*. Über den entsprechenden Feldern kann definiert werden, ob der Mantel, bzw. die obere, oder die untere

Grundfläche sichtbar sein sollen. Das Beispiel des Zylinders wird in Abbildung 2.7 visualisiert.

```
Cylinder : X3DGeometryNode {
  SFNode  [in,out] metadata NULL [X3DMetadataObject]
  SFBool  []       bottom   TRUE
  SFFloat []       height   2     (0,infinite)
  SFFloat []       radius   1     (0,infinite)
  SFBool  []       side     TRUE
  SFBool  []       solid    TRUE
  SFBool  []       top      TRUE
}

Beispiel:
<Scene>
  <Shape>
    <Cylinder height="2.5" radius="1"/>
  </Shape>
</Scene>
```

Quelltext 2.5: Spezifikation des Knotens *Cylinder* mit Beispiel.

Der Knoten *Sphere*, siehe Quelltext 2.6, definiert eine Kugel. Mit dem Feld *radius* wird der Radius der Kugel bestimmt. Das Beispiel wird in Abbildung 2.8 dargestellt.

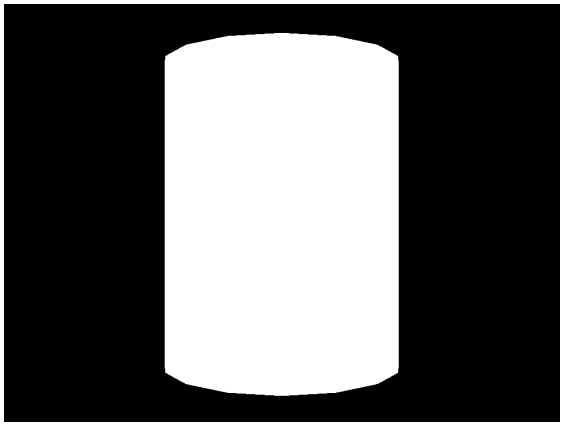

Abbildung 2.7: Zeigt die entsprechende Visualisierung des Zylinders.

```
Sphere : X3DGeometryNode {
  SFNode   [in,out] metadata NULL [X3DMetadataObject]
  SFFloat  []       radius   1    (0,infinite)
  SFBool   []       solid    TRUE
}

Beispiel:
<Scene>
  <Shape>
    <Sphere radius="3.0"/>
  </Shape>
</Scene>
```

Quelltext 2.6: Spezifikation des Knotens *Sphere* mit Beispiel.

Über den Knoten *ElevationGrid* lässt sich ein einheitliches Gitter mit unterschiedlichen Höhen definieren. Es ist sehr gut geeignet für die Gestaltung von Terrains.

Der Knoten *IndexedFaceSet* definiert einen dreidimensionalen Körper, der aus einzelnen Flächen konstruiert ist. Diese Flächen sind wiederum aus einer Menge von Punkten zusammengesetzt. Mittels *IndexedFaceSet* ist es mögliche freie dreidimensionale Formen darzustellen. [1]

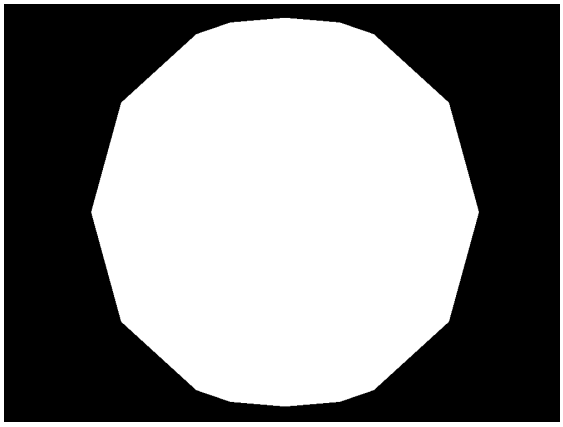

Abbildung 2.8: Zeigt die entsprechende Visualisierung der Kugel.

Material und Beleuchtung

Mittels des *Appearance*-Knotens wird das Aussehen eines Objektes beschrieben. Mit den untergeordneten Knoten *Material* und *ImageTexture* lassen sich die Oberflächeneigenschaft und die Texturierung eines Objektes definieren.

Der Knoten *Material*, siehe Quelltext 2.7, spezifiziert Materialeigenschaften. Zum Beispiel definiert das *diffuseColor*-Feld die Farbe des Objektes und *transparency* bestimmt, wie transparent ein Objekt ist. Der Wert *1* bedeutet komplett transparent und *0* vollkommen undurchsichtig. In Abbildung 2.9 wird das Beispiel dargestellt.

```
Material : X3DMaterialNode {
  SFFloat [in,out] ambientIntensity 0.2          [0,1]
  SFColor [in,out] diffuseColor     0.8 0.8 0.8  [0,1]
  SFColor [in,out] emissiveColor    0 0 0        [0,1]
  SFNode  [in,out] metadata         NULL         [
    X3DMetadataObject]
  SFFloat [in,out] shininess        0.2          [0,1]
  SFColor [in,out] specularColor    0 0 0        [0,1]
  SFFloat [in,out] transparency     0            [0,1]
}

Beispiel:
<Scene>
  <Shape>
    <Sphere radius="3"/>
    <Appearance>
      <Material diffuseColor="0 1 0"/>
    </Appearance>
  </Shape>
</Scene>
```

Quelltext 2.7: Spezifikation des Knotens *Material* mit Beispiel.

Durch den *ImageTexture*-Knoten, siehe Quelltext 2.8, ist es möglich, eine Textur auf eine Geometrie abzubilden. Die Textur wird definiert durch die Angabe eines Pfades (*url*) zu einem Bild. Das Beispiel wird in Abbildung 2.10 dargestellt.

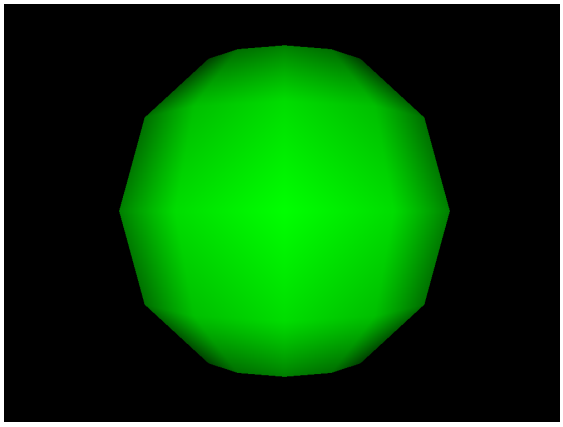

Abbildung 2.9: Zeigt die entsprechende Visualisierung des Material-Knotens.

```
ImageTexture : X3DTexture2DNode {
  SFNode    [in,out] metadata          NULL [X3DMetadataObject]
  MFString  [in,out] url                 []    [urn]
  SFBool    []       repeatS             TRUE
  SFBool    []       repeatT             TRUE
  SFNode    []       textureProperties   NULL [
    TextureProperties]
}

Beispiel:
<Scene>
  <Shape>
    <Sphere radius="3.0"/>
    <Appearance>
      <ImageTexture url="earth.jpg"/>
    </Appearance>
  </Shape>
</Scene>
```

Quelltext 2.8: Spezifikation des Knotens *ImageTexture* mit Beispiel.

Durch die Knoten *DirectionalLight*, *PointLight* und *SpotLight* spezifiziert X3D

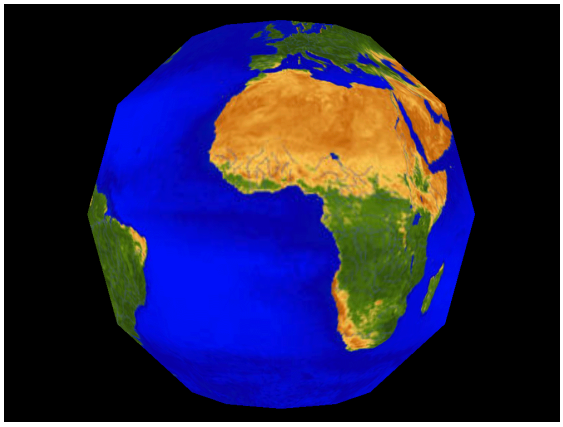

Abbildung 2.10: Zeigt die entsprechende Visualisierung des ImageTexture-Knotens.

unterschiedliche Möglichkeiten zur Beleuchtung. Für nähere Details siehe [51].

Transformation

Der Knoten *Transform*, siehe Quelltext 2.9, gehört zum *Group*-Knoten und definiert für seine Unterknoten ein Koordinatensystem, relativ zum Koordinatensystem des übergeordneten Knotens. Mittels der Datenfelder *translation*, *rotation* und *scale* sind eine Rotation, Verschiebung und Skalierung spezifiziert. Die Verschiebung ist definiert durch einen dreidimensionalen Vektor, der das Objekt entlang der x-, y- und z-Achse verschiebt. Bei der *rotation* bestimmen die ersten drei Werte die Rotationsachse und der vierte Wert den Rotationswinkel. Bei *scale* erfolgt durch die Angabe eines dreidimensionalen Vektors eine Skalierung entlang der x-, y- und z-Achse. Werden drei gleiche Werte verwendet, bleibt die Proportion des Objektes erhalten. [24] In Abbildung 2.11 wird das Beispiel zur Transformation dargestellt.

```
Transform : X3DGroupingNode {
  MFNode      [in]      addChildren             [X3DChildNode]
  MFNode      [in]      removeChildren          [X3DChildNode]
  SFVec3f     [in,out]  center            0 0 0 (-inf,inf)
  MFNode      [in,out]  children          []    [X3DChildNode]
  SFNode      [in,out]  metadata          NULL  [
     X3DMetadataObject]
  SFRotation  [in,out]  rotation          0 0 1 0  [-1,1] or
     (-inf,inf)
  SFVec3f     [in,out]  scale             1 1 1    (-inf,inf)
  SFRotation  [in,out]  scaleOrientation 0 0 1 0  [-1,1] or
     (-inf,inf)
  SFVec3f     [in,out]  translation       0 0 0    (-inf,inf)
  SFVec3f     []        bboxCenter        0 0 0    (-inf,inf)
  SFVec3f     []        bboxSize          -1 -1 -1 [0,inf) or
     -1 -1 -1
}

Beispiel:
<Scene>
  <Shape>
    <Box/>
    <Appearance>
      <Material diffuseColor="1 0 0"/>
    </Appearance>
  </Shape>
  <Transform scale="1.8 1.8 1.8" translation="-4 0 0">
    <Shape>
      <Sphere/>
      <Appearance>
        <Material diffuseColor="0 1 0"/>
      </Appearance>
    </Shape>
  </Transform>
```

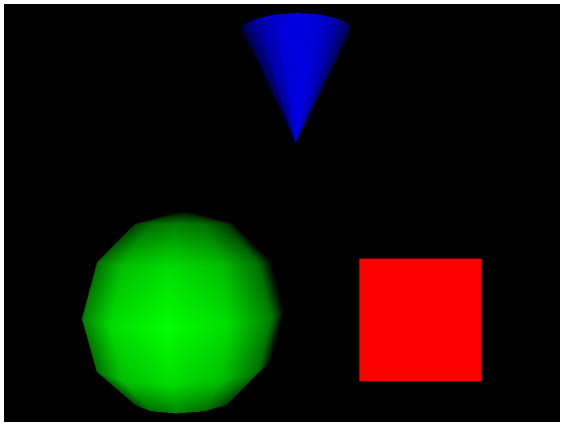

Abbildung 2.11: Zeigt die entsprechende Visualisierung des Transform-Knotens.

```
  <Transform rotation="0 0 1 3.1415" translation="-2 4 0">
    <Shape>
      <Cone bottomRadius="1" height="2"/>
      <Appearance>
        <Material diffuseColor="0 0 1"/>
      </Appearance>
    </Shape>
  </Transform>
</Scene>
```

Quelltext 2.9: Spezifikation des Knotens *Transform* mit Beispiel.

Navigation

Der Knoten *Navigationinfo* definiert die Möglichkeiten, die sich dem Benutzer bei der Navigation bieten. Es werden unterschiedliche Arten von Navigation angeboten: *ANY*, *WALK*, *EXAMINE*, *FLY*, *LOOKAT*, und *NONE*, wobei die Kombination aus *EXAMINE ANY FLY* die beste Einstellung bietet.

Durch den *Viewpoint*-Knoten können vordefinierte Standpunkte erstellt werden, die dem Benutzer eine sehr schnelle und angenehme Navigation durch eine Welt ermöglichen. [13]

Mit dem *LOD*-Knoten können von einem Objekt unterschiedlich detaillierte Versionen definiert werden. Je nach Abstand vom Benutzer zum Objekt werden die verschiedenen Varianten eines Gegenstandes eingeblendet. Diese werden innerhalb von *LOD* aufgelistet. Die Angabe der Objekte erfolgt von sehr detaillierter bis weniger detaillierter Darstellung. Mit dem Attribut *range* wird die Distanz definiert, in der der Wechsel zwischen den unterschiedlichen Detaillierungsgraden erfolgt, siehe Quelltext 2.10.

```
LOD : X3DGroupingNode {
  MFNode  [in]      addChildren           [X3DChildNode]
  MFNode  [in]      removeChildren        [X3DChildNode]
  MFNode  [in,out]  children      []      [X3DChildNode]
  SFNode  [in,out]  metadata      NULL    [X3DMetadataObject]
  SFVec3f []        bboxCenter    0 0 0   (-inf,inf)
  SFVec3f []        bboxSize      -1 -1 -1 [0,inf) or -1 -1 -1
  SFVec3f []        center        0 0 0   (-inf,inf)
  MFFloat []        range         []      [0,inf) or -1
}
```

Quelltext 2.10: Spezifikation des Knotens *LOD*.

X3D Event Model

Eine dreidimensionale Szene kann auf unterschiedliche Art und Weise beeinflusst werden. Entweder durch die Interaktion eines Benutzers mit der 3D-Welt (wird registriert durch Sensoren) oder durch vordefinierte, Zeit gesteuerte Aktionen (Interpolatoren). Als Basis dient das *X3D Event Model*, indem eine Nachricht von einem Knoten zu einem anderen Knoten gesendet wird. Realisiert wird dieses Prinzip durch *ROUTEs*. Mit einer *ROUTE* ist es möglich Verbindungen zwischen Knoten herzustellen. So kann ein Event vom Datenfeld eines Knotens zum Datenfeld eines anderen Zielknotens geschickt werden, womit sich der Zustand des Knotens ändert. X3D spezifiziert vier Zugriffsarten auf die Datenfelder eines Knotens:

initializeOnly der Wert eines Datenfeldes kann zur Laufzeit nicht geändert werden.

inputOnly der Empfang eines Events ist möglich, jedoch sind keine Leserechte vergeben.

outputOnly das Senden eines Events ist möglich, jedoch sind keine Schreibrechte vergeben.

inputOutput sowohl der Empfang als auch das Senden von Events ist möglich.

Das *ROUTE* Statement, siehe Quelltext 2.11, kann überall im X3D-Dokument definiert werden. Das Feld <*fromNodeName*> gibt den Namen des Knotens an, der einen Event generiert und <*fromFieldName*> den Namen des Datenfeldes. Dieses Feld beinhaltet den zu übermittelten Wert. Mit <*toNodeName*> und <*toFieldName*> werden der Knoten, der den Event erhält, und das entsprechende Datenfeld spezifiziert. [13] [18]

```
ROUTE <fromNodeName> <fromFieldName> <toNodeName>
      <toFieldName>
```

Quelltext 2.11: Spezifikation von *ROUTE*.

Durch *Sensoren* werden Aktionen registriert, die von Benutzern getätigt werden. [18] Die Übermittlung der Werte erfolgt über eine *ROUTE*. In X3D werden verschiedene Typen von Sensoren spezifiziert: *Pointing device sensors*, *Key device senors*, *Environmental sensor* und *Picking sensor component*. Mittels *TimeSensor* ist es möglich periodische Aktivitäten, Animationen oder auf Zeit basierende Events zu kontrollieren. So kann über diesen Sensor ein Zeitintervall in einer Schleife wiederholt, gestoppt oder gestartet werden. [5] Die Position und den Zustand eines *pointing devices* (z.B. Maus) wird durch den *TouchSensor* getrackt, dieser lokalisiert ob der Betrachter auf ein Objekt zeigt. Dem Datenfeld *isOver* wird der Wert *TRUE* zugewiesen, wenn auf ein Objekt mittels Eingabegerät gezeigt wird. Der Zustand wird auf *FALSE* gesetzt, wenn sich die Maus wieder vom Objekt wegbewegt.

Mit *Interpolatoren* ist es möglich Zeit gesteuerte Animationen zu erhalten. Durch die Interpolation von vordefinierten Schlüsselwerten (*key values*) werden die erforderlichen Zwischenwerte berechnet. [24] X3D spezifiziert verschiedene *Interpolatoren*, zum Beispiel *ColorInterpolator*, *OrientationInterpolator*, *PositionInterpolator*.

Um eine Rotation zu erzeugen, wird der Knoten *OrientationInterpolator* verwendet. Die Interpolation erfolgt durch die Berechnung des kürzesten Wegs zwischen den Werten. Der *PositionInterpolator* wird verwendet, um Positionen innerhalb einer Animation zu berechnen.

Durch den Knoten *Script* ist die Einbindung von Programmen möglich, die in einer anderen Programmiersprache geschrieben wurden. [24] Dabei kann der Programmteil direkt im X3D-Dokument integriert werden oder extern in einer Datei. [18] Mit *SAI* (*Scene access interface*) wird ein Programm Interface angeboten, das den Zugriff auf den Szenegraphen zur Laufzeit ermöglicht. SAI bietet eine API für interne Skripts und externe Programme. [50]

Geospatial

Die Komponente *Geospatial* bietet Unterstützung für die Darstellung von geografischen Daten mit Bezug auf andere Informationen. X3D unterstützt hier unter anderem die Angabe von Koordinaten oder die Modellierung von Terrains.

Mit *GeoCoordinate* ist die Definition von Koordinaten möglich, die im Feld *coord* von Knoten, wie *IndexedFaceSet*, *IndexLineSet* und *PointSet*, verwendet werden. Mittels *geoSystem* wird ein räumliches Bezugssystem definiert, wie GD - Geodetic spatial reference frame (latitude/longitude), UTM - Universal Transverse Mercator, GC - Earth-fixed Geocentric. Durch *GeoElevationGrid* ist die Spezifizierung eines einheitlichen Gitters von Höhenwerten gegeben. Diese Werte werden in eine geozentrische, gekrümmte Darstellung der Erde umgewandelt. Damit ist es möglich Koordinaten durch Breitengrad, Längengrad und durch Höhe darzustellen. Die Positionierung eines Objektes auf dieser Erde ist mittels *GeoLocation* spezifiziert. *GeoLOD* ermöglicht es unterschiedlich detaillierte Ebenen von Objekten einzusetzen. Der Grad der Genauigkeit hängt davon ab, wie nahe der Betrachter sich befindet. Somit ist es möglich Texturen für die Erde mit unterschiedlich hoher Auflösung zu verwenden. Je weiter sich der Benutzer einzoomt, umso detailliertere Texturen werden eingesetzt. Die Beschreibung von *Geospatial*-Knoten wird von *GeoMetadata* unterstützt. Mittels *GeoOrigin* kann von geografischen Koordinaten in das lokale kartesische Koordinatensystem konvertiert werden. *GeoPositionInterpolator* ermöglicht eine Interpolation von geografischen Koordinaten. Der *GeoProximity-*

Sensor generiert Events, sobald der Betrachter einen Bereich betritt bzw. verlässt oder sich in diesem bewegt. *GeoTouchSensor* bietet dieselben Funktionalitäten wie *TouchSensor*. Zusätzlich werden jedoch auch die Koordinaten retourniert. Durch *GeoViewpoint* ist es möglich Viewpoints durch Längen- und Breitengrade zu definieren.

2.2 Extended Stylesheet Language Transformation - XSLT

XSLT bezeichnet eine Sprache, mit der ein XML-Dokument in ein anderes XML-Dokument umgewandelt werden kann. XSLT ist ein Teilbereich von *XSL (Extensible Stylesheet Language)*, einer Stylesheet Sprache für XML. In der XML-Datei wird der Inhalt definiert, für die Darstellung ist ein XSL-Dokument erforderlich. In der XSL-Datei sind Regeln für die Transformation festgelegt. [43] [44]

Quelltext 2.12 zeigt ein XML-Dokument, das mittels XSL (siehe Quelltext 2.13) in eine X3D-Datei (siehe Quelltext 2.14) transformiert wird.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<?xml-stylesheet type="text/xsl" href="transform.xsl"?>
<kugel>
  <verschiebung>4 0 0</verschiebung>
  <radius>2.0</radius>
  <farbe>0 0 1</farbe>
  <beschreibung>Kugel</beschreibung>
</kugel>
```

Quelltext 2.12: Beispiel eines XML-Dokuments.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org
   /1999/XSL/Transform">
  <xsl:output method="xml" version="1.0"
    encoding="UTF-8" .../>

  <xsl:template match="/">
    <X3D profile="Immersive" version="3.0" ...>
      <head>
        <meta content="Autor" name="Angelika Preissler"/>
      </head>
      <Scene>
        <xsl:apply-templates/>
      </Scene>
    </X3D>
  </xsl:template>

  <xsl:template match="kugel">
    <Transform>
      <xsl:attribute name="DEF">
          <xsl:value-of select="beschreibung"/>
        </xsl:attribute>
      <xsl:attribute name="translation">
          <xsl:value-of select="verschiebung"/>
      </xsl:attribute>
      <Shape>
        <Sphere>
          <xsl:attribute name="radius">
            <xsl:value-of select="radius"/>
          </xsl:attribute>
        </Sphere>
```

```
        <Appearance>
          <Material>
            <xsl:attribute name="diffuseColor">
              <xsl:value-of select="farbe"/>
            </xsl:attribute>
          </Material>
        </Appearance>
      </Shape>
    </Transform>
  </xsl:template>
</xsl:stylesheet>
```

Quelltext 2.13: Beispiel eines XSL-Dokuments.

Für diese Transformation ist ein *XSLT*-Prozessor (z.B. *Xalan-Java*, *SAXON*, *XT*, *Oracle XDK*) notwendig. [37] Nähere Details zur Verwendung der XSL-Transformation, in Zusammenhang mit der Applikation, folgen in Kapitel 4.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE X3D PUBLIC "http://www.web3d.org/specifications/
  x3d-3.1.dtd"
  "file:///www.web3d.org/TaskGroups/x3d/translation/
  x3d-3.1.dtd">
<X3D profile="Immersive" version="3.0"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema-instance"
  xsd:noNamespaceSchemaLocation="http://www.web3d.org/
  specifications/x3d-3.1.xsd">
  <head>
    <meta content="Autor" name="Angelika Preissler"/>
  </head>
  <Scene>
    <Transform DEF="Kugel" translation="4 0 0">
      <Shape>
        <Sphere radius="2.0"/>
        <Appearance>
          <Material diffuseColor="0 0 1"/>
        </Appearance>
      </Shape>
    </Transform>
  </Scene>
</X3D>
```

Quelltext 2.14: Transformiertes X3D-Dokument.

2.3 Wikis

Ein Wiki ist eine Sammlung von Webseiten (Artikel), die von Benutzern erstellt, bzw. bearbeitet werden und die untereinander verlinkt sind. Aus einer Vielzahl von Artikeln entsteht ein umfangreiches Nachschlagewerk. [19] Der Unterschied zwischen einer herkömmlichen Webseite und einem Wiki liegt in den Möglichkeiten, neue Seiten zu erstellen und existierende Seiten zu bearbeiten. Wikis verwirk-

Web 1.0	Web 2.0
DoubleClick	Google AdSense
Ofoto	Flickr
Akamai	BitTorrent
mp3.com	Napster
Britannica Online	Wikipedia
personal websites	blogging
evite	upcoming.org and EVDB
domain name speculation	search engine optimization
page views	cost per click
screen scraping	web services
publishing	participation
content management systems	wikis
directories (taxonomy)	tagging („folksonomy“)
stickiness	syndication

Tabelle 2.3: Web 2.0 anhand von Beispielen. [38]

lichen den Grundgedanken vom Informationsaustausch im World Wide Web und ermöglichen somit ein kollaboratives Arbeiten. [21] [10] Die Basis des historischen 3D-Atlasses bildet ein *Wiki* (*MediaWiki*). Durch eine Vielzahl von Benutzern ist es möglich, eine große Datenmenge von historischen Inhalten mit geografischen Daten zu erhalten. Das Wiki-Projekt *Wikipedia* ist das beste Beispiel dafür.

Web 2.0

Wikis bilden einen wichtigen Bestandteil von Web 2.0. Tim O'Reilly beschreibt in seinem Bericht [38] den Begriff von Web 2.0 als eine Art Zusammenfassung der neuesten Tendenzen im Web-Bereich. In Tabelle 2.3 wird der Begriff anhand von Vergleichen illustriert. Web 2.0 ist keine neue Technologie, sondern lässt sich als eine Menge von Prinzipien und Methoden beschreiben, siehe Abbildung 2.12. *Web als eine Plattform* wurde als eine dieser Prinzipien definiert. Es dreht sich alles um die Interaktivität im World Wide Web. [38][3]

Gemeinsame Informationen(*Harnessing Collective Intelligence*) nutzbar zu machen, ist das Erfolgskonzept einiger großen Firmen wie Google, Yahoo!, eBay oder Amazon. Die gemeinsame Nutzung von Informationen beschreibt zugleich auch

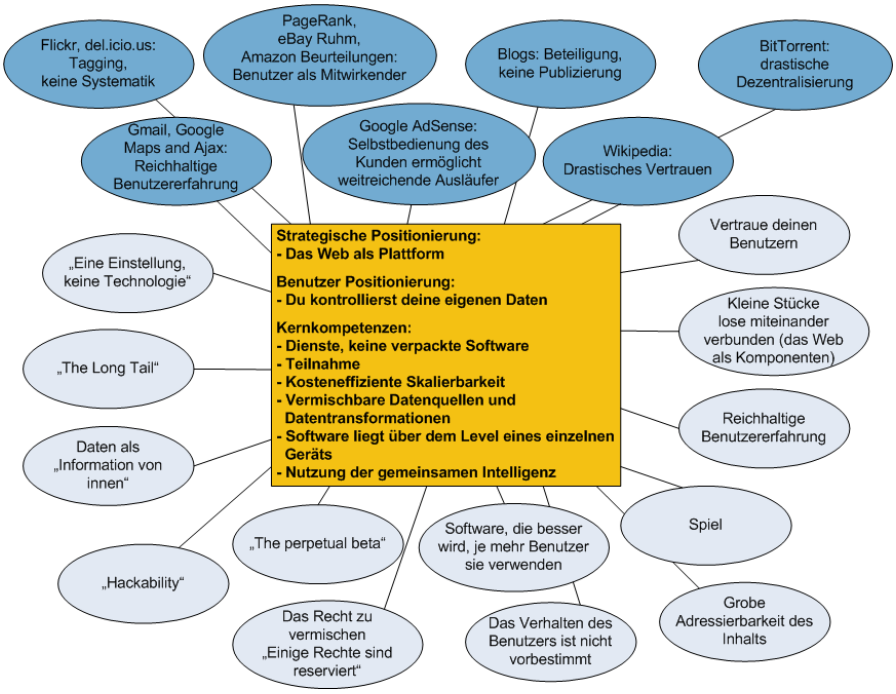

Abbildung 2.12: Tim O'Reilly's Gedanken zu Web 2.0. [38]

das zentrale Prinzip von Web 2.0. [38] Der Inhalt wird von Benutzern erstellt, nicht von Experten. Amazon publiziert zum Beispiel Rezensionen von Kunden in ihrem Onlineshop. Die Idee ist es, nicht selbst Inhalte zu erstellen, sondern die Beiträge der Benutzer zu verwenden, diese zu strukturieren und Vorteile daraus zu ziehen bzw. dadurch die Qualität zu steigern. Auch Wikis folgen diesem Prinzip. Jeder Benutzer kann einen Eintrag in einem Wiki erstellen bzw. diesen editieren oder auch andere Artikel bearbeiten. Wikipedia gehört zu einer der erfolgreichsten Webseiten in den letzten Jahren. [19]

Ein weiterer wichtiger Aspekt von Web 2.0 sind Dienstschnittstellen, welche die Integration von Inhalten anderer Webseiten in die eigene ermöglichen. Beispiele dazu wären, Fotos von *Flickr* oder News von Zeitungen in die eigene Webseite einzubinden. [3]

2.3.1 Wiki-Konzept

Das erste Wiki wurde von Ward Cunningham 1995 entwickelt und wurde als WikiWikiWeb bezeichnet. Die ursprüngliche Idee war es, ein neues Dokumentationssystem für Software-Entwicklung zu schaffen. Die Bezeichnung „WikiWiki“ ist hawaiianisch und bedeutet übersetzt „schnell“ oder „sich beeilen“ und bezieht sich auf die Schnelligkeit, in der Informationen über ein Wiki bereitgestellt werden können. [10] Aus diesem WikiWikiWeb haben sich weitere Wiki-Engines entwickelt, z.B. MediaWiki, UseModWiki, PmWiki, FlexWiki, TWiki, JSPWiki. [21]

Das Konzept eines Wikis besticht durch die Einfachheit in der neue Seiten erstellt bzw. vorhandene Seiten bearbeitet werden können. Über jedem Eintrag steht ein Verweis auf „Seite bearbeiten“, der es ermöglicht jeden vorhandenen Artikel zu bearbeiten und sofort online zu stellen. Neue Seiten können durch das Setzen eines neuen Links generiert werden. Existiert die Seite, auf die verlinkt wird, noch nicht, kann ein neuer Eintrag erstellt werden. Über die Auflistung der Versionen und Autoren können Veränderungen nachvollzogen, verglichen und auch rückgängig gemacht werden. [10][21]

2.3.2 Anwendungsgebiete

Die Hauptanwendungsmöglichkeit von Wikis liegt in der Nutzung als *Wissensdatenbank*. Wikipedia, die erfolgreichste Wiki-Seite, stammt aus diesem Anwendungsgebiet. Bei Wikipedia werden Informationen zu jedem erdenklichen Thema gesammelt. Der Aufbau eines Wikis entspricht dem eines Lexikons. Über Stichworte wird auf den Inhalt zugegriffen, durch Links werden Verweise auf andere Begriffe gesetzt. Die Gestaltung und das Layout rücken in den Hintergrund, die Erstellung der Information wird somit erleichtert und einheitlich gehalten. [21] Wikis dienen immer öfter der Erstellung von Dokumentationen und kommen auch vermehrt in Schulen zum Einsatz. [10]

Private Webseiten werden sehr oft durch ein Wiki ersetzt, aufgrund ihrer einfachen Syntax und Wartungsmöglichkeiten. [21]

Weitere Anwendungsgebiete für Wikis sind als Groupware in Unternehmen, als Plattform für Diskussionen oder als öffentlich zugängliches Notizbuch. [10][21]

2.3.3 Funktionalitäten

Generell gibt es bei Wikis keinen Standard für das eingesetzte Textformat, es wird jedoch an einem Standard namens WikiCreole gearbeitet. Fast alle vorhandenen Wikis leiten ihre Syntax vom ersten Wiki (WikiWikiWeb) ab, somit besteht eine Ähnlichkeit bei den wichtigsten Funktionalitäten. Ist die Formatierung mit der Wiki-Syntax nicht ausreichend, ist es auch möglich, zusätzlich HTML-Elemente oder Cascading Style Sheets (CSS) [42] zu verwenden. [21] Nachfolgend werden die wichtigsten Funktionalitäten eines Wikis beschrieben.

Bearbeitung der Seiten

Das Grundprinzip eines Wikis ist, dass jeder Artikel für jeden Benutzer verfügbar ist. Somit kann jeder Benutzer neue Artikel erstellen und beliebige Artikel von anderen Autoren bearbeiten. Aufgrund dieses Prinzips entsteht eine ständige Verbesserung und Erweiterung der Artikel. Durch die Mitarbeit von vielen Benutzern kann eine umfangreiche Datensammlung erstellt werden. Über die Versionskontrolle ist es möglich, Änderungen zu verfolgen und damit die Qualität der Artikel durch Selbstkontrolle der Benutzer sicher zu stellen. Falsche Informationen und willkürliche Veränderungen können von Benutzern selbst rückgesetzt werden.

Verlinkung der Seiten

Durch Links (Hyperlinks) können verschiedene Seiten (Hypertext) miteinander verbunden werden. Durch das Setzen von Links wird die Navigation durch die einzelnen Artikel erleichtert, da miteinander verlinkte Seiten zu ähnlichen Themen gehören. Somit kann ein großer Themenbereich nicht nur auf einer, sondern auf mehreren Seiten abgedeckt werden. Verweist ein Link auf eine Seite, die noch nicht vorhanden ist, wird dieser Link anders dargestellt, als bei existierenden Seiten. Besteht also noch kein Artikel, so kann der Benutzer diesen erstellen. [10][21]

Es wird zwischen drei Verlinkungen in einem Wiki unterschieden: Wiki-interne Links, InterWiki: Links zu anderen Wikis und externen Links ins Web. [10][21]

Wiki-interne Links verweisen auf Seiten innerhalb eines Wikis. In den Anfängen des Wikis wurden diese durch gemischte Groß-/Kleinschreibung gesetzt, die

Schreibweise wurde als *WikiWords* oder *CamelCase* bezeichnet, zum Beispiel „NeuerLink“. Diese Variante wird jedoch nicht mehr von allen Wikis unterstützt. Als Alternative zu den WikiWords gibt es so genannte *FreeLinks*, die von allen Wikis akzeptiert werden. Das Wort, das verlinkt wird, wird in doppelten eckigen Klammern geschrieben.

```
[[Seitentitel]]
```

Es besteht auch die Möglichkeit, anstatt des Titels der Seite eine alternative Beschreibung als Linkverweis anzugeben.

```
[[Seitentitel|Beschreibung]]
```

InterWiki ist die Bezeichnung für die Verlinkung in andere Wikis. Die Grundidee ist, verschiedenste Wikis einfach miteinander zu vernetzen. Dazu ist es notwendig, externe Wikis in einer Konfigurationsdatei mit Bezeichnung und URL zu definieren, zum Beispiel:

```
WikiPediaDE http://de.wikipedia.org/wiki/
```

Danach wird die gewünschte Seite mit folgender Syntax aufgerufen:

```
[[WikiPediaDE:Seitentitel]]
```

Der Seitentitel wird von der Wiki-Engine direkt an jene URL angehängt, die in der Konfigurationsdatei angegebenen wurde.

Externe Links verweisen auf beliebige externe Inhalte, dabei kann es sich auch um andere Wikis handeln. Der Link zu einer Webseite wird in einfachen eckigen Klammern geschrieben.

```
[http://www.tuwien.ac.at]
```

2.4 Grundlagen zu geografischen Daten

In den folgenden Kapiteln werden grundlegende Begriffe zur Verarbeitung und Standardisierung von geografischen Daten erläutert. Abschließend werden verschiedene Geo-Browser näher betrachtet.

2.4.1 Grundbegriffe

Geo-Objekte

Geometrische Informationen sind in einem Punkt verankert, dieser ist die Grundlage aller weiteren räumlichen Objekte (*Geo-Objekte*) wie Linie, Fläche und Körper. [2] Geo-Objekte sind definiert durch ihre Geometrie, Topologie, Thematik und Dynamik.

Die *Geometrie* beinhaltet sämtliche Informationen zur Lage und Größe eines Geo-Objektes. Diese Angaben erfolgen auf der Grundlage eines eindeutigen räumlichen Bezugssystems. Es können verschiedene Bezugssysteme herangezogen werden, wie zum Beispiel kartesisches Koordinatensystem, GPS-Koordinaten. Geo-Objekte selbst werden als Vektor- oder Rastermodell dargestellt. Das Vektormodell besteht aus gerichteten Vektoren, diese sind durch einen Start- und Endpunkt (Koordinaten) eindeutig definiert. Neben den Koordinaten müssen auch Informationen gespeichert werden, die die Zugehörigkeit von Punkten (Vektoren) zu Linien und von Linien zu Flächen beschreiben. Beim Rastermodell wird eine Fläche (zumeist ein Quadrat - Pixel) als Grundelement herangezogen. Durch die Angabe eines Spalten- und Zeilenindexes wird jeder Pixel eindeutig definiert. So wird ein Punkt durch einen Pixel, eine Linie und eine Fläche durch mehrere aneinander gereihte Pixel dargestellt. [8]

Die *Topologie* beschreibt die relative Lage und die räumlichen Beziehungen zu anderen Geo-Objekten. Die relative Lage von Objekten ändert sich nicht. Die Geometrie der Geo-Objekte wird abstrahiert, wenn die Topologie betrachtet wird. Nachbarschaften, Teilmengen und Überschneidungen bilden zentrale Konzepte der Topologie. Die sechs möglichen topologischen Beziehungen der Grundformen Punkt, Linie und Fläche im zweidimensionalen Bereich zeigt Abbildung 2.13.

Weitere Eigenschaften von Geo-Objekten sind die Thematik und die Verände-

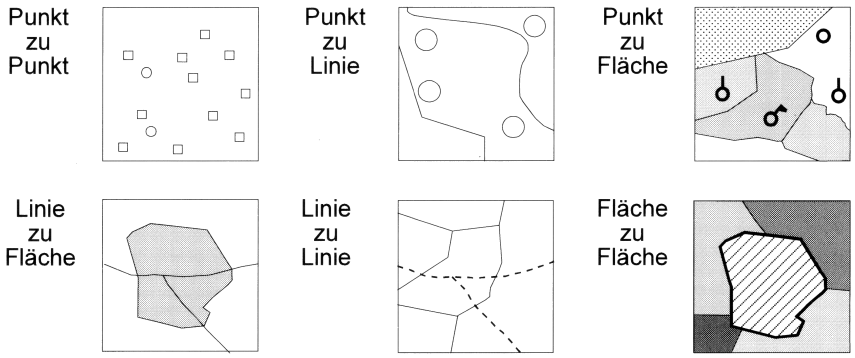

Abbildung 2.13: Topologische Beziehungsarten der Grundformen Punkt, Linie und Fläche. [8]

rung (Dynamik), wobei zwischen zeitlichen und räumlichen Veränderungen unterschieden wird. [8]

Koordinatensysteme

Die Darstellung der Geo-Objekte, entweder im Vektor- oder Rastermodell definiert, erfolgt in einem *kartesischen Koordinatensystem*. Neben diesem System sind *Polarkoordinaten* von großer Bedeutung. Ein Vergleich zwischen den beiden Systemen wird in der Abbildung 2.14 gezeigt.

Die Umrechnung zwischen den zwei Koordinatensystemen im dreidimensionalen Fall erfolgt durch folgende Formeln: [8]

Polarkoordinaten in kartesische Koordinaten:

$$\begin{aligned} P(r,\alpha,\beta): \qquad x &= r\ sin\beta\ cos\alpha \\ y &= r\ sin\beta\ sin\alpha \\ z &= r\ cos\beta \end{aligned}$$

In einigen Bereichen wird auch diese Umrechnung verwendet:

$$\begin{aligned} P(r,\alpha,\beta): \qquad x &= r\ cos\beta\ cos\alpha \\ y &= r\ cos\beta\ sin\alpha \\ z &= r\ sin\beta \end{aligned}$$

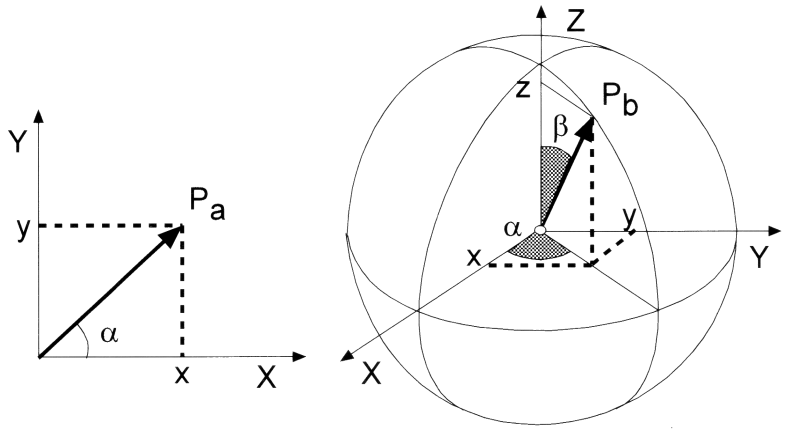

Abbildung 2.14: Darstellung eines Punktes in Polarkoordinaten und in kartesischen Koordinaten, jedes in 2D und 3D. [8]

Kartesische Koordinaten in Polarkoordinaten:

$$
\begin{aligned}
P(x,y,z): \qquad r &= \sqrt{(x^2+y^2+z^2)} \\
\alpha &= \arctan\ y/x \text{ (für } x \neq 0) \\
\beta &= \arctan \sqrt{(x^2+y^2}/z \\
\text{für } x &= 0 \text{ und } y = r \text{ ist } \alpha = \Pi/2 \\
\text{für } x &= 0 \text{ und } y = -r \text{ ist } \alpha = -\Pi/2
\end{aligned}
$$

Jeder Punkt wird eindeutig auf einer Kugel definiert, wenn der Radius r konstant ist. Im Falle der Erde wird dieser Radius auf 6.371 km gesetzt. Die Koordinaten eines Punktes ergeben sich aus den Winkeln φ (*Geografische Breite*) und λ (*Geografische Länge*). Das Geografische Koordinatensystem besteht aus diesen beiden Winkeln, siehe Abbildung 2.15. [8]

- Das Geografische Gradnetz besteht aus Breiten- und Längenkreisen.
- Der Kreis, dessen Ebene durch den Erdmittelpunkt geht und senkrecht zur Rotationsachse der Erde steht, ist der *Äquator*.
- Als Breitenkreise oder Parallelkreise werden die Kreise bezeichnet, die parallel zum Äquator verlaufen. Der Winkel zwischen einem Punkt auf der

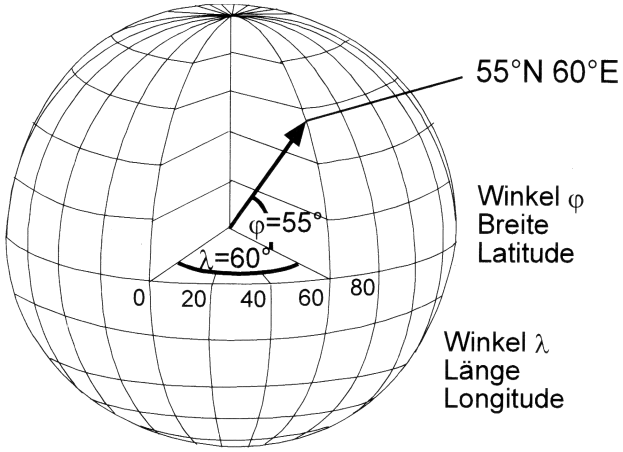

Abbildung 2.15: Zeigt das geografische Koordinatensystem. [8]

Oberfläche der Kugel und der Äquatorebene, entlang des entsprechenden Längenkreises, wird als Geografische Breite bezeichnet, 0° = Äquator, -90° = Südpol, 90° = Nordpol.

- Kreise, die durch beide Pole und vertikal zum Äquator verlaufen, werden als Längenkreise oder Meridiane bezeichnet. Die Sternwarte von Greenwich wurde als Nullmeridian definiert, von diesem ausgehend wird je 180° nach Osten und Westen gerechnet.

- Der Äquator und alle Meridiane werden als Großkreise bezeichnet, weil sie denselben Radius wie die Erde besitzen.

Ellipsoide

Bei exakten Koordinatenbestimmungen darf nicht davon ausgegangen werden, dass die Erde der Form einer Kugel entspricht. Hier muss die Erde mit Ellipsoiden angenähert werden. Unter anderem finden hier Ausbuchtungen am Äquator und Abflachungen an den Polen Berücksichtigung. Diese Ellipsoide werden durch die Hauptachse (längere Achse) und die Nebenachse (kürzere Achse) definiert und entstehen durch die Drehung um die Nebenachse. Aufgrund weiterer Abweichungen

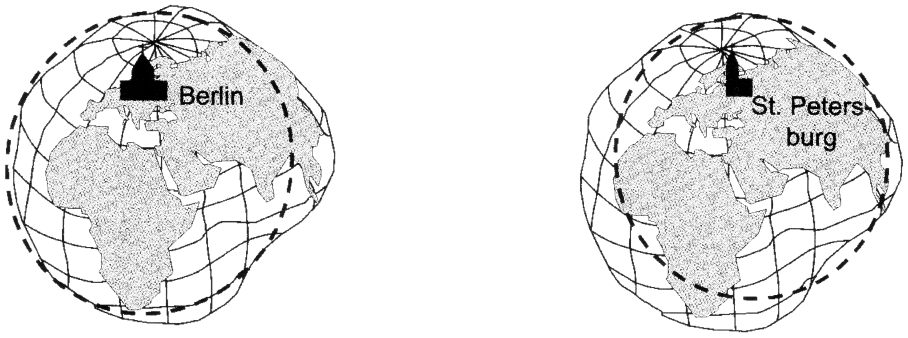

Abbildung 2.16: Je nach Region werden unterschiedliche Referenzellipsoide verwendet. [8]

werden in unterschiedlichen Gebieten verschiedene Ellipsoide eingesetzt. Abbildung 2.16 zeigt die Verwendung von unterschiedlichen Ellipsoiden. [8]

2.4.2 Formate für Geodaten

Die Nachfrage nach Geodaten ist in den letzten Jahren stark gestiegen. Nicht nur traditionelle Abnehmer, wie Umwelt- und Planungsämter sind an diesen Daten interessiert, sondern auch Wirtschaftsunternehmen zur Organisation von Vertriebsgebieten, Location-based Services, Verkehrsnavigation oder zur Optimierung ihrer Leitungssysteme. [8]

Aufgrund der Komplexität von geografischen Daten ist es notwendig diese zu standardisieren, um eine sinnvolle Weiterverarbeitung im Internet zu gewährleisten. Zu einem *Geoinformationssystem* (*GIS*) gehören Anwendungen und entsprechende Daten. Ein GIS ist für die Erfassung, Analyse, Verwaltung und Präsentation von Geodaten zuständig. Nachteile dieser Systeme sind die komplexe Installation und Handhabung. Durch die gestiegene Nachfrage nach Geodaten mussten diese Systeme überarbeitet und leichter bedienbar gemacht werden, zum Beispiel eine Aufteilung in Funktionseinheiten. Das *Open Geospatial Consortium* (OGC) arbeitet an einer Integration von Geoinformationen in die IT-Technologie. Das OGC besteht aus über 300 Firmen, Behörden und Universitäten. Weiters beschäftigt sich auch ISO mit der Standardisierung von Geoinformationen und Geodaten. [16]

GML

GML (Geography Markup Language) ist ein von OCG entwickelter Standard für geografische Informationen. Dieser Standard ist ein offenes Format um geografische Objekte und Anwendungen zu beschreiben, und dient gleichzeitig als Austauschformat. GML (derzeit Version 3.1.1) basiert auf XML und ist für die Strukturierung des geografischen Inhalts zuständig. Die Darstellung der Daten kann durch unterschiedliche Art und Weise erfolgen, z. B. X3D, SVG.

Objekte werden durch *Features* beschrieben, wobei zwischen räumlichen und nichträumlichen Features unterschieden wird. Räumliche Features beinhalten eine geografische Position. Ein Feature wiederum besteht aus einer Liste von Geometrien und Eigenschaften. Eine *Geometrie* beschreibt die geometrischen Eigenschaften eines Objekts unter Verwendung von geometrischen Formen (Punkt, Linie, Kurven, Polygone). *Eigenschaften* werden durch *Name*, *Typ* und *Wert* definiert. Die Spezifikation von GML ist in Profilen (wie X3D, siehe Kapitel 2.1.4) unterteilt, z.B. *Point Profile*, *GML Simple Feature Profile*. Jedes Profil beinhaltet eine Teilmenge von definierten Elementen. [56] [20] Quelltext 2.15 zeigt ein Beispiel in dem ein Gebäude über die Koordinaten eines Punktes definiert ist. Durch die Angabe von *srsName* wird auf ein Bezugssystem verwiesen. [20] [2] Die Spezifikation zu GML kann unter [36] nachgelesen werden.

```
<Feature fid="1" ftype="gebaeude">
  <Property name="stockwerke" type="integer" value="3" />
  <gml:location>
    <gml:Point gml:srsName="urn:EPSG:geographicCRS:
      62836405">
      <gml:pos>-31.936 115.834</gml:pos>
    </gml:Point>
  </gml:location>
</Feature>
```

Quelltext 2.15: Beispiel eines GML-Dokuments.

KML

KML (Keyhole Markup Language) wurde ursprünglich von der Firma Keyhole entwickelt. KML ist ein offenes Format (basiert auf XML, teilweise auf GML) und wird als Datenaustauschformat für GoogleEarth verwendet. In KML wird sowohl der Inhalt als auch die Darstellung des Inhaltes definiert, ähnlich wie in HTML. Ein KML-Dokument kann in zwei Varianten abgespeichert werden: .kml, .kmz. Die Visualisierung einer kml-Datei erfolgt in *Google Earth*. KML bietet Möglichkeiten Orte durch Icons oder Bezeichnungen zu bestimmen, Blickpunkte und Kamerapositionen zu definieren, die Erdoberfläche mit Bildern zu überlagern, Stile zu definieren, Elemente zu gruppieren, Animationen zu erzeugen und KML Dateien dynamisch nachzuladen. [55] [15]

Quelltext 2.16 zeigt ein Beispiel einer KML Datei. *Placemark* bildet das zentrale Element und wird für die Markierung einer geografischen Position verwendet. Es können Geometrien (Punkt, Linie, Fläche), die dazugehörigen Koordinaten (*coordinates*), Icons (*IconStyle*) und die Darstellung des Placemarks definiert werden. Mittels *LookAt* kann ein Blickpunkt angegeben werden, durch *longitude* und *latitude* wird dieser über seinen Längen- und Breitengrad definiert. Optional können *heading* (Orientierung), *range* (Höhe des Blickpunktes) und *tilt* (Neigung des Blickwinkels) angegeben werden. [55] [15]

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://earth.google.com/kml/2.1">
<Placemark>
  <name>Stonehenge, England</name>
  <description>
    Stonehenge was built around 2500BC
  </description>
  <LookAt>
    <longitude>-1.826752</longitude>
    <latitude>51.179045</latitude>
    <range>1500</range>
    <tilt>0</tilt>
    <heading>0</heading>
  </LookAt>
  <Style>
    <IconStyle>
      <Icon>
        <href>beispiel.png</href>
      </Icon>
    </IconStyle>
  </Style>
  <Point>
    <coordinates>-1.826752,51.179045,0 </coordinates>
  </Point>
</Placemark>
</kml>
```

Quelltext 2.16: Beispiel eines KML-Dokuments.

2.4.3 Geo-Browser

Für die Darstellung von geografischen Daten gab es bereits genügend Anwendungen für Spezialisten. Jedoch fehlte es am Zugang und an der Aufbereitung dieser Geodaten für die breite Masse. Seit der Existenz von Geo-Browser interessiert sich

auch die breite Öffentlichkeit für geografische Daten. Dabei wird eine dreidimensionale Erde für die Visualisierung dieser Informationen herangezogen. [57]

Zwei derzeit sehr bekannte Geo-Browser, Google Earth und World Wind von NASA, werden nachfolgend näher betrachtet.

Google Earth

Seit 1993 arbeitete die aus Deutschland stammende Firma Art+Com am ersten Geo-Browser, Terravision. Der Öffentlichkeit wurde dieses Projekt 1994 auf der ITU-Konferenz (International Telecommunications Union) in Kyoto vorgestellt. Nach einer Präsentation auf der Siggraph wurde 1995 eine Installation im Demonstrationszentrum von SGI übernommen. Terravision wurde in die SGI-Grafikbibliothek Performer integriert. Der damalige Direktor der Grafikbibliotheken von SGI, T. Jones, gründete ein paar Jahre später die Firma Keyhole, die die Applikation Earth Viewer entwickelte. Keyhole wurde 2004 von Google gekauft und der Namen der Applikation in Google Earth geändert. [40][57] Die erste Betaversion von Google Earth wurde im Juni 2005 veröffentlicht. [22]

Google Earth basiert auf einem Server, der das entsprechende Kartenmaterial liefert und einem Client, der die Daten visualisiert. [22] Google Earth ist in vier Varianten erhältlich: die kostenlose Google Earth Version, siehe Abbildung 2.17, Google Earth Plus, Google Earth Pro und die Lösung für Unternehmen, Google Earth Enterprise. [14] Das Kartenmaterial dafür wird von Google zugekauft bzw. wurde von der früheren Firma Keyhole übernommen. Derzeit gibt es nur für einen kleinen Bereich der Erde hochauflösendes Kartenmaterial. Mittlerweile beauftragt Google selbst Überflüge um geeignetes Bildmaterial zu Verfügung zu haben. [57] Google Earth bietet eine Zoommöglichkeit für eine detaillierte Darstellung der Erdoberfläche, es werden dabei immer hochauflösendere Satelliten- bzw. Luftbilder geladen. [22] Der Benutzer kann nach Orten und Plätzen suchen, indem der Name oder die Koordinaten des Ortes eingegeben werden. Jedoch sind einige durchaus bedeutende Sehenswürdigkeiten der Ortsdatenbank nicht bekannt, wie zum Beispiel der Ayers Rock in Australien. Durch Placemarks ist es möglich einen geografischen Punkt zu markieren und mit Information zu versehen, siehe dazu KML (Kapitel 2.4.2). Weiters können auch dreidimensionale Objekte (Großstädte

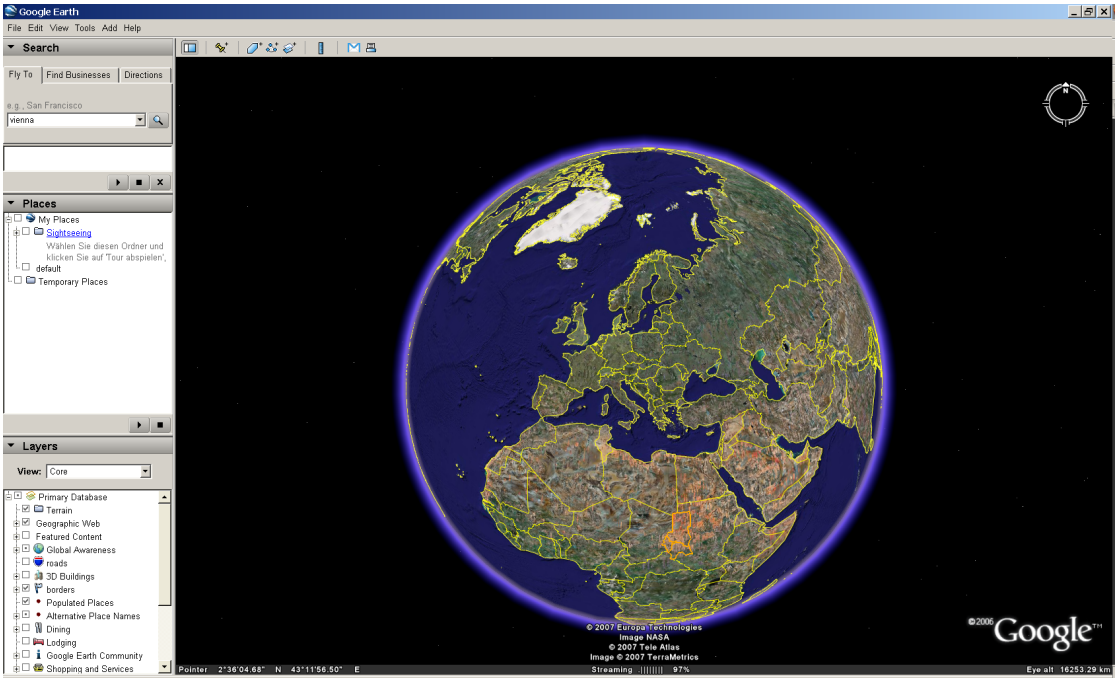

Abbildung 2.17: Google Earth.

in Amerika) oder Straßenkarten angezeigt werden. Auch die Suche nach Adressen ist möglich. [57]

NASA World Wind

World Wind kann kostenlos heruntergeladen werden. Der Geo-Browser unterliegt einer Open-Source-Lizenz. Das verwendete Bildmaterial stammt aus frei verfügbaren Datenquellen. Vor allem für die USA, Frankreich und Deutschland steht hochauflösendes Kartenmaterial zu Verfügung. Wie bei Google Earth ist auch bei World Wind ein stufenloses Zoomen möglich. Der große Unterschied zu Google Earth ist, dass World Wind für wissenschaftliche Zwecke entwickelt wurde. Die Menüleiste wird so klein wie möglich gehalten, um eine bestmögliche Ansicht des Bildmaterials zu gewähren, siehe Abbildung 2.18. [22][40][57] Es kann nach Orten durch den Namen (nur auf Englisch) oder den Koordinaten gesucht werden, weiters können bevorzugte Orte als Hotspots gespeichert werden. Auch bei World Wind ist die Ortsdatenbank sehr umfangreich, es werden auch sehr kleine, abgelegene Orte gefunden. Von der Applikation werden jedoch keine Orte gefunden, die unbewohnt,

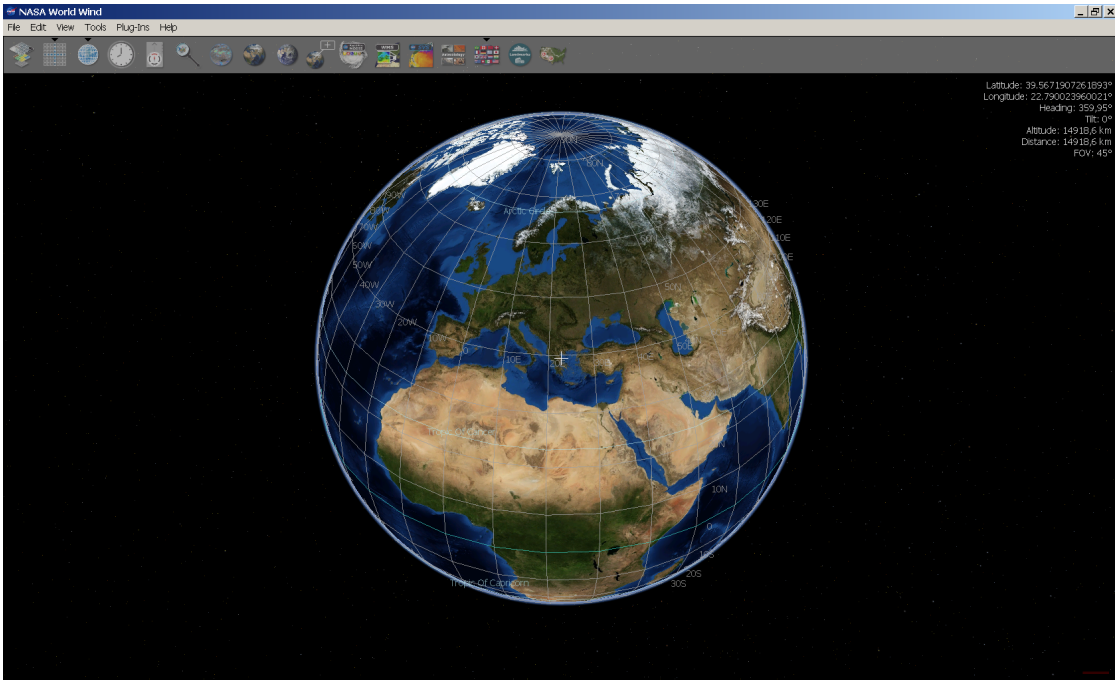

Abbildung 2.18: NASA World Wind.

aber bekannt sind, zum Beispiel der Ayers Rock. Auf WMS-Dienste (Web Mapping Service), ein offener Standard von OCG, kann direkt zugegriffen werden. Dadurch ist der Abruf von Daten aus verschiedenen Quellen möglich. So liefert die NASA Informationen zu Naturphänomenen und Animationen, in denen Vorgänge auf der Erde gezeigt werden. Derzeit können keine dreidimensionalen Objekte dargestellt werden. Da World Wind als Open-Source frei zur Verfügung steht, kommen jedoch laufend Erneuerungen hinzu.

Durch definierte Formate ist es möglich Daten untereinander auszutauschen. Google Earth und World Wind liefern sehr gute Benutzerschnittstellen, bieten aber nur eingeschränkte Möglichkeiten zur Mitarbeit an der Datenbasis. Bei beiden Geo-Browsern ist ein spezieller Client zum Zugriff auf die Daten notwendig. Der im Rahmen dieser Arbeit erstellte Prototyp bietet durch ein Wiki die Möglichkeit der Mitarbeit von vielen Benutzern. Zusätzlich ist die Benutzerschnittstelle über einen beliebigen Webbrowser verfügbar.

3 Entwurf des technischen Systems

Das Ziel des praktischen Teils der Arbeit ist es, eine technische Umgebung zur Erstellung eines historischen 3D-Weltatlasses zu implementieren. Als Grundgerüst für dieses System fungiert ein Wiki. In diesem werden die historischen Geodaten von Wiki-Benutzern erstellt und mittels X3D visualisiert. Die technischen Anforderungen an das System, das Konzept und die Architektur werden in diesem Kapitel detailliert dargestellt.

3.1 Anforderungen

Durch Benutzer-Eingaben in ein Wiki sollen geografische Daten mit textuellen Annotationen auf einer 3D-Weltkugel dargestellt werden. Ziel ist es, viele Benutzer zu erreichen und die Eingabe so einfach wie möglich zu gestalten. Die erstellten Inhalte im Wiki müssen in eine entsprechende Struktur gebracht werden, um diese effizient weiterverarbeiten zu können. Aus diesem Grund werden die Daten in einer XML-Struktur zusammengefügt und mittels Stylesheet Transformation ein X3D-Dokument generiert. Weitere Anforderungen bestehen darin, über einen Zeitbalken die historische Entwicklung der Welt zu verfolgen und nach bestimmten Zeitperioden zu filtern.

In Abbildung 3.1 wird der gesamte Prozess von der Eingabe bis zur Ausgabe der Daten dargestellt. Nachfolgend eine detaillierte Auflistung der Anforderungen:

1. Ein Wiki dient als Datenbasis für die Bearbeitung und Verwendung durch eine große Anzahl von Benutzern. Für die Beschreibung der historischen und geografischen Daten gibt es eine definierte Eingabemöglichkeit.

2. Extrahieren der Eingabedaten (Längengrad, Breitengrad, Beschreibungen, ...) um eine strukturierte Darstellungsweise zu erhalten (XML-Dokument).

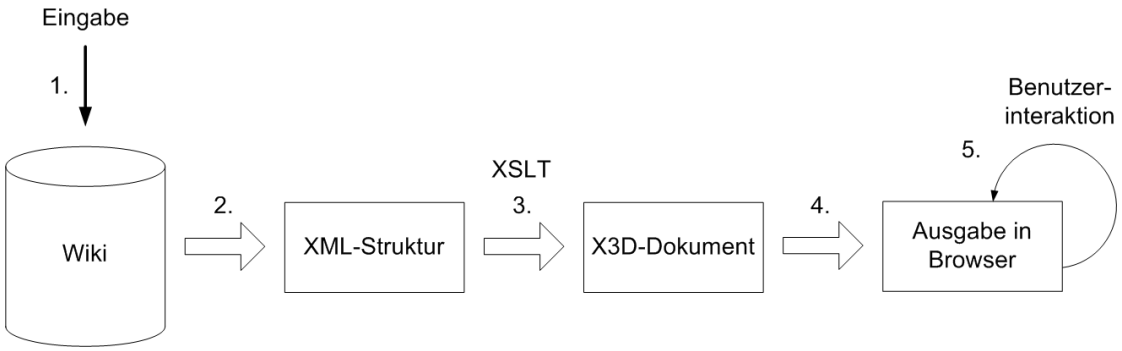

Abbildung 3.1: Ablauf des Visualisierungsprozesses.

3. XSL-Transformation eines XML-Dokuments um ein X3D-Dokument zu generieren.

4. Visualisierung des X3D-Dokuments im Browser durch die Integration eines X3D-Plug-ins.

5. Integration eines Zeitbalkens, um die historischen Entwicklungen verfolgen zu können, bzw. Eingabemöglichkeiten von Zeitperioden, um die Ausgabe auf bestimmte Zeitspannen festlegen zu können.

3.2 Konzept

3.2.1 Wiki-Engine

Als Wiki-Engine wird MediaWiki verwendet, diese wird unter anderem auch für die Online-Enzyklopädie Wikipedia eingesetzt. MediaWiki unterliegt der GNU General Public License (GPL) und ist in PHP implementiert. MediaWiki wird als Serverbasierende Software über einen Webserver (z.B. Apache [41] oder IIS [33]) betrieben. Als Datenbank kann MySQL [34] oder PostgreSQL [39] verwendet werden. [32] In Abbildung 3.2 wird die Architektur von MediaWiki dargestellt.

MediaWiki unterstützt JavaScript, XHTML, Editier- und Formatierungsmöglichkeiten, Cascading Stylesheets (CSS), Bildergalerien, Darstellung von mathematischen Formeln und Schutzmaßnahmen gegen Vandalismus und Spamming. Durch

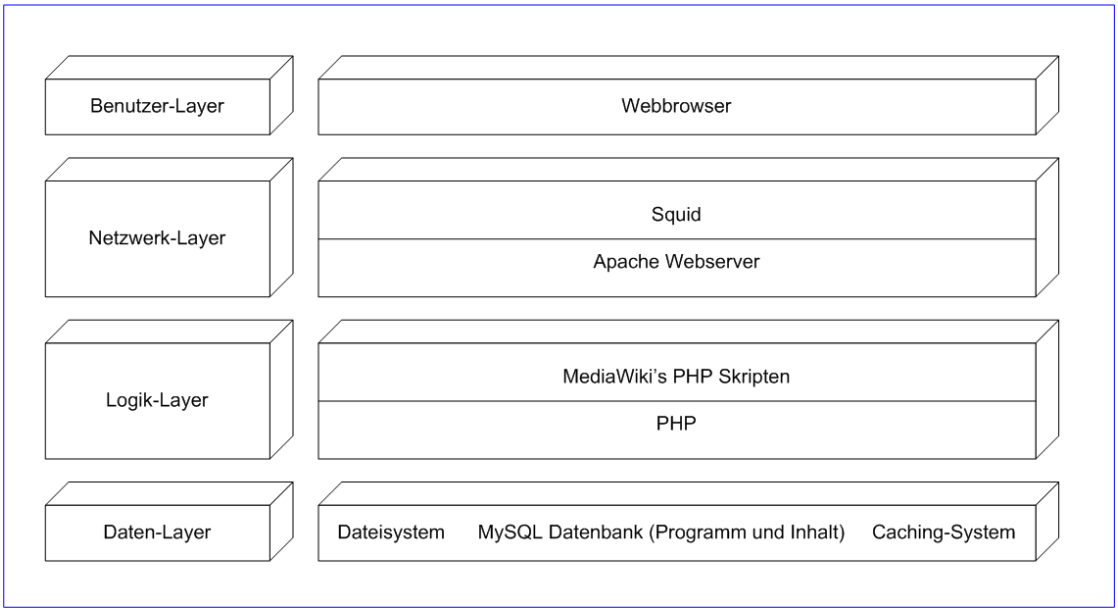

Abbildung 3.2: Architektur von MediaWiki. [54]

Extensions und Spezialseiten ist es möglich MediaWiki mit Funktionen zu erweitern, ohne direkt im Software-Kern Veränderungen vornehmen zu müssen. [21]

3.2.2 Seitentypen

Für den historischen Weltatlas wurde eine Klassifizierung der Seiten in unterschiedliche Seitentypen vorgenommen.

Geografische Seiten

Geografische Seiten beinhalten das XML-Element `points`, das im Wikitext inkludiert wird. Dadurch erkennt das System, dass Daten auf der 3D-Weltkugel darzustellen sind. Dieser Seitentyp wird in zwei Arten unterschieden:

1. Geo-Definitionsseite

   Anhand von `points`-Elementen werden Geo-Objekte definiert; dabei bestehen folgende Möglichkeiten:

   - Punkte: z.B. Städte, Orte

- Linien: z.B. Landesgrenzen
- Flächen: z.B. Länder, Teile eines Landes

```
<points>Geo-Objekt</points>
```

Ein wichtiges Attribut dieser Geo-Objekte ist `time`, womit die zeitliche Komponente einer geografischen Information definiert werden kann.

2. Geo-Containerseite

   Werden zwischen `points` keine geografischen Objekte definiert, dient dieses XML-Element als Container. Dabei werden die Informationen aus den verlinkten Seiten geholt, die sich im Artikel befinden. Die geografische Visualisierung wird dann aus den verlinkten Begriffen (`Link 1`, `Link 2`) generiert, sofern diese Seiten Geo-Definitionsseiten sind.

   ```
   <points></points>
   Link 1
   Link 2
   ```

Nicht geografische Seiten

Diese Seiten beinhalten kein XML-Element `points`. Sie entsprechen einer üblichen Wiki-Seite. Diese Artikel dienen zur Beschreibung von Inhalten und zum Anzeigen von zweidimensionalen Grafiken. Es wird dabei keine X3D-Darstellung generiert.

3.2.3 Linktypen

Abbildung 3.3 zeigt ein Beispiel, welche mögliche Verlinkungen von einem Artikel ausgehen können und wie diese verarbeitet werden. Artikel (Seiten) werden als Knoten und Verlinkungen als Kanten dargestellt. Der aktuell dargestellte Artikel ist der Wurzel-Knoten. Alle Seiten, durch die die verstärkte durchgezogene Linie führt, werden auf der dreidimensionalen Weltkugel visualisiert.

Bei jedem verlinkten Artikel wird geprüft, ob es sich um eine geografische Seite handelt. Nur solche werden weiter verarbeitet und dargestellt. Bei einer nicht

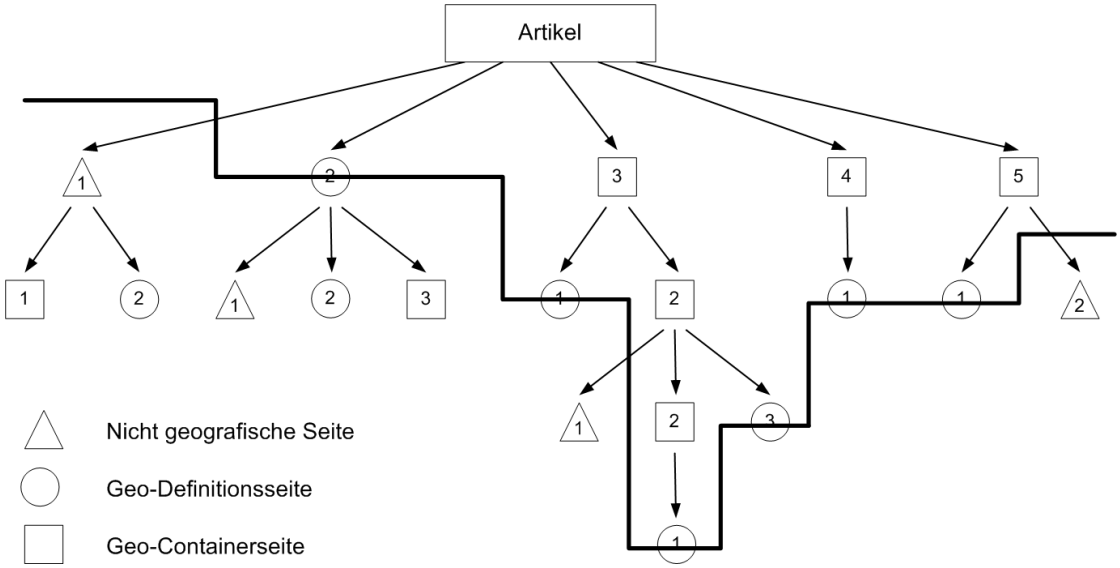

Abbildung 3.3: Skizzierung eines Verlinkungsbaums unter Verwendung von unterschiedlichen Seitentypen. Alle Artikel, die sich auf der verstärkten durchgezogenen Linie befinden, werden am Weltatlas dargestellt.

geografischen Seite wird der gesamte Pfad verworfen, wie Abbildung 3.3 bei den Pfaden *Artikel.1.1* und *Artikel.1.2* zeigt.

Handelt es sich um eine Geo-Definitionsseite, werden die Informationen aus dem `points`-Element in X3D dargestellt, weitere verlinkte Artikel werden nicht verfolgt, zum Beispiel die Pfade *Artikel.2.1*, *Artikel.2.2* und *Artikel.2.3*.

Bei einer Geo-Containerseite wiederum werden alle verlinkten Seiten so lange durchlaufen, bis eine Geo-Definitionsseite (Pfad *Artikel.3.2.2.1*) oder eine nicht geografische Seite (Pfad *Artikel.3.2.1*) gefunden wird. Durch die Betrachtung von Pfaden innerhalb der Baumstruktur wird die wiederholte Verwendung derselben geografischen Information verhindert.

3.2.4 Integration von zeitlichen Verläufen

Historische Darstellungen werden aufbauend auf den in Kapitel 3.2.2 und 3.2.3 beschriebenen Konzepten erstellt. Dazu wird eine neue Geo-Containerseite erstellt, die auf bereits vorhandene Geo-Definitionsseiten oder Geo-Containerseiten verlinkt, siehe Quelltext 3.1.

Durch die Definition eines Zeitbereiches anhand der Attribute `from` und `until` ist es möglich, sich nur jene Artikel anzeigen zu lassen, die sich innerhalb dieser Zeitperiode befinden. Dabei wird das Attribut `time` der Geo-Objekte als Filter herangezogen und nur jene Objekte angezeigt, bei denen die Jahreszahl, die in dem Attribut definiert ist, sich zwischen `from` und `until` befindet. Die Visualisierung der geografischen Informationen erfolgt anhand der Verarbeitung der vorhandenen Links, wie in Kapitel 3.2.3 detailliert beschrieben. Durch Interaktionselemente ist es möglich, sich den historischen Verlauf der Seiten anzusehen und zu navigieren.

```
Further information ...
<points from="1911" until="2007"></points>

[[Austria]]
[[Austria-Hungary 1911]]
[[Austria 1935]]
[[Austria 1885]]
```

Quelltext 3.1: Zeitlicher Verlauf.

3.3 Architektur

Dieses Kapitel beschreibt die Integration der Applikation innerhalb von MediaWiki. Die Verteilung der einzelnen Bereiche des Systems ist in Abbildung 3.4 skizziert. Dabei ist ersichtlich, dass Erweiterungen sowohl auf Client- als auch auf Serverseite erfolgt sind. Für die Umsetzung werden die von MediaWiki vorgesehenen Möglichkeiten verwendet, wie *Extension* und *Spezialseite*.

3.3.1 Wiki-Extension

Für die Einbettung des X3D-Plug-ins in die HTML-Ausgabe an der Position des `points`-Elements ist die Extension `Point` verantwortlich. Die Extension registriert eine Callback-Funktion im Wiki, wodurch beim Auftreten des XML-Elements `points` im Wikitext diese Funktion aufgerufen wird. [31] Wenn das Element nicht

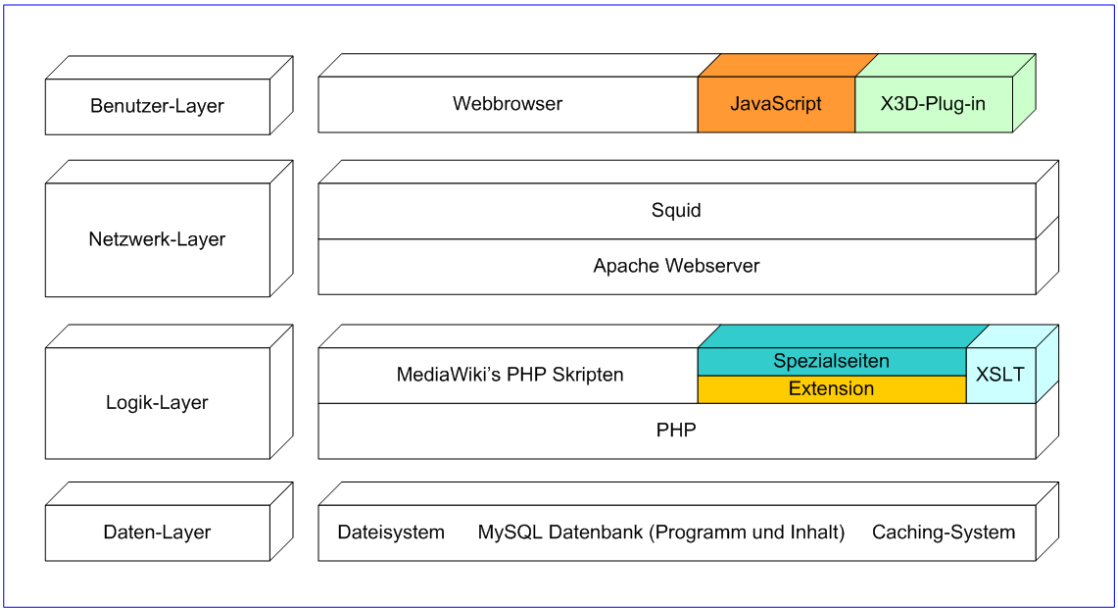

Abbildung 3.4: Architektur des technischen Systems mit Integration in MediaWiki.

in der Wiki-Seite inkludiert ist, wird die Extension nicht ausgeführt und die X3D-Anzeige für diesen Artikel nicht aktiviert. Dies ist bei nicht geografischen Seiten der Fall.

3.3.2 Wiki-Spezialseite

Im MediaWiki können so genannte Spezialseiten erstellt werden. Diese bieten die Möglichkeit, die Ausgabe vollständig zu definieren und dadurch verschiedene Medientypen (MIME-Types) anzugeben. [30] Die Applikation definiert eine Spezialseite, welche den X3D-Inhalt aus den Artikeln generiert. Dabei werden die in Kapitel 3.2.2 und 3.2.3 beschriebenen Funktionalitäten verwendet, um geografische Daten und Links zu verarbeiten und daraus ein XML-Dokument zu erstellen. Anschließend wird auf diesem XML-Dokument eine XSL-Transformation durchgeführt, um damit eine X3D-Ausgabe zu generieren. Diese wird an den Browser gesendet und mittels Plug-in angezeigt. Der detaillierte Ablauf wird in Abbildung 3.5 dargestellt.

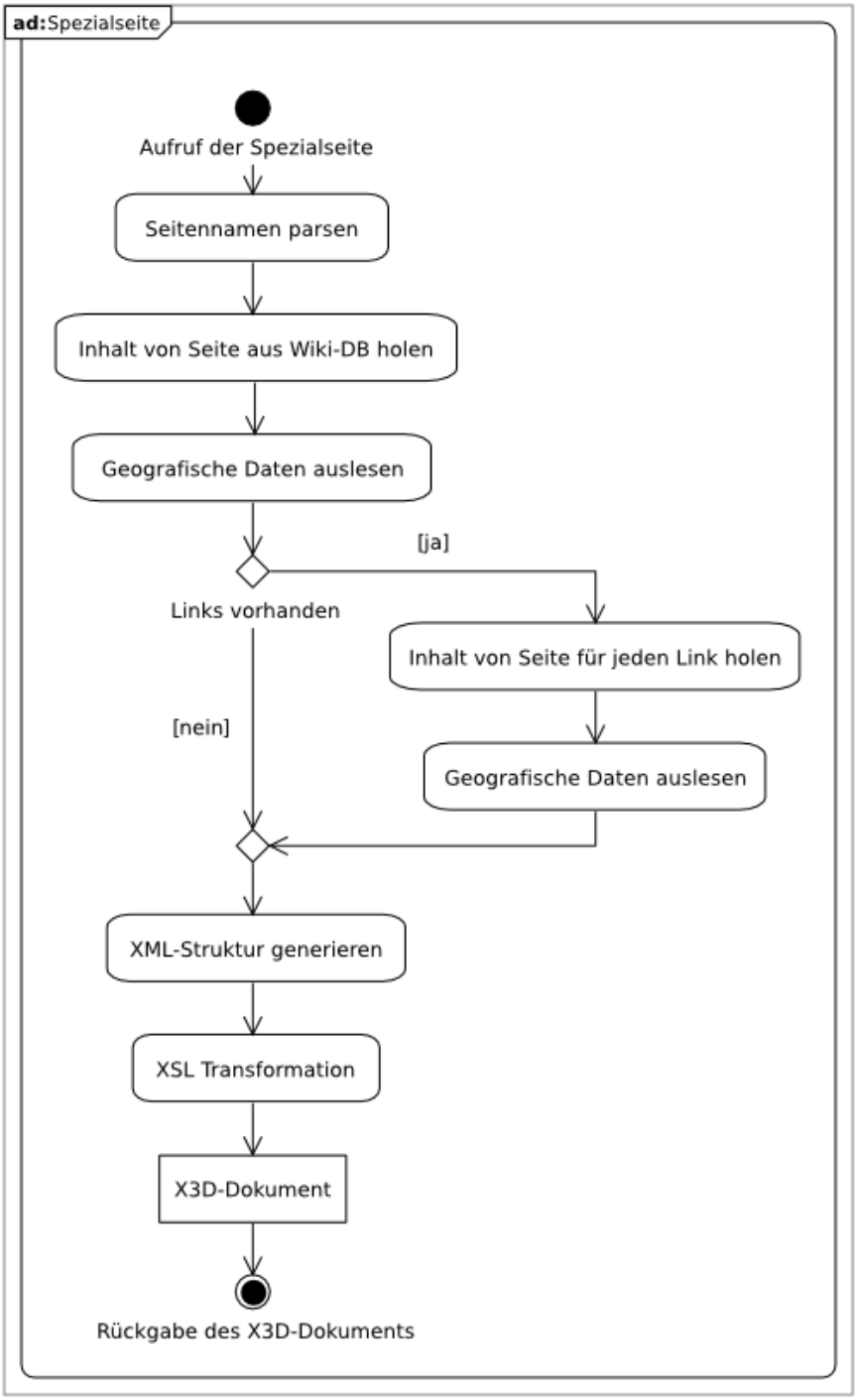

Abbildung 3.5: Ablaufdiagramm für die Abarbeitung der Spezialseite.

JavaScript

Mittels JavaScript werden sämtliche Interaktionen des Benutzers mit der X3D-Szene verarbeitet. Dies ist der Fall bei der Benutzung der Interaktionselemente, um durch einen zeitlichen Verlauf zu navigieren. Dabei wird eine Schnittstelle (`getNodeEventOut`, `setNodeEventIn`) zum X3D-Plug-in für den Datenaustausch verwendet.

X3D-Plug-in

Für die Darstellung einer X3D-Szene ist ein X3D-Plug-in notwendig. Dafür kann jedes Plug-in verwendet werden, das die X3D-Spezifikation erfüllt. Nähere Details zu vorhandenen Plug-ins sind in Kapitel 2.1.5 zu finden.

3.3.3 Benutzereingabe

Der Benutzer erstellt oder bearbeitet eine Wiki-Seite, zum Beispiel *Wien*. Zusätzlich zum textuellen Inhalt definiert der Autor die geografischen Informationen durch das Hinzufügen des XML-Elements `points`. Quelltext 3.2 zeigt eine mögliche Eingabe.

```
Description of the historical event ...

<points>
<point def="Vienna" trans="0" desc="Vienna 1858..."
  url="http://www.wien.gv.at/ma08/geschichte/whiseinl.htm"
  time="1858">
  <lat>48.208616458792086</lat>
  <long>16.372697353363037</long>
</point>
</points>

Other information ...
```

Quelltext 3.2: Beispiel einer Wikiseite mit geografischen und historischen Daten.

Nachdem die Seite bearbeitet und gespeichert wurde, werden die Daten in der Datenbank abgelegt. Bei den Eingabedaten handelt es sich um einen Wikitext.

3.3.4 Seite anzeigen

Abbildung 3.6 zeigt ein detailliertes Ablaufdiagramm für die Darstellung einer Wiki-Seite, mit und ohne geografischem Inhalt. Beim Aufruf einer Wiki-Seite erfolgt zuerst die Überprüfung, ob es sich um einen Artikel mit oder ohne Geodaten handelt. Bei einem geografischen Inhalt wird das Plug-in aufgerufen, welches die X3D-Szene anzeigt.

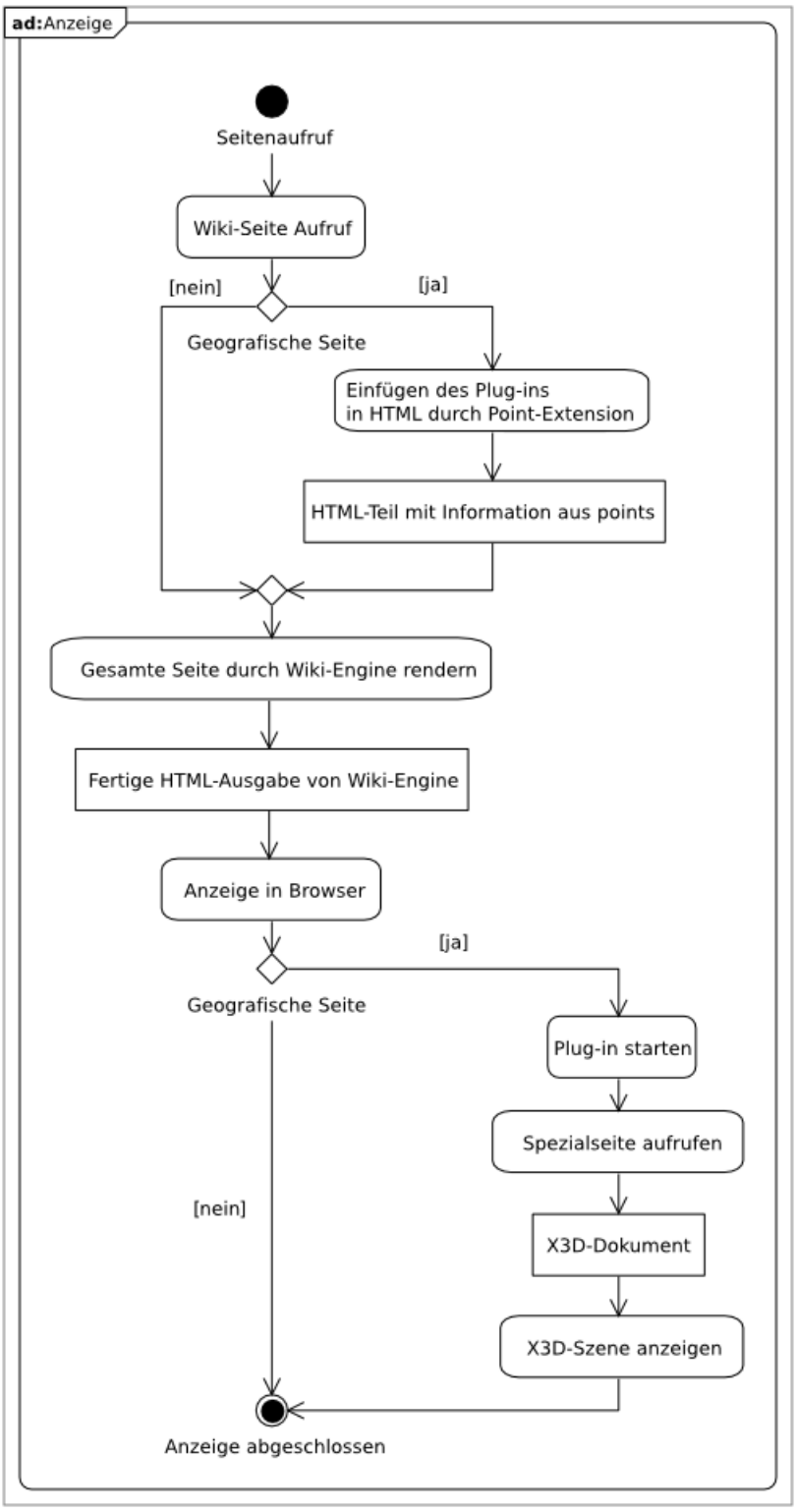

Abbildung 3.6: Ablaufdiagramm für die Anzeige einer Wiki-Seite.

4 Implementierung des technischen Systems

Diese Kapitel beschreibt die technische Umsetzung der bereits zuvor erwähnten Konzepte.

4.1 Definition von Geo-Objekten

Mögliche Geo-Objekte sind Punkt, Linie und Fläche. Für jedes dieser Objekte kann durch die Angabe von verschiedenen Parametern das Aussehen bestimmt bzw. eine Beschriftung, Beschreibung und Verlinkung hinzugefügt werden.

In den folgenden Unterkapiteln wird die Syntax der einzelnen Elemente dargestellt.

4.1.1 Punkt

Ein Punkt wird durch das Element `point` definiert und durch folgende Attribute beschrieben:

- **def:** Angabe des Titels
- **trans:** Transparenz (1 ... durchsichtig, 0 ... undurchsichtig - Dezimaler Wertebereich zwischen 0 und 1)
- **desc:** Beschreibung des Elements
- **url:** Link zu einer internen oder externen Seite
- **time:** Definiert das Jahr des historischen Ereignisses (Integer)

Das Element `point` enthält weitere Sub-Elemente wie `lat` und `long`. Wobei `lat` die Angabe des Breitengrads (latitude) und `long` die Angabe des Längengrads (longitude) beschreibt, siehe Quelltext 4.1.

```
<points>
<point def="Salzburg" trans="0.4" desc="Salzburg..."
  url="" time="2000">
  <lat>47.8001839</lat>
  <long>13.0427455</long>
</point>
</points>
```

Quelltext 4.1: Beispiel des Geo-Objekts: *Punkt*.

4.1.2 Linie

Eine Linie wird durch das Element `line` definiert. Wie bei `point` gibt es auch hier mehrere Attribute:

- **def:** Angabe des Titels
- **color:** Farbe (in RGB-Schreibweise angeben, zum Beispiel 1 0 0 - für Rot)
- **trans:** Transparenz (1 ... durchsichtig, 0 ... undurchsichtig - Dezimaler Wertebereich zwischen 0 und 1)
- **desc:** Beschreibung des Elements
- **url:** Link zu einer internen oder externen Seite
- **time:** Definiert das Jahr des historischen Ereignisses (Integer)

Eine Linie wird anhand mehrerer `point`-Elemente definiert. Dabei werden die möglichen Attribute von `point` nicht verarbeitet. Die Koordinaten aus den Sub-Elementen `lat` und `long` werden für die Darstellung einer Linie verwendet, siehe Quelltext 4.2.

```
<points>
<line def="Austria-Hungary" color="1 0 0" trans="0.3"
  desc="Dual Monarchy Austria-Hungary 1911"
  url="http://en.wikipedia.org/wiki/Austria-Hungary"
  time="1911">
  <point>
    <lat>50.15</lat>
    <long>16.7</long>
  </point>
  <point>
    <lat>50.45</lat>
    <long>16.25</long>
  </point>
  ...
</line>
</points>

Dual Monarchy Austria-Hungary in 1911 ...
```

Quelltext 4.2: Beispiel des Geo-Objekts: *Linie*.

4.1.3 Fläche

Eine Fläche wird durch das Element `area` definiert, das dieselben Attribute besitzt wie zuvor bei `line` beschrieben. Wie auch bei der Linie, wird eine Fläche durch die Angabe mehrerer Punkte (`point`) dargestellt, siehe Quelltext 4.3.

```
<points>
<area def="Luxembourg" color="0 0 1" trans="0.1"
  desc="borders of benelux countries (BEL, NL, LUX)"
  url="http://en.wikipedia.org/wiki/Benelux" time="1990">
  <point>
    <lat>49.462</lat>
    <long>6.352</long>
  </point>
  <point>
    <lat>49.567</lat>
    <long>6.361</long>
  </point>
  ...
</area>
</points>
```

Quelltext 4.3: Beispiel des Geo-Objekts: *Fläche*.

4.2 Integration in MediaWiki-Umgebung

In diesem Kapitel werden technische Details zur Implementierung der Extension und der Spezialseite beschrieben.

4.2.1 Wiki-Extension

Das Wiki ruft die Callback-Funktion `renderPoints()` bei Auftreten eines `points`-Elements auf. Diese Funktion wird beim Laden der Extension (*Point.php*) über den Event Handler registriert, siehe Quelltext 4.4. Dabei wird als erster Parameter der Name des Elements (`points`) angegeben, siehe *Codezeile 3*.

```
function wfPoints() {
    global $wgParser;
    $wgParser->setHook( "points", "renderPoints" );
}
```

Quelltext 4.4: Registrierung der Callback-Funktion.

Die Funktion `renderPoints()` ist für die HTML-Ausgabe verantwortlich. Dieser werden drei Argumente übergeben [31], siehe Quelltext 4.5:

$input: Der Inhalt, der zwischen `<points>` und `</points>` definiert wurde bzw. NULL, wenn das Element geschlossen ist (`<points/>`).

$argv: Array der Attribute des Elements `points`: `from` und `until`.

$parser: Wikitext Parser, dieser verfügt über Informationen über den gesamten Artikel. Der Parser wird verwendet um den Titel und den Wikitext zu erhalten.

In `renderPoints()` wird überprüft, ob die Attribute `from` und `until` definiert sind, siehe Codezeile *8 und 13*. Trifft dies zu, werden zusätzlich zur X3D-Ausgabe die Navigationselemente eingeblendet. Dafür wird eine weitere Funktion (`createEmbedded()`) aufgerufen, siehe Codezeile *25*. Neben den Attributen `from`, `until` und dem Status ob ein Slider eingeblendet wird oder nicht, wird auch der Name des Artikels mit übergeben.

```
function renderPoints( $input, $argv, $parser ) {
  // get title of page
  $title = $parser->mTitle->getText();

  $showSlider = true;

  // check if from and until are set, when not, show no
     slider
  if(!isset($argv["from"]) && !isset($argv["until"])) {
    // show no slider
    $showSlider = false;
    $until = "0";
    $from = "0";
  } else if(!isset($argv["until"])) {
    // if until is not set, set until to current year
    $until = date("Y");
    $from = $argv["from"];
  } else {
    $from = $argv["from"];
    $until = $argv["until"];
  }

  $site='/wiki/index.php/Special:GeoExtension/
  '.$title.'';
  // embed X3D plugin into wiki article
  $output=createEmbedded($from, $until, $showSlider,
                         $site);

  return $output;
}
```

Quelltext 4.5: Funktion *renderPoints()*.

Die Funktion `createEmbedded()` ist für die Einbindung des X3D-Plug-ins verantwortlich. Wobei als X3D-Dokument ein Verweis auf die Spezialseite mit dem Namen des Artikels als Parameter übergeben wird, siehe Quelltext 4.6 (*Codezeile 7 und 10*).

```
1  function createEmbedded($from, $until, $showSlider, $site)
2  {
3    $output = '
4      <div>
5        ...
6          <object width="720" height="540" id="X3DWorld">
7            <param name="src" value="'.$site.'"/>
8            <param name="type" value="model/x3d+xml"/>
9            <param name="align" value="center"/>
10             <embed src="'.$site.'" width="720"
11             height="540" TYPE="model/x3d+xml"
12             name="X3DWorld" style="align:center"/>
13         </object>
14       ...
15     </div>';
16
17   return $output;
18 }
```

Quelltext 4.6: Integration des X3D-Plug-ins.

In Quelltext 4.7 wird die Integration der Navigationselemente (Slider und Buttons) und die Ausgabe der Jahreszahlen dargestellt. Der Slider wird in *Codzeile 10 bis 13* definiert, wobei für die Umsetzung die *Yahoo! UI Library* eingesetzt wurde; für nähere Details siehe Kapitel 4.4.2. Als zusätzliche Möglichkeit für die Navigation können auch Buttons verwendet werden, die in *Codezeile 6-8, 15-18* und *22-27* definiert werden. Die aktuelle Jahreszahl (*Codezeile 20*) wird durch JavaScript angepasst.

```
function createEmbedded($from, $until, $showSlider, $site)
{
  ...
  if($showSlider){
    $output.='...
      <form>
        <input type="button" class="btn" name="previous"
            value="<<" onclick="moveLeft()" ...>
      </form>
      ...
      <div class="yui-skin-sam">
        ... // Definition des Sliders
        <script type="text/javascript">sliderInit('.$from.'
            ,'.$until.');</script>
      </div>
      ...
      <form>
        <input type="button" class="btn" name="later"
          value=">>" onclick="moveRight()" ...>
      </form>
      ...
      Year: <span id="slider-value">'.$from.'</span>
      ...
      <form>
        <input type="button" class="btn" name="all"
          value="Show all" onclick="showAllElements()"...>
        <input type="button" class="btn" name="animation"
          value="Animation" onclick="startAnimation()" ...>
      </form> ...';
  } ...
  return $output;
}
```

Quelltext 4.7: Definition der Navigation mittels Slider und Buttons.

4.2.2 Wiki-Spezialseite

Die Spezialseite (*Geo_extension_body.php*) ist für das Generieren der X3D-Ausgabe zuständig, siehe Quelltext 4.8. Als Erstes wird durch den übergebenen Seitennamen der Inhalt des entsprechenden Artikels geholt (*Codezeile 3*). Da diese Spezialseite nur für Geo-Definitions- oder Geo-Containerseiten aufgerufen wird, ist ein `points`-Element vorhanden. Dieses Element wird durch *Codezeile 5* extrahiert (parse_points). Alle Links des übergebenen Artikels werden aus dem Wikitext ausgelesen (*Codezeile 7*). Anschließend werden diese Links rekursiv abgearbeitet (*Codezeile 8*) und die jeweiligen Geo-Objekte verwendet. Dabei erfolgt die Durchführung anhand des Algorithmus, wie in Kapitel 3.2.3 beschrieben. Die Funktion `collectLinks()` retourniert die aneinander gefügten `points`-Elemente aller verlinkten Artikel.

Die Geo-Objekte der Verlinkungen sowie des aktuellen Artikels werden kombiniert (*Codezeile 10*). Um eine vollständige XML-Struktur zu erhalten, werden zusätzlich der XML-Header und das Wurzel-Element `geoinformation` hinzugefügt (*Codezeile 12-16*).

Auf das erhaltene XML-Dokument wird eine XSL-Transformation angewendet (*Codezeile 18*). Details zur Transformation werden im nachfolgenden Kapitel 4.3 behandelt. Die daraus resultierende X3D-Ausgabe wird an das Plug-in zurückgegeben (*Codezeile 20*).

```
1  function execute( $par ) {
2    ...
3    $content = get_content_of_article(NS_MAIN, $par);
4
5    $output1 = parse_points($content,"<points","</points>");
6
7    $link_list = parse_values($content, "[[", "]]");
8    $output = self::collectLinks($link_list);
9
10   $output = $output1.$output;
11   ...
12   $xml_doc = "<?xml version=\"1.0\"
13     encoding=\"ISO-8859-1\"?>
14   <?xml-stylesheet type=\"text/xsl\"
15     href=\"geoinformation.xsl\"?>
16   <geoinformation>".$output."</geoinformation>";
17
18   $x3d_doc = xsl_transformation($xml_doc);
19   ...
20   echo $x3d_doc;
21 }
```

Quelltext 4.8: Spezialseite *GeoExtension_body.php*.

4.3 XSL-Transformation

Nachdem die Eingabedaten in eine XML-Struktur gebracht wurden, wird aus dieser mittels XSL-Transformation ein X3D-Dokument generiert. Nachfolgend wird die Transformierung der Daten näher erläutert.

Bei der Transformation eines Punktes werden die eingegebenen Werte für Längen- und Breitengrad in kartesische Koordinaten umgerechnet, siehe Quelltext 4.9. Diese dienen zur Darstellung des Punktes auf einer Kugel. Das Bogenmaß (*rad*) von Längen- und Breitengrad muss noch jeweils berechnet werden, dies erfolgt durch

folgende Formel: $Winkel\ (in\ rad) = 2\ \pi\ lat/360$.

```
1  <xsl:template match="points/point">
2    ...
3    <Transform>
4    ...
5      <xsl:attribute name="translation">
6        <xsl:value-of select="$radius*math:cos(2*$PI*lat div
             360)*math:cos(2*$PI*long div 360)"/>
7        <xsl:text> </xsl:text>
8        <xsl:value-of select="$radius*math:sin(2*$PI*lat div
             360)"/>
9        <xsl:text> </xsl:text>
10       <xsl:value-of select="-($radius*math:cos(2*$PI*lat
             div 360)*math:sin(2*$PI*long div 360))"/>
11     </xsl:attribute>
12     ...
13   </Transform>
14 </xsl:template>
```

Quelltext 4.9: Berechnung der kartesischen Koordinaten eines Punkts.

Des Weiteren werden die vom Benutzer definierten Attribute in die dafür zuständigen X3D-Attribute geschrieben, siehe Quelltext 4.10.

Jedem X3D-Element eines Geo-Objekts wird eine ID vergeben, siehe Quelltext 4.11 (*Codezeile 8 und 18*). Damit können Objekte explizit angesprochen werden und so aus- und eingeblendet werden, wie zum Beispiel der Titel oder die Geometrie eines Objekts.

```
<xsl:template match="points/point">
  ...
    <Transform>
      <xsl:attribute name="DEF">
        <xsl:value-of select="@def"/>
      </xsl:attribute>
    </Transform>
  ...
</xsl:template>
```

Quelltext 4.10: Transformation der eingegebenen Attribute.

```
<xsl:template match="points/point">
  <xsl:variable name="cnt">
    <xsl:number level="any" count="points/point" />
  </xsl:variable>
  ...
    <Shape>
      <Appearance>
      <Material diffuseColor="1 0 0">
        <xsl:attribute name="DEF">
          <xsl:value-of select="concat('e_1_',$cnt)"/>
        </xsl:attribute>
         ...
      </Material>
      </Appearance>
      <Sphere radius="0.0015"/>
    </Shape>
  ...
</xsl:template>
```

Quelltext 4.11: Vergabe von IDs.

4.3.1 Punkt

Ist `point` ein direktes Sub-Element von `points` wird das Template `points/point` verwendet, siehe Quelltext 4.12. Diese führt die Transformation eines Punktes durch. Die Darstellung erfolgt anhand der Geometrie `Sphere` (*Codezeile 16*).

```
1  <xsl:template match="points/point">
2    ...
3    <Transform>
4      ...
5          <Shape>
6            <Appearance>
7            <Material diffuseColor="1 0 0">
8              <xsl:attribute name="DEF">
9                <xsl:value-of select="concat('e_1_',$cnt)"/>
10             </xsl:attribute>
11             <xsl:attribute name="transparency">
12               <xsl:value-of select="@trans"/>
13             </xsl:attribute>
14           </Material>
15           </Appearance>
16           <Sphere radius="0.0015"/>
17         </Shape>
18       ...
19       <Billboard axisOfRotation="0 0 0">
20         ...
21       </Billboard>
22     ...
23   </Transform>
24 </xsl:template>
```

Quelltext 4.12: Transformation eines Punkts.

4.3.2 Linie

Die Transformation von Linien erfolgt nach dem gleichen Prinzip wie zuvor bei den Punkten. Anstatt des geometrischen Elements `Sphere` wird ein `IndexedLineSet` verwendet, das sich aus einer Menge von Punkten zusammensetzt, siehe Quelltext 4.13. In *Codezeile 13-17* werden die Koordinaten der einzelnen Punkte definiert. Die Indizes aus dem Attribut `coordIndex` *Codezeile 7-12* werden verwendet, um die einzelnen Punkte miteinander zu verknüpfen und damit eine Linie zu bilden.

```
<xsl:template match="points/line">
  ...
  <Transform>
    ...
      <Shape>
        <IndexedLineSet colorPerVertex="false">
          <xsl:attribute name="coordIndex">
            <xsl:for-each select="point">
              <xsl:number level="single" count="point"
                  format="1 " value="position()-1" />
            </xsl:for-each>
            <xsl:text>-1</xsl:text>
          </xsl:attribute>
          <Coordinate>
            <xsl:attribute name="point">
              <xsl:apply-templates/>
            </xsl:attribute>
          </Coordinate>
          <Color>...</Color>
        </IndexedLineSet>
        <Appearance>
          ...
        </Appearance>
      </Shape>
    ...
  </Transform>
  ...
</xsl:template>
```

Quelltext 4.13: Transformation einer Linie.

Für jeden Punkt der Linie werden die Koordinaten durch das Template `points/line/point` berechnet, siehe Quelltext 4.14.

```
<xsl:template match="points/line/point">
    <xsl:value-of select="$radius*math:cos(2*$PI*lat div
        360)*math:cos(2*$PI*long div 360)"/>
    <xsl:text> </xsl:text>
    <xsl:value-of select="$radius*math:sin(2*$PI*lat div
        360)"/>
    <xsl:text> </xsl:text>
    <xsl:value-of select="-($radius*math:cos(2*$PI*lat div
        360)*math:sin(2*$PI*long div 360))"/>
</xsl:template>
```

Quelltext 4.14: Berechnung der einzelnen Punkte einer Linie.

4.3.3 Fläche

Eine Fläche besteht aus einer Menge von Punkten, die zu einer Fläche zusammengefügt werden. Als Geometrie wird hier kein `IndexedLineSet` sondern ein `IndexedFaceSet` verwendet. Ansonsten erfolgt die Verarbeitung wie bei der Linie.

4.3.4 Level of Detail

Bei der Transformation der Benutzereingaben wird unterschieden, ob ein Level of Detail Element (nachfolgend als `LOD`-Element bezeichnet) angegeben wurde oder nicht. Definiert der Benutzer kein `LOD`-Element, wird von der Spezialseite ein `nolod`-Element in das XML-Dokument eingefügt. Dies ist notwendig, um im Stylesheet die Textur der Erdoberfläche einzufügen, siehe Quelltext 4.15.

```
<xsl:template match="points/nolod">
  <xsl:if test="count(//points/lod) = 0">
    <Group DEF="Earth" containerField="children">
      <xsl:call-template name="world_tex"/>
    </Group>
  </xsl:if>
</xsl:template
```

Quelltext 4.15: Verarbeitung bei nicht Definition des *LOD*-Elements.

Wird ein `LOD`-Element definiert, werden die kartesischen Koordinaten der vier Eckpunkte der detaillierten Karte berechnet. Wie in Kapitel 2.1.6 beschrieben, werden unterschiedlich detaillierte Geometrien eingeblendet. Die Unterscheidung erfolgt durch das Attribut `range`, siehe Quelltext 4.16 (*Codezeile 3*). Dabei wird außerhalb dieses Bereichs die gesamte Textur der Erde angezeigt (*Codezeile 20-22*). Zoomt der Benutzer jedoch innerhalb dieses Bereichs, blendet das detailliertere Kartenmaterial (*Codezeile 5-17*) und die gesamte Textur der Erdoberfläche ein (*Codezeile 18*).

```
<xsl:template match="points/lod">
  <Group DEF="Earth" containerField="children">
    <LOD center="0 0 0" range="1.2">
      <Group>
        <Shape >
          <IndexedFaceSet solid="false">
            ...
          </IndexedFaceSet>
          <Appearance>
            <ImageTexture containerField="texture"
              repeatS="true" repeatT="true">
              <xsl:attribute name="url">
                <xsl:value-of select="@urlLOD"/>
              </xsl:attribute>
            </ImageTexture>
          </Appearance>
        </Shape>
        <xsl:call-template name="world_tex"/>
      </Group>
      <Group>
        <xsl:call-template name="world_tex"/>
      </Group>
    </LOD>
  </Group>
</xsl:template>
```

Quelltext 4.16: XSL-Transformation des LOD-Elements.

4.4 Navigation

Die Applikation bietet verschiedene Navigationsmöglichkeiten: Navigation innerhalb des Plug-ins, Interaktionselemente (Slider, Buttons) oder durch eine erweiterte Suche nach Zeitperioden.

4.4.1 X3D-Plug-in

Ein X3D-Plug-in bietet eine Vielzahl von Interaktionsmöglichkeiten mit der X3D-Welt. Es besteht die Möglichkeit die Art der Bewegung zu definieren (gehen, gleiten, betrachten, fliegen, etc.) und die gewünschte Geschwindigkeit festzulegen. Durch definierte Viewpoints kann die X3D-Szene aus verschiedenen Positionen betrachtet werden. Innerhalb der XSL-Transformation werden diese Viewpoints eingefügt, wie in Quelltext 4.17 dargestellt.

```
...
<Transform DEF="Europe" rotation="0 1 0 0.3">
  <Transform rotation="0 0 1 0.9">
    <Viewpoint description="Europe"
      orientation="0 1 0 1.57" position="2 0 0"/>
  </Transform>
</Transform>
...
```

Quelltext 4.17: Definition von Viewpoints im XSL-Dokument.

4.4.2 Navigationselemente

Um die historische Entwicklung der Welt verfolgen zu können, muss den einzelnen Geo-Objekten eine zeitliche Komponente hinzugefügt werden. Dies erfolgt durch das Verwenden des Attributs `time`. In Quelltext 4.18 wird die Integration des Zeitattributs dargestellt (*Codezeile 3*).

```
<points>
<point def="Graz" trans="0" desc="Graz ..." url=""
   time="1988">
   <lat>...</lat>
   <long>...</long>
</point>
</points>
```

Quelltext 4.18: Geo-Objekt mit dem Attribut *time*.

Durch die Angabe der Zeit ist es möglich die Geo-Objekte zeitlich zu sortieren. Diese chronologische Sortierung dient als Basis für die Visualisierung des zeitlichen Verlaufs. Durch die Definition der Attribute `from` und `until` bei Geo-Containerseiten werden nur jene Geo-Objekte der verlinkten Seiten angezeigt, deren `time`-Attribut innerhalb des Intervalls liegt, siehe Quelltext 4.19.

```
<points from="1950" until="2008"></points>

[[Graz]]
[[Innsbruck]]
```

Quelltext 4.19: Geo-Containerseite mit den Attributen *from* und *until*.

Werden die beiden Attribute `from` und `until` angegeben, fügt die Extension die Navigationselemente (Slider und Buttons) ein. Für die Implementierung des Sliders wurde die *Yahoo! UI Library* [58] eingesetzt, da HTML keine Slider-Funktionalität anbietet. Beim Aufruf des Sliders werden die Werte der beiden Attribute `from` und `until` übergeben, siehe Quelltext 4.20 (*Codezeile 7 und 8*).

```
<div class="yui-skin-sam">
  <div id="slider-bg" tabindex="-1" title="Slider">
    <div id="slider-thumb">
      <img src="/wiki/extensions/GeoExtension/thumb-n.gif">
    </div>
  </div>
  <script type="text/javascript">sliderInit('.$from.',
    '.$until.');</script>
</div>
```

Quelltext 4.20: Aufruf der Slider-Funktion.

Beim Initialisieren des Sliders werden diese für den Minimal- und Maximalwert herangezogen, wie in Quelltext 4.21 dargestellt. Die Skalierung des Sliders erfolgt je nach Zeitspanne. Das heißt, je größer diese Spanne ist, desto größer ist auch die Schrittweite beim Navigieren. Bei jeder Zustandsänderung des Sliders wird die Funktion `changeEvent()` aufgerufen (*Codezeile 11*).

```
1  function sliderInit(min, max) {
2    minYear = min;
3    maxYear = max;
4    var diff = max - min;
5    var bgWidth = Dom.get(bg);
6    factor = 400/diff;
7
8    bgWidth.style.width=""+(Math.ceil(diff*factor)+17)+"px";
9    slider = YAHOO.widget.Slider.getHorizSlider(bg, "slider-
       thumb", 0, Math.ceil(diff*factor), Math.ceil(factor));
10   useSlider = true;
11   slider.subscribe("change", changeEvent);
12 }
```

Quelltext 4.21: Funktion *sliderInit()*.

In dieser Funktion wird das neue Jahr berechnet und im Browser angezeigt, siehe Quelltext 4.22 (*Codezeile 8-9*). Des Weiteren wird das Jahr der Funktion `showYear()` als Parameter übergeben (*Codezeile 7*).

```
1  function changeEvent(offsetFromStart) {
2    if (offsetFromStart > Math.ceil(diff*factor))
3      slider.setValue(Math.ceil(diff*factor));
4
5    var year = Math.ceil(minYear + (offsetFromStart/factor));
6
7    showYear(year);
8    var valnode = Dom.get(valuearea);
9    valnode.innerHTML = year;
10 }
```

Quelltext 4.22: Funktion *changeEvent()*.

Für die Änderung der X3D-Szene ist die Funktion `showYear()` zuständig, die sich aufgrund der veränderten Slider-Position ergibt. Wird also der Slider verschoben, werden die entsprechenden Geo-Objekte aus- bwz. eingeblendet. Durch die erfolgte Vergabe von IDs bei der XSL-Transformation (siehe Kapitel 4.3) ist jedes Objekt eindeutig ansprechbar.

Die JavaScript Funktionen `getNodeEventOut(nodeName, eventOutName)` und `setNodeEventIn(nodeName, eventInName, value)` ermöglichen das Auslesen und Verändern einzelner Elemente der X3D-Szene.

4.4.3 Suche nach Zeitperioden

Durch eine erweiterte Suche ist es möglich, sich nur jene historische Ereignisse (Geo-Objekte) anzeigen zu lassen, die der eingegebenen Zeitperiode entsprechen. Diese erweiterte Suchmöglichkeit wurde durch zwei Spezialseiten implementiert. In der ersten (*SearchTimePeriodExtension_body.php*) wird zunächst die Eingabemaske definiert. Die Verarbeitung der Formulardaten wird von der zweiten Spezialseite

(*GeoSearchTimePeriodExtension_body.php*) übernommen. Diese ist von der Funktionalität ähnlich der Spezialseite in Kapitel 4.2.2. Der Unterschied besteht darin, dass bei dieser Seite sämtliche im Wiki vorhandene Artikel betrachtet werden und nicht nur die verlinkten. Es werden jedoch nur jene Geo-Objekte angezeigt, die sich innerhalb des Intervalls befinden, das der Benutzer über die Suchmaske definiert hat.

5 Anwendungsbeschreibung

Dieses Kapitel beschreibt die Verwendung der Applikation aus Benutzersicht. Es werden mögliche Anwendungen der Geo-Objekte innerhalb der verschiedenen Seitentypen (Geo-Definitionsseite, Geo-Containerseite) dargestellt. Des Weiteren werden unterschiedliche Navigationsmöglichkeiten gezeigt. Abschließend werden technische Voraussetzungen für die Inbetriebnahme der Applikation angeführt.

5.1 Erstellung von Artikeln

Nachfolgend wird das Erstellen von historischen Artikeln mithilfe von Geo-Objekten, anhand eines konkreten Beispiels, diskutiert. Dabei werden die in Kapitel 3.2.2 vorgestellten Seitentypen im praktischen Zusammenhang behandelt.

In Quelltext 5.1 wird ein Artikel (*Austria*) gezeigt, in dem verschiedene Seitentypen zum Einsatz kommen. Zuerst werden die Landesgrenzen von *Austria* durch das Geo-Objekt `line` definiert (*Codezeile 2-18*). Danach wird zu anderen Artikeln verlinkt: *Salzburg*, *Mozart* und *Cities of Austria*.

Anhand der Abbildung 5.1 ist ersichtlich, dass es sich bei *Salzburg* um eine Geo-Definitionsseite, bei *Cities of Austria* um eine Geo-Containerseite und bei *Mozart* um eine nicht geografische Seite handelt. Aus der Skizzierung ist weiters ersichtlich, dass *City of Austria* auf die Geo-Definitionsseiten *Graz* und *Innsbruck* verlinkt. Die verstärkte durchgezogene Linie markiert, welche Artikel auf dem Weltatlas dargestellt werden.

Abbildung 5.2 zeigt die dazugehörige grafische Darstellung in X3D. Wobei *Mozart* nicht dargestellt wird, da es sich um eine nicht geografische Seite handelt.

In den nachfolgenden Kapiteln wird näher auf die einzelnen Artikel eingegangen.

```
<points>
<line def="Austria" color="0 1 1" trans="0.3"
  desc="borders of Austria" url="http://www.oesterreich.at"
  time="2007">
  <point>
    <lat>48.718</lat>
    <long>16.901</long>
  </point>
  <point>
    <lat>48.814</lat>
    <long>16.540</long>
  </point>
  ...
  <point>
    <lat>48.718</lat>
    <long>16.901</long>
  </point>
</line>
</points>

A beautiful city of Austria is [[Salzburg]], which is
known for the famous composer [[Mozart]].
Further cities can be found under [[Cities of Austria]].
```

Quelltext 5.1: Artikel mit geografischem Inhalt (Geo-Objekt) und Verlinkungen auf weitere Artikel.

5.1.1 Geo-Definitionsseite

Der Wikitext zum Artikel *Salzburg* wird in Quelltext 5.2 aufgelistet. Dieser beinhaltet für die Darstellung der Stadt das Geo-Objekt `point`. Mittels der Definition des Attributs `time` wird den geografischen Daten eine zeitliche Information hinzugefügt. Dadurch ist es möglich die einzelnen Geo-Objekte zeitlich anzusprechen. Das ist für zeitliche Verläufe und der Suche nach Zeitperioden erforderlich.

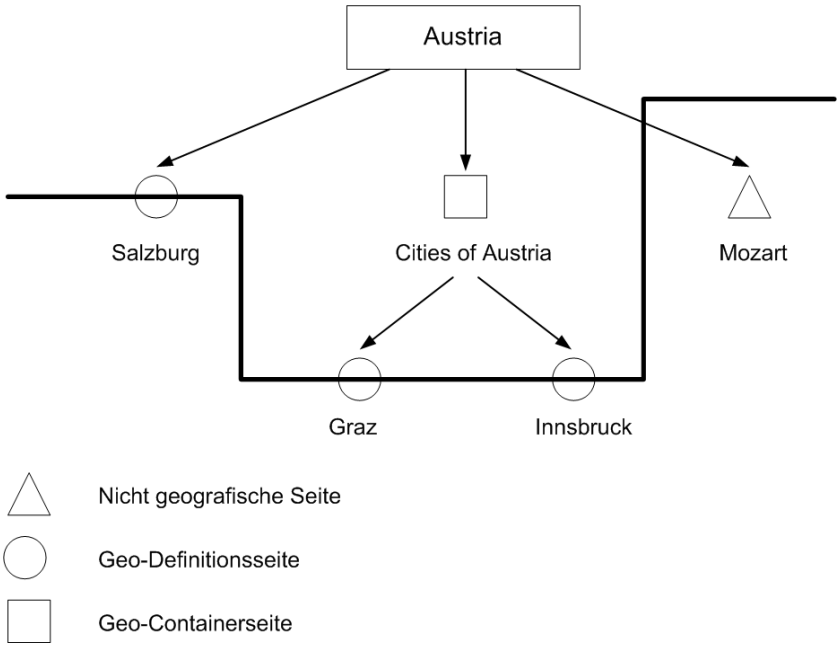

Abbildung 5.1: Verlinkungsbaums für die grafische Darstellung von *Austria*.

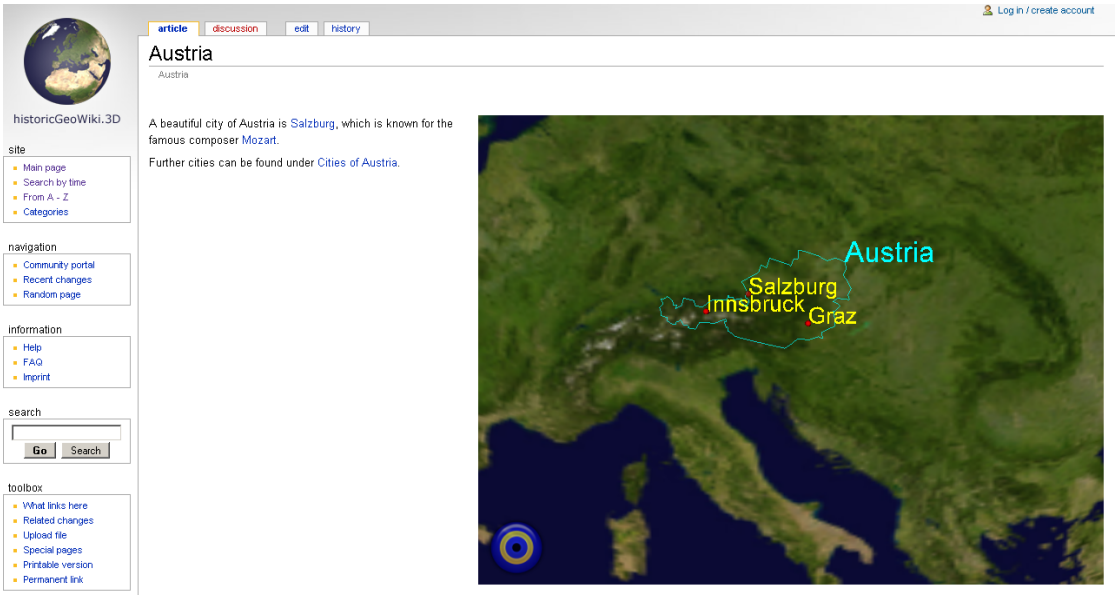

Abbildung 5.2: Grafische Darstellung des Artikels *Austria*.

```
<points>
  <point def="Salzburg" trans="0.4" desc="Salzburg..."
    url="" time="2000">
    <lat>47.8001839</lat>
    <long>13.0427455</long>
  </point>
</points>
```

Quelltext 5.2: Wikitext der Geo-Definitionsseite *Salzburg*.

5.1.2 Geo-Containerseite

In Quelltext 5.3 wird die Darstellung der Geo-Containerseite *Cities of Austria* aufgelistet. Die dreidimensionale Darstellung wird hier aus den verlinkten Seiten (*Graz* und *Innsbruck*) generiert. Dabei wird jede verlinkte Seite, wie in Kapitel 3.2.3 beschrieben, verarbeitet und die darin definierten Geo-Objekte angezeigt. Die beiden Geo-Definitionsseiten *Graz* und *Innsbruck* beinhalten jeweils das Geo-Objekt `point`, siehe Abbildung 5.3.

```
<points></points>

[[Graz]]
[[Innsbruck]]
```

Quelltext 5.3: Wikitext der Geo-Containerseite *Cities of Austria*.

Zeitlicher Verlauf

Durch die Verwendung einer Geo-Containerseite ist es möglich, bestehende Artikel für einen zeitlichen Verlauf heranzuziehen. In Quelltext 5.4 wird ein Beispiel (*History of Austria*) dazu dargestellt. Dieser beinhaltet Links auf die Artikel *Austria*, *Austria-Hungary 1911*, *Austria 1935* und *Austria 1885*.

Abbildung 5.4 zeigt den Verlinkungsbaum für die grafische Darstellung. Durch das Hinzufügen der beiden Attribute `from` und `until` wird ein Zeitintervall defi-

Abbildung 5.3: Grafische Darstellung des Artikels *Cities of Austria*.

niert, siehe Quelltext 5.4 (*Codezeile 1*). Es werden nur jene Geo-Objekte der verlinkten Artikel auf dem Weltatlas dargestellt, die sich innerhalb dieses Intervalls befinden, diese sind *Austria*, *Austria 1911* und *Austria 1935*. Der Artikel *Austria 1885* wird nicht dargestellt, da das Attribut *time* nicht innerhalb des Zeitbereichs liegt, siehe Quelltext 5.5 (*Codezeile 2*).

Die Visualisierung des Artikels *History of Austria* wird in Abbildung 5.5 dargestellt.

```
1 <points from="1911" until="2007"></points>
2
3 [[Austria]]
4 [[Austria-Hungary 1911]]
5 [[Austria 1935]]
6 [[Austria 1885]]
```

Quelltext 5.4: Artikel *History of Austria* mit zeitlichem Verlauf.

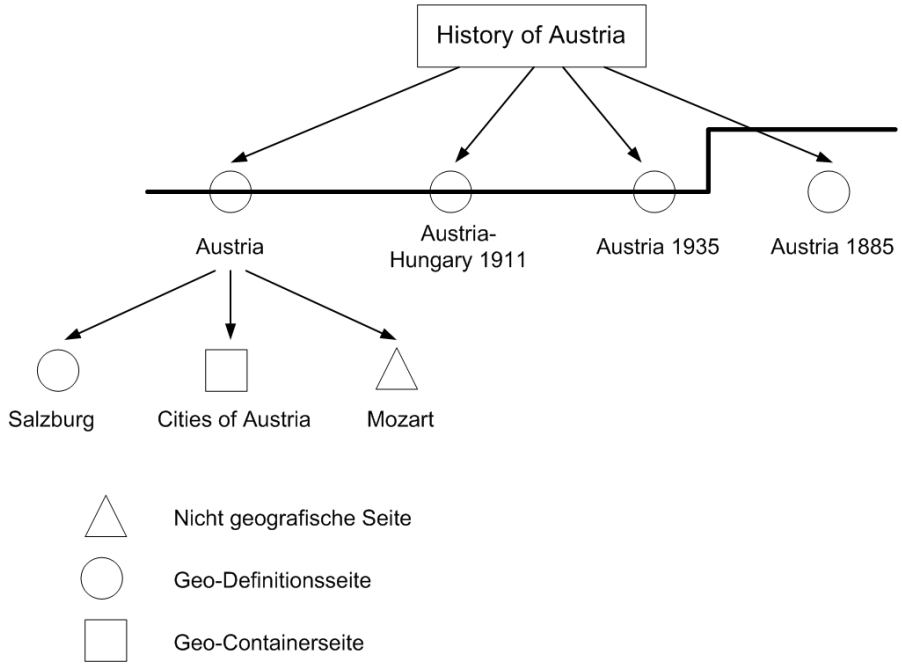

Abbildung 5.4: Verlinkungsbaum für die grafische Darstellung von *History of Austria*.

```
<points>
<line def="Austria 1885" color="1 1 0" trans="0.3" desc="
    Austria 1885 borders" url="" time="1885">
  <point>
    <lat>48.718</lat>
    <long>16.901</long>
  </point>
  ...
  <point>
    <lat>47.739</lat>
    <long>12.255</long>
  </point>
</line>
</points>
```

Quelltext 5.5: Wikitext des Artikels *Austria 1885*.

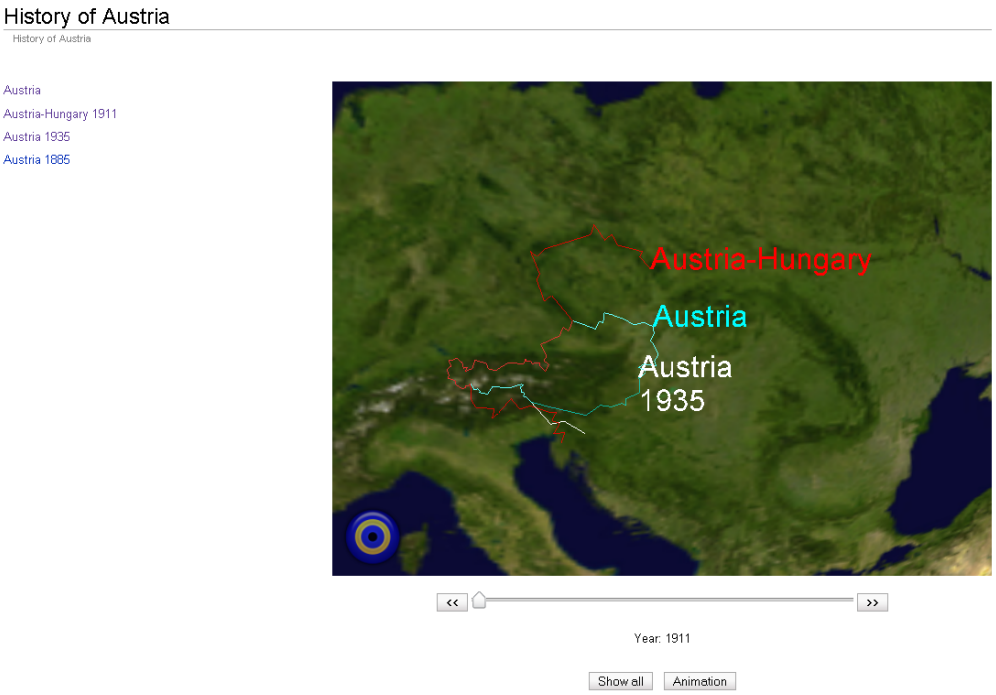

Abbildung 5.5: Visualisierung des Artikels *History of Austria*.

5.1.3 Nicht geografische Seite

Auf einer nicht geografischen Seite befinden sich textuelle Informationen mit eventuell hinzugefügten zweidimensionalen Abbildungen, wie der Artikel *Mozart* von Quelltext 5.1. Dieser beinhaltet kein `points`-Element und damit auch keine Informationen für die dreidimensionale Darstellung.

5.2 Navigation

Dieses Kapitel beschreibt die einzelnen Navigationsmöglichkeiten innerhalb der Benutzerschnittstelle.

5.2.1 X3D-Plug-in

Die Funktionalitäten von X3D-Plug-ins sind je nach Anbieter unterschiedlich. Die nachfolgende Beschreibung erfolgt anhand des Plug-ins *BS Contact VRML/X3D*.

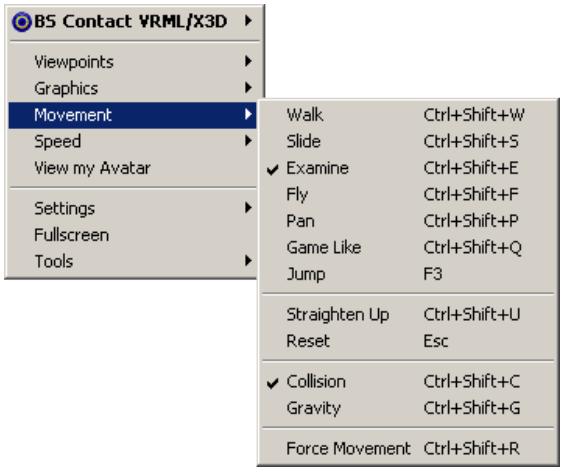

Abbildung 5.6: Mit dem Kontextmenü eines X3D-Plug-ins können Einstellungen für die Navigation vorgenommen werden.

[4] Dieses Plug-in wurde bei der Erstellung der Applikation verwendet.

Beim Klick mit der rechten Maustaste ins Plug-in öffnet sich ein Kontextmenü, siehe Abbildung 5.6. Dieses bietet verschiedene Auswahlmöglichkeiten für den Benutzer an. So kann zwischen unterschiedlichen Viewpoints ausgewählt werden. Dabei ist es möglich, diese mittels Viewpoint Tour aufeinander folgend abzuspielen und zwischen den einzelnen Viewpoints nach vor und zurück zu navigieren (Prev Viewpoint, Next Viewpoint). Des Weiteren werden verschiedene Darstellungsweisen der X3D-Szene (Drahtmodel, Knotenmodel, Festkörper, ...) und eine Auswahl von unterschiedlichen Bewegungsarten (*Movement*) angeboten.

5.2.2 Navigationselemente

Durch Slider und Buttons ist ein Navigieren zwischen den einzelnen Geo-Objekten möglich. Mit den Vorwärts- und Rückwärtsbuttons wird der zeitliche Verlauf schrittweise verändert, siehe Abbildung 5.7. Die Anzeige von allen Geo-Objekten ist durch den Button *Show all* umgesetzt, mittels des Buttons *Animation* werden die einzelnen Objekte automatisch in chronologischer Reihenfolge durchlaufen.

Abbildung 5.7: Interaktionselemente: Slider und Buttons.

5.2.3 Suche nach Zeitperioden

Durch die Angabe von Zeitperioden kann nach geografischen Objekten mit zeitlichem Hintergrund gesucht werden. Dazu ist es erforderlich die beiden Eingabefelder `from` und `until` auszufüllen. Die Applikation durchsucht alle Artikel um jene anzuzeigen, die sich innerhalb dieser Zeitspanne befinden. Die Navigation erfolgt, wie zuvor beschrieben, durch den Slider bzw. den Buttons. In Abbildung 5.8 wird die Darstellung eines Suchergebnisses gezeigt.

5.2.4 Level of Detail

Durch die Verwendung des `LOD`-Elements kann detaillierteres Kartenmaterial integriert werden. Dabei wird neben der Definition eines Geo-Objekts zusätzlich das `LOD`-Element eingebunden, siehe Quelltext 5.6 (*Codezeile 2-19*). Durch das Attribut `urlLOD` (*Codezeile 2*) wird der Pfad zur einzublendenden Textur definiert. Der Benutzer muss zuvor das Kartenmaterial über die MediaWiki-Funktionalität *Datei hochladen* auf den Server laden. Durch die Angabe von den Koordinaten der vier Eckpunkte der Karte (*Codezeile 3-18*) ist die Applikation imstande das neue Bild auf der Weltkugel einzufügen. In Abbildung 5.9 wird dazu ein Beispiel dargestellt. Das linke Bild zeigt eine X3D-Szene ohne Level of Detail. Im rechten Bild wird derselbe Ausschnitt nochmals dargestellt, jedoch mit detaillierterem Kartenmaterial. Dies wird erreicht, indem der Benutzer eine bestimmte Zoomtiefe erreicht.

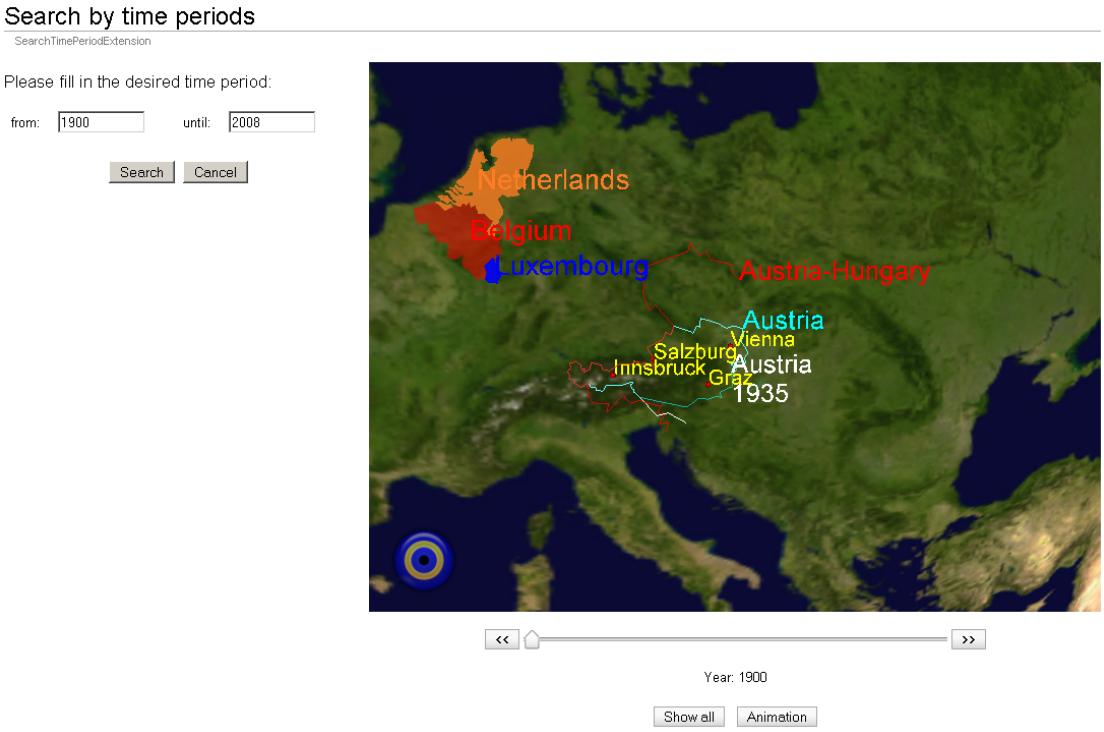

Abbildung 5.8: Darstellung des Suchergebnisses nach einer Zeitperiode.

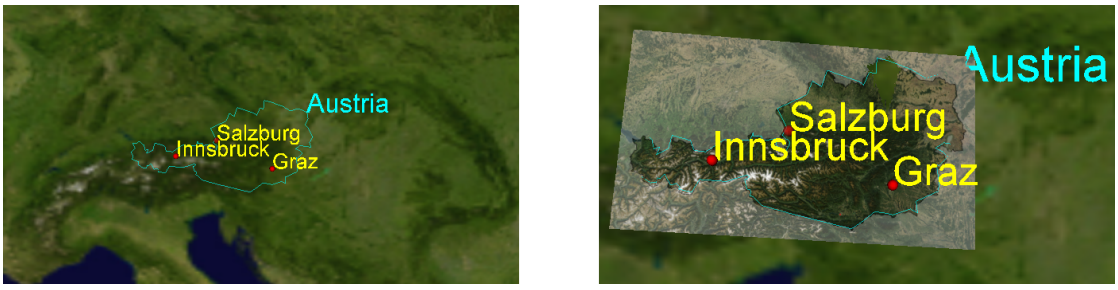

Abbildung 5.9: Beispiel einer Verwendung von Level of Detail in einer X3D-Szene.

```
<points>
  <lod urlLOD="/wiki/images/Austria.jpg">
    <point>
      <lat>46.10</lat>
      <long>9.35</long>
    </point>
    <point>
      <lat>46.10</lat>
      <long>17.35</long>
    </point>
    <point>
      <lat>49.10</lat>
      <long>17.35</long>
    </point>
    <point>
      <lat>49.10</lat>
      <long>9.35</long>
    </point>
  </lod>
  <line def="Austria" color="0 1 1" trans="0.3"
    desc="borders" url="http://www.oesterreich.at"
    time="2007">
    <point>
      <lat>48.718</lat>
      <long>16.901</long>
    </point>
    ...
  </line>
</points>
```

Quelltext 5.6: Artikel mit integriertem Level of Detail (LOD).

5.3 Technische Voraussetzungen

Die Voraussetzungen zur Installation und Benutzung der Applikation auf Server- und Clientseite werden in diesem Kapitel näher erläutert.

5.3.1 Anforderungen an die Inbetriebnahme der Applikation am Server

Installation von MediaWiki

Die Applikation basiert auf MediaWiki Version 1.10.0. Um MediaWiki erfolgreich zu installieren sind folgende Anforderungen gegeben: [28]

- MediaWiki ist in PHP programmiert, für diese Version ist die Verwendung von PHP 5 erforderlich.
- Apache oder ISS kann als Webserver eingesetzt werden. Für die Entwicklung wurde Apache 2 (Version 2.2.3) verwendet.
- Als Datenbank kann MySQL oder PostgreSQL zum Einsatz kommen. Für die Applikation wurde mit MySQL 5 (Version 5.0.27) gearbeitet.

Nähere Details zur Installation sind unter [29] zu finden.

Erweiterungen

Um die Extension und die Spezialseiten in MediaWiki zu installieren, müssen diese in der Datei *LocalSettings.php*, zu finden im Wurzelverzeichnis, angegeben werden, siehe Quelltext 5.7.

```
require_once("$IP/extensions/GeoExtension/Point.php");
require_once( "$IP/extensions/GeoExtension/
  GeoExtension.php");
require_once( "$IP/extensions/GeoExtension/
  SearchTimePeriodExtension.php");
require_once( "$IP/extensions/GeoExtension/
  GeoSearchTimePeriodExtension.php");
```

Quelltext 5.7: Installation der Erweiterungen: Extension und Spezialseiten.

Bilder hochladen

Das Laden von Bildern in MediaWiki ist standardmäßig deaktiviert. Durch Änderungen in den Konfigurationen kann diese Funktionalität jedoch aktiviert werden. Nähere Details zu den Einstellungen und zur Verwendung der hoch geladenen Bilder im Wiki sind unter [26] und [27] zu finden.

5.3.2 Anforderungen an den Client

Betriebssystem

Die Applikation ist unabhängig vom Betriebssystem, da das System als Web-Applikation konzipiert wurde. Als Entwicklungsumgebung wurde Windows XP verwendet.

Browser

Der 3D-Atlas kann in allen gängigen Browsern benutzt werden, sofern ein X3D-Plug-in installiert wird. Für die Entwicklung kam Internet Explorer 7 zum Einsatz.

X3D-Plug-in

Für die Darstellung des X3D-Dokuments ist ein X3D-Plug-in erforderlich. In Kapitel 2.1.5 werden verschiedene Plug-ins vorgestellt. Für die Entwicklung der Applikation wurde das Plug-in *BS Contact VRML/X3D* [4] verwendet.

6 Schlussfolgerungen

Im Laufe der Arbeit hat sich die Komplexität und Vielseitigkeit dieses Themengebiets herausgestellt, vor allem durch die Kombination unterschiedlicher Technologien und das Beachten verschiedener Aspekte, die bei der Erstellung eines historischen Weltatlasses erforderlich sind. Dabei waren Kenntnisse in Web 2.0, in der Visualisierung von Objekten in X3D und in der Verarbeitung von geografischen Informationen erforderlich.

Die Herausforderungen lagen unter anderem in der Darstellung der Geo-Objekte auf einer dreidimensionalen Kugel. Die X3D-Spezifikation bietet spezielle Funktionalitäten (Geospatial-Komponente) für die Visualisierung von geografischen Daten, jedoch gibt es keine Plug-ins, die diese auch in einem Webbrowser unterstützen. Bei der Weiterentwicklung von X3D-Plug-ins besteht ein großer Nachholbedarf, obwohl Intentionen für eine Implementierung dieser X3D-Funktionalitäten bestehen, siehe BS Contact Geo.

Die Applikation wurde in ein bestehendes Wiki (MediaWiki) integriert. Dabei war ein weiteres Kriterium die Generierung einer Struktur für die Darstellung in X3D aus den Wiki-Daten. Die Verwendung einer Extension ermöglichte das Einbetten der Daten innerhalb des Wikitexts. Zum Konstruieren komplexer geografischer Darstellungen wurde auf das Grundprinzip von Wikis, dem Verlinken, zurückgegriffen. Dabei wird die Visualisierung aus den Inhalten der verlinkten Artikel generiert.

Eine zukünftige Erweiterung der Applikation besteht in der Erhöhung der Genauigkeit bei der Darstellung von Geo-Objekten sowie bei der Integration von detaillierterem Kartenmaterial. Bei ausreichender Unterstützung der X3D-Spezifikation seitens der X3D-Plug-ins, würde eine Restrukturierung des Systems auf die Geospatial-Komponente höhere Flexibilität bringen. Eine sehr hilfreiche Funktionalität wäre das Integrieren einer Benutzerschnittstelle, die es ermöglicht, die

Koordinaten für Geo-Objekte per Mausklick auf den 3D-Globus einzugeben, anstatt diese direkt in den Wikitext eintragen zu müssen.

Der X3D-Standard hat sich noch nicht etabliert. Jedoch lassen die unterschiedlichen Arbeitsgruppen des Web3D-Konsortiums und den daraus resultierenden Spezifikationen für unterschiedlichen Einsatzgebiete auf eine interessante Zukunft hoffen. Projekte wie Wikipedia haben gezeigt, dass durch die Einbeziehung vieler Benutzer eine große Datensammlung geschaffen werden kann. Durch die Verwendung ähnlicher Technologien kann dieses Potenzial für diese Applikation ausgeschöpft werden, um eine umfangreiche historische Datenbasis, verknüpft mit geografischen Informationen, zu erhalten.

Abbildungsverzeichnis

Quelltextverzeichnis

Literaturverzeichnis

[1] Asbeck, Axel und Sebastian Beyl: *Stereoskopische Visualisierung des VRML Subsets von X3D auf der Basis einer DirectX Architektur.* Diplomarbeit, Universität Köln, 2006.

[2] Bartelme, Norbert: *Geoinformatik : Modelle, Strukturen, Funktionen.* Springer, Berlin ; Heidelberg [u.a.], 4., vollst. überarb. Auflage, 2005.

[3] Behme, Henning: *Das letzte Siebtel - Interaktivität und Anwendungsfelder.* iX Special, (1):6–7, 2007.

[4] Bitmanagement Software GmbH: *BS Contact VRML / X3D, BS Contact Geo.* `http://www.bitmanagement.com/index.de.html`, Stand vom 10.02.2008.

[5] Bullard, Len: *Extensible 3D: XML Meets VRML*, 2003. Online Version: `http://www.xml.com/pub/a/2003/08/06/x3d.html`, Stand vom 10.02.2008.

[6] Carey, Rikk: *The Virtual Reality Modeling Language Explained.* IEEE MultiMedia, 5(3):84–93, 1998.

[7] Carey, Rikk und Gavin Bell: *The annotated VRML 2.0 reference manual.* Addison-Wesley Longman Ltd., Essex, UK, 1997.

[8] de Lange, Norbert: *Geoinformatik in Theorie und Praxis.* Springer, Berlin; Heidelberg [u.a.], 2., aktualisierte u. erw. Auflage, 2006.

[9] Dässler, Rolf: *Das Einsteigerseminar VRML.* BHV Verlag, Kaarst, 1999.

[10] Ebersbach, Anja, Markus Glaser und Richard Heigl: *Wiki-Tools: Kooperation im Web.* Springer, Berlin, 2005.

[11] Eibl, Maximilian: *X3D-Standard: 3D im Internet.* tecChannel, 2001. `http://www.tecchannel.de/entwicklung/programmierung/401713`, Stand vom 10.02.2008.

[12] Elza, Dethe: *XML Matters: The Web ain't just for 2D any more.* 2005. `http://www-128.ibm.com/developerworks/web/library/x-matters43`, Stand vom 10.02.2008.

[13] Geroimenko, Vladimir und Chaomei Chen: *Visualizing Information Using SVG and X3D.* Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2004.

[14] Google Earth: *Erforschen, suchen und entdecken.* `http://earth.google.com`, Stand vom 10.02.2008.

[15] Google Earth: *What is KML?* `http://earth.google.com/kml/whatiskml.html`, Stand vom 10.02.2008.

[16] Kiehle, Christian: *Lokales abstrakt - Geodaten standardisiert integrieren.* iX Magazin für profenssionelle Informationstechnik, Seiten 50–55, Dezember 2006.

[17] Kloss, Jörg, Robert Rockwell, Kornél Szabó und Martin Duchrow: *VRML97 - Der neue Standard fuer interaktive 3D-Welten im World Wide Web.* Addison Wesley Longman Verlag GmbH, Bonn, Screen-Multimedia Auflage, 1998.

[18] Krone, Oliver: *Webfähige interaktive 3D-Visualisierung von Proteinstrukturen.* Diplomarbeit, Universität Osnabrück, 2003.

[19] Kunze, Michael: *Verflochtenes Leben Web 2.0 - der nächste Schritt.* c´t Magazin für Computer Technik, (1):174–178, 2006.

[20] Lake, Ron: *Introduction to GML Geography Markup Language.* `http://www.w3.org/Mobile/posdep/GMLIntroduction.html`, Stand vom 10.02.2008.

[21] Lange, Christoph: *Wikis und Blogs. Planen. Einrichten. Verwalten.* C & L Computer- U. Literaturverlag, 2007.

[22] Löhr, Stefan: *Thematische 3D-Kartographie unter Verwendung von CommonGIS und Google Earth.* Diplomarbeit, Fachhochschule Mainz Fachbereich Geoinformatik, 2006.

[23] Matsuba, Stephen N. und Alan Hudson: *The Xj3D Toolkit: A Position Paper.* `http://w3-mmt.inf.tu-dresden.de/web3d2003/Yumetech-X3D-Tool-PositionPaper.pdf`, Stand vom 10.02.2008.

[24] Mayer, Katja: *Extended 3D (X3D) - Untersuchung der vom Web3D-Konsortium vorgelegten Spezifikation zur Erstellung von 3D-Welten in XML.* Diplomarbeit, Fachhochschule Karlsruhe, 2002.

[25] Media Machines: *Flux Player.* `http://www.mediamachines.com`, Stand vom 10.02.2008.

[26] MediaWiki: *Manual:Configuring file uploads.* `http://www.mediawiki.org/wiki/Manual:Configuring_file_uploads`, Stand vom 10.02.2008.

[27] MediaWiki: *Manual:Image Administration.* `http://www.mediawiki.org/wiki/Manual:Image_Administration`, Stand vom 10.02.2008.

[28] MediaWiki: *Manual:Installation requirements.* `http://www.mediawiki.org/wiki/Manual:Installation_requirements`, Stand vom 10.02.2008.

[29] MediaWiki: *Manual:Installing MediaWiki.* `http://www.mediawiki.org/wiki/Manual:Installing_MediaWiki`, Stand vom 10.02.2008.

[30] MediaWiki: *Manual:Special pages.* `http://www.mediawiki.org/wiki/Manual:Special_pages`, Stand vom 10.02.2008.

[31] MediaWiki: *Manual:Tag extensions.* `http://www.mediawiki.org/wiki/Manual:Tag_extensions`, Stand vom 10.02.2008.

[32] MediaWiki: *MediaWiki.* `http://www.mediawiki.org`, Stand vom 10.02.2008.

[33] Microsoft Corporation: *IIS Internet Information Services*. `http://www.iis.net`, Stand vom 22.02.2008.

[34] MySQL AB: *MySQL*. `http://www.mysql.com`, Stand vom 22.02.2008.

[35] NIST (National Institute of Standards and Technology): *VRML Plugin and Browser Detector*. `http://cic.nist.gov/vrml/vbdetect.html`, Stand vom 10.02.2008.

[36] OGC Open Geospatial Consortium: *Geography Markup Language*. `http://www.opengeospatial.org/standards/gml`, Stand vom 10.02.2008.

[37] OIO - Orientation in Objects: *XSLT Prozessoren im Überblick*. `http://www.postgresql.org`, Stand vom 22.02.2008.

[38] O'Reilly, Tim: *What Is Web 2.0, Design Patterns and Business Models for the Next Generation of Software*. `http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html?page=1`, Stand vom 10.02.2008.

[39] PostgreSQL Global Development Group: *PostgreSQL*. `http://www.postgresql.org`, Stand vom 22.02.2008.

[40] Steinmann, Harald: *3D-Online-Visualisierungen von Geodaten für digitale Schulatlanten (mit Beispielanwendungen für die geplante interaktive Version des Schweizer Weltatlas)*. Diplomarbeit, Hochschule Karlsruhe - Technik und Wirtschaft Fakultät Geomatik - Studiengang Kartographie und Geomatik, 2006.

[41] The Apache Software Foundation: *Apache*. `http://httpd.apache.org`, Stand vom 22.02.2008.

[42] W3C: *Cascading Style Sheets*. `http://www.w3.org/Style/CSS/`, Stand vom 22.02.2008.

[43] W3C: *XSL Transformations (XSLT) Version 1.0 W3C Recommendation 16 November 1999*. `http://www.w3.org/TR/xslt`, Stand vom 22.02.2008.

[44] W3Schools: *XSL Language.* http://www.w3schools.com/xsl/xsl_languages.asp, Stand vom 10.02.2008.

[45] Web3D Consortium: *VRML97 Functional specification and VRML97 External Authoring Interface (EAI), ISO/IEC 14772-1:1997 and ISO/IEC 14772-2:2004 - Virtual Reality Modeling Language (VRML).* http://www.web3d.org/x3d/specifications/vrml/ISO-IEC-14772-VRML97, Stand vom 10.02.2008.

[46] Web3D Consortium: *Web3D Consortium Working Groups.* http://www.web3d.org/x3d/workgroups, Stand vom 10.02.2008.

[47] Web3D Consortium: *What is X3D?* http://www.web3d.org/about/overview, Stand vom 10.02.2008.

[48] Web3D Consortium: *X3D and Related Specifications.* http://www.web3d.org/x3d/specifications, Stand vom 10.02.2008.

[49] Web3D Consortium: *X3D-Edit for Extensible 3D (X3D) Graphics - README.* http://www.web3d.org/x3d/content/README.X3D-Edit.html, Stand vom 10.02.2008.

[50] Web3D Consortium: *X3D Specifications, Encodings and Language Bindings, ISO/IEC 19775:2004/FDAM Am1:2006 - X3D Architecture and base components with Amendment 1, Part 2.* http://www.web3d.org/x3d/specifications/ISO-IEC-19775-X3DAbstractSpecification_+_Amendment1_to_Part1, Stand vom 10.02.2008.

[51] Web3D Consortium: *X3D Specifications, Encodings and Language Bindings, ISO/IEC CD 19775-1r1:200x - X3D Architecture and base components Revision 1.* http://www.web3d.org/x3d/specifications/ISO-IEC-19775-X3DAbstractSpecification_Revision1_to_Part1, Stand vom 10.02.2008.

[52] Web3D Consortium: *X3D Tools and Applications.* http://www.web3d.org/tools, Stand vom 10.02.2008.

[53] Web3D Consortium: *Xj3D*. `http://www.xj3d.org`, Stand vom 10.02.2008.

[54] Wikimedia: *Help:MediaWiki architecture*. `http://meta.wikimedia.org/wiki/Help:MediaWiki_architecture`, Stand vom 10.02.2008.

[55] Wilk, Christian: *Überlagert - Mit KML Google Earth erweitern*. iX Magazin für profenssionelle Informationstechnik, (12):58–63, Dezember 2006.

[56] Wilk, Christian: *Komplexe Nische - GML: geografischer Markup-Standard*. iX Magazin für profenssionelle Informationstechnik, (12):66–68, Dezember 2006.

[57] Wilk, Christian: *Welt in Händen - Arbeiten mit Google Earth und World Wind*. iX SPECIAL, (1):69–76, 2007.

[58] Yahoo: *Yahoo! UI Library: Slider*. `http://developer.yahoo.com/yui/slider`, Stand vom 10.02.2008.