



MASTERARBEIT / MASTER'S THESIS

Titel der Masterarbeit / Title of the Master's Thesis

„Optimierungsvorschläge zur
Real Time-Visualisierung von Punktsignaturen gezeigt
am Beispiel von Luftfahrzeugdaten“

verfasst von / submitted by

Thomas Smutny BSc

angestrebter akademischer Grad / in partial fulfilment of the requirements for the degree of
Master of Science (MSc)

Wien, 2017 / Vienna, 2017

Studienkennzahl lt. Studienblatt /
degree programme code as it appears
on the student record sheet:

A 066 856

Studienrichtung lt. Studienblatt /
degree programme as it appears
on the student record sheet:

Masterstudium
Kartographie und Geoinformation,
UG2002

Betreut von / Supervisor:

Ass.- Prof. Mag. Dr. Andreas Riedl

Inhaltsverzeichnis

Eidesstattliche Erklärung	v
Abbildungsverzeichnis	vi
Tabellenverzeichnis	xi
Abkürzungsverzeichnis	xii
Kurzfassung	xiii
Abstract	xiv
Vorwort	xv
1 Einleitung	1
1.1 Problematik der Vektordatendarstellung am Hyperglobus (Stand der Forschung)	1
1.2 Wissenschaftliche Fragestellung	3
1.3 Motivation	3
1.4 Aufbau der Arbeit	4
1.5 Zielsetzung	5
1.6 Methoden	5
2 Globen	6
2.1 Analoge Globen	7
2.2 Digitale Globen	10
2.3 (Taktile) Hyperglobus	12
2.3.1 Projektionsverfahren beim taktilen Hyperglobus	18
2.4 Eigenschaften des taktilen Hyperglobus	22
2.5 Taktile Hyperglobus der Hyperglobe Research Group (HRG)	24

3 Real Time-Visualisierung	26
3.1 Begriffsabgrenzung Visualisierung	26
3.1.1 Visualisierung in der Wissenschaft	26
3.2 Allgemein Real Time (Echtzeit)	30
3.3 Arten von Echtzeit	32
3.3.1 Weiche Echtzeit (soft Real Time)	32
3.3.2 Harte Echtzeit (hard Real Time)	32
4 Vektor versus Raster	34
4.1 Vektordaten(-modell)	34
4.2 Rasterdaten(-modell)	37
4.2.1 Rasterdatenformate	37
4.3 Vergleich zwischen Vektor- und Rasterdaten	44
5 Mathematische Grundlagen am Globus	46
5.1 Koordinatensystem am Globus	46
5.2 Quadratische Plattkarte.....	48
5.3 Tissot´sche Indikatrix.....	49
5.4 Orthodrome und Loxodrome.....	53
6 Datengewinnung vom Flug Real Time-Daten	56
6.1 Flightradar24	56
6.1.1 Allgemein.....	56
6.1.2 Funktionsweise	56
6.1.3 Flightradar24 - Mitgliedschaft.....	60
7 Die herkömmliche Methode der Generierung von Vektordaten	64
7.1 OmniSuite Allgemein.....	64
7.2 Datenintegration über Rasterdaten.....	66
7.3 Datenintegration über Raster- und Vektordaten.....	67
7.3.1 Raster- und Vektordaten extern verschneiden	67
7.3.2 Raster- und Vektordaten intern verschneiden.....	78

8 Optimierungsvorschläge zur Generierung von Vektordaten	93
8.1 Interaktionsmöglichkeiten mit Punkt-Datensatz	93
8.1.1 Beispiel Städte-Quiz	94
8.2 Integration von Linien	98
8.3 Interner Aufruf von Daten	106
9 Zusammenfassung und Ausblick	109
10 Literatur	111
11 Anhang	115
11.1 server_new.py	115
11.2 schleife.py	118
11.3 flugsuche2.py	120
11.4 orthodrome.py	122

Eidesstattliche Erklärung

Hiermit versichere ich,

dass ich die vorliegende Masterarbeit selbstständig verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt und mich auch sonst keiner unerlaubter Hilfe bedient habe,

dass ich dieses Masterarbeitsthema bisher weder im In- noch im Ausland in irgendeiner Form als Prüfungsarbeit vorgelegt habe

und dass diese Arbeit mit der vom Begutachter beurteilten Arbeit vollständig übereinstimmt.

Wien, am

Unterschrift

Abbildungsverzeichnis

Abbildung 1 - Graphisches Globenprofil eines analogen Globus, Quelle: [RIE-00], S. 39	7
Abbildung 2 - Atlas Farnese, Quelle: [SVA-16]	7
Abbildung 3 - Behaims „Erdapfel“, Quelle: [SVA-16]	8
Abbildung 4 - Erd- und Himmelsglobus von Mercator, Quelle: https://www.meinbezirk.at/donaustadt/lokales/erd-und-himmelsglobus-von-gerard-mercator-einem-der-beruehmtesten-kartographen-und-globenbauer-im-16-jahrhundert-m5184899,714130.html , abgerufen am 14.08.2017	9
Abbildung 5 - Globen nach deren Beschaffenheit, Quelle: [RIE-10], S.9	11
Abbildung 6 - Digitaler Behaim Globus, Quelle: [RIE-11a], S. 2	13
Abbildung 7 - EarthBrowser, Quelle: http://www.earthbrowser.com/desktop , abgerufen am 18.09.2017	13
Abbildung 8 - GeoSphereGlobe, Quelle: [RIE-11a], S. 4.....	14
Abbildung 9 - Hyperglobus, Quelle: [RIE-11a], S. 3.....	14
Abbildung 10 - OmniGlobe, Quelle: [RIE-11a], S. 6	16
Abbildung 11 - Magic Planet, Quelle: [RIE-11a], S.6	16
Abbildung 12 - Science on a sphere, Quelle: [RIE-11a], S.7	17
Abbildung 13 - Außenprojektion, Quelle: [RIE-08], S. 5.....	18
Abbildung 14 - Fischaugenbasierte Innenprojektion, Quelle: [RIE-08], S. 6	19
Abbildung 15 - Spiegelbasierte Innenprojektion, Quelle: [RIE-08], S. 7	20
Abbildung 16 - Graphisches Globenprofil eines taktilen Hyperglobus im Vergleich zu einem analogen Globus, Quelle: [WIN-12], S. 9	22
Abbildung 17 - Omniglobe, Quelle: [KRI-12], S. 17	24
Abbildung 18 - Erdbebenbeispiel, Quelle: https://sos.noaa.gov/datasets/earthquakes-2001-2015/ , abgerufen am 18.09.17	25

Abbildung 19 - Visualisierungsmethoden, Quelle: https://www.e-education.psu.edu/geog486/book/export/html/1407 , abgerufen am 22.05.17	28
Abbildung 20 - Kartographischer Kommunikationsprozess, Quelle: [LAN-13], S. 277	29
Abbildung 21 - Visualisierungspipeline, Quelle: [HOF-14] und https://www.cg.tuwien.ac.at/studentwork/VisFoSe98/pet/ger/pipeline.htm , abgerufen am 22.05.17	29
Abbildung 22 - Das dehnbare Konstrukt der Echtzeit, Quelle: verändert nach [MAY-07], S. 31 – Quelle: Eigene Darstellung	31
Abbildung 23 - Weiche Echtzeit, Quelle: [BRA-o.J.], S. 4.....	32
Abbildung 24 - Harte Echtzeit, Quelle: [BRA-o.J.], S. 4.....	33
Abbildung 25 - Gegenüberstellung von hard Real Time versus soft Real Time systems, Quelle: [BRA-o.J.], S. 4.....	33
Abbildung 26 - Übersicht von Vektordaten, Quelle: http://www.gis-news.de/knowhow/gis-grundlagen/vektordatenmodell/ , abgerufen am 24.05.17.....	34
Abbildung 27 - Spaghetti-Datenmodell, Quelle: [HAK-02], S. 158	35
Abbildung 28 - Topologisches Datenmodell, Quelle: [HAK-02], S. 159	36
Abbildung 29 - Hierarchisches Vektor-Datenmodell, Quelle: [HAK-02], S. 15	36
Abbildung 30 - Raster-Format, Quelle: http://ivvgeo.uni-muenster.de/vorlesung/FE_Script/Bilder/abb3_1_1.gif , abgerufen am 10.05.17	37
Abbildung 31 - Bildbeispiel für Test 1.....	41
Abbildung 32 - Bildbeispiel für Test 2.....	42
Abbildung 33 - Bildbeispiel für Test 3.....	43
Abbildung 34 - Longitude und Latitude, Quelle: http://www.geographyalltheway.com , abgerufen am 06.12.2016	47
Abbildung 35 - Unterschied zwischen Positionen im geographischen Netz, Quelle: [IBM-o.J.].....	48
Abbildung 36 - Quadratische Plattkarte, Quelle: [KAI-12]	48
Abbildung 37 - Tissot´sche Indikatrix bei einer quadratischen Plattkarte, Quelle: [LIE-12], S. 2.....	50

Abbildung 38 - Punktsignaturen ohne Berücksichtigung von Verzerrungseigenschaften, Quelle: [LIE-12], S. 2.....	50
Abbildung 39 - Kontextmenü des Plug-In, Quelle: Eigene Darstellung.....	52
Abbildung 40 - QGIS-Plug In, Quelle: Eigene Darstellung.....	52
Abbildung 41 - Orthodrome – Flugstrecke Wien-Los Angeles, Quelle: https://www.luftlinie.org/Wien,AUT/Los-Angeles,CA,USA , abgerufen am 25.09.2017.....	53
Abbildung 42 - Abbildung der Orthodromen auf der Erdkugel, Quelle: http://kartenkunde- leichtgemacht.de/handbuch.php?page=Koordinatensysteme , abgerufen am 25.09.2017.....	54
Abbildung 43 - Berechnung Orthodrome, Quelle: [KAI-12].....	54
Abbildung 44 - Loxodrome-Skizze, Quelle: https://upload.wikimedia.org//wikipedia/commons/d/d6/ Loxodrome.png , abgerufen am 19.04.2017.....	55
Abbildung 45 - Loxodrome-Berechnung, Quelle: [KAI-12].....	55
Abbildung 46 - Übersicht der Funktion von ADS-B Einrichtungen, Quelle: [RIC-10].....	57
Abbildung 47 - Übersichtskarte der Abdeckung von Flightradar24, Quelle: [FLI-16b].....	58
Abbildung 48 - Funktion von MLAT, Quelle: [AUS-16].....	59
Abbildung 49 - Übersicht der Funktion von Sekundärradar, Quelle: [AUS-16].....	60
Abbildung 50 - Übersicht der Mitgliedschaftsformen, Quelle: [FLI-16c].....	60
Abbildung 51 - Übersicht der Funktionen I, Quelle: [FLI-16c].....	61
Abbildung 52 - Übersicht der Funktionen II, Quelle: [FLI-16c].....	62
Abbildung 53 - Übersicht der Flüge und Upload-Funktion, Quelle: [FLI-16d].....	63
Abbildung 54 - Plattform OmniSuite, Quelle: [RIE-08], S. 14.....	65
Abbildung 55 - Funktionsweise taktiler Hyperglobus, Quelle: [LIE-12], S. 9.....	66
Abbildung 56 - Schema der externen Vektordatenintegration, Quelle: [LIE-12], S. 47.....	68
Abbildung 57 - Optimierung der Vektorenintegration mittels Python-Skripts, Quelle: Eigene Darstellung.....	69
Abbildung 58 - JSON-Ausgabe, Quelle: Eigene Darstellung.....	70
Abbildung 59 - lokale Webseite, Quelle: Eigene Darstellung.....	71

Abbildung 60 - Funktion Abfrage, Quelle: Eigene Darstellung	72
Abbildung 61 - Funktion Schreibe, Quelle: Eigene Darstellung	73
Abbildung 62 - Bibliotheken und Definition des Ports des HTTP-Servers, Quelle: Eigene Darstellung.....	73
Abbildung 63 - GET-Request, Quelle: Eigene Darstellung	74
Abbildung 64 - POST-Request, Quelle: Eigene Darstellung	74
Abbildung 65 - WebServer definieren und Eingabe verarbeiten, Quelle: Eigene Darstellung.....	75
Abbildung 66 - Inhalt des Textfiles nach Aufrufen des Webservers, Quelle: Eigene Darstellung.....	75
Abbildung 67 - Setzen von Modulen und Variablen, Quelle: Eigene Darstellung.....	76
Abbildung 68 - Schleife und Ausgabe als (.lyr)-File, Quelle: Eigene Darstellung.....	77
Abbildung 69 - Ausgabe als PNG-File, Quelle: Eigene Darstellung.....	78
Abbildung 70 - Schema der internen Vektordatenintegration, Quelle: [LIE-12], S. 46.....	79
Abbildung 71 - Erstellung Textfile, Quelle: [GLO-14], S. 91.....	80
Abbildung 72 - Variation von Punktgrößen, Quelle: Eigene Darstellung.....	80
Abbildung 73 - Darstellung der Punkte, Quelle: [GLO-14], S. 95.....	82
Abbildung 74 – Städtenamen für Beschriftung der Punkte, Quelle: [GLO-14], S. 96.....	82
Abbildung 75 - Darstellung der Punkte und Namen, Quelle: [GLO-14], S. 97	83
Abbildung 76 - Kategorisierung der Punkte, Quelle: [GLO-14], S. 98.....	84
Abbildung 77 - Kategorisierung mit Bevölkerungsdaten, Quelle: [GLO-14]	85
Abbildung 78 - Erstellung animierter Layer, Quelle: [GLO-14], S. 103.....	85
Abbildung 79 - Optimierung von Variante 2, Quelle: Eigene Darstellung	87
Abbildung 80 - Funktion "schreibe", Quelle: Eigene Darstellung.....	88
Abbildung 81 – Funktionen „abfrage“, Quelle: Eigene Darstellung.....	88
Abbildung 82 – Skript zur Anzeige der Flugdaten mit Karte, Quelle: Eigene Darstellung.....	89
Abbildung 83 – Webseite, Quelle: Eigene Darstellung.....	89
Abbildung 84 – JavaScript Funktion, Quelle: Eigene Darstellung	90

Abbildung 85 - Interface des Städte Quiz, Quelle: Eigene Darstellung	95
Abbildung 86 - HTML Startseite Städte Quiz, Quelle: Eigene Darstellung	95
Abbildung 87 - Interface der Webseite Städte-Quiz, Quelle: Eigene Darstellung.....	98
Abbildung 88 – Orthodromendarstellung als PNG mittels Python-Skript, Quelle: Eigene Darstellung.....	99
Abbildung 89 – Daten der Flughäfen der Welt, links: ohne Transponieren, rechts: mit Transponierung, Quelle: Eigene Darstellung	100
Abbildung 90 - Allgemeine Einstellungen, Quelle: Eigene Darstellung	101
Abbildung 91 - Koordinaten auslesen, Quelle: Eigene Darstellung	101
Abbildung 92 - Koordinaten in Tabelle schreiben und Orthodrome erstellen, Quelle: Eigene Darstellung	
Abbildung 93 - Speichern als PNG, Quelle: Eigene Darstellung	102
Abbildung 94 - Ergebnis Orthodrome, Quelle: Eigene Darstellung	103
Abbildung 95 - Orthodrome nach Ablauf des Werkzeugs, Quelle: Eigene Darstellung	105
Abbildung 96 - Button zur Aktualisierung im OmniSuite Controller, Quelle: Eigene Darstellung.....	106
Abbildung 97 - Timer zur Aktualisierung in OmniSuite Controller, Quelle: Eigene Darstellung.....	106
Abbildung 98 – Zoom-Toolbar, Quelle: Eigene Darstellung	107
Abbildung 99 - Darstellung der fliegenden Langstreckenflotte von Austrian Airlines, Quelle: Eigene Darstellung.....	108

Tabellenverzeichnis

Tabelle 1 - Unterscheidungsmerkmale von Globen, Quelle: [RIE-10], S. 1	10
Tabelle 2 - Übersicht der Anbieter taktiler Hypergloben, Quelle: [RIE-08], S. 8.....	17
Tabelle 3 - Vor- und Nachteile der Projektionsverfahren bei den digitalen Globen, Quelle: [RIE-10], S. 3 (+ positiv, - negativ).....	21
Tabelle 4 - Übersicht der Farbtiefe bei Rasterbildern.....	38
Tabelle 5 - Übersicht der in OmniSuite verwendeten Bildformate	39
Tabelle 6 - Vergleich der Ladezeit von Test 1	41
Tabelle 7 - Vergleich der Ladezeit von Test 2	42
Tabelle 8 - Vergleich der Ladezeit von Test 3.....	43
Tabelle 9 - Vor- und Nachteile von Vektordaten, Quelle: [RIE-12], [HAR-09] und [LAN-13].....	44
Tabelle 10 - Vor- und Nachteile von Rasterdaten, Quelle: [RIE-12], [HAR-09] und [LAN-13].....	44
Tabelle 11 – Langstreckenverbindungen von Austrian Airlines, Quelle: Austrian Airlines Timetable (+1 = Landung am nächsten Kalendertag).....	91

Abkürzungsverzeichnis

ADS-B - Automatic Dependent Surveillance – Broadcast

API - Application programming interface

CSS - Cascading Style Sheets

ESRI - Environmental Systems Research Institute

HRG - Hyperglobe Research Group

JSON - JavaScript Object Notation

KML - Keyhole Markup Language

MHz - Megahertz

MLAT - Multilateration

NOAA - National Oceanic and Atmospheric Administration

PHP - Hypertext Preprocessor

PNG - Portable Network Graphics

SOS - Science on a Sphere

URL - Uniform Resource Locator

UTC - Coordinated Universal Time

Kurzfassung

Ein sphärisches Display ist eine neue Form der Präsentationstechnologie, wobei das Display als Eingabe- und Ausgabegerät dient, das heißt, dass die Daten über einen PC mittels einer speziellen Globensoftware geladen und als bildhafte Inhalte wiedergegeben werden. Die Inhalte können Raster- und/oder Vektordaten sein, welche mittels diverser Projektionsverfahren auf die Innenseite der Globuskugel projiziert werden, sodass die verschiedenen Themen angezeigt werden.

In dieser Masterarbeit wird untersucht, wie die derzeit bestehende Vektordatenintegration optimiert beziehungsweise automatisiert werden kann. Ausgehend von vorhandenen Bearbeitungsabläufen, soll mittels Python-Skripte, Vektordaten geladen und diese direkt am sphärischen Display visualisiert werden. Die Umsetzung soll mit Flugdaten von FlightRadar24 durchgeführt werden. Es werden bewusst Real Time-Daten verwendet, um einen realen Eindruck für die/den AnwenderIn möglich zu machen. Außerdem wird das Problem der Verzerrungen ausgehend von der geographischen Breite erläutert und in die Optimierungs- und Aktualisierungsmethoden intern mit einbezogen, sodass dieses Problem bei einer direkten Integration von Vektordaten automatisch gelöst wird.

Im praktischen Teil der Arbeit werden die optimierten Ablaufschemata und Python-Skripte näher erläutert und Neuerungen hinsichtlich der möglichen Optimierung mittels der Flugdaten umgesetzt und vorgestellt.

Abstract

A spherical display is a new form of presentation technology with the display serving as an input and output device, meaning that the data are loaded on a computer using special globe software and rendered as pictorial contents. These may be raster and / or vector data that are projected onto the inside of the globe by different projection methods so that the various topics are displayed.

This master thesis investigates how the existing vector data integration can be optimized or automated. Based on existing processes, vector data are loaded by Python scripts and visualized directly on the spherical display. The implementation is to be carried out from FlightRadar24 with real time flight data, so that the user can win a real impression. Furthermore the distortions problem is explained and the calculation is integrated in the optimization methods so that this problem is solved automatically by directly integrating vector data.

In the practical part of the thesis the optimized flowcharts and Python scripts are explained in greater detail and innovations regarding the flight data are implemented and presented.

Vorwort

Ein ganz besonderer Dank gilt natürlich meiner Familie, die es mir ermöglicht hat, diese Ausbildung zu genießen und mich während meines gesamten Studiums immer unterstützt hat. Vielen Dank.

Ein besonders herzlicher Dank richtet sich an meinen Betreuer Herrn Mag. Dr. Andreas Riedl, Assistenzprofessor an der Universität Wien, für die Möglichkeit, eine Masterarbeit über ein Thema zu schreiben, das mein Hobby als auch mein Interesse in der Luftfahrt, beinhaltet. Außerdem möchte ich mich für die Betreuung meiner Anliegen bedanken. Danke Andreas.

Weiters danke ich Herrn Mag. Jürgen Kristen für die Hilfe bei der Umsetzung von Neuerungen und für die Betreuung meiner Anliegen. Danke Jürgen.

Des Weiteren danke ich Frau Mag. Dr. Doris Riedl für die konstruktive Kritik und für das Korrekturlesen der Masterarbeit. Danke Doris.

Außerdem möchte ich mich bei meiner Freundin Ines sowie bei Freunden und StudienkollegInnen, vor allem bei Herrn Thomas Binder BA, Herrn Rudolf Churanek MSc, Herrn Mag. Lukas Nebel und Herrn Mag. Karl Trummer bedanken, die mich mit konstruktiver Kritik sowie mit Gesprächen über dieses Thema bereichert, beziehungsweise mich während des Studiums begleitet haben.

Vielen Dank euch allen.

1 Einleitung

Ein sphärisches Display ist eine Kugeloberfläche, welche als Eingabe- und Ausgabegerät dient, das heißt, dass die Daten über eine Software eingespielt werden und die Kugel selber die Präsentationsfläche ist, auf der die NutzerInnen die Inhalte sehen können. An der Universität Wien wird im Rahmen der Hyperglobe Research Group seit dem Jahr 2005 aktiv an dieser Thematik geforscht und es wurde eine plattformunabhängige Abspielplattform entwickelt, um globale Thematiken abspielen zu können.

1.1 Problematik der Vektordatendarstellung am Hyperglobus (Stand der Forschung)

Als moderne Art der Wissensvermittlung im Bereich des sphärischen Displays zeigt der derzeitige Stand der Forschung eine sehr starke Fokussierung auf die Visualisierung von globalen Phänomenen. Um Global Stories für das sphärische Display aufzubereiten werden Rasterdaten benötigt, in diesem Fall wird sehr oft auf die quadratische Plattkarte (rasterbasiert) als Kartengrundlage zurückgegriffen (Kapitel 5.2.). Die beiden großen Vorreiter, welche Thematiken für sphärische Displays aufbereiten und erstellen, sind unter anderem die amerikanische Wetterbehörde National Oceanic and Atmospheric Administration (NOAA) mit der Implementierung der „Science On a Sphere“ (SOS) und die Firma Globocces. Im Bereich der Real Time-Darstellungen gibt es bei SOS derzeit über 50 Anwendungen (vgl. [NOA-17]). Ein Beispiel stellt die Wolkenbedeckung dar, bei der jede Stunde das aktuelle Satellitenbild der Erde heruntergeladen und neu eingebunden werden kann, um die Weiterbewegung der Wolken zu sehen. Im Vergleich dazu gibt es bei Globocces derzeit drei Real Time-Anwendungen (vgl. [GLO-17a]).

Nur mit Rasterdaten alleine wird es nicht möglich sein, eine globale Thematik gut zu erklären, darum werden oft auch Vektordaten für die Visualisierung am Hyperglobus verwendet. Diese Vektordatensätze können punkthafte (Koordinaten von Luftfahrzeugen), linienhafte (Gradnetz) und flächenhafte Signaturen (Staatsflächen) sein und diese werden in einem extra Schritt, zum Beispiel mittels eines Geoinformationssystems (GIS), mit den Rasterdaten überlagert. Die Aufbereitung von Vektordaten und das Zusammenfügen mit den Rasterdaten können in einem GIS sehr rasch erledigt werden. Das Problem welches hier aber entsteht ist, dass quadratische Plattkarten nach dem Zusammenfügen nur für eine zweidimensionale Verwendung geeignet sind, jedoch nicht für den Hyperglobus. Eine

quadratische Platkarte weist mit zunehmender geographischen Breite eine immer stärkere Verzerrung auf. Dieses Problem kann mit einem Geoinformationssystem auch nicht automatisch gelöst werden (Kapitel 5.3).

Johannes LIEM beschäftigte sich im Jahr 2012 im Rahmen seiner Diplomarbeit, mit dem Titel „Vektordaten am taktilen Hyperglobus – über Verfahren zur Aufbereitung und Visualisierung geographischer Vektordatensätze am sphärischen Display“, mit Methoden und Lösungsansätzen, wie Vektordaten am Hyperglobus verwendet werden können und welche Probleme berücksichtigt werden müssen. Ausgehend von dieser Diplomarbeit möchte ich in meiner Masterarbeit, mit dem Titel „Optimierungsvorschläge zur Real Time-Visualisierung von Punktsignaturen gezeigt am Beispiel von Luftfahrzeugdaten“, die von Johannes LIEM durchgeführten Methoden und Lösungen einer Prüfung unterziehen und diese Methoden erweitern und zum Teil automatisieren. In dieser Arbeit soll der Versuch unternommen werden, Real Time-Flugdaten auf einem Globus verzerrungsfrei abzubilden. Des Weiteren sollen Optimierungsvorschläge vorgestellt und der Ist- und zukünftige Stand im Bereich der Generierung von Vektordaten untersucht werden.

Im Bereich der Luftfahrt ist die seit dem Jahr 2008 bekannte World Air Traffic Animation im Umlauf, welche die Züricher Hochschule für Angewandte Wissenschaften in Zusammenarbeit mit dem Technorama Winterthur erstellte¹. Bei dieser Darstellung wird der gesamte weltweite Flugverkehr in einem Zeitraffer dargestellt, wobei die An- und Abflugzeiten aller Flughäfen und die Flugpunkte dazwischen interpoliert werden und somit eine bewegte Darstellung entsteht. Aufgrund des Zusammenfassens der Daten handelt es sich allerdings um keine Real Time-Anwendung, da die Daten nicht innerhalb einer Zeitspanne übertragen werden, sondern vorab bereits bearbeitet wurden. Real Time-Anwendungen für Flugbewegungen wurden bereits von FlightRadar24 sowie FlightAware und einigen anderen Firmen umgesetzt. Negativ zu bewerten ist jedoch, dass die derzeitige Wiedergabe der Flugbewegungen nur auf Flachbildschirmen von statten geht, da die Darstellung am Globus aufgrund der Verzerrungsproblematik bei Vektordaten einer Lösung harret. Auf dieses Problem soll es am Ende dieser Arbeit eine Antwort beziehungsweise einen Lösungsweg geben.

¹ <http://radar.zhaw.ch/worldwide.html>, abgerufen am 26.06.2017

1.2 Wissenschaftliche Fragestellung

Aus diesen Überlegungen ergibt sich die folgende Forschungsfrage, aus der heraus Arbeitsfragen abgeleitet werden, welche im Rahmen dieser Masterarbeit erörtert werden sollen:

Forschungsfrage

- Wie lässt sich die Vektordatenintegration am sphärischen Display optimieren bzw. automatisieren?

Arbeitsfragen

- Wie unterscheiden sich die Methoden der Datengewinnung sowie -verarbeitung hinsichtlich des Zieldatenformats (Vektor versus Raster)?
- Wie können Verzerrungen bei der Visualisierung (Punkte, Linien) vermieden werden?

Schwerpunkt dieser Arbeit ist die Untersuchung des Problems der direkten Verarbeitung von Vektordaten auf den Hyperglobus mit Real Time-Daten sowie das Aufzeigen neuer Methoden, um problemlos Vektordaten integrieren zu können und das Problem der Verzerrungen einer Lösung zuzuführen.

1.3 Motivation

Motivation für das Thema dieser Masterarbeit ist das große Interesse an der Thematik Hyperglobus und der Versuch, komplexe globale Inhalte leichter und verständlicher darstellen zu können. Ein weiterer Grund ist meine eigene Begeisterung an der zivilen Luftfahrt. Daher stellt diese Masterarbeit für mich eine Kombination von Studium, Arbeit und Hobby dar. Weiters besteht meinerseits seit der Lehrveranstaltung Multimediatechnologie und Geokommunikation großes Interesse an dieser Fragestellung, da im Rahmen dieses Kurses selbst eine Global Story zur Thematik „Jetstreams und deren Auswirkungen“ erstellt und somit ein Thema für eine Nutzergruppe leichter und verständlich präsentiert wurde.

1.4 Aufbau der Arbeit

Am Beginn wird eine Übersicht über diverse Arten von Globen (Kapitel 2) wiedergegeben, wobei zwischen analogen und digitalen Globen und weiters nach Riedl [RIE-11a] zwischen virtuellen und taktilen Hypergloben und Hologloben unterschieden wird. Es werden die verschiedenen Möglichkeiten hinsichtlich der Abbildungsvermittlung mittels diverser Projektionssysteme näher beschrieben. Nach Riedl [RIE-08] wird zwischen Außen-, Innen- und Direkter Projektion unterschieden. Darüber hinaus soll auf die Vor- und Nachteile beziehungsweise Probleme der einzelnen Projektionsverfahren eingegangen werden. In weiterer Folge soll ein kurzer Überblick über den Hyperglobus des Instituts für Geographie und Regionalforschung folgen. Anschließend wird der Institutsglobus in die zuvor erwähnte Übersicht der diversen Globen integriert, wobei der Begriff der Real Time-Visualisierung (Kapitel 3) allgemein abgegrenzt und in Bezug auf den Hyperglobus erklärt werden soll.

Im Anschluss wird allgemein auf das Vektor- beziehungsweise Rasterdatenformat (Kapitel 4) eingegangen, um die Probleme hinsichtlich der Integration von Vektor- und Rasterdaten am Hyperglobus aufzuzeigen. Weiters sollen die diversen Möglichkeiten der Kombination der Verschneidungen von Daten erläutert, sowie auf die Visualisierungsprobleme eingegangen werden. Derzeit werden Vektordaten (Punkte, Linien, Flächen) über einen zusätzlichen Schritt auf der Rasterkarte eingezeichnet und verortet. Das Ziel soll sein, Vektordaten ohne vorheriger händischer Bearbeitung, mittels Adobe Photoshop oder ESRI ArcGIS direkt am Globus zu visualisieren.

Bei der Visualisierung von Flugrouten auf einem Globus, welcher als einziges kartographisches Produkt Flächen-, Winkel- sowie Längentreue besitzt, müssen Abbildungsvorschriften abhängig von Verzerrung und Darstellung gewählt werden. Es wird auf geographische Koordinatensysteme (WGS84) und auf die quadratische Plattkarte eingegangen (Kapitel 5), da diese den Raumbezug der derzeitigen Kartengrundlage darstellt, welche nach Riedl [RIE-10] *„aufgrund der einfachsten Erstellung und allgemeinen Bekanntheit [...] für das Mapping von globalen Daten auf einem digitalen Globus empfohlen wird“*.

Hier sollen auch die Verzerrungsprobleme und Lösungswege über die Tissot'sche Indikatrix angesprochen werden (vgl. [LAM-11]). Flugzeuge fliegen im Optimalfall entlang eines Teilstückes des Großkreises, der so genannten Orthodrome, welche die kürzeste Verbindung zwischen zwei Punkten darstellt, um die wirtschaftlichen Kosten klein zu halten. Mit einer

optimalen Route kann zum Beispiel Treibstoff gespart oder die Flugzeiten reduziert werden (vgl. [KRE-70, S. 162]. An dieser Stelle soll neben der allgemeinen Begriffsdefinition auch auf die Berechnung der Orthodrome eingegangen sowie ein Vergleich zur Loxodrome gegeben und die daraus entstehenden Probleme, Vor- und Nachteile diskutiert werden.

Am Beginn des praktischen Teils der Arbeit wird neben einer kurzen Vorstellung der Firma Flightradar24 auch die Funktionsweise der Datenaufnahme über drei Aufnahmeverfahren (ADS-B, MLAT und Radar) näher ausgeführt (vgl.[FLI-16b]). Anschließend wird auf diverse Funktionen in den unterschiedlichen Mitgliedschaften eingegangen und der Datendownload vorgestellt (Kapitel 6).

Des Weiteren wird eine Übersicht über die derzeitigen Bearbeitungsmethoden (Kapitel 7) zur Implementierung von Vektordaten am Hyperglobus gegeben. Außerdem wird der Autor einen neuen Lösungsweg aufzeigen, um Real Time-Flugdaten über zwei verschiedene Programmiersprachen am Hyperglobus integrieren zu können. Die Globensoftware OmniSuite wurde im Rahmen einer Diplomarbeit im Jahr 2012 von Mag. Jürgen Kristen [KRI-12] entwickelt und wird laufend weiterentwickelt. Das letzte Kapitel beschäftigt sich mit neu umgesetzten Methoden und gibt einen Überblick über Neuerungen die von Seiten von OmniSuite nun möglich sind.

1.5 Zielsetzung

Das Ziel dieser Arbeit ist Optimierungsvorschläge zu den bestehenden Methoden der Vektordatenintegration auszuarbeiten und dies mit Real Time-Flugdaten zu testen und am sphärischen Display anzuzeigen.

1.6 Methoden

Dem ersten Teil der Arbeit, welcher sich mit den Themen Globen, Vektordaten, Rasterdaten, Real Time und mathematische Grundlagen beschäftigt, liegt eine Literaturrecherche zugrunde. Der zweite Teil beschäftigt sich am Anfang mit einer Vorstellung der derzeitigen Methoden, um Vektordaten am Hyperglobus zu integrieren. Anschließend werden neue Methoden mittels der Skriptsprachen Python und PHP umgesetzt, um Real Time-Flugdaten für die Implementierung am Hyperglobus anwenden zu können. Am Ende der Arbeit werden neue Ansätze vorgestellt, um Vektordaten am Globus automatisch zu integrieren.

2 Globen

In diesem Kapitel wird ein Überblick über die verschiedenen Arten von Globen gegeben. Am Anfang wird auf die geschichtliche Entwicklung von analogen Globen eingegangen. Anschließend erfolgt eine Einteilung der digitalen Globen nach unterschiedlichen Parametern, welche wiederum eine detailliertere Abgrenzung in Virtuellen-, Taktile- und Hologlobus, ermöglichen. Gegen Ende dieses Kapitels wird der Taktile Hyperglobus, das Medium für die spätere praktische Umsetzung, in dieser Arbeit näher vorgestellt und ein Überblick der unterschiedlichen Projektionsverfahren gegeben.

Nach Riedl [RIE-09], S. 177 ist ein Globus wie folgt definiert:

„Ein Globus präsentiert ein maßstabsgebundenes und strukturiertes Modell eines Himmelskörpers (bzw. der scheinbaren Himmelskugel) in seiner unverzerrten dreidimensionalen Ganzheit.“

Ein weiterer Vorteil nach Kristen [KRI-12], S. 4 ist folgender:

„Darstellungen am Globus sind sowohl längen-, flächen- als auch winkeltreu, da die Kugeloberfläche der Erde nicht auf einen zweidimensionalen Zeichenträger verebnet und damit verzerrt werden muss.“

Abbildung 1 gibt eine graphische Übersicht über die Eigenschaften eines analogen Globus. Anhand dieses graphischen Globenprofils lässt sich gut erkennen, dass Repräsentativität, Treueigenschaften und die Didaktikeignung von analogen Globen sehr gut sind, beziehungsweise als Vorteile gesehen werden können. Nachteile hingegen sind die geringe Größe und die damit verbundene starke Generalisierung des Kartenbildes. Der Transport von Globen ist außerdem als eher kompliziert anzusehen (vgl. [MOK-05], S. 10). Im Kapitel 2.3.1. wird ein Vergleich zwischen Hypergloben und analogen Globen vorgenommen.

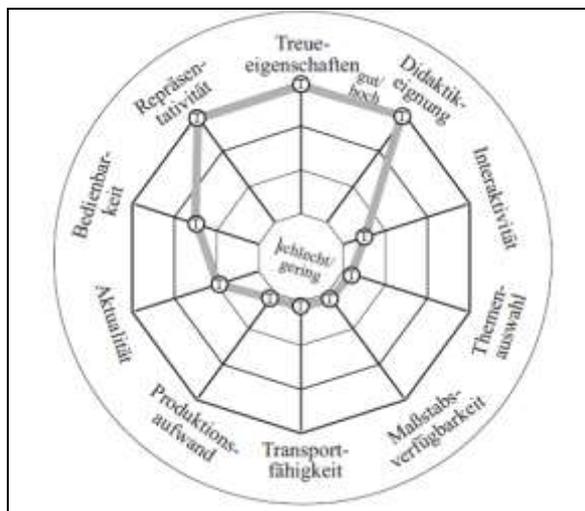


Abbildung 1 - Graphisches Globenprofil eines analogen Globus, Quelle: [RIE-00], S. 39

2.1 Analoge Globen

Globen haben eine sehr lange Tradition, da bereits in der griechischen Antike Himmelsgloben um 360 v. Chr. von Eudoxes von Knidos (408-355 v. Chr.) beschrieben wurden. Der „Atlas Farnese“ (siehe Abb. 2), eine zwei Meter hohe Marmorstatue, welche eine römische Kopie nach einem hellenistischen Original darstellt und im heutigen Archäologischen Nationalmuseum in Neapel aufbewahrt wird, visualisiert das Wissen der damaligen Zeit (vgl. [MOK-05], S. 15).



Abbildung 2 - Atlas Farnese, Quelle: [SVA-16]

Erstaunlich ist es, dass in dieser Zeit die Bedeutung der Kugelgestalt kaum wahrgenommen wurde, obwohl einige bekannte griechische Wissenschaftler, unter anderem Pythagoras (6. Jh. v. Chr.), Aristoteles (384-322 v. Chr.) und Eratosthenes (276-192 v. Chr.), der den Erdumfang genau berechnete, sich mit der Kugelgestalt der Erde auseinandersetzten (vgl. [MOK-05], S. 15).

Im späten Mittelalter erlangte die Kugelgestalt wieder mehr an Bedeutung und einige wenige Globen sind aus dieser Zeit erhalten geblieben. Der bekannteste und weltweit älteste Erdglobus (Ø 51 cm) ist der von Martin Behaim (1459-1507), welcher um 1492 einen Erdglobus für die Stadt Nürnberg anfertigte (siehe Abb. 3), welcher als ältester Globus und als Behaims „Erdapfel“ bekannt wurde. Das Weltbild, das auf diesem Globus dargestellt wird, wurde vor den Entdeckungsfahrten von Christoph Kolumbus und Amerigo Vespucci angefertigt (vgl. [MOK-05], S. 16, [RIE-11a] und [SVA-16]).



Abbildung 3 - Behaims „Erdapfel“, Quelle: [SVA-16]

Das 16. Jahrhundert war das Jahrhundert der spanischen und portugiesischen Entdeckungsfahrten. Ab dieser Zeit wurde die Kugelgestalt der Erde von der römisch-katholischen Kirche anerkannt, Amerika als neuer Kontinent entdeckt und Globen wurden als wissenschaftliche Instrumente und als Lehrmittel angewandt. Die Vervielfältigung von Globen wurde durch Holzschnitt- und Kupferstichverfahren immer weiter vorangetrieben. Als Vorreiter für die Globenherstellung zu dieser Zeit galt Johannes Schöner (1477-1547) aus Nürnberg. Das bekannteste Globenpaar wurde von Gerhard Mercator (1512-1594) angefertigt (Abb. 4) (vgl. [MOK-05], S. 17-18).



Abbildung 4 - Erd- und Himmelsglobus von Mercator, Quelle: <https://www.meinbezirk.at>

Ab dem 17. Jahrhundert wurde der Globus immer mehr zum Verlagsprodukt beziehungsweise zum Prunkobjekt und weist eine immer größere Realitätstreue als die Globen zuvor auf. Die Globenherstellung konzentrierte sich auf Europa, vor allem auf England, Italien und die Niederlande. Der bekannteste Globenbauer zu dieser Zeit war Vincenzo Coronelli (1650-1718), welcher Prunkgloben für den französischen König Ludwig XIV in verschiedenen Größen anfertigte (vgl. [MOK-05], S. 18-19).

Im 18. Jahrhundert ließ die Euphorie über Globen stark nach. Der Globus wurde zur Massenware und daher für jedermann erschwinglich, wobei die Qualität stark nachließ (vgl. [MOK-05], S. 19-20).

Ab dem Beginn des 19. Jahrhunderts veränderte sich das Kartenbild der Globen erneut. Durch anhaltende Entdeckungsreisen wurden weitere Teile der Welt entdeckt und kartographiert. Im gleichen Zeitraum veränderte sich die Herstellung: Weg vom teuren Kupferstich hin zur Lithografie (vgl. [MOK-05], S. 22).

Das 20. Jahrhundert ist durch die endgültige Massenproduktion von Globen mit verschiedenen Thematiken als Kartenbild gekennzeichnet. Beinahe jeder Haushalt hat einen Globus, der jedoch eher als Dekorationselement als zur Wissensvermittlung dient. Gegen Ende des 20. Jahrhunderts beschäftigte man sich aufgrund der einsetzenden Digitalisierung mit einem neuen Globustyp, nämlich dem digitalen Globus (mehr dazu anschließend ab Kapitel 2.2.) (vgl. [MOK-05], S. 22-23).

2.2 Digitale Globen

Digitale Globen gibt es seit den 1990er Jahren und stellen das Erdbild als Ganzes dar. Im Laufe der Zeit wurden diese Globen immer beliebter. Es kam zum Streben nach mehr Interaktivität. Man wollte dem Betrachter beziehungsweise Anwender selbst Funktionen anwenden lassen - es kam es zur Geburtsstunde von *Google Earth*.

Google Earth wurde am Anfang als digitaler Globus bezeichnet, das ist jedoch nicht ganz richtig, da es sich eigentlich um einen Geo-Browser handelt und vom Anwender ein kleiner detaillierter Ausschnitt der Erde betrachtet werden kann. Heutzutage wird zum Beispiel *Google Earth* für die rasche Suche von Orten verwendet, um einen realen Eindruck eines ausgesuchten Ortes zu bekommen (vgl. [RIE-10], S. 1 ff).

Eine Möglichkeit Globen zu gliedern, ist die Abgrenzung nach Riedl [RIE-11a], S. 1 (vgl. Tab. 1/Abb. 5), in dem der Autor die Globen nach drei Parametern unterteilt:

1. Globenkörper (materiell vs. virtuell)
2. Raum (real vs. virtuell)
3. Abbild (analog vs. digital)

Daraus lässt sich folgende Einteilung ableiten:

	Abbild	Raum	Globenkörper
Analoger Globus	analog	real	materiell
Virtueller Hyperglobus	digital	virtuell	virtuell
Taktile Hyperglobus	digital	real	materiell
Hologlobus	digital	real	virtuell

Tabelle 1 - Unterscheidungsmerkmale von Globen, Quelle: [RIE-10], S. 1

Aus der Ableitung der drei oben genannten Parameter ergeben sich nach Riedl [RIE-09] eine Unterteilung der Globen hinsichtlich des Namens und Abbilds:

- **Analoger Globus:** „Darstellung des analogen Abbildes auf einem materiellen Globenkörper im realen Raum.“
- **Virtuelle Hypergloben (VH):** „Darstellung des digitalen Abbildes auf dem virtuellen Globenkörper im virtuellen Raum.“
- **Taktile Hypergloben (THG):** „Darstellung des digitalen Abbildes auf einem materiellen (berührungssensitiven) Globenkörper im realen Raum.“
- **Hologloben (HG):** „Darstellung des digitalen Abbildes auf einem virtuellen Kugelkörper im realen Raum.“

Die Abgrenzung nach dem Globenkörper zeigt, dass bei analogen Globen und bei taktilen Hypergloben die Kugel, auf der die Inhalte präsentiert werden, materiell ist, das heißt physisch im Raum vorhanden ist, hingegen wird beim Virtuellen Hyperglobus und Hologlobus der Globenkörper virtuell in den Raum projiziert. Der Parameter Abbild zeigt, dass ausschließlich beim analogen Globus die Fläche der Informationen analog vorhanden ist, bei allen anderen Globen ist dies digital. Bei der räumlichen Abgrenzung wird ausschließlich beim virtuellen Hyperglobus ein virtueller Raum verwendet, bei allen anderen Globenarten befinden sich die Globen im realen Raum.

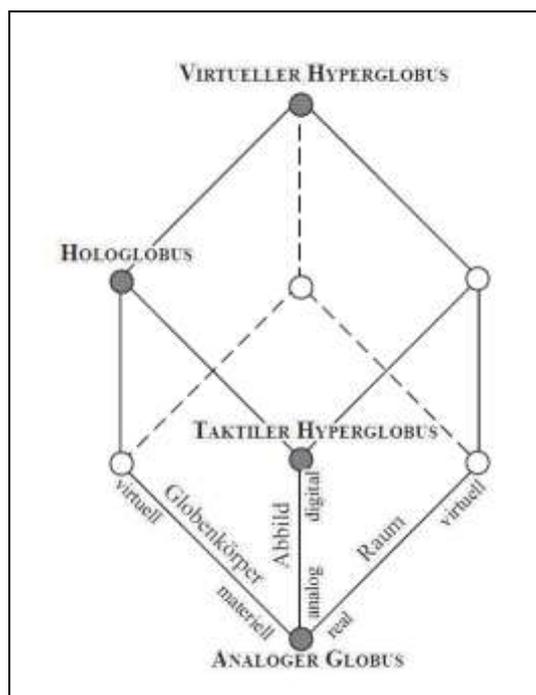


Abbildung 5 - Globen nach deren Beschaffenheit, Quelle: [RIE-10], S.9

Riedl [RIE-00] verweist auf zehn Eigenschaften nach denen Globenarten bewertet werden können:

- Treueeigenschaften
- Repräsentativität
- Didaktikeignung
- Aktualität
- Themenvielfalt
- Interaktivität
- Maßstabsverfügbarkeit
- Transportierbarkeit
- Bedienbarkeit
- Produktionsaufwand

Detaillierte Informationen dazu erhalten sie ab Kapitel 2.4.1.

2.3 (Taktiler) Hyperglobus

Taktile Hypergloben können als Nachfolger analoger Globen bezeichnet werden, wobei diese sich dadurch auszeichnen, dass der Globuskörper selbst das Visualisierungsgerät (sphärisches Display) ist, auf dem der thematische Inhalt für den Anwender und Betrachter sichtbar wird (vgl. [RIE-11b], S. 2). Der Begriff „Hyperglobus“ leitet sich vom Wort „Hyper“ ab, welcher wiederum vom Begriff Hypermedia abstammt und so viel wie Interaktivität bedeutet. Des Weiteren heißt „taktil“ nicht, dass der Hyperglobus haptisch und somit Inhalte abgetastet werden können, sondern es besteht die Möglichkeit die Erdkugel real zu berühren (vgl. [RIE-10], S. 2).

Der erste digitale Globus ist der digitale Behaim Globus (siehe Abb. 6), welcher im Rahmen einer Diplomarbeit aus dem Jahr 1996 an der TU Wien entstanden ist. Dieser wurde anlässlich des 500. Jahrestages von Behaim erstellt (vgl. [RIE-11a], S. 2).

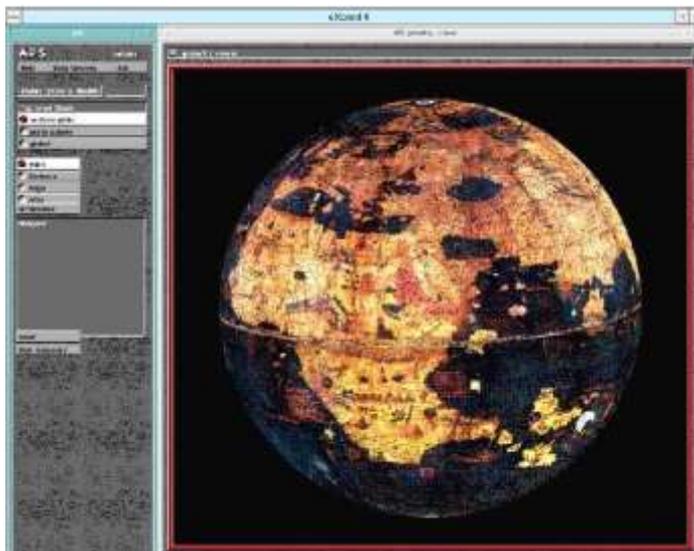


Abbildung 6 - Digitaler Behaim Globus, Quelle: [RIE-11a], S. 2

Ebenfalls im Jahr 1996 wurde der virtuelle Hyperglobus „Planet Earth“ bekannt. Dieser Globus unterschied sich vom Behaim Globus, dahingehend, dass dieser der erste Hyperglobus war, welcher mit einer Echtzeit-Visualisierung ausgestattet war. Am Anfang beschränkten sich die Thematiken auf die aktuelle Wettersituation und die Bewölkung. Nach der Umbenennung auf „EarthBrowser“ wurden weitere Funktionen hinzugefügt und können noch heute unter der Webseite www.earthbrowser.com aufgerufen und installiert werden, da es sich um eine Anwendung für Handy und andere mobile Geräte handelt. Abbildung 7 zeigt das heutige Interfacedesign dieses virtuellen Hyperglobus. Die Thematiken sind seit Beginn von nur Wetter auf die Anzeige von Naturkatastrophen (Vulkane, Feuer, Stürme, etc.), Straßennetzwerke (OpenStreetMap), Webcams der Erde, Zeitzonen und vieles mehr ausgeweitet worden (vgl. [RIE-11a], S. 2).



Abbildung 7 - EarthBrowser, Quelle: <http://www.earthbrowser.com>

Der erste taktile Hyperglobus wurde im Jahr 1992 in Brasilien am National Institute for Space Research aufgebaut und über eine halbtransparente Acrylgaskugel (\varnothing 2 m) war es möglich, globale Themen zu projizieren. Die Technik von GeoSphereGlobe war in dieser Zeit noch nicht so weit wie bei heutigen taktilen Hypergloben. Aus diesem Grund war ein Dauerbetrieb der Projektoren nicht möglich. Deshalb wurde ein halbtransparentes Satellitenbild angebracht, sodass auch bei „Offline-Betrieb“ die Erdkugel zu sehen war. Es handelte sich somit um eine Mischung zwischen analogem und digitalem Globus (vgl. [RIE-09], S. 177 und [RIE-11a], S. 4).



Abbildung 8 - GeoSphereGlobe, Quelle: [RIE-11a], S. 4

Als erster wissenschaftlicher Autor beschäftigte sich Andreas Riedl im Rahmen seiner Dissertation aus dem Jahre 2000 mit virtuellen Globen in der Geovisualisierung und dabei untersuchte er den Einsatz von Multimediatechniken in der Geopräsentation. Es wurde ein Hyperglobus (siehe Abb. 8) bei dieser Arbeit angefertigt, wo diverse Thematiken (Illustration der Erdgestalt, Auswirkungen auf die Erdachsenneigung, Echtzeit-Darstellungen, etc.) dargestellt wurden und somit diente dieser Hyperglobus als Instrument zur Wissensvermittlung (vgl. [RIE-11a], S. 2).



Abbildung 9 - Hyperglobus, Quelle: [RIE-11a], S. 3

Ab dem Jahr 2002 (Pionierphase) wurden diverse taktile Hypergloben serienmäßig, mit unterschiedlichen Projektionsverfahren gefertigt, weltweit verkauft und weiterentwickelt (vgl. [RIE-11a], S. 5).

Durch das große Interesse an 3-dimensionalen Darstellungen der Erdkugel gab es bereits zwei Jahre (Übergangsphase) später in jedem elektronischen Weltatlas einen virtuellen Hyperglobus als Element. In der Reifephase wurde versucht, virtuelle Hypergloben mit anderen Softwareprodukten zu verbinden. Im Jahr 2004 wurde von der Firma ESRI (Environmental Systems Research Institute) im Rahmen der ArcGIS Version 9 das Zusatzprodukt ArcGlobe implementiert, um Geodaten auf der Erdkugel zu visualisieren. In den derzeit laufenden ArcGIS Versionen steht ArcGlobe weiterhin zur Verfügung. ESRI hat bereits eine neue und zwar Gratis-Software für alle Nutzer, nämlich ArcGIS Earth entwickelt, um ein Pendantprodukt zu Google Earth zu haben, bei dem aber mehr Funktionalitäten im Geoinformationsbereich (Laden von Shapefiles, etc.) möglich sind (vgl. [RIE-11a], S. 3).

Nach Riedl [RIE-o8], S. 4 besteht noch keine direkte Konkurrenz zu den analogen Globen, da der Kostenfaktor zum Ankauf eines taktilen Hyperglobus eine Hemmschwelle darstellt. Werden diese jedoch leistbarer, könnten sie, aufgrund der Informationsvermittlung, den Markt erobern und die starren analogen Globen verdrängen. Aktuell kann gesagt werden, dass der weltweite Verkauf von taktilen Hypergloben seit den letzten acht Jahren zugenommen hat, aber noch immer nicht an den analogen Globus herankommt, da die Kosten für sphärische Displays noch immer sehr hoch sind und sie deswegen ausschließlich von Firmen, Museen oder dergleichen als Informationsvermittlungsgeräte angekauft werden.

Die bekanntesten Produzenten von taktilen Hypergloben sind unter anderem die Firma ARC Science Simulation (<http://arcscience.com/>) aus Colorado. Von ARC stammt der bekannte OmniGlobe (siehe Abb. 10), welcher das anzuzeigende Bild mittels spiegelbasierter Innenprojektion auf die Außenseite der Acrylglaskugel projiziert, sodass für den Nutzer der wiedergegebene Inhalt sichtbar wird. OmniGloben wurden ab dem Jahr 2002 in den Durchmessern 80 cm, 1,5 m und 2 m produziert. Von Europa ostwärts werden diese Globen von der deutschen Firma Globocess vertrieben (vgl. [RIE-11a], S. 5).

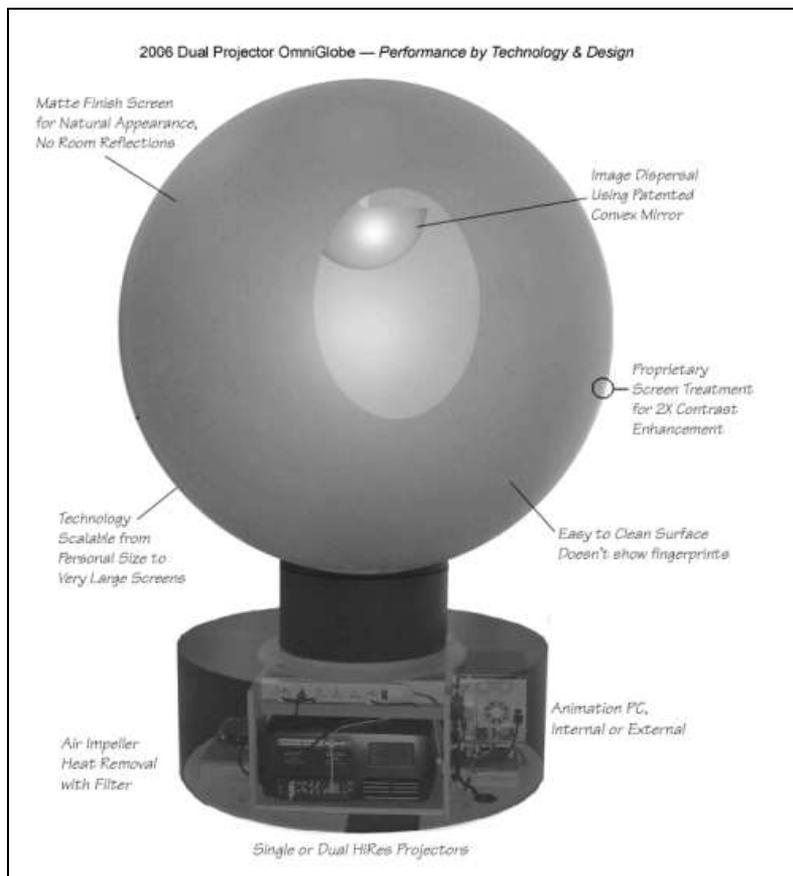


Abbildung 10 - OmniGlobe, Quelle: [RIE-11a], S. 6

Im selben Jahr wurde der „Magic Planet“ (siehe Abb. 11) von der Firma Global Imagination (<http://globalimagination.com/>) aus Kalifornien auf dem Markt gebracht. Bei dieser Art von Globen handelt sich um ein Innenprojektionssystem. Magic Planet wurde in drei verschiedenen Größenvarianten angeboten, nämlich in 41-46 cm, 61-91 cm und 1,2-1,8 m) (vgl. [RIE-11a], S. 5).

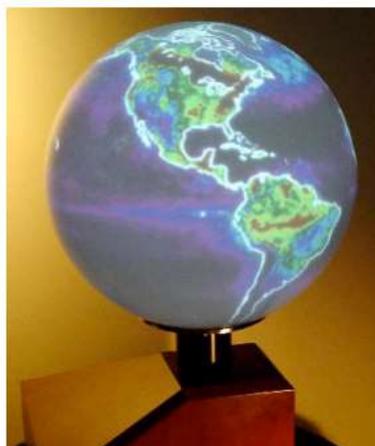


Abbildung 11 - Magic Planet, Quelle: [RIE-11a], S.6

Der dritte und letzte Vertreter ist die Nationale Wetterbehörde der USA (NOAA) aus Boulder in Colorado, welche mit der Science on a Sphere (SOS) https://sos.noaa.gov/What_is_SOS/ ein Außenprojektionssystem entwickelte. Die Kugeldurchmesser sind größer als bei den anderen beiden Unternehmen und zwar beginnen die Globen bei 1,5 bis 2 m, können aber auch noch größer sein (vgl. [RIE-11a], S. 5).



Abbildung 12 - Science on a sphere, Quelle: [RIE-11a], S.7

Tabelle 2 gibt nochmals einen Gesamtüberblick über die verschiedenen Hersteller von taktilen Hypergloben von 2008 bis heute. Anzumerken ist, dass die Firmen ag4 und Pufferfish nicht mehr am Markt vorhanden sind und somit keine Globen mehr herstellen.

Anbieter	Web	Modellvielfalt	System	inflatable/solid
ag4	www.ag4.de	+	DP	s
ARC Science Simulation	www.arcscience.com	+++	IPS	s
Global Imagination	www.globalimagination.com	+++	IPF	s
Globe4D	www.globe4d.com	+	DP	s
Globoccess	www.globoccess.com	+++	IPS	s
MESO	www.meso.net	+	AP	i
Pufferfish	www.pufferfishdisplays.co.uk	++	AP	i
QuasarsVision	www.quasarsvision.com	++	IP	i
NOAA - SOS	sos.noaa.gov	+	AP	s

AP – Außenprojektion; IPF – Innenprojektion Fischauge; IPS – Innenprojektion Spiegel; DP – Direkte Projektion

Tabelle 2 - Übersicht der Anbieter taktiler Hypergloben, Quelle: [RIE-08], S. 8

Zu den angesprochenen Projektionsverfahren von taktilen Hypergloben erfahren Sie in den anschließenden Kapiteln mehr.

2.3.1 Projektionsverfahren beim taktilen Hyperglobus

Wie schon zuvor erwähnt, wurden ab dem Jahr 2002 unterschiedliche Projektionsverfahren bei taktilen Hypergloben (weiter)entwickelt und verkauft. Diese Verfahren sollen im Folgenden näher beschrieben werden:

2.3.1.1 Außenprojektion

Bei der Außenprojektion wird zur Abbildung des Bildes eine allgemeine perspektivische Azimutalabbildung verwendet, bei der mindestens vier Projektoren (Bereich Äquator) ein Bild auf die Außenseite des Globus projizieren (vgl. Abb. 13). Aufgrund der vielen Projektoren, die in Verwendung sind, entstehen an den Übergangsbereichen, zwischen den einzelnen projizierten Bildern, Verzerrungen, sodass ein „Edge-blending“ Verfahren angewendet werden muss, um zwischen zwei benachbarten Bildern die Verzerrung zu mindern und ein verwertbares Bild zu bekommen. Außerdem muss bei diesem Verfahren beachtet werden, dass sich während der Bestrahlung des Globenkörpers kein Objekt zwischen dem Beamer und der Globuskugel befindet, da sonst der abgedeckte Bereich des Bildes nicht dargestellt werden kann. Diese Art von Projektion eignet sich am besten für Globen ab einem Durchmesser von 1,5 m, um die Fläche zwischen Anwender und Globuskörper zu vergrößern, sodass ein Bild überhaupt sichtbar wird. Zusammenfassend lässt sich sagen, dass die Außenprojektion die besten Werte bei der Auflösung erzielt und somit der gesamte Globus mit Inhalten abgedeckt werden kann (vgl. [RIE-09], S. 177-178 und [RIE-08], S. 5-6).

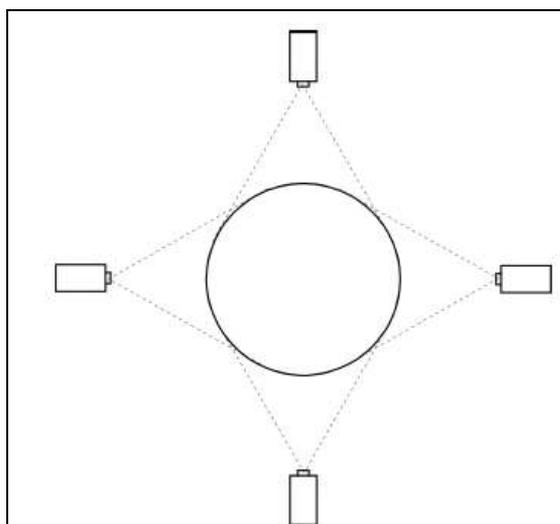


Abbildung 13 - Außenprojektion, Quelle: [RIE-08], S. 5

2.3.1.2 Innenprojektionen (fischaugen- und spiegelbasiert)

Bei der Innenprojektion wird ein Bild angezeigt wenn es sich um eine vermittelnde Azimutalabbildung handelt, bei der der Projektionsstrahl von unten in das Kugellinnere projiziert wird und entweder durch Spiegel (spiegelbasiertes System) oder mittels speziellen Objektiven (fischaugenbasiertes System) auf den Globenkörper verteilt und nach außen hin für den Anwender sichtbar wird (vgl. [RIE-09], S. 178).

Beim Fischaugenprinzip befindet sich der Beamer, wie in Abb. 14 zu sehen ist, im Sockel, auf dem sich der Globenkörper befindet. Über ein Loch in der Mitte dieses Körpers wird der Strahl des Beamers in das Innere der Kugel geleitet und über ein spezielles Weitwinkelobjektiv werden die Strahlen auf die gesamte Innenseite des Globenkörpers abgelenkt. So wird der projizierte Inhalt für den Anwender sichtbar. Ein Nachteil dieses Systems ist, dass aufgrund des Loches in der Mitte des Globenkörpers an dieser Stelle kein Bild dargestellt werden kann und deshalb ein „blind spot“ entsteht. Des Weiteren kann dieses System nur mittels eines Beamers betrieben werden und ist deshalb nur für kleinmaßstäbige Globen empfehlenswert (vgl. [RIE-08], S. 6).

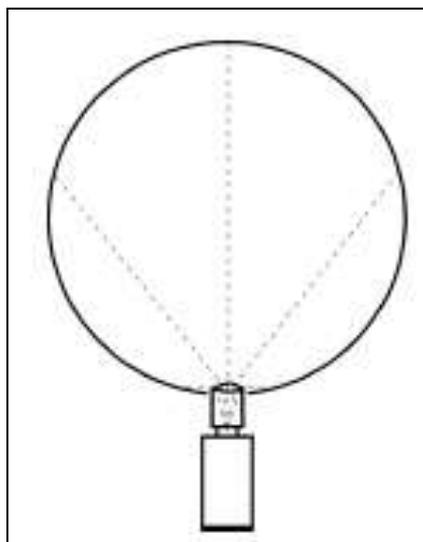


Abbildung 14 - Fischaugenbasierte Innenprojektion, Quelle: [RIE-08], S. 6

Die spiegelbasierte Version der Innenprojektion funktioniert ähnlich wie die fischaugenbasierte Projektion. Der Strahl des Beamers wird von unten aus der Kugelmittle in das Innere des Globus projiziert. In der Mitte des Globuskörpers befindet sich ein konvexer Spiegel, mit dem die Projektionsstrahlen auf die Innenseiten abgelenkt werden, und dadurch für den Anwender nach außen hin sichtbar werden (vgl. Abb. 15). Der Nachteil ist hier natürlich größer, da nicht nur im unteren, sondern auch im oberen Bereich des Globuskörpers ein „blind spot“ entsteht und an diesen Stellen kein Bild projiziert werden kann. Der Vorteil ist, dass hier ein zweiter Beamer anstatt des Spiegels montiert werden kann. Wird ein zweiter Beamer montiert, spricht man von einem Dual-Beamer-System (vgl. [RIE-08], S. 7).

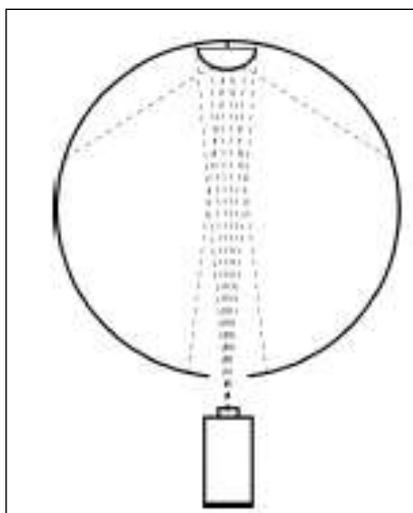


Abbildung 15 - Spiegelbasierte Innenprojektion, Quelle: [RIE-08], S. 7

2.3.1.3 Direkte Projektion

Hierbei sind Projektor und Projektionsfläche ein zusammengehörendes System. Vorteil bei der direkten Projektion ist, dass es zu keinem „blind spot“ durch Anbringung von Projektoren oder zu Bildstörungen durch Objekte kommen kann (vgl. [RIE-08], S. 8).

In Tabelle 3 nach Riedl [RIE-10], S. 3 werden weitere Vor- und Nachteile hinsichtlich der zuvor diskutierten Projektionsverfahren nochmals zusammengefasst:

	Außen- projektion AP	Innen- projektion - Fisheye IP _f	Innen- projektion - Mirror IP _m	Innen- projektion - Direkt IP _d	Direkte Projektion LEDs DP
Auflösung	+++	+	++	+++	-
Bildqualität	+++	+	++	+++	-
Installation/Deinstallation	-	++	+	-	++
Unabhängigkeit vom Umgebungslicht	++	+	+	++	+++
Raumbedarf	-	+++	+++	+++	-
Hardwarevoraussetzung	-	+	+	+	+
Blind Spot	+++	++	+	+++	+
Abschattung des Projektionsstrahls	-	+++	+++	+++	+++

Tabelle 3 - Vor- und Nachteile der Projektionsverfahren bei digitalen Globen,

Quelle: [RIE-10], S. 3 (+ positiv, - negativ)

Die Außenprojektion punktet mit der Auflösung, Bildqualität und dem Hintergrund. Es entstehen keine „blind spots“. Negativ anzumerken sind die hohen Anforderungen an Installation/Deinstallation, den Raumbedarf, der Abschattung des Projektionsstrahls und der Hardwarevoraussetzung (vgl. [WIN-12], S. 15).

Das Fischaugenprojektionsverfahren zeichnet sich dadurch aus, dass es kaum Platz benötigt, da der Beamerstrahl von unten in den Globenkörper strahlt, weshalb die Abschattung des Strahls zu vernachlässigen ist, da nur der Südpol als „blind spot“ vorhanden ist. Dieser „blind spot“ ist systembedingt nicht lösbar. Die weiteren Eigenschaften wie Auflösung, Bildqualität und Hardwarevoraussetzungen können ebenfalls als positiv angesehen werden. Außerdem stellt dieses System keine negativen Punkte dar und kann für kleinmaßstäbige Anwendungen gut verwendet werden (vgl. [WIN-12], S. 16).

Das spiegelbasierte Projektionsverfahren weist ähnliche Eigenschaften wie das fischaugenbasierte Projektionsverfahren auf. Durch die Einstellung des Spiegels mit dem Beamerstrahl entsteht jedoch mehr Aufwand. Außerdem entsteht dadurch ein zweiter „blind spot“, welcher vom Spiegel hervorgerufen wird, aber durch ein Dual-Beamer-System kompensiert werden kann (vgl. [WIN-12], S. 16).

2.4 Eigenschaften des taktilen Hyperglobus

Der taktile Hyperglobus kann auch nach denselben Eigenschaften wie in Kap. 2.2. graphisch dargestellt werden (siehe Abb. 16):

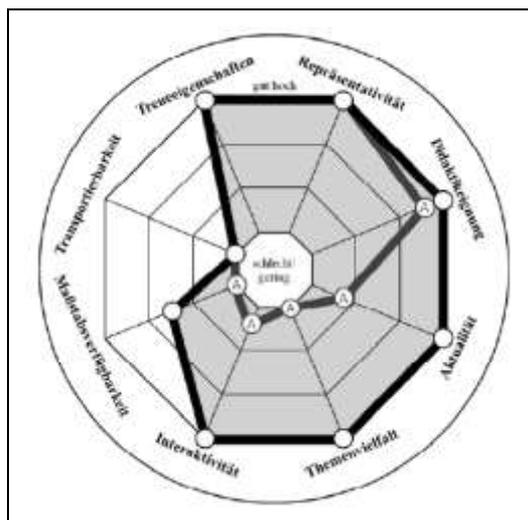


Abbildung 16 - Graphisches Globenprofil eines taktilen Hyperglobus im Vergleich zu einem analogen Globus, Quelle: [WIN-12], S. 9

Wie bereits im Kapitel 2.1. (Abb. 1) aufgezeigt werden konnte, ist unter anderem die Treueeigenschaft sehr gut einzustufen, da der Globus als einzige kartographische Ausdrucksform die Erde unverzerrt wiedergeben kann und somit alle Treueeigenschaften gegeben sind, auch wenn interaktive Inhalte auf dem Globus gezeigt werden (vgl. [WIN-12], S. 9-11).

Aufgrund der enormen Größe des Hyperglobus ist die Transportierbarkeit eines solchen Körpers eingeschränkt und daher schwer möglich. Es kann festgehalten werden, dass das Präsentationsgerät (die Globenkugel) und die anzuzeigenden Daten getrennt sind, da letztere auf einem externen PC vorhanden sind (vgl. [WIN-12], S. 9-11).

Der Maßstab eines taktilen Hyperglobus kann nicht variiert werden, da dieser aufgrund des starren Globenkörpers durch den Durchmesser bestimmt ist. Um bei speziellen Thematiken näher an das integrierte Kartenmaterial heranzukommen, würde sich die Implementierung einer kartographischen Lupe, welche bereits am Institut für Geographie und Regionalforschung – Arbeitsgruppe Kartographie und Geoinformation – umgesetzt wurde, anbieten (vgl. [WIN-12], S. 9-11).

Im Bereich der Interaktivität gibt es keine Grenzen nach oben. Das Wechseln von Karten, Bildern sowie Grafiken ist jederzeit möglich und steuerbar. Der Bereich von touchfähigen

Globen ist noch in der Entwicklungsphase. Im Rahmen einer Diplomarbeit von Mag. Michael Holzapfel [HOL-12] wurde ein Lösungsweg entwickelt, am bestehenden Hyperglobus des Instituts für Geographie und Regionalforschung mittels Infrarot-Technologie ein System zu entwickeln, welches Berührungen erkennen kann (vgl. [WIN-12], S. 9-11 und [HOL-12], S. VII).

Auf Hypergloben können eine Vielzahl von Themen dargestellt werden, da die Verfügbarkeit von Geodaten immer mehr zunimmt und viele auch frei zur Verfügung stehen. Die NOAA bietet unter anderem über ihr Projekt SOS fertige Stories an, die via Hyperglobus präsentiert werden können. Ein weiterer Vorteil ist, dass nicht nur – wie am analogen Globus – statische Bilder dargestellt, sondern auch dynamische Inhalte aufgezeigt werden können (vgl. [WIN-12], S. 9-11).

Die Aktualität bei digitalen Globen spielt eine sehr große Rolle. Im Gegensatz zu den analogen Globen stellt dies bei digitalen Globen kein Problem dar, weil aufgrund von Initiativen im Bereich von OpenData jederzeit ein aktuelles Bild der Erde zur Verfügung steht. Außerdem können andere Thematiken, bei denen Echtzeitdaten benötigt werden, so automatisiert werden, dass kein direktes Eingreifen des Nutzers notwendig wird (vgl. [WIN-12], S. 9-11).

Taktile Hypergloben sind sehr repräsentativ. Aufgrund ihrer Größe und ihres Erscheinungsbildes stellen sie einen Blickfang dar (vgl. [WIN-12], S. 9-11).

Im Rahmen des Geographie und Wirtschaftskunde Unterrichts kann ein taktiler Hyperglobus gut eingesetzt werden, da globale Thematiken für Schülerinnen und Schüler leichter als auf analogen Karten fassbar werden. Grund für dieses Problem ist, dass die Kinder sich aufgrund der Verebnung der Erde schwer tun. Außerdem ist die Wahl des Mittelmeridians für einige schwer begreifbar. Ein klassisches Beispiel hierfür ist, dass bei einem Mittelmeridian in Europa Russland und die USA getrennt dargestellt werden, aber die Länder eigentlich sehr nahe aneinander liegen. Leider gibt es in ganz Österreich keine Schule, die so einen Globus im eigenen Inventar hätte, da das Budget einer Schule für so eine Anschaffung nicht ausreichen würde. Nach Mag. Stefan Glaser, Diplomarbeit [GLA-12] *„Der taktile Hyperglobus und seine Wirkung auf den Lernzuwachs am Beispiel von Schülerinnen und Schülern der Oberstufe“*, könnte ein taktlier Hyperglobus gut in der Schule eingesetzt werden. Da jedoch die Kosten exorbitant hoch sind, ist es derzeit noch nicht möglich, einen Globus im Regelunterricht einzusetzen.

2.5 Taktile Hyperglobus der Hyperglobe Research Group (HRG)

Im Jahr 2005 wurde die Hyperglobe Research Group (HRG) am Institut für Geographie und Regionalforschung unter der Leitung von Ass.-Prof. Dr. Andreas Riedl an der Universität Wien gegründet. Ein Jahr später wurde ein taktile Hyperglobus für Forschungszwecke angekauft. Dieser war somit der erste taktile Hyperglobus an einer europäischen Universität. Der spiegelbasierte Globus wurde von der Firma ARC Science Simulation aus den USA erworben und hat einen Durchmesser von 1,5 Meter und einen Maßstab, verglichen zur Erde, von 1:8,5 Mio. (vgl. [RIE-10], S. 3 und [RIE-11b], S. 2).

Abbildung 17 zeigt den Aufbau dieses Omniglobes, welcher aus Globenkörper, Sockel, einem oder zwei Projektoren, Umlenkspiegel und einem konvexen Spiegel besteht. Wie ersichtlich, befinden sich die Projektoren und die Umlenkspiegel im Sockel des Globus, wodurch die Lichtstrahlen in den Globenkörper abgelenkt werden. Ganz oben befindet sich der konvexe Spiegel, der die Lichtstrahlen, welche vom Projektor ausgesandt werden, auf die Innenseite der Kugel verteilt. Wie bereits in Kapitel 2.3.1. bei den Projektionsverfahren angesprochen, entstehen bei einem spiegelbasierten System zwei blind-spots. Außerdem ist der Globenkörper mit einem speziellen Material beschichtet, sodass es möglich ist, das auf die Innenseite projizierte Bild auch auf der Außenseite für den Betrachter sichtbar zu machen (vgl. [KRI-12], S. 18).



Abbildung 17 - Omniglobe, Quelle: [KRI-12], S. 17

Nach dem Aufbau und der Installation beschäftigte sich die HRG gemeinsam mit der deutschen Firma Globocess mit der Programmierung einer plattformunabhängigen Software mit dem Namen „OmniSuite“, welche im Rahmen einer Diplomarbeit im Jahr 2012 von Mag. Jürgen Kristen (vgl. [KRI-12]) umgesetzt wurde und derzeit laufend weiterentwickelt wird (vgl. [RIE-08], S. 180 und [RIE-11b], S. 2). Mehr zur Software erfahren Sie in Kapitel 7.1.

Eigentlich können alle globalen Thematiken als „global stories“ auf einem taktilen Hyperglobus dargestellt werden. Am besten eignen sich dabei komplexe Thematiken, da die Inhalte auf einem realen Globuskörper problemlos raum-zeitlich verändert werden können und somit für den Betrachter einfacher und greifbarer werden.

Nach Riedl [RIE-08], S. 13 sind dies folgende Themenbereiche, welche sich für die Betrachtung am taktilen Hyperglobus eignen:

Im Prinzip eignen sich alle globalen Thematiken zur Visualisierung am taktilen Hyperglobus, unter anderem das Erscheinungsbild der Erde und alle anderen Planeten. Außerdem gibt es Animationen zur Plattenverschiebung und deren Auswirkungen auf die Erde. Daraus entstehen wiederum Thematiken zu Naturkatastrophen wie zum Beispiel Vulkantätigkeit, Erdbeben (siehe Abb. 18), Stürme, Feuer, etc. auf der Erde. Es können auch wirtschaftliche Themen am Hyperglobus betrachtet werden, zum Beispiel die Wirtschaftsexporte aus einzelnen Ländern oder die diversen Routen von Luftfahrzeugen und vieles mehr. Von Seiten des Projekts SOS und von der Firma Globocess wurden Datenkataloge angefertigt, um einen Überblick aller Thematiken kategorisiert zu bekommen. Diese Datenkataloge können unter <https://sos.noaa.gov/datasets/catalog/datasets> und <http://globocess.at/showroom> abgerufen werden.

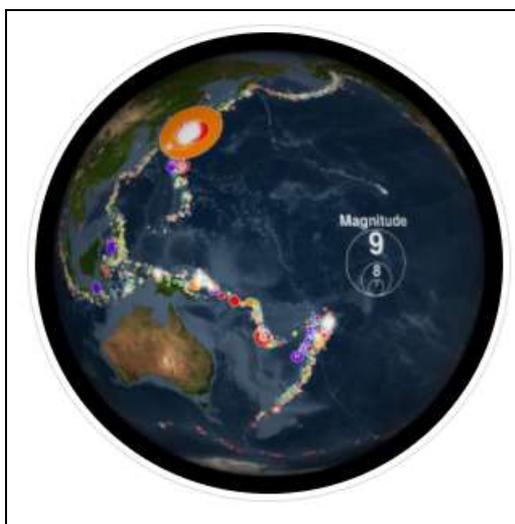


Abbildung 18 - Erdbebenbeispiel, Quelle: <https://sos.noaa.gov/datasets/>

3 Real Time-Visualisierung

In diesem Kapitel sollen die Begriffe „Real Time“ und „Visualisierung“ erläutert werden. Am Beginn wird der Begriff der Visualisierung allgemein abgegrenzt und im Bezug zur Wissenschaft erläutert. Anschließend wird auf den Begriff der Geo-Visualisierung näher eingegangen und die Bedeutung für den kartographischen Kommunikationsprozess hinsichtlich der Visualisierung von Daten am Hyperglobus. Danach wird auf den Begriff der Echtzeit eingegangen und die verschiedenen Arten von Echtzeit erläutert. Abschließend wird für die praktische Umsetzung eine mathematische Berechnung durchgeführt, um zu verstehen, dass Real Time abhängig von der Thematik gesehen werden muss und es sich nicht nur im Sekunden beziehungsweise Minutenbereich abspielt, sondern Jahre oder Jahrhunderte auch Echtzeit sein kann.

3.1 Begriffsabgrenzung Visualisierung

Der Begriff „Visualisierung“ beschreibt die Erstellung graphischer Darstellungen von Daten, Strukturen und Zusammenhängen, um diese leichter und verständlicher transportieren zu können. Außerdem ist anzumerken, dass die Visualisierung keine neue Erfindung der modernen Welt, sondern bereits seit Jahrhunderten Tradition ist (vgl. [SCH-00], S. 1).

3.1.1 Visualisierung in der Wissenschaft

Im Bereich der Wissenschaften, wie zum Beispiel in der Astronomie, Meteorologie und Kartographie wurde schon immer visualisiert. In den Anfängen diente dabei die Visualisierung vor allem der Unterstützung von Seefahrt und dem Militär. Die bekanntesten Beispiele in diesem Bereich sind ein Kartenwerk über Sternpositionen aus dem Jahr 1603 und die Verwendung von Höhenlinien ab dem Anfang des 18. Jahrhunderts (Carte Géométrique de la France). Außerdem wurden zur Veranschaulichung der magnetischen Deklination (Edmond Halley, 1701) Isolinien im wissenschaftlichen Kontext zur Visualisierung verwendet (vgl. [SCH-00], S. 1).

Mit Beginn des Computerzeitalters wurden die meisten Darstellungen digital erstellt. Das bekannteste Beispiel war hier das System SAGE, welches im Jahr 1958 für die Luftraumüberwachung eingesetzt wurde. Ziel dieser Forschung war es, aus der Beschreibung

von Geometriedaten Anwendungen zu erstellen. Erst ab den 2000er Jahren, als die Geräte eine verbesserte Rechen- und Darstellungsleistung erreichten, ist man dazu übergegangen, die Visualisierung von Daten nicht-geometrischer Natur auch am Computer zu visualisieren (vgl. [SCH-00], S. 2).

Ab 1987 haben Mc Cormick, De Fanti und Brown die „Visualization in Scientific Computing“ geprägt, welche das Ziel der Visualisierung vorschreibt, deren Ziele bis heute noch Gültigkeit haben (vgl. [SCH-00], S. 1).

Diese zwei Ziele sind unter anderem:

„[Die Visualisierung] soll zum einen Ergebnisse präsentieren und damit das Verständnis und die Kommunikation über die Daten und die zugrundeliegenden Modelle und Konzepte erleichtern. Zum anderen soll sie die Analyse der Daten unterstützen, indem die Bilder so aufgebaut werden, da[ss] der Betrachter in der Lage ist, nicht nur zu sehen, sondern auch zu erkennen, zu verstehen und zu bewerten. Innere, sonst verborgene Zusammenhänge sollen aufgezeigt werden, die allein aus der Interpretation von Zahlenkolonnen nicht ableitbar wären.“ [SCH-00], S. 2

Diese beiden Ziele müssen aber mit großer Sorgfalt umgesetzt werden, da es schnell zu falschen Interpretationen, ausgelöst durch falsche Abbildungen, kommen kann (vgl. [SCH-00], S. 2).

Der Begriff Geographic Visualization (Geovisualisierung, GVIS) wurde im Jahr 1994 von MacEachren geschaffen und beschreibt eine Spezialform der Visualisierung, welche von einer Person ausgeht und ausgehend von einer Karte einen Kommunikationsprozess einleitet (siehe Abb. 19). Im Vordergrund stehen wie bei der klassischen Visualisierung die Begriffe Exploration, Analyse, Synthese und Präsentation von räumlichen Daten (vgl. [LAN-13], S. 276).

„Geovisualization integrates approaches from visualization in scientific computing (ViSC), cartography, image analysis, information visualization, exploratory data analysis (EDA), and geographic information systems (GISystem) to provide theory, methods, and tools for visual exploration, analysis, synthesis, and presentation of geospatial data [...]“. [LAN-13], S. 276

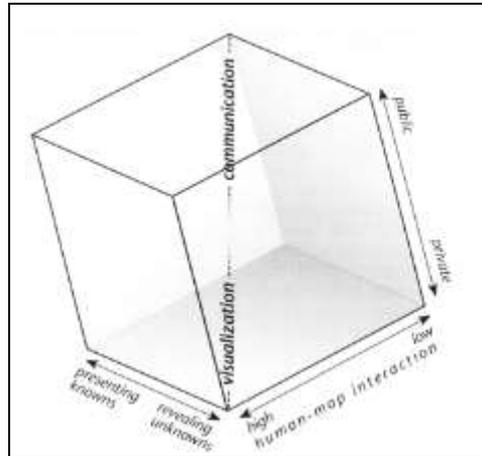


Abbildung 19 - Visualisierungsmethoden, Quelle: <https://www.e-education.psu.edu>

Dieser Ansatz der (Geo)Visualisierung findet in der Kartographie bereits Anwendung, da die Darstellung von Daten von großer Bedeutung ist. Nach de Lange [LAN-13], S. 277 werden beispielsweise Lagebeziehungen direkt und intuitiv erfasst. Des Weiteren können Nachbarschaften und Distanzen schnell visualisiert werden. Weitere Vorteile sind unter anderem die rasche Erkennung von räumlichen Strukturen und der Vergleich anderer räumlicher Erscheinungsformen. Die Wissensvermittlung setzt einen funktionierenden Kommunikationsprozess voraus.

Abbildung 20 zeigt den klassischen kartographischen Kommunikationsprozess. Am Beginn des Prozesses steht die Entwicklung eines Primärmodells, welches von der Realität ausgehend entwickelt wird. Es werden diverse Vereinfachungen bzw. Generalisierungen bei der Entwicklung durchgeführt. Der Fachmann (Topograph) nimmt die Daten aus der Umwelt auf und speichert sie mithilfe des Geoinformatikers im Primärmodell. Die Daten sind noch sehr datenreich und eventuell wenig generalisiert, aber die Daten haben bereits eine Struktur. Der Kartograph nimmt diesen Datensatz und verarbeitet ihn zu einem graphischen Produkt weiter; dann spricht man von einem Sekundärmodell. Festzuhalten ist, dass in jedem Schritt der Kommunikation Fehler und Interpretationsspielräume möglich sind und eine „korrekte“ der Realität entsprechende Wiedergabe der Realität nicht gewährleistet werden kann. Der Transfer der graphischen Umsetzung zum Konsumenten und die Erstellung des Tertiärmodells (Modell der Umwelt) bilden den letzten Schritt im kartographischen Kommunikationsprozess. Der Kommunikationsprozess ist somit abgeschlossen. Da der Benutzer Teilnehmer dieses Kommunikationsprozesses ist und mitunter wenig Vorwissen hat, ist es leicht möglich, dass Lese- und Interpretationsfehler auftreten (vgl. [LAN-13], S. 279-280).

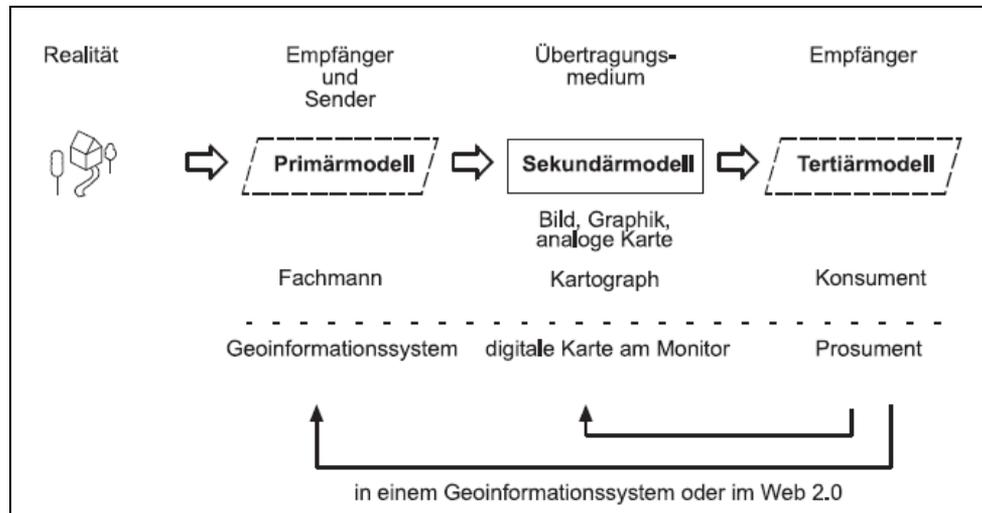


Abbildung 20 - Kartographischer Kommunikationsprozess, Quelle: [LAN-13], S. 277

Abbildung 21 zeigt eine technische Betrachtung des Visualisierungsprozesses. Im Rahmen des Datenimports/der Datengewinnung werden die Daten erfasst und gespeichert. Im nächsten Schritt erfolgt im Rahmen der Filterbereinigung die Reduzierung der Datenmenge. Dieser Schritt kann mit einer Generalisierung im kartographischen Prozess gleichgesetzt werden. Bei der Konvertierung erfolgt die Übertragung der Geometrie und der Attribute. Vergleichbar ist dieser Schritt mit dem Sekundärmodell (vgl. Abb. 20). Sobald die Daten dargestellt werden, handelt es sich um ein Tertiärmodell.

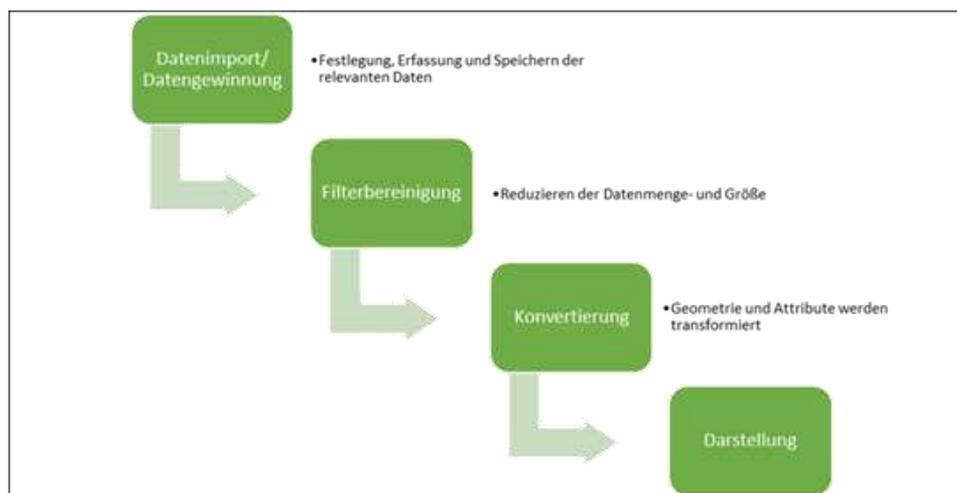


Abbildung 21 - Visualisierungspipeline, Quelle: [HOF-14] und <https://www.cg.tuwien.ac.at>

3.2 Allgemein Real Time (Echtzeit)

Der Begriff „Real Time“ stammt aus dem Englischen und bedeutet eigentlich nichts anderes als Echtzeit. Echtzeit als Begriff stammt aus der Informatik und beschreibt ein System, welches Aufgaben in einer bestimmten Zeitspanne ausführt (vgl. [MAY-07], S. 30). Das heißt so viel, dass Echtzeitsysteme nicht nur einen richtigen Wert berechnen, sondern auch noch zur richtigen Zeit liefern müssen. Wenn dies nicht passiert, ist das System gescheitert.

Hermann KOPETZ [KOP-11] grenzt den Begriff Echtzeitsystem wie folgt ab:

„A Real Time (computer) system is a (computer) system in which the correctness of the system behavior depends not only on the logical results of the computations, but also on the physical time when these results are produced“[KOP-11], S. 2

Ein Real Time-System ist kein System, welches einzeln vorhanden ist, sondern ist meist Teil eines größeren Computersystems.

An ein Echtzeitsystem werden folgende Anforderungen gestellt:

- Rechtzeitigkeit
- Gleichzeitigkeit
- Spontane Reaktion auf Ereignisse

In einer weiteren Erläuterung versteht Bill [BIL-96] unter Echtzeit, eine im Sekundenbereich ablaufende Anwendung. Im Rahmen dieser Arbeit wird der Begriff aber weiter gefasst, da dies vom Autor als nicht korrekt bzw. als eine zu enge Sichtweise auf die Thematik angesehen wird. Beispiele welche hier gerne herangezogen werden, sind unter anderem die Navigation, bei der im Sekundenbereich neue Daten verfügbar sind. Hingegen dauert es bei der Plattentektonik Jahrtausende, bis Veränderungen sichtbar werden. Abbildung 22 zeigt einen Überblick über das dehnbare Konstrukt der Echtzeit und Beispiele für alle zeitlichen Abschnitte. Natürlich kann darüber diskutiert werden, ob alle Ereignisse, die über den Minutenbereich hinausgehen noch als Echtzeitsysteme betrachtet werden können, oder nicht. Der Verfasser ist der Meinung, dass diese weiterhin zu den Echtzeitsystemen gehören, da es zum Beispiel bei der Plattentektonik sekundlich zu kleinen Verschiebungen kommen kann, aber diese in einem anderen Zeitintervall abgebildet werden müssen, um eine Veränderung wahrzunehmen. Ein Erdbeben, hervorgerufen durch eine größere Verschiebung von Platten, kennzeichnet eine große Auswirkung von kleinen im

Sekundenbereich stattgefundenen Bewegungen. Meiner Meinung nach ist es wichtig, dass gerade Erdbebenaufzeichnungen in Real Time stattfinden, da dies auch Vorteile für den Katastrophenschutz hat.

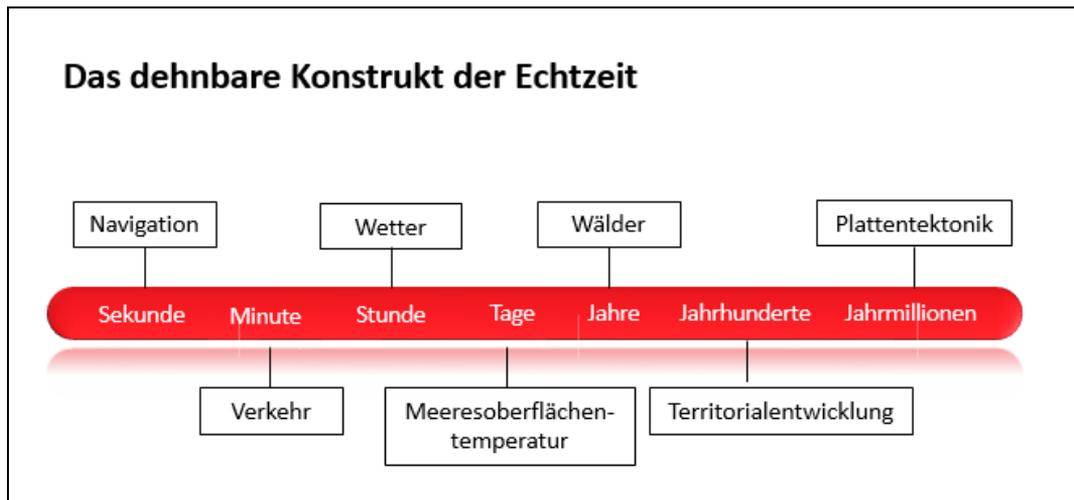


Abbildung 22 - Das dehnbare Konstrukt der Echtzeit, Quelle: verändert nach [MAY-07], S. 31 –
Quelle: Eigene Darstellung

Als nächster Schritt muss das Zeitintervall berechnet werden, in welchem die Real Time-Flugdaten aktualisiert werden müssen, um die Signatur des Luftfahrzeuges um 1 Pixel im Rasterbild weiter zu verschieben.

Berechnung des Zeitintervalls:

Bildgröße: 4096x2048 Pixel

Äquatorumfang: 40075,017 km

Ausbreitung von 1 Pixel am Äquator: $40075,017 / 4096 = 9,78$ km

Ø Geschwindigkeit eines Langstreckenflugzeuges: 800 km/h = 13 km/Minute

Bewegung um 1 Pixel: 44 Sekunden

Bei einer Bildgröße von 4096x2048 Pixel und einem Äquatorumfang von 40.075,017 km beträgt die Ausbreitung von 1 Pixel am Äquator 9,78 km. Wenn die durchschnittliche Geschwindigkeit eines Langstreckenflugzeuges mit einbezogen wird, beträgt die Bewegung um 1 Pixel auf der oben angesprochenen Bildgröße 44 Sekunden. Näheres zur konkreten Umsetzung wird in Kapitel 8 erläutert.

3.3 Arten von Echtzeit

Aufgrund des Verpassens von gesetzten Fristen unterscheidet man zwischen harter Echtzeit (hard Real Time) und weicher Echtzeit (soft Real Time). Diese beiden Unterscheidungen sind wichtig zu erklären, da beide Arten von Echtzeit bei Hypergloben vorkommen können.

3.3.1 Weiche Echtzeit (soft Real Time)

Weiche Echtzeitsysteme sind dadurch gekennzeichnet, dass die Eingaben schnell, im Sekundenbereich, verarbeitet werden. Wenn es zum Überschreiten der Fristensetzung kommt, kann dies zu einem gewissen Maß toleriert werden und ist weniger schlimm als bei harten Echtzeitsystemen. Sollte es zu keiner Reaktion des Systems kommen, wird der Vorgang als fehlgeschlagen angesehen. Als repräsentativstes Beispiel für solche Systeme ist das System für Videokonferenzen zu nennen. Hierbei kommt es lediglich zu einem Bildrauschen, sollte die Frist verpasst werden und das Bildsignal zu spät ankommen (vgl. [BRA-o.J.], S. 3-4 und [KOP-11], S. 2, 13-15).

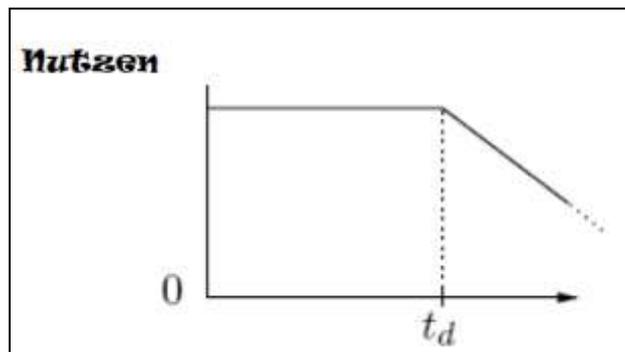


Abbildung 23 - Weiche Echtzeit, Quelle: [BRA-o.J.], S. 4

Abbildung 23 zeigt eine schematische Darstellung eines Ereignisses. Das Ereignis startet beim Zeitpunkt 0 und läuft ab, bei t_d soll die gesetzte Frist für die Umsetzung überschritten werden, sollte diese nicht erreicht werden, nimmt das Nutzen ab, hat aber bei weichen Echtzeitsystemen keine negativen Einfluss auf das gesamte System (vgl. [BRA-o.J.], S. 4).

3.3.2 Harte Echtzeit (hard Real Time)

Bei harten Echtzeitsystemen ist eine Überschreitung der gesetzten Frist ein Problem, da dies als Fehler gewertet wird. Zeitschranken müssen eingehalten werden, um das System funktionieren zu lassen. Abbildung 24 zeigt ein hartes Echtzeitsystem. Das Ereignis startet

wiederum beim Zeitpunkt 0 und läuft ab, bei t_d soll die Umsetzung des Ereignisses innerhalb einer harten Frist erreicht werden. Sollte diese Umsetzung nicht erfolgen, nimmt das Nutzen schlagartig ab und es entsteht ein großer Schaden, da die Zeitschranke eingehalten werden muss. Ein aktuelles Beispiel ist das rechtzeitige Anhalten von autonomen Fahrzeugen vor einer Ampel oder einer auftretenden Gefahr auf der Straße. Harte Echtzeitsysteme arbeiten im Millisekundenbereich, um bei dem genannten Beispiel sehr rasch reagieren zu können (vgl. [BRA-o.J.], S. 4).

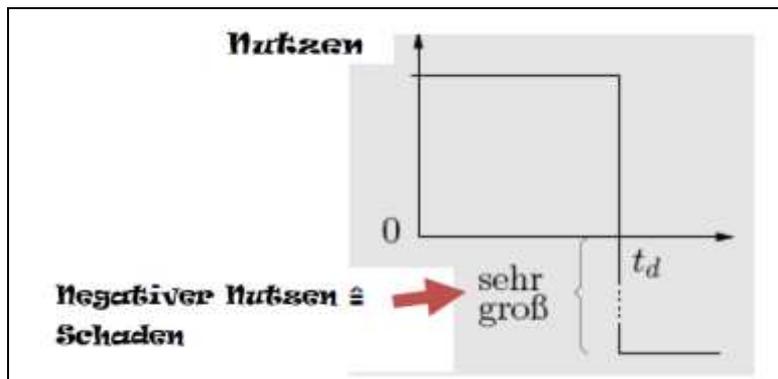


Abbildung 24 - Harte Echtzeit, Quelle: [BRA-o.J.], S. 4

Abbildung 25 fasst nach Braun [BRA-o.J.], S. 4 die Unterschiede zwischen harter Echtzeit und weicher Echtzeit nochmals tabellarisch zusammen:

Charakteristik	Harte Echtzeit	Weiche Echtzeit
Antwortzeit	hart	weich
Kontrolle des Tempos	Umgebung	Computer
Fehlererkennung	System	Benutzer
Sicherheit	kritisch	unkritisch
Redundanztyp	aktiv	Stand-by
Granularität	Millisekunde	Sekunde

Abbildung 25 - Gegenüberstellung von hard Real Time versus soft Real Time-Systems, Quelle: [BRA-o.J.], S. 4

Zusammenfassend kann festgehalten werden, dass hauptsächlich Daten verwendet werden, die der weichen Echtzeit zugeschrieben werden können, da bei keiner Anzeige von Daten am sphärischen Display es zu keiner Abnahme des Nutzens kommt und kein großer Schaden dadurch eintritt, sondern nur keine Inhalte für die/den AnwenderIn ersichtlich werden.

4 Vektor versus Raster

In diesem Kapitel wird am Beginn ein theoretischer Überblick über Vektor- und Rasterdaten gegeben, um auf die Unterschiede hinzuweisen. Außerdem wird auf Rasterdatenformate, welche für die Globensoftware OmniSuite (siehe Kap. 7) relevant sind näher eingegangen und Performance Tests mit unterschiedlichen Rasterbilder mit unterschiedlichen Inhalt durchgeführt. Am Ende werden Vektor- sowie Rasterdaten gegenübergestellt und deren Vor- und Nachteile ausgearbeitet.

4.1 Vektordaten(-modell)

Bei Vektordaten handelt es sich um Punkte, Linien oder Flächen (siehe Abb. 26). Die Lage der Vektoren wird durch x-/y-Koordinaten beschrieben. Bei Punkten ist es ein Koordinatenpaar (x/y), welches die Lage im Raum beschreibt. Bei Linien wird der Startpunkt mit x-/y-Koordinaten angegeben, zusätzlich noch der Winkel und eine Länge für die Richtung der Linie (vgl. [GEO-11], S. 69). Zusätzlich können auch Höhenwerte (z-Werte) bei den Vektoren inkludiert sein.

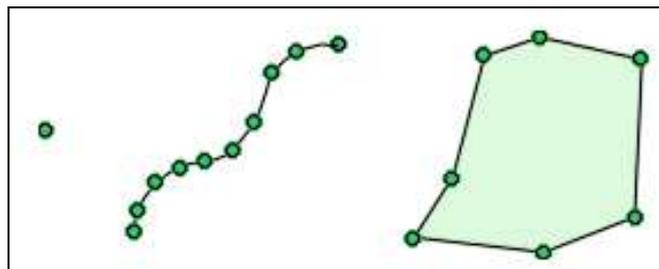


Abbildung 26 - Übersicht von Vektordaten, Quelle: <http://www.gis-news.de>

Aus Vektordaten entstehende Geoobjekte sind durch Attribute und Eigenschaften gekennzeichnet, die die Daten aufgrund der unterschiedlichen Thematik bekommen (vgl. [LAN-13], S. 139). Daraus ergeben sich nach De Lange [LAN-13], S. 139 unter anderem folgende statistische Skalenbereiche:

- Nominalskala – Bsp.: Postleitzahlen
- Ordinalskala – Bsp.: Schulnoten
- Intervallskala – Bsp.: Grad Celsius (kein absoluter Nullpunkt)

- Ratioskala – Bsp.: Alter in Jahren (absoluter Nullpunkt vorhanden)

Diese Geoobjekte können sich auch zeitlich verändern. Hier unterscheidet de Lange [LAN-13], S. 140 zwischen zwei Arten der Veränderung:

- Zeitliche Variabilität
- Räumliche Variabilität

Bei der zeitlichen Variabilität werden neue Daten aufgenommen oder aktualisiert, aber der Standort der Aufnahme wird nicht geändert und bleibt konstant. Als Beispiel kann eine Wetterstation herangezogen werden. Bei der räumlichen Variabilität hingegen wird ausschließlich die Lage des Geoobjekts bei der Aufnahme oder Aktualisierung verändert. Bei dem in der Masterarbeit verwendeten Flugdaten handelt es sich um eine Kombination von den beiden oben aufgeführten Variabilitäten, nämlich um eine raumzeitliche Variabilität, da sich das Luftfahrzeug nach jeder Aktualisierung der Daten (Zeit) auch weiterbewegt (Raum) hat. Diese kann mittels Geoinformationssystem nur sehr schwer dargestellt werden, deswegen kommt es zur Teilung der räumlichen und zeitlichen Daten. Um die raumzeitlichen Thematiken einfach darstellen zu können, werden meist dynamische Modelle herangezogen (vgl. [LAN-13], S. 140).

Im Folgenden werden die drei vektororientierten Datenmodelle nach Hake [HAK-02], S. 158-159 näher erläutert:

„Das **Spaghetti-Datenmodell** (siehe Abb. 27) entsteht bei der linienweisen Digitalisierung von Landkarten. Nachbarschaftsbeziehungen können dabei nur aus den redundant gespeicherten Koordinaten errechnet werden. Deshalb ist dieses Datenmodell für die Kartenherstellung gut geeignet.“ [HAK-02], S. 158-159

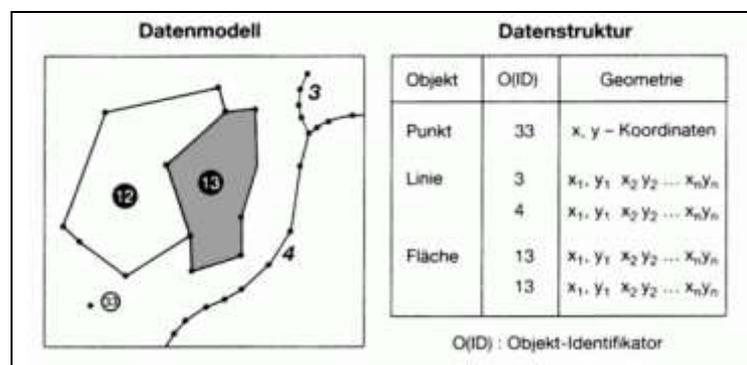


Abbildung 27 - Spaghetti-Datenmodell, Quelle: [HAK-02], S. 158

„Beim **topologischen Datenmodell** (siehe Abb. 28) werden die Nachbarschaftsbeziehungen explizit modelliert, indem jeder Kante die Identifikation (Objektnamen oder -nummern) der Objekte zur Linken bzw. zur Rechten zugeordnet werden. Durch die Auswertung des mittels Knoten und Kanten beschriebenen Modells (planarer Graph) lassen sich flächenhafte Objekte bilden. Aufgrund der redundanzfreien Speicherung stellt dieses Datenmodell bereits eine erhebliche Verbesserung gegenüber dem Spaghetti-Modell im Hinblick auf raumbezogene Analysen dar, jedoch sind bei großräumigen Auswertungen zeitraubende sequentielle Suchprozesse erforderlich, bis zum Beispiel die Menge aller zu einer Masche gehörenden Kanten gefunden worden ist.“ [HAK-02], S. 158-159

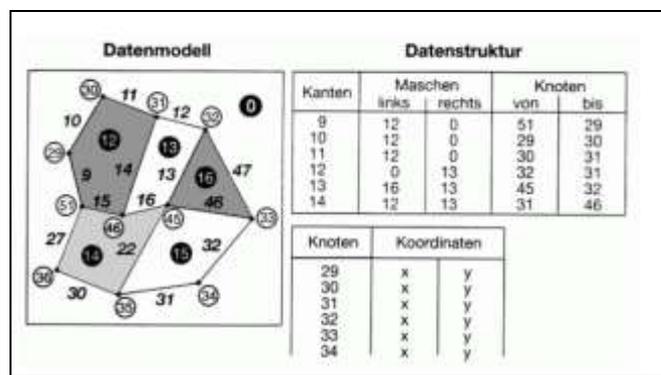


Abbildung 28 - Topologisches Datenmodell, Quelle: [HAK-02], S. 159

„Dieser Nachteil ist beim **hierarchischen Vektor-Datenmodell** (siehe Abb. 29) als Weiterentwicklung des topologischen Datenmodells beseitigt worden (Peucker/Chrisman 1975). Dazu wurde die Kette als logisches Grundelement eingeführt. Jede Kette ist durch einen Anfangsknoten und einen Endknoten begrenzt. Die Anzahl der Ketten hängt lediglich von der Anzahl der Maschen ab, nicht aber von der Anzahl der Stützpunkte, d.h. der Form der Ketten. Die durch Listen realisierte Hierarchie von Knoten, Ketten und Maschen unterstützt auch großräumige Auswertungen effizient.“ [HAK-02], S. 158-159

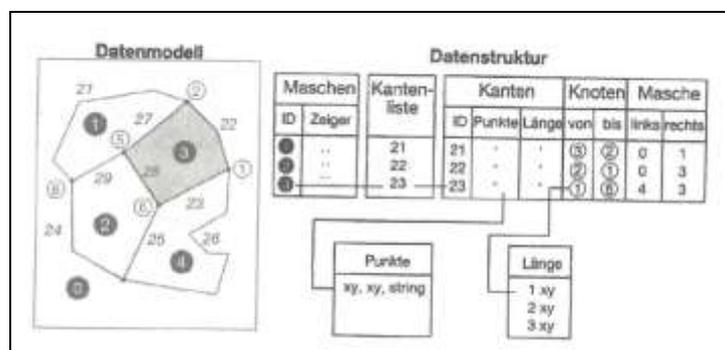


Abbildung 29 - Hierarchisches Vektor-Datenmodell, Quelle: [HAK-02], S. 15

Das hierarchische Modell lässt sich von den anderen beiden aufgrund der Art der erhobenen Daten unterscheiden. Durch Kanten und Knoten entstehende Flächen machen es möglich, direkt Bezug auf Nachbarschaften zu nehmen. Um den Rechenaufwand der dabei entsteht zu vermindern, wird im hierarchischen Vektordatenmodell eine Fläche als Kette (Kanten und Knoten mit identem Start- und Endpunkt) herangezogen, die aber den gleichen Informationsgehalt hat.

4.2 Rasterdaten(-modell)

Bei Rasterdaten handelt es sich um keine Vektoren, bei denen Informationen gespeichert werden, sondern um eine Matrix aus Zellen, so genannte Pixel (siehe Abb. 30). Die Mitte (x/y) des linken oberen Pixels beschreibt die richtige Lage im Raum. Pro Pixel wird ein Wert abgespeichert, dieser kann zum Beispiel ein Höhen- oder Farbwert sein (vgl. [GEO-11], S. 70).

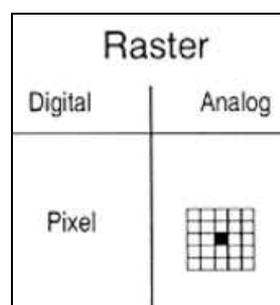


Abbildung 30 - Raster-Format, Quelle: <http://ivvgeo.uni-muenster.de/>

4.2.1 Rasterdatenformate

Das richtige Bildformat ist für das Laden von Bildern beim Hyperglobus essentiell, da die Bildberechnung durch Echtzeit (Real Time-Rendering) geprägt ist. Es werden bis zu 30 Bilder pro Sekunde geladen, gelegentlich kann dies auf mehreren Ebenen vorkommen. Außerdem ist die Farbtiefe der verwendeten Bilder wichtig. Der taktile Hyperglobus verwendet den Farbraum RGB, das heißt es werden die Farbwerte der Kanäle Rot, Grün und Blau angewandt. Am häufigsten werden 8 Bit pro Kanal verwendet, dies entspricht einem 24 Bit oder True-Color Bild, was mehr als 16,7 Mio. darstellbare Farben bedeutet (siehe Tab. 4). Um Rasterbilder am Hyperglobus anzeigen zu können, werden nur zwei Datenformate unterstützt. Auf der einen Seite ist es das Portable Network Graphics-Format (PNG) und das Direct Draw Surface-Format (DDS).

Farbtiefe (bpp)	Farbanzahl	Fotorealistisch	Bezeichnung
1	2	nein	Bitmap
4	16	nein	
8	256	eingeschränkt	Pseudo Color
16 (15)	65 536 (32 768)	annähernd	High Color
24	16 777 216	ja	True Color
32	16 777 216 plus Sonderkanal	ja	True Color plus Sonderkanal

Tabelle 4 - Übersicht der Farbtiefe bei Rasterbildern

4.2.1.1 PNG-Format (Portable Network Graphics)

Das PNG-Format weist eine verlustfreie Kompressionsart auf, das heißt es kann die Dateigröße so oft reduziert werden, ohne dass Informationen verloren gehen. Das Pendant zum PNG ist das JPEG-Format, welches eine verlustbehaftete Kompressionsart aufweist. Es werden weniger wichtige Informationen in der komprimierten Datei nicht mehr gespeichert. Daher ist es nicht möglich die Ursprungsqualität des Bildes zurückzugewinnen. JPEG Bilder sind somit weniger gut geeignet für die Verwendung in OmniSuite.

4.2.1.2 DDS-Format (Direct Draw Surface)

Das Direct Draw Surface-Format wurde von Microsoft gezielt für Echtzeitrendering Anwendungen entwickelt und ist im Direct X inkludiert. DDS ermöglicht eine Kompression, die direkt in der Grafikkarte verwendet werden kann. Anders als beim JPEG- oder PNG-Format ist keine Dekodierung notwendig, da die komprimierten Daten direkt in den Speicher der Grafikkarte geladen werden können; dadurch wird dort auch Speicherplatz gespart. Als Kompressionsalgorithmen stehen die S3 Texture Compression Algorithmen zur Auswahl, welche verlustbehaftet sind (vgl. [BRO-13]).

Der S3TC Algorithmus wurde in fünf Varianten implementiert und sind nach den ihnen in DirectX zugewiesenen Codes (FourCC) benannt: DXT1 bis DXT5. DXT2 sowie DXT4 werden kaum verwendet (vgl. [BRO-13]).

Bei der Verwendung der DXT-Kompression wird die Textur in 4x4 Pixel-Blöcke zerlegt, welche anschließend komprimiert werden. Von diesen 16 Pixeln wird eine Farbtabelle aus 4 Farben bestimmt und jedem Pixel wird ein Farbwert aus dieser Farbtabelle zugewiesen. In der Farbtabelle werden nur die zwei Extremwerte gespeichert, die beiden anderen Farbwerte werden aus den Extremen interpoliert. Die Farbwerte selbst werden als 16 Bit Farbwerte gespeichert (Rot: 5 Bit, Grün 6 Bit, Blau 5 Bit) (vgl. [BRO-13]).

Für DXT1 resultieren die zwei 16-Bit RGB Farbwerte und eine 4x4 2-Bit Lookup-Tabelle in 64-Bit für den 4x4 Pixel-Block. Dies ergibt eine 1:8 Kompression gegenüber den 512 Bits ohne Kompression (vgl. [BRO-13]).

DXT1 mit Alpha ermöglicht das Speichern von 1 Bit im Alpha Kanal. Dies bedeutet entweder volle oder keine Transparenz. Einer der vier Farben der Farbtabelle wird dazu als transparent definiert und limitiert damit weiter den Farbumfang (vgl. [BRO-13]).

DXT3 und DXT5 speichern die Farbwerte wie DXT1, ermöglichen aber das Speichern eines Transparenzkanals. Beide speichern diesen Transparenzkanal mit zusätzlichen 64 Bit, was in einer Kompressionsrate von 1:4 resultiert. DXT3 speichert die Transparenzinformation explizit mit 4 Bit. Dies ergibt 16 Transparenzstufen. DXT5 speichert die Transparenzinformation ähnlich wie die Farbinformation mit Hilfe einer „Farb“-Tabelle und einer 4x4 3-Bit Lookup-Tabelle. Gespeichert werden zwei Extremwerte (8-Bit Alphawerte), 6 weitere Alphawerte werden interpoliert. Dies ermöglicht einen gleichmäßigeren Transparenzverlauf (vgl. [BRO-13]).

Für die Bildqualität bedeutet das, dass die Datenkompressionsrate fix ist – d.h. die Dateigröße ist bei gleicher räumlicher Auflösung für jedes Bild gleich. Dies bedeutet auch, dass der Qualitätsverlust vom Inhalt des Bildes abhängig ist. Die Tatsache, dass 16 Bit für das Speichern der Farbwerte verwendet werden, bedeutet den Verlust von feinen Farbnuancen im Vergleich zu 24 Bit Farbwerten. Die Farbtabelle ergibt sich aus zwei gespeicherten Farbwerten und zwei weiteren Farben, die linear interpoliert werden. Das bedeutet, wenn Farben nicht auf einer Linie im Farbraum liegen, gehen Farbinformationen verloren. Bilder mit vielen verschiedenen Farben auf kleinem Raum verlieren daher stärker an Qualität (vgl. [BRO-13]). Tabelle 5 zeigt eine Übersicht der am häufigsten genutzten Varianten von Bildformaten in OmniSuite.

	JPEG	PNG	DXT1	DXT1A	DDS		RGB
					DXT3	DXT5	
Farbtiefe (R-G-B bit)	8-8-8	8-8-8	5-6-5	5-6-5	5-6-5	5-6-5	8-8-8
Alphakanal (bit)	-	8	-	1	4 explizit	4 interpoliert	-
Verlustfreie Komprimierung	Nein	Ja	Nein	Nein	Nein	Nein	Ja

Tabelle 5 – Übersicht der in OmniSuite verwendeten Bildformate

Im nächsten Abschnitt soll eine Übersicht der Richtlinien zur Verwendung von Bildformaten in OmniSuite gegeben werden:

1) Datenmenge (Bilddaten müssen in Echtzeitgeladen werden)

- a. Bei Animationen mit hoher Veränderungsrate ist das DDS-DXT1 Format zu verwenden, bei geringer Veränderungsrate kann eventuell ein verlustfreies Format (z.B. PNG) verwendet werden
- b. Wenn eine Ebene keine globale Ausdehnung besitzt kann eventuell ein verlustfreies Format verwendet werden

2) Bildinhalte

- a. Bei einem Bildinhalt mit vielen Linien und Schrift ist ein verlustfreies Format zu bevorzugen, wenn die Datenmenge dies zulässt
- b. Bei Satellitenbildern oder vergleichbaren Karten ist das DDS-DXT1 Format ausreichend

3) Ebenen

- a. Wenn Teile des Kartenbildes selten verändert werden (z.B. Grenzen, Gradnetz, Namengut, Legenden..) sollte dieser Inhalt in einer eigenen Ebene und verlustfrei dargestellt werden. Die Qualität des eigentlichen Kartenbildes im DDS-DXT1 Format wird dadurch ebenfalls höher.

Hinsichtlich der Performance und Ladezeit wurden drei Tests mit den zuvor vorgestellten Bildformaten (JPEG, PNG und DDS) durchgeführt. Für die Vergleiche wurden unterschiedliche Farben, Signaturen und Texte des untersuchten Bildes verwendet. Allgemein kann am Anfang festgestellt werden, dass ein Consumer PC mit Festplatte (PC 1) und eine Workstation mit SSD (PC 2) verwendet wurde. Die Zeile „Colors“ entspricht der Zahl der im Bild verwendeten Farben. Weiters wird die Ladezeit in Millisekunden angegeben (entspricht der Ladezeit nach dem Warmstart – Mittelwert aus 4 Versuchen). Die Tabellen 6 bis 8 und die Abbildungen 34 bis 36 geben eine Übersicht über die Ladezeiten der unterschiedlichen Formate und zeigen die verwendeten Beispielbilder.

Test 1: Feine Farbschattierungen

Bildgröße: 4096 x 2048 Pixel (Vorschau im Bild ist 2-fach vergrößert)

	JPEG	PNG	DXT1	DXT1A	DDS DXT3	DDS DXT5	RGB
Colors	30071	9215	2020	-	1600	1600	9215
Disk Size	625,14 KB	3,87 MB	4,0 MB	-	8,0 MB	8,0 MB	24,0 MB
Ladezeit PC 1	164	297	163,75	-	253,5	281,5	344
Ladezeit PC 2	146,8	312	140	-	203	218	168,4

Tabelle 6 - Vergleich der Ladezeit von Test 1

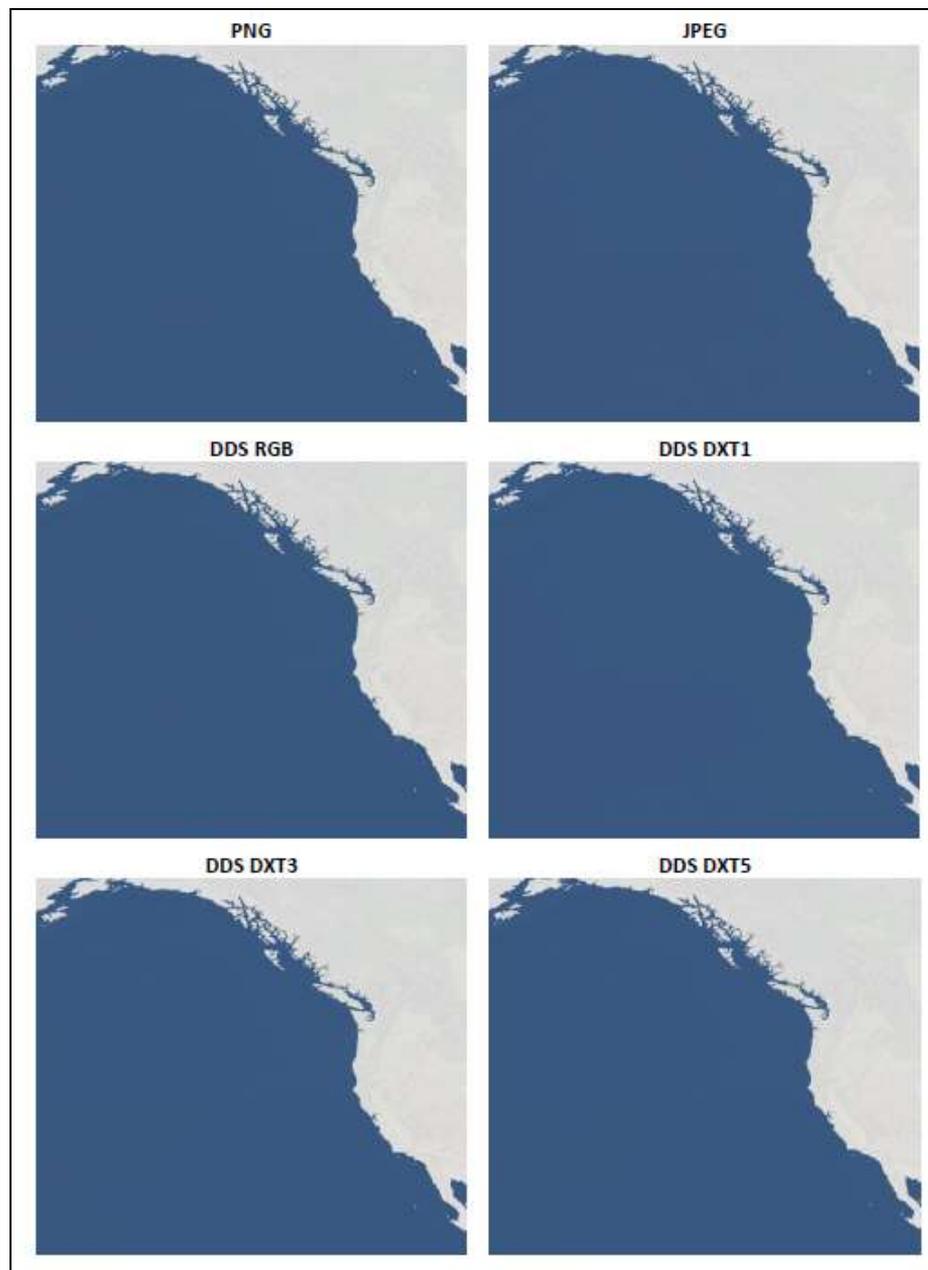


Abbildung 31 - Bildbeispiel für Test 1

Test 2: Naturnahe Farbgebung + Punktsignaturen + Text

Bildgröße: 2048 x 1024 Pixel (Vorschau im Bild ist 4-fach vergrößert)

	JPEG	PNG	DXT1	DXT1A	DDS DXT3	DDS DXT5	RGB
Colors	142999	202024	33787	-	32927	32927	202024
Disk Size	330,41 KB	3,17 MB	1,0 MB	-	2,0 MB	2,0 MB	6,0 MB
Ladezeit PC 1	49	78	51,5	-	73	78,66	99
Ladezeit PC 2	47	81	47	-	59,4	59,2	49,8

Tabelle 7 - Vergleich der Ladezeit von Test 2



Abbildung 32 - Bildbeispiel für Test 2

Test 3: Transparenz + Text

Bildgröße: 4096 x 2048 Pixel (Transparente Bereiche wurden rosa eingefärbt)

	JPEG	PNG	DXT1	DXT1A	DDS DXT3	DXT5	RGB
Colors	-	933	-	254	373	854	-
Disk Size	-	848,3 KB	-	4,0 MB	8,0 MB	8,0 MB	-
Ladezeit PC 1	-	180	-	156,33	250	250	-
Ladezeit PC 2	-	173	-	134,6	203	203	-

Tabelle 8 - Vergleich der Ladezeit von Test 3

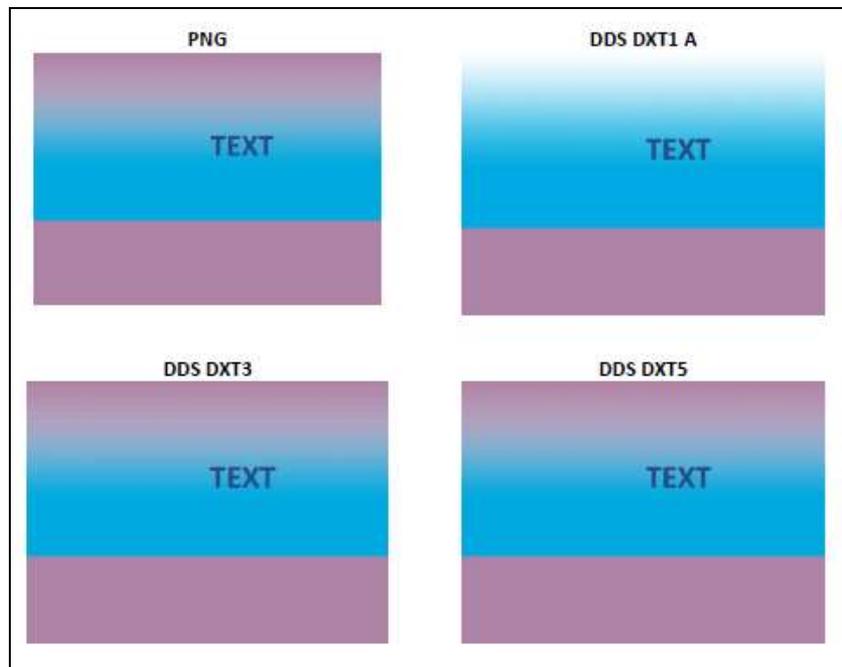


Abbildung 33 - Bildbeispiel für Test 3

Bei der Ladezeit konnte beobachtet werden, dass PNG-Dateien bei erneutem Laden die gleiche Ladezeit aufweisen. Bei den DDS-Dateien konnte bei erneutem Laden eine drastische Verkürzung der Ladezeit festgestellt werden.

4.3 Vergleich zwischen Vektor- und Rasterdaten

In diesem Kapitel sollen die Unterschiede zwischen Vektor- und Rasterdaten herausgearbeitet und eine Empfehlung abgegeben werden.

Laut Riedl [RIE-12], Harcke [HAR-09] und de Lange [LAN-13] ergeben sich einige Vor- und Nachteile:

Vektordaten

Vorteile	Nachteile
geringe Datenmenge und geringer Speicherbedarf	Hohe Erfassungszeit
kurze Rechenzeit	komplexe Datenstruktur
logische Strukturierung und Objektbezug einfach realisierbar	
hohe Punktgenauigkeit	
geringer Qualitätsverlust	

Tabelle 9 - Vor- und Nachteile von Vektordaten, Quelle: [RIE-12], [HAR-09] und [LAN-13]

Rasterdaten

Vorteile	Nachteile
einfache Datenstruktur	große Datenmenge und großer Speicherbedarf
flächenhafte Betrachtungsweise	lange Rechenzeiten
niedrige Erfassungszeit	logische Strukturierung und Objektbezug kaum möglich
Möglichkeit einfacher räumlicher Analysen	Verschlechterung der Qualität bei Skalierung (Treppeneffekt)

Tabelle 10 - Vor- und Nachteile von Rasterdaten, Quelle: [RIE-12], [HAR-09] und [LAN-13]

Allgemein kann festgehalten werden, dass sich Vektordaten grundsätzlich für großmaßstäbige Thematiken besser eignen als für kleinmaßstäbige Bereiche. Flächenhaft kontinuierliche Elemente sind teilweise besser mit Rasterdaten darstellbar, besonders weil

zum Beispiel an den Übergängen zu anderen Klassen die Möglichkeit besteht, eine Übergangsklasse zu erstellen. Bei linearen und punkthaften Elementen sind Vektoren wiederum besser geeignet. Dagegen werden Rasterdaten hauptsächlich bei Anwendungen, bei denen es um großräumige Überblicke von Regionen (Satellitenbilder) geht, sowie für die digitale Bildverarbeitung verwendet (vgl. [LAN-13], S. 356).

Zusammenfassend kann festgehalten werden, dass über die Vor- und Nachteile von Vektor- beziehungsweise Rasterdaten keine eindeutige Aussage über deren Eignung getroffen werden kann. Es muss angemerkt werden, dass beide Datenformate für die eine oder andere Fragestellung besser geeignet sind und sich somit nicht gegenseitig ausschließen, sondern gleichermaßen Verwendung finden (vgl. [LAN-13], S. 356).

Für das sphärische Display sind Rasterdaten sehr wichtig, da diese die Kartengrundlage (Plattkarte) und somit die Grundlage des Abbildes der Erde bilden. Vektordaten alleine können natürlich auch abgebildet werden, nur ist es dann meist schwierig zu erkennen, wo sich die Daten auf der Erde befinden, da die Kartengrundlage im Hintergrund (Raster) fehlt. Der Autor ist der Meinung, dass ohne Raster- oder Vektordaten keine Darstellung am sphärischen Display möglich wäre. Darum ist es essentiell mit Raster- und Vektordaten zu arbeiten und eine Kombination von Raster- und Vektordaten für eine gute und inhaltliche Abbildung für das sphärische Display zu empfehlen.

5 Mathematische Grundlagen am Globus

Dieses Kapitel beschäftigt sich mit dem Koordinatensystem am Globus und mit der quadratischen Platkarte als Abbildungsgrundlage des Globenbildes, welche bei der Globensoftware verwendet wird. Außerdem soll auf die Verzerrungsprobleme, hervorgerufen durch die Eigenschaft der quadratischen Platkarte eingegangen werden. Mittels der Formel der Tissot'schen Indikatrix soll erläutert werden wie das Verzerrungsproblem durch Berechnung gelöst werden kann und wie breit die Signaturen sein müssen um korrekt am Hyperglobus angezeigt zu werden. Des Weiteren soll auf die Darstellungen von Flugrouten (Orthodrome versus Loxodrome) und auf mögliche Probleme bei der Visualisierung am Globus eingegangen werden.

5.1 Koordinatensystem am Globus

Am Globus wird grundsätzlich ein geographisches Koordinatensystem verwendet, um die Position der Objekte am Globus zu verorten (vgl. [COZ-11], S. 13).

Bei geographischen Koordinaten, meist in Grad, wird die Position $[-180^\circ, +180^\circ]$ von West nach Ost angegeben. Diese Parallelen werden als Längengrade (longitude) bezeichnet. Die latitude verläuft von Süden nach Norden $[-90^\circ, +90^\circ]$ und diese Kreise werden als Breitengrade bezeichnet (siehe Abb. 34). Der Äquator teilt die Erde in zwei gleich große Hälften und ist der längste Breitengrad mit einem Wert von 0° . Der Nullmeridian wird als der Ursprung der Längengradkoordinaten mit 0° definiert. Im 2. Jahrhundert n. Chr. wurde der Nullmeridian von Claudius Ptolemäus auf die kanarische Insel El Hierro (Ferro) gelegt und bis ins 19. Jahrhundert verwendet. Der heute häufigste verwendete Nullmeridian ist die Linie, welche durch den englischen Ort Greenwich verläuft. Der Nullmeridian wurde 1884 bei der Internationalen Meridiankonferenz in Washington festgelegt. Anzumerken ist, dass auch andere Nullmeridiane bekannt sind zum Beispiel durch Bern, Bogota und Paris (vgl. [SCH-17], S. 34).

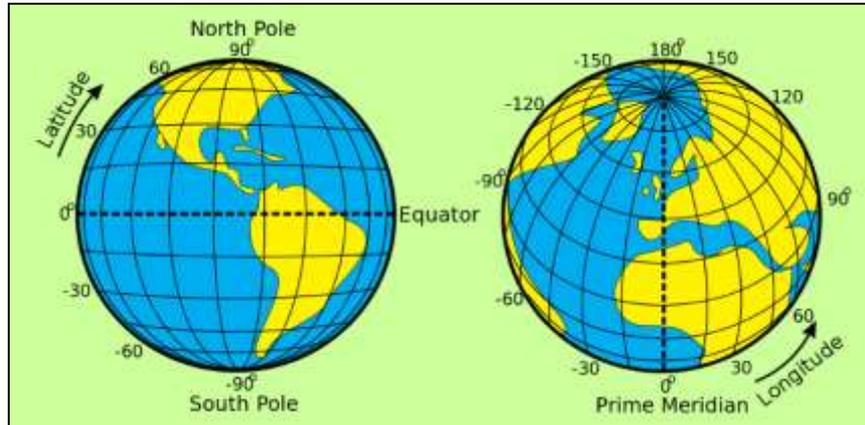


Abbildung 34 - Longitude und Latitude, Quelle: <http://www.geographvalltheway.com>

Breitenkreise und Meridiane umspannen den Globus und werden im weiteren Verlauf als Linien gesehen. Als geographisches Netz bezeichnet man die Überlagerung beider Linienarten. Dieses geographische Netz hat seinen Koordinatenursprung dort, wo der Nullmeridian den Äquator schneidet. Dieser Punkt westlich vom Kontinent Afrika hat die Koordinaten (0,0). Entlang des Äquators ist der Breitenkreis ungefähr so lang wie ein Meridian (ca. 111,321 km). In Richtung der Pole schrumpft die Länge der Breitenkreise. Da die Längengradlinien an den Polen konvergieren, ist der Abstand zwischen den einzelnen Meridianen an jedem Breitenkreis unterschiedlich. Je näher man zu den Polen rückt, desto größer ist der Unterschied zwischen einem Breitengrad entsprechender Strecke im Vergleich zu der Strecke eines Längengrades. Die Länge eine Breitengradlinie mithilfe des geografischen Netzes zu ermitteln, ist daher, mit Ausnahme des Äquators, kompliziert. Die Breitengradlinien sind anders als Meridiane konzentrische Kreise, die mit abnehmendem Abstand zu den Polen immer kleiner werden. An den Polen besteht ein Kreis nur noch aus einem einzigen Punkt, der zugleich der Ursprung der Meridiane ist. Am Äquator ist 1 Grad ca. 111,321 km lang. Am 60. Breitengrad beträgt der Wert für 1 Grad Länge hingegen nur noch 55,802 km (siehe Abb. 35). Zurückzuführen ist dies auf das Jahr 1866, in dem Clarke ein Sphäroid definierte, das die Erdkugel als berechenbaren Körper beschreibt. Dadurch, dass die Breiten- und Längengrade nicht einheitlich lang sind, können die Abstände zwischen Punkten mithilfe von Winkelmaßen nicht exakt ermittelt werden (vgl. [IBM-o.J.]).

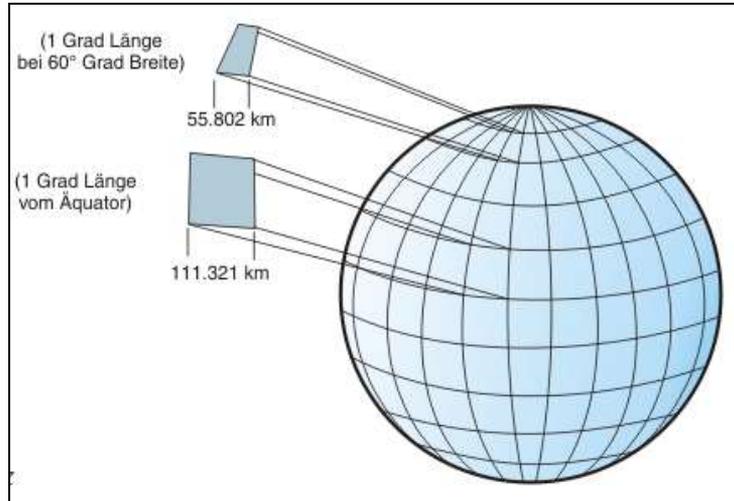


Abbildung 35 - Unterschied zwischen Positionen im geographischen Netz, Quelle: [IBM-o.J.]

5.2 Quadratische Plattkarte

Die quadratische Plattkarte (vgl. Abb. 36) ist die einfachste und älteste Projektionsform in der Kartographie und besteht aus quadratischen Kacheln, von denen auch die Bezeichnung abgeleitet wurde. Marinus von Tyros hat bereits rund um 100 n. Chr. diese Kartenprojektion das erste Mal erwähnt. In der verwendeten Software OmniSuite wird eine quadratische Plattkarte (abstandstreue Zylinderprojektion in normaler Lage mit dem Berührungskreis am Äquator) als Grundlage für das Erdbild verwendet (vgl. [LAM-11] und [SNY-93], S. 5-8).

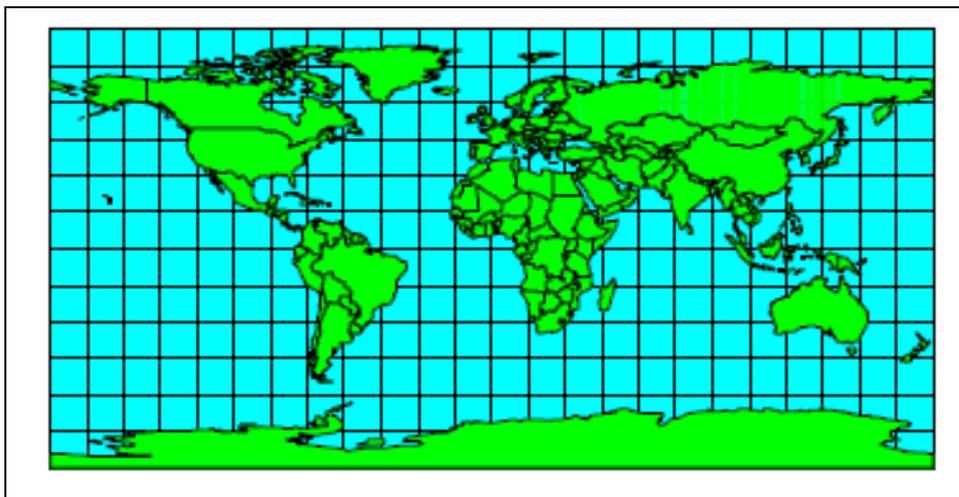


Abbildung 36 - Quadratische Plattkarte, Quelle: [KAI-12]

Ein Vorteil dieser Abbildung ist, dass die geographischen Koordinaten direkt aus der Plattkarte herausgelesen werden können, weshalb diese Art der Projektion gerne für computerbasierte Anwendungen verwendet wird (vgl. [RIE-10], S. 5).

Allgemein werden bei einer quadratischen Plattkarte der Äquator längentreu, die Meridiane und Parallelkreise als Geraden, die einander rechtwinkelig schneiden, abgebildet (vgl. [KAI-12]). Außerdem muss angemerkt werden, „dass das Seitenverhältnis einer quadratischen Plattkarte 2:1 beträgt, [weshalb] sämtliche Breitenkreise doppelt so lang dargestellt [werden] als sämtliche Längenkreise. Aufgrund der Tatsache, dass die Breitenkreise auf der Kugel mit wachsender Entfernung zum Äquator abnehmen, entstehen hier Probleme bei der Darstellung.“ [LAM-11], S. 19

Wie schon beschrieben, ist das Seitenverhältnis konstant 2:1. Da die Länge der Breitenkreise mit wachsender Breite auf der Erde (bzw. generell auf einer Kugel) immer geringer wird, auf der Karte jedoch gleich bleibt, entstehen Verzerrungen. Im Extremfall ist es so, dass für die Darstellung des Äquators genauso viele Pixel zur Verfügung stehen wie für die Abbildung des Poles. Dadurch werden in höheren Breiten mehr Pixel gespeichert als überhaupt verwendet werden können und es muss eine Interpolation der Pixel vorgenommen werden, was zu unsaubereren Darstellungen am Globus führen kann. [LAM-11], S. 19

5.3 Tissot'sche Indikatrix

Nicolas Auguste Tissot (1824-1897) war ein französischer Mathematiker, der die nach ihm benannte Tissot'sche Indikatrix erfand. Hier werden mit Hilfe von Verzerrungsellipsen Kartennetzentwürfe auf ihre Verzerrungseigenschaften mittels der Formel

$$\text{Verzerrung} = \frac{1}{\cos(\text{Breite in Grad})} \quad \text{überprüft.}$$

Dies bedeutet, dass bei einer geographischen Breite von 60°, die Signatur doppelt so breit wie hoch sein müsste, um diese korrekt anzuzeigen.

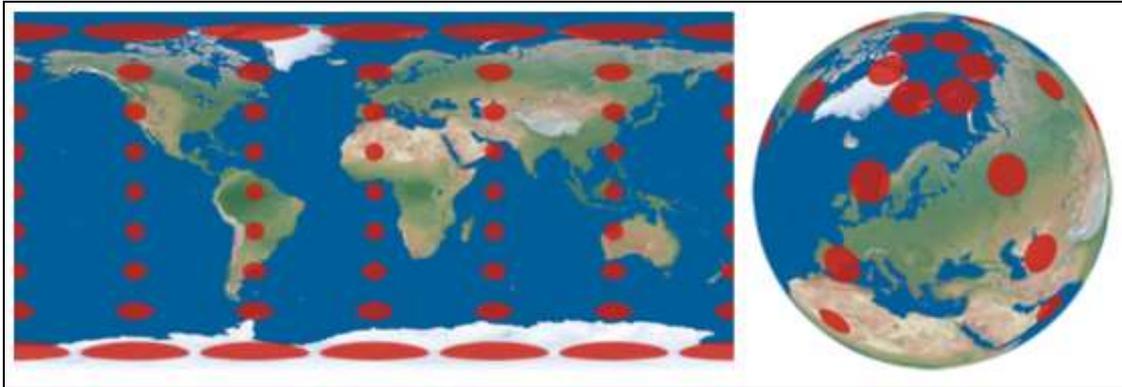


Abbildung 37 - Tissot'sche Indikatrix bei einer quadratischen Plattkarte, Quelle: [LIE-12], S. 2

Abbildung 37 zeigt, dass am und in der Nähe des Äquators die Symbole einen richtigen Kreis darstellen, sonst sind es Ellipsen. Je weiter man sich in Richtung der Pole bewegt, desto stärker ist die Verzerrung bei den Kreisen. Es muss hier nochmals erwähnt werden, dass sich nur die Breite des Kreises verändert und die Höhe konstant bleibt. Würde man die Verzerrung nicht berücksichtigen und alle Kreissignaturen auf der quadratischen Plattkarte gleich einzeichnen, würde sich folgendes Ergebnis am Globus zeigen (siehe Abb. 38) (vgl. [LAM-11], S. 20 und [LIE-12], S. 2).

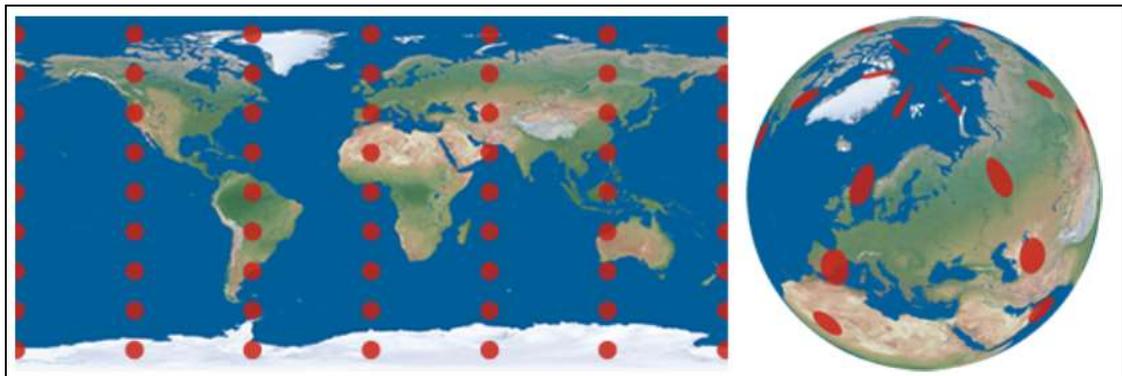


Abbildung 38 - Punktsignaturen ohne Berücksichtigung von Verzerrungseigenschaften, Quelle: [LIE-12], S. 2

Als Vorteil hat sich herausgestellt, dass Legenelemente oder andere Symbole zwischen 25° nördlicher und 25° südlicher Breite angebracht werden sollten, da hier keine Korrekturen notwendig sind, da eine Veränderung nicht sichtbar wäre.

Um zum Beispiel kreisförmige Signaturen der Verzerrung entsprechend korrekt einzuzeichnen, gibt es zwei Möglichkeiten:

Die erste Möglichkeit ist es Punktsignaturen mittels Adobe Photoshop und dem Werkzeug Ellipse Tool zu erstellen. Um die Signaturen lagerichtig zu positionieren, muss die

Ortsangabe von geographischen Koordinaten in Pixelkoordinaten (4096x2048 Pixel) mittels dieser Formeln umgerechnet werden:

$$X = \frac{\text{Breite (Plattkarte)} * (180 + \text{Longitude})}{360}$$

$$Y = \frac{\text{Breite (Plattkarte)}}{2} - \frac{\frac{\text{Breite (Plattkarte)}}{2} * (90 + \text{Latitude})}{180}$$

Nehmen wir folgende Koordinaten (57,5° N, 10,3° E) für den Austrian Airlines Flug (OS81) von Wien nach Los Angeles an. Die Pixelkoordinaten würden nach der Berechnung folgende Werte betragen: X = 1082,6 und für Y = 184,8. Als letzter Schritt muss noch mit der Formel der Verzerrung die Korrektur der Länge des Breitenkreises berechnet werden. In unserem Fall wäre dies eine Verzerrung der Signatur die annähernd doppelt (1,86 = 186 %) so breit wie hoch sein müsste. Anzumerken ist, dass diese Methode nur die Problematik erläutern soll, wie dies manuell möglich ist; es sollte auf eine automatisierte Methode zurückgegriffen werden.

Die zweite Möglichkeit besteht darin, mittels Geoinformationsprogrammen, zum Beispiel ESRI ArcGIS, die entsprechenden Punktsignaturen darzustellen. Dabei wird, ausgehend von Punktdaten, welche Städte oder andere Ereignisse darstellen können, ein geodätischer Buffer erstellt, der die Verzerrungseigenschaften einer Plattkarte innehat.

Beim Freeware Produkt QGIS wurde von Herrn Ervin Wirth und Herrn Peter Kun, beide vom Institut für Geodäsie, Kartographie und Fernerkundung von der Technischen und Wirtschaftswissenschaftlichen Universität Budapest ein Plug-In mit dem Namen „Indicatrix Mapper“ entwickelt. Es werden keine Kreise, sondern das Sphäroid zur Berechnung verwendet. Außerdem wird eine andere Formel zur Berechnung bei diesem Werkzeug verwendet, nämlich die Vincenty-Formel von Thaddeus Vincenty (1920-2002)². Die Anpassung der Kreise an die verwendete Projektion wird „on the fly“ projiziert. Der Unterschied zur Tissot’schen Indikatrix ist jener, dass es sich hier um keine reine mathematische Methode handelt, sondern um eine direkte Umrechnung von fixen Radien für die Kreise, bei unterschiedlichen Projektionen, welche sofort sichtbar sind³. Abbildung 39 zeigt das Menü und die möglichen Einstellungen des Indicatrix Mapper Werkzeugs.

² polnisch-amerikanischer Geodät, Mitbegründer des NAD-83 (North American Datum)

³ Quelle: http://geomatica.com.polimi.it/workbooks/n12/FOSS4G-eu15_submission_214.pdf, abgerufen am 25.09.2017

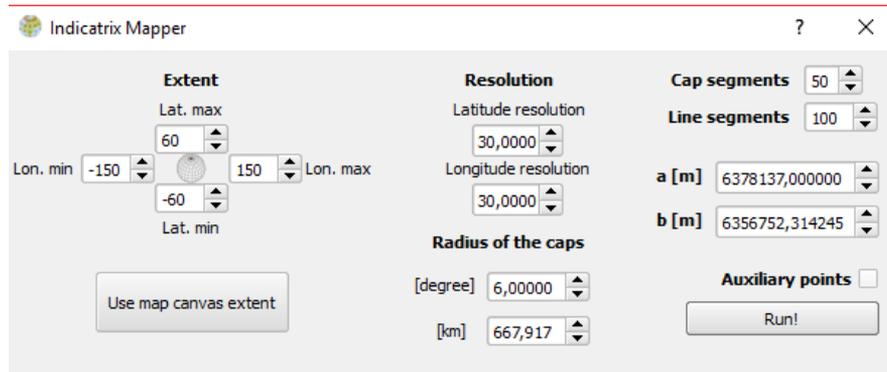


Abbildung 39 - Kontextmenü des Plug-In, Quelle: Eigene Darstellung

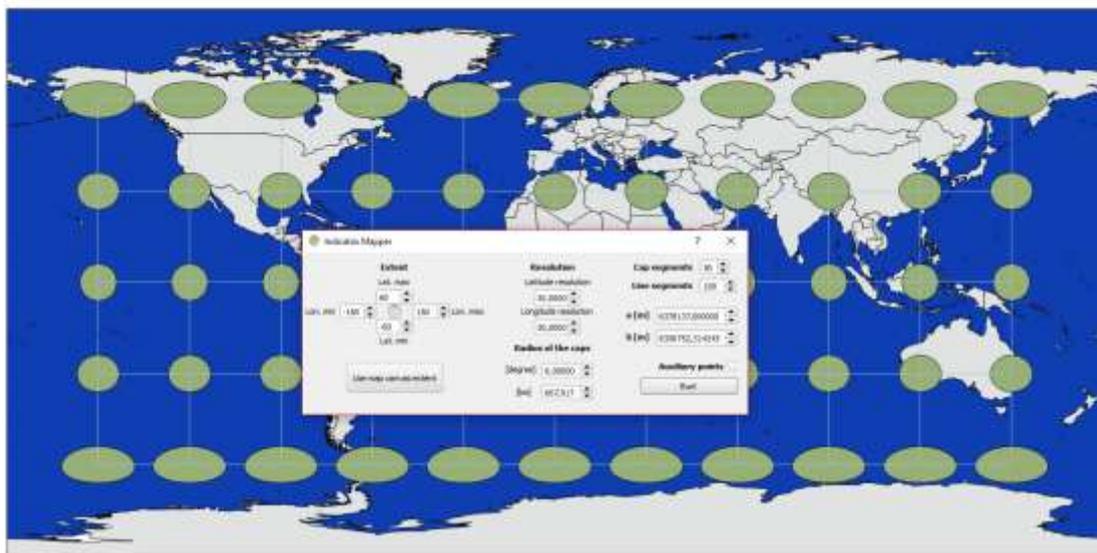


Abbildung 40 - QGIS-Plug In, Quelle: Eigene Darstellung

Der Vorteil dieser Variante ist es, dass sie automatisch sehen, wie die kreisförmigen Signaturen bei den unterschiedlichen Projektionen entzerrt werden müssen (siehe Abb. 40). Nachteil ist, dass dieses Plug-In nicht auf selbst erstellte Punktdaten angewendet werden kann, um die Daten richtig in einer Plattkarte im zwei-dimensionalen Raum zu entzerren und korrekt am Globus anzuzeigen.

5.4 Orthodrome und Loxodrome

„Die Orthodrome stellt die kürzeste Verbindung zweier Punkte der Erdoberfläche dar und ist daher für das Verkehrswesen von grundlegender Wichtigkeit. Ihre Bedeutung wächst mit zunehmender Größe der Entfernungen. Obwohl sie auf dem Globus dann leicht aufzufinden ist, wenn man durch diese beiden Punkte den Großkreis legt, ist ihre Darstellung in Karten von dem verwendeten Netzentwurf abhängig. Um den kürzesten Weg zwischen zwei Punkten ausfindig zu machen, brauchen wir deshalb ein Netz, das die Orthodrome als gerade Linie abbildet. Verbinden wir nämlich in einem solchermaßen ausgezeichneten Netz zwei Punkte durch eine Gerade, so ist damit der kürzeste Weg festgelegt und die Großkreisbestimmung in der Ebene erreicht.“ [KRE-70, S. 162]

Abbildung 46 zeigt die Flugstrecke von Wien nach Los Angeles. Auffallend ist, dass die Orthodrome, dies ist die kreisförmige Linie in der Darstellung, die kürzeste Verbindung zwischen den beiden Städten ist, die Gerade hingegen, die Loxodrome, um einige Tausend Kilometer länger ist. Deshalb fliegen Luftfahrzeuge immer entlang der Großkreise, um so die zu fliegende Distanz beziehungsweise die Flugzeit geringer zu halten. Ein weiterer Grund ist der wirtschaftliche Faktor, welcher hier auch mit hineinspielt, da Treibstoffmengen und -kosten reduziert und somit eingespart werden können.

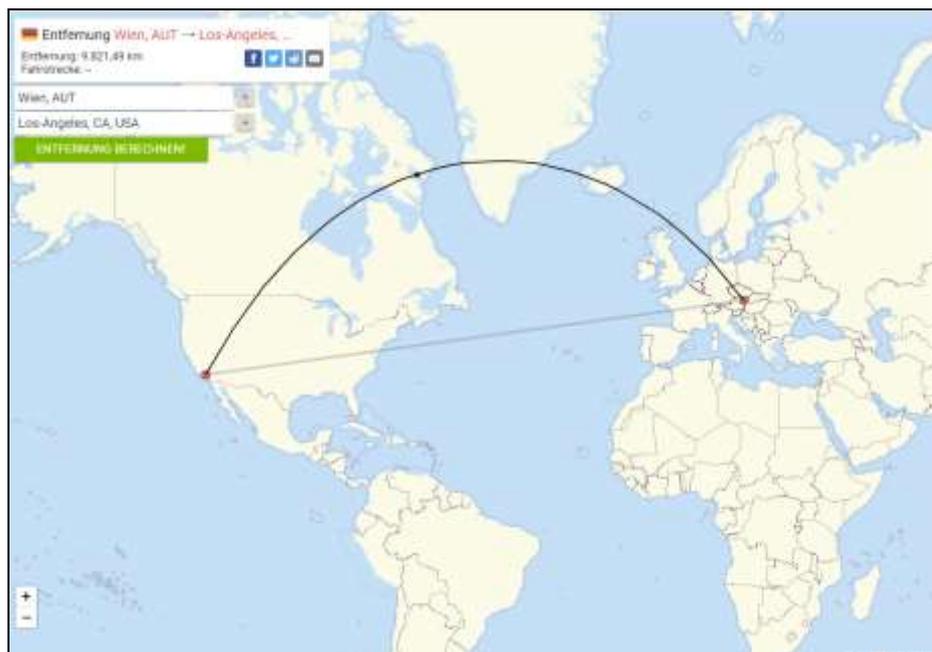


Abbildung 41 - Orthodrome – Flugstrecke Wien-Los Angeles, Quelle: <https://www.luftlinie.org>

Wenn man sich die Erde als dreidimensionale Kugel ansieht, so merkt man, dass die in Abbildung 42 violette Linie eigentlich eine Gerade ist und die orangene Linie die gekrümmte Linie ist, das heißt, dass die Projektion ausschlaggebend ist, wie die Linien dargestellt werden. Nur in der gnomonischen Abbildung wird die Orthodrome als Gerade abgebildet (siehe Abb. 42).

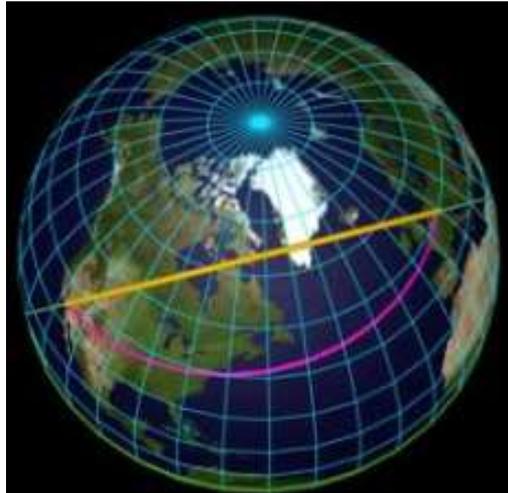


Abbildung 42 - Abbildung der Orthodromen auf der Erdkugel, Quelle: <http://kartenkunde-leichtgemacht.de>

Abbildung 43 zeigt die Berechnung der Distanz der Orthodromen in einem Python-Skript zwischen zwei Städten in Kilometer.

```
def orthodrome (p1 ,p2) :

    phi1      =p1 [0]
    lambda1   =p1 [1]
    phi2      =p2 [0]
    lambda2   =p2 [1]
    gamma     =radians (lambda2-lambda1)
    a         =radians (90-phi1)
    b         =radians (90-phi2)
    cosc=cos (a)*cos (b)+sin (a)*sin (b)*cos (gamma)
    c         =degrees (acos (cosc))
    orthodrome = 6370000.0*pi*c/180.0
    return orthodrome

from math import *
p1 = input("phi1")
l1 = input("lambda1")
p2 = input("phi2")
l2 = input("lambda2")

A=(p1,l1)
B=(p2,l2)

print "Die kuerzeste Entfernung betraegt", orthodrome (A,B)/1000.0, "
Kilometer"
```

Abbildung 43 - Berechnung Orthodrome, Quelle: [KAI-12]

Zur Vollständigkeit wird in diesem Kapitel auch auf die Loxodrome kurz eingegangen.

„Das Pendant zur Orthodromen ist die Loxodrome, jene schiefsläufige Linie, die die Meridiane immer unter dem gleichen Winkel schneidet, verdankt ihre Bedeutung vor allem dem Bestreben in der Seefahrt, den einmal bestimmten Kurs möglichst lange beizubehalten. Somit wird klar, da[ss] nur jener Netzentwurf [Mercatorprojektion], der die Loxodrome als gerade Linie abbildet, für die Navigation auf See ausgezeichnete Bedeutung besitzt, da aus ihm der Kurs unmittelbar abgelesen werden kann. Da allerdings bei der Bewegung über größere Entfernungen der Unterschied zwischen Orthodrome und Loxodrome immer stärker auseinanderklafft, wird der Großkreis in der Kartenebene des entsprechenden Netzes aufgesucht und dieser durch Loxodromenstückchen approximiert.“ [KRE-70, S. 162]

Abbildung 44 zeigt die spiralförmige Linie vom Äquator zum Pol.

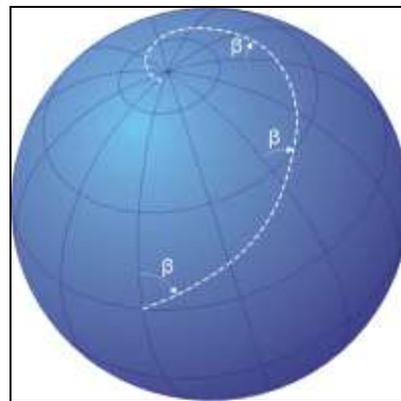


Abbildung 44 - Loxodrome-Skizze, Quelle: <https://upload.wikimedia.org/>

Abbildung 45 wird die dazugehörige Berechnungsmethode des Winkels einer Loxodrome dargestellt.

$$\begin{array}{l}
 P_1(\varphi_1, \lambda_1) \\
 P_2(\varphi_2, \lambda_2) \\
 \lambda_2 - \lambda_1 = \tan \alpha \left[\operatorname{Intan} \left(\frac{\pi + \varphi_2}{4} \right) - \operatorname{Intan} \left(\frac{\pi + \varphi_1}{4} \right) \right] \\
 \text{Abstand } s \text{ zwischen } P_1 \text{ und } P_2 \\
 s = \frac{R}{\cos \alpha} (\varphi_2 - \varphi_1) \\
 \text{Winkel in Radiant}
 \end{array}$$

Abbildung 45 - Loxodrome-Berechnung, Quelle: [KAI-12]

Die Orthodrome ist aber jene Linie, welche im praktischen Teil (siehe Kap. 8.2) weiterverwendet wird, um neue Optimierungsvarianten vorzustellen.

6 Datengewinnung vom Flug Real Time-Daten

Dieses Kapitel gibt einen Überblick über das Live-Tracking der Seite FlightRadar24. Näher erläutert werden sollen unter anderem Automatic Dependent Surveillance – Broadcast (ADS-B), Multilateration (MLAT) und Radar Daten, um die Position der diversen Luftfahrzeuge abzufragen und diese für die User über deren Homepage und App für Android und iOS zur Verfügung zu stellen. Des Weiteren wird kurz auf deren Entstehungsgeschichte und die mögliche Datengewinnung für die Visualisierung von Live-Daten am Hyperglobus eingegangen.

6.1 Flightradar24

6.1.1 Allgemein

Im Jahr 2006 entwickelten zwei schwedische Luftfahrtenthusiasten Flightradar24 und bauten ein Netzwerk mit ADS-B (Automatic Dependent Surveillance – Broadcast) Empfängern in Europa auf. Zwei Jahre später wurde das gesamte System für die Öffentlichkeit frei zugänglich. Damit wurde es möglich, derartige Sendeeinrichtungen zu Hause aufzustellen, um die weltweite Netzabdeckung zu verstärken (vgl. [FLI-16a]).

6.1.2 Funktionsweise

Wie schon in der Einleitung erwähnt, bedient sich Flightradar24 drei unterschiedlicher Bezugsquellen für die Datengewinnung, die im Folgenden näher erörtert werden.

6.1.2.1 ADS-B

Die erste Art der Datengewinnung sind ADS-B Sender und Empfänger Einrichtungen, welche den Luftverkehr noch sicherer machen und eine noch genauere Vorhersage gewährleisten, auf welchem Punkt der Erde sich das Luftfahrzeug gerade befindet. Vor allem in Gebieten, die bislang mittels Radar nicht abgedeckt waren, wie zum Beispiel die Hudson

Bay in Kanada, ist dies sehr wichtig. Abbildung 46 zeigt die Funktionsweise von ADS-B. Dabei kann festgestellt werden, dass dieses System drei Komponenten (Globale Navigationssatellitentechnik [EGNOS und GLONASS], Sende- und Empfangseinrichtungen) verwendet, um die Daten der Luftfahrzeuge zu erhalten. Vom Luftfahrzeug wird ein Signal (1090 MHz) zu den Bodenstationen oder zu den Luftfahrzeugen, die sich in der Nähe befinden, gesendet. In diesem Signal befinden sich Informationen zur aktuellen Position, Flughöhe, Fluggeschwindigkeit, Flugnummer sowie zum Abflug- beziehungsweise Ankunftsflughafen. Flugsicherung und Pilot bekommen dieselben Daten für den gerade durchgeführten Flug. Dadurch erhöht sich nicht nur die Sicherheit im Luftverkehr, sondern es erlaubt auch das direktere Routing in den diversen Lufträumen der Welt. Aus diesem Grund sind in Europa von Seiten der Europäischen Kommission, der Eurocontrol und diversen nationalen Luftfahrbehörden Initiativen wie zum Beispiel Single European Sky oder Single European Sky ATM Research Programm entstanden, mit dem Ziel einer größtmöglichen Optimierung sowie Vereinheitlichung und Harmonisierung des Luftverkehrsmanagements (vgl. [RIC-10]).

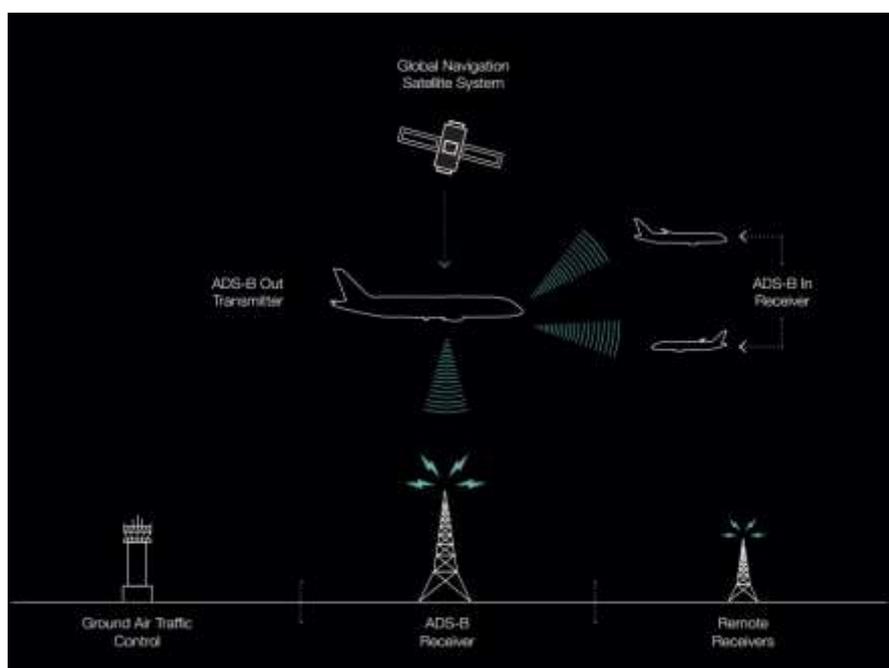


Abbildung 46 - Übersicht der Funktion von ADS-B Einrichtungen, Quelle: [RIC-10]

In der Zukunft soll ADS-B das herkömmliche Radarsignal zur Überwachung in der Luftfahrt ablösen. Flightradar24 hat derzeit über 10.000 ADS-B Einrichtungen auf der ganzen Welt aufgebaut. Damit wurde es möglich, eine bestmögliche Abdeckung (vgl. Abb. 47) zu gewährleisten. Die gelben Flieger sind Real-Time Darstellungen und die orangenen Flieger, 5 Minuten verzögert, sind vor allem im nordamerikanischen Luftraum zu finden. Diese Daten

werden über die nordamerikanische Luftfahrtbehörde (Federal Aviation Administration – FAA) abgewickelt. Natürlich werden in naher Zukunft immer mehr dieser Einrichtungen benötigt, da man die ganze Welt abdecken möchte. Ein Problem stellen jedoch die Ozeane dar, da diese Empfangseinrichtungen nur eine maximale Reichweite von 250 bis 450 km in alle Richtungen haben und die Aufstellung im Wasser sich als sehr kompliziert darstellt. Warum gibt es dann trotzdem Positionsdarstellungen über den Ozeanen, obwohl hier keine Antennen aufgestellt werden können oder die Reichweite der letzten Station am Festland nicht ausreicht, da vor allem die nordamerikanische Luftfahrtbehörde Daten aus den Luftfahrzeugen bekommt und diese Daten verwendet und unter anderem auch Flightradar24 zur Verfügung stellt. Meiner Meinung handelt es sich hier bei den Daten um soft-real-time Daten (vgl. Kap. 3.3.1), da es hier immer wieder durch zu wenig Abdeckung durch Antennen von Seiten von FlightRadar24 zu Ausfällen kommen kann, aber dies keine gravierenden Auswirkungen hat und somit das Nutzen abnimmt. Würde es sich um ein von der Flugsicherung abhängiges System handeln, würde ich der Meinung sein, dass es sich hier um ein hard-real-time System (vgl. Kap. 3.3.2) handelt, da bei Verschwinden eines Luftfahrzeuges angenommen werden muss, dass das Luftfahrzeug abgestürzt sei und somit das Nutzen schlagartig abnimmt und ein großer Schaden entsteht.



Abbildung 47 - Übersichtskarte der Abdeckung von Flightradar24, Quelle: [FLI-16b]

6.1.2.2 MLAT

Multilateration (MLAT) ist die zweite Methode, welche von Flightradar24 verwendet wird, um die Positionen der Luftfahrzeuge anzuzeigen zu können. Diese Art der Positionsermittlung wird hauptsächlich in Gebieten eingesetzt, wo das normale Radar aufgrund der Topographie zu viel Abschattung hat und somit keine richtigen Daten liefern kann. In Österreich wird dies zum Beispiel in Innsbruck verwendet. MLAT misst mittels ausgesendeter Daten des Transponders im Luftfahrzeug die Laufzeitunterschiede zwischen mehreren Bodenstationen und berechnet so den aktuellen Standort und die Geschwindigkeit des Luftfahrzeugs (siehe Abb. 48) (vgl. [AUS-16]).



Abbildung 48 - Funktion von MLAT, Quelle: [AUS-16]

6.1.2.3 Radar

Bei Radarsystemen unterscheidet man zwischen Primär- und Sekundärradaranlagen. Bei den klassischen Radaranlagen (Primärradar) sendet die Anlage einen Impuls aus, welcher vom Luftfahrzeug reflektiert wird. Dessen Position wird über die Laufzeit des reflektierten Impulses berechnet. Das Sekundärradar sendet wie das Primärradar einen Impuls aus, welcher am Luftfahrzeug reflektiert. Zusätzlich wartet diese Art von Radar auf einen Impuls von Seiten des Luftfahrzeuges, genauer gesagt vom Transponder, welcher Informationen zur Identität und Flughöhe mitsendet. In Österreich gibt es insgesamt drei Stationen: Buschberg, Feichtberg und Koralpe (vgl. [AUS-16], siehe Abb. 49).

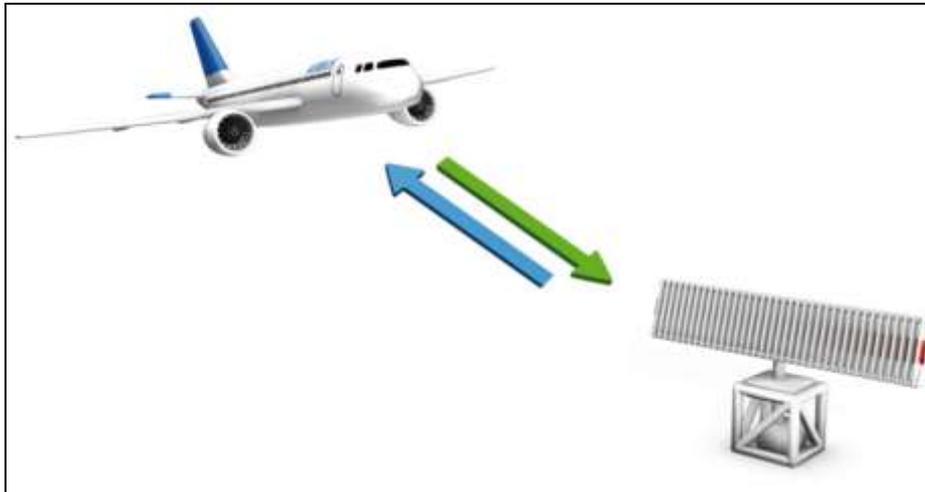


Abbildung 49 - Übersicht der Funktion von Sekundärradar, Quelle: [AUS-16]

6.1.3 Flightradar24 - Mitgliedschaft

Bei Flightradar24 gibt es verschiedene Arten der Mitgliedschaft (vgl. Abb. 50). Entweder Basic, Silver, Gold oder Business. Ab Silver wird dies kostenpflichtig mit \$9,99 pro Jahr.

Basic	Silver	✓ Gold	Business
<ul style="list-style-type: none"> ✓ Live flight tracking ✗ Ads removed ✗ Extra map labels ✗ Alerts ✗ Full aircraft details ✗ Aeronautical charts ✗ Weather layers ✗ Additional flight history 	<p>Everything in Basic</p> <ul style="list-style-type: none"> ✓ Ads removed ✓ Alerts ✓ 60 days of past flights ✓ Text labels ✓ More aircraft details 	<p>Everything in Silver</p> <ul style="list-style-type: none"> ✓ Aeronautical charts ✓ Aviation weather layers ✓ More flight details ✓ ATC boundaries ✓ 180 days of past flights 	<p>Everything in Gold</p> <ul style="list-style-type: none"> ✓ Licensed for businesses ✓ High-level significant weather layer ✓ Lightning layer ✓ AIRMETS/SIGMETs layer ✓ Airport view ✓ Fleet view
	<p>See all Silver features</p> <p>\$1.49 /Month*</p> <p>or</p> <p>\$9.99 /Year*</p> <p>Downgrade to Silver</p>	<p>See all Gold features</p> <p>\$3.99 /Month*</p> <p>or</p> <p>\$34.99 /Year*</p> <p>Current plan Make changes</p>	<p>See all Business features</p> <p>\$49.99 /Month*</p> <p>or</p> <p>\$499.99 /Year*</p> <p>Upgrade to Business</p>

Abbildung 50 - Übersicht der Mitgliedschaftsformen, Quelle: [FLI-16c]

Durch die unterschiedlichen Mitgliedschaftsformen bekommt der User/die Userin auch verschiedene Funktionen zur Verfügung gestellt. Abbildung 51 und 52 geben einen Überblick über diese Funktionen. Ab der Mitgliedschaft Gold bekommt der Anwender/die Anwenderin die Möglichkeit auf den Zugriff von Streckenkarten für den unteren und oberen Luftraum. Im österreichischen Luftraum liegt die Grenze der Aufteilung bei Flughöhe 24.500 Fuß (ca. 7.500 Meter). Eine weitere gute Darstellungsmethode ist die Anzeige der Segmentierung der

einzelnen Luftsicherungsbehörden, um zu sehen, welches Land für das Luftfahrzeug zuständig ist. Außerdem werden über den beiden großen Ozeane (Atlantik und Pazifik) zweimal täglich die so genannten Oceanic Tracks aktualisiert, welche eine Sicherheit für die Überquerung der Ozeane sicherstellen, da während des Überfluges kein ständiger Funkkontakt zur Flugsicherung möglich ist. Ein weiteres Feature ab der Gold Mitgliedschaft ist die Darstellung von Wetterereignissen mit der Wolkenbedeckung und Niederschlagsvorhersage für die nächsten 12 Stunden. Eine noch detailliertere Anzeige von Winddaten und Vorkommnissen von Blitzschlag bietet erst die Business Version. Die Details zum ausgewählten Flug sind im Gegensatz zur frei nutzbaren Version auch genauer, da die genaue GPS-Höhe und die tatsächliche Fluggeschwindigkeit, passend zu Rücken- oder Gegenwind, angezeigt wird. Außerdem gibt es Informationen zur aktuellen Außentemperatur im Luftraum, in dem sich das Luftfahrzeug derzeit befindet (vgl. [FLI-16c]).

The image shows a screenshot of the Flightradar24 subscription page. At the top, there are four subscription plans: Basic, Silver, Gold, and Business. Below the plans is a table comparing various features across the different tiers. The Gold plan is currently selected.

	Basic	Silver	Gold	Business
<p>You can use your Flightradar24 subscription on any computer, phone, or tablet.</p> <p> Feature is available in iOS and Android apps.</p>				
		\$1.49 /Month*	\$3.99 /Month*	\$49.99 /Month*
		\$9.99 /Year*	\$34.99 /Year*	\$499.99 /Year*
		Downgrade to Silver	Current plan Make changes	Upgrade to Business
General				
Ads removed	+	✓	✓	✓
Prioritized customer support	+	Yes	Yes	Yes
Sessions	+	2	3	3
Types of usage allowed	Private	Private	Private	Private & business
Timeout removed	+	✓	✓	✓
Aircraft on map				
Flight status	✓	✓	✓	✓
Live flight tracking	✓	✓	✓	✓
Aircraft details				
Registration	✓	✓	✓	✓
Mode-S Code	✓	✓	✓	✓
Aircraft type	✓	✓	✓	✓
Aircraft serial number	+	✓	✓	✓
Aircraft age	+	✓	✓	✓

Abbildung 51 - Übersicht der Funktionen I, Quelle: [FLI-16c]

Flight details					
Ground speed	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Position	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Track	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Calibrated altitude	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Vertical speed	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Squawk code	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
GPS altitude	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
True airspeed	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Wind	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Indicated airspeed	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Temperature	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
FIR/UIR	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Mach	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Past flights					
Flight history	<input type="checkbox"/>	7 days	60 days	180 days	365 days
Aircraft history	<input type="checkbox"/>	7 days	60 days	180 days	365 days
Playback	<input type="checkbox"/>	7 days	60 days	180 days	365 days
Aircraft on ground	<input type="checkbox"/>	Last 60 minutes	Last 7 days	Last 30 days	Last 90 days
Download CSV/KML files	<input type="checkbox"/>	<input checked="" type="checkbox"/>	10 per month	100 per month	1,000 per month
Weather Layers					
Volcanic eruption	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Current weather	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Cloud	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Total precipitation	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Intense precipitation	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Winds	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
AIRMETS/SIGMETs	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
High level significant weather	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Lightning	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Map Layers					
ATC boundaries	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Oceanic tracks	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Aeronautical Charts	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Map Views					
Delay view	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Multi-select view	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Radar view	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Full-screen view	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
List view	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Airport view	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Fleet view	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

Abbildung 52 - Übersicht der Funktionen II, Quelle: [FLI-16c]

Der Verfasser dieser Arbeit besitzt die Mitgliedschaft „Gold“, aber nicht ausschließlich wegen der Masterarbeit, sondern auch aufgrund des Eigeninteresses an der zivilen Luftfahrt und der Arbeit bei der Flughafen Wien AG.

Ab der Mitgliedschaft Silver ist es möglich, CSV und/oder KML-Dateien über vergangene Flüge (vgl. Abb. 53), aber auch Flüge zu downloaden, die sich gerade in der Luft befinden. Die bereitgestellten Daten aktualisieren sich sekundlich automatisch und werden bei erneutem Download aktualisiert, dadurch befinden sich wieder mehr Wegpunkte in den Dateien. Bei der Mitgliedschaft Gold ist der einzige Nachteil die Beschränkung der Upload-Funktion auf 100 pro Monat. Dies reicht aber völlig aus um die Machtbarkeit aufzuzeigen.

25 Oct	Vienna (VIE)	New York (GRO)	B772 (DF-LPE)	-	10:45 AM	10:58 AM	2:15 PM	Estimated 2:03 PM	KML	CSV	Play
24 Oct	Vienna (VIE)	New York (GRO)	B763 (DF-LAX)	9:17	10:45 AM	11:29 AM	2:15 PM	Landed 2:46 PM	KML	CSV	Play
23 Oct	Vienna (VIE)	New York (GRO)	B763 (DF-LAX)	9:17	10:45 AM	11:02 AM	2:15 PM	Landed 2:20 PM	KML	CSV	Play
22 Oct	Vienna (VIE)	New York (GRO)	B763 (DF-LAX)	9:03	10:45 AM	11:59 AM	2:15 PM	Landed 3:03 PM	KML	CSV	Play
21 Oct	Vienna (VIE)	New York (GRO)	B763 (DF-LAX)	9:05	10:45 AM	11:05 AM	2:15 PM	Landed 2:10 PM	KML	CSV	Play
20 Oct	Vienna (VIE)	New York (GRO)	B763 (DF-LAX)	8:59	10:45 AM	11:03 AM	2:15 PM	Landed 2:02 PM	KML	CSV	Play
19 Oct	Vienna (VIE)	New York (GRO)	B763 (DF-LAX)	8:56	10:45 AM	11:24 AM	2:15 PM	Landed 2:20 PM	KML	CSV	Play
18 Oct	Vienna (VIE)	New York (GRO)	B763 (DF-LAX)	8:52	10:45 AM	10:58 AM	2:15 PM	Landed 1:50 PM	KML	CSV	Play
17 Oct	Vienna (VIE)	New York (GRO)	B763 (DF-LAX)	9:13	10:45 AM	11:04 AM	2:15 PM	Landed 2:17 PM	KML	CSV	Play
16 Oct	Vienna (VIE)	New York (GRO)	B763 (DF-LAX)	8:33	10:45 AM	12:08 PM	2:15 PM	Landed 2:42 PM	KML	CSV	Play

Abbildung 53 - Übersicht der Flüge und Upload-Funktion, Quelle: [FLI-16d]

7 Die herkömmliche Methode der Generierung von Vektordaten

Am Anfang dieses Kapitels wird die Globensoftware OmniSuite näher vorgestellt und die Grundstruktur erklärt. Dadurch, dass sich diese Masterarbeit schwerpunktmäßig mit der Optimierung der Vektordatengenerierung für den Hyperglobus beschäftigt, folgt in diesem Kapitel zuerst ein Überblick die beiden aktuellen Methoden (intern vs. extern) für die Generierung von Vektordaten und deren Weiterverarbeitung mittels OmniSuite. Anschließend sollen aufbauend neue Methoden geschaffen werden beziehungsweise die bestehenden Methoden teilweise automatisiert werden. Es wird mit unterschiedlichen Skriptsprachen, unter anderem Python und PHP die Daten von FlightRadar 24 heruntergeladen und verarbeitet, um Real Time-Flugdaten am Hyperglobus anzuzeigen.

7.1 OmniSuite Allgemein

OmniSuite ist ein Softwarepaket, welches im Auftrag der Firma Globocess von der HRG (Hyperglobe Research Group) unter der Projektleitung von Andreas Riedl als plattformunabhängigen Authoring- und Abspielplattform entwickelt und von Herrn Jürgen Kristen programmtechnisch umgesetzt und regelmäßig weiterentwickelt wird. Die Software ist dafür verantwortlich, dass die Materialien, die Projektionen sowie die Wiedergabe, welche vom Nutzer eingegeben werden, richtig und in Echtzeit am Globus wiedergegeben werden können. Die Software selber teilt sich in zwei Elemente auf, auf der einen Seite ist OmniSuite eine Autorenumgebung und auf der anderen Seite eine Präsentationsplattform für einen taktilen Hyperglobus (siehe Abb. 54). Die in Kapitel 2.5. erläuterten Themenbereiche werden über so genannte „global stories“ mittels OmniSuite erstellt und geben so eine komplexe globale Thematik am Hyperglobus wieder. OmniSuite wurde in der Programmiersprache C++ entwickelt und verwendet Ogre3D als Grafik-Engine (vgl. [GLO-17], S. 7).

Folgende Module stehen nach Riedl [RIE-11b], S. 6-8 zur Verfügung (vgl. Abb. 54):

- **Controller:** „[...] Schnittstelle zwischen Anwender und taktilen Hyperglobus.“
- **Material Editor:** *Materialien bilden die Basis einer „Global Story“. Ein Material definiert Globenbild(er) (statisch oder animiert) und ordnet diesem entsprechende Layer zu [...].*

- **Story Editor:** „[Es] werden die eigentlichen Stories entsprechend eines Storybooks erstellt. So lassen sich zuvor erstellte Materialien via Zeitleiste steuern, [...].“
- **Render Engine:** „Die 3D-Engine ist das eigentliche Herz von OmniSuite und wird vom Controller gesteuert. Sie ist für die Projektion des Globenbildes verantwortlich. Ausgehend von einer quadratischen Platte, generiert sie eine vermittelnde Azimutalabbildung und garantiert so die geometrisch korrekte Wiedergabe am sphärischen Display. Die 3D-Engine rendert [...] auch Echtzeit-Inhalte (z. B. aktuelle Wolkenanimation) [...].“
- **Catalog:** „[...] dient im Wesentlichen zur Verwaltung der Story Library (Hinzufügen und Löschen von Stories) und zum Storytransfer zwischen Rechnern z. B. vom Authoring-Rechner zum Globus-Rechner.“
- **Preferences:** „Dieses Modul dient zur Konfiguration sowohl der Hardware als auch der Software. Die Optionen betreffen Bereiche wie Globustyp [...], Displayeinstellungen [...].“

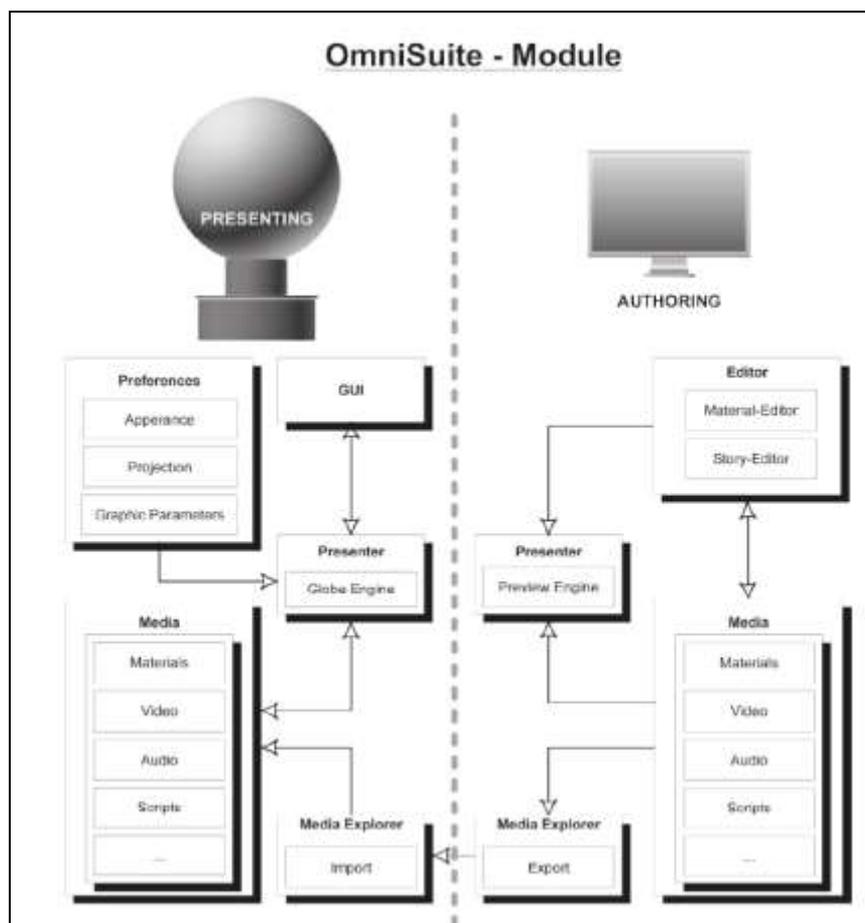


Abbildung 54 - Plattform OmniSuite, Quelle: [RIE-08], S. 14

7.2 Datenintegration über Rasterdaten

Bei der Erstellung einer Global Story dient bei den meisten Globensystemen die quadratische Plattkarte als Basismedium, da diese einfach für die Darstellung und Verarbeitung ist. Diese quadratische Plattkarte ist rasterbasiert und besteht im Idealfall, aufgrund der aktuellen technischen Beschränkungen der Projektoren, aus 4096 x 2048 Pixel und dient als Informationsträger. Abbildung 55 zeigt den Verlauf beziehungsweise die Bearbeitungsschritte der Software von der Eingabe (a) über die Verarbeitung (b, c) bis zur Ausgabe (d) am sphärischen Display. Nach der Eingabe des Basismediums wird die Darstellung der verwendeten Projektion berechnet. Wenn dies abgeschlossen ist, wird das Globenbild vom Beamer ausgestrahlt und über einen Spiegel auf die Außenhülle des Globenkörpers abgelenkt und dadurch für den Nutzer sichtbar (vgl. [LIE-12], S. 8-11).

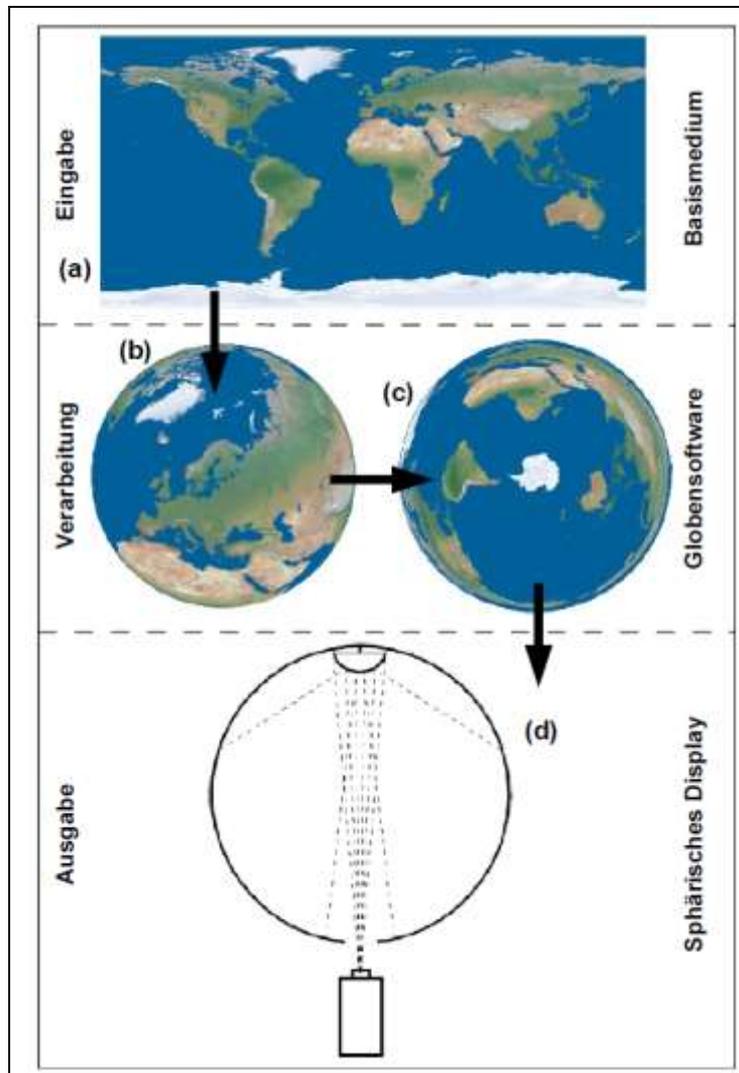


Abbildung 55 - Funktionsweise taktile Hyperglobus, Quelle: [LIE-12], S. 9

Am einfachsten kann eine quadratische Plattkarte mit Hilfe einer GIS-Software, wie zum Beispiel ArcGIS Desktop, erstellt werden. Hierzu werden freie Ländergrenzen der Welt und eventuelle Daten der Meeresgrenzen von der Webseite <http://www.naturearthdata.com/> heruntergeladen. Im ArcGIS Desktop werden diverse Größeneinstellungen beim Seitenlayout getroffen. Es gilt hier zu beachten, dass das Seitenverhältnis von 2:1 immer erhalten bleiben muss, da es sonst zu Verzerrungen kommen kann. Des Weiteren wird der Datenrahmen der Karte auf die Projektion der Plattkarte eingestellt und anschließend kann die Karte exportiert werden.

7.3 Datenintegration über Raster- und Vektordaten

Global Stories haben mehr Informationsgehalt, wenn zusätzlich die Rasterdaten mit Vektordaten verknüpft werden. Johannes Liem [*LIE-12*] unterscheidet hier zwischen *interner und externer Verschneidung*.

7.3.1 Raster- und Vektordaten extern verschneiden

Die derzeitige Bearbeitung von Raster- und Vektordaten läuft wie folgt ab (siehe Abb. 56):

Den Ausgangspunkt stellen auf der einen Seite die darzustellenden Vektordaten in Form eines Textfiles mit Koordinaten dar oder sie werden selber auf einer weiteren Ebene eingezeichnet. Auf der anderen Seite die Rasterkarte. In diesem Fall eine quadratische Plattkarte der Erde. Um die Vektordaten lagerichtig auf der Plattkarte zu integrieren, wird der Schritt meist über Photoshop durchgeführt, indem beide Ebenen überlagert und miteinander verschnitten werden. Anschließend wird das entstehende Bild abgespeichert und kann mittels OmniSuite eingespielt und am Globus dargestellt werden (vgl. [*LIE-12*], S. 46-47).

Möchte man Animation erstellen, müssen im Photoshop einzelne Ebenen der Vektordaten und die quadratische Plattkarte als Hintergrundebene erstellt werden. Anschließend werden mittels einer Automatisierungsfunktion in Photoshop (Actions – Batch Processing) einzelne Rasterbilder rausgespeichert, sodass eine zeitbasierte Darstellung in OmniSuite möglich ist.

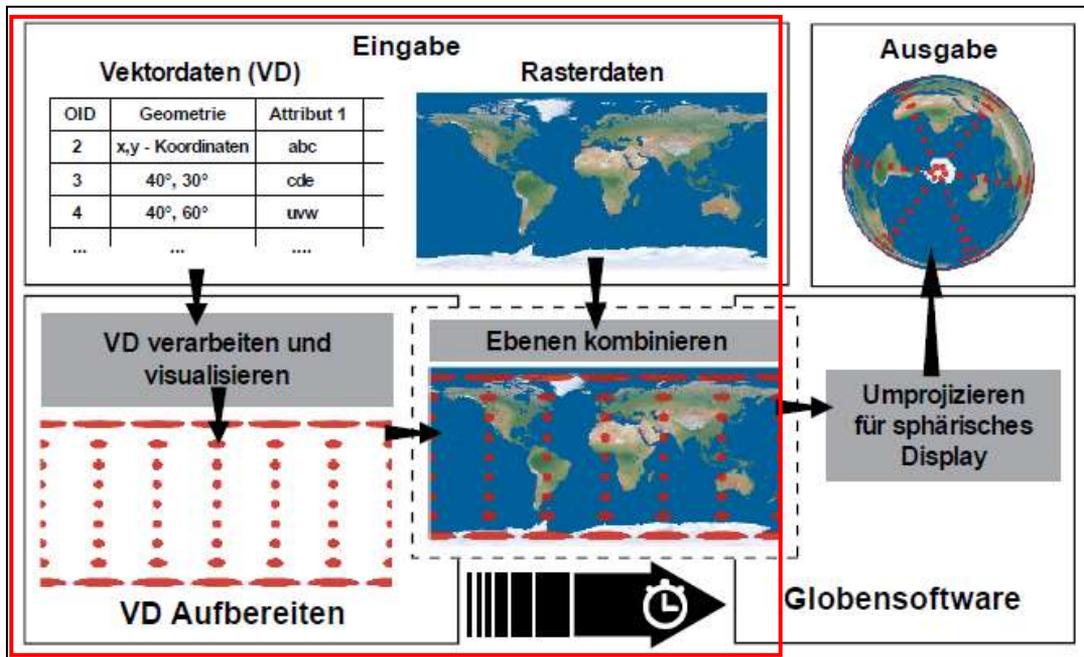


Abbildung 56 - Schema der externen Vektordatenintegration, Quelle: [LIE-12], S. 47

Im Rahmen dieser Arbeit wurde für die Optimierung und Verbesserungen der externen Visualisierung aus Abbildung 56 ein Python-Skript erstellt. Der rote Rahmen kennzeichnet jenen Bereich der mittels dem Python-Skript automatisiert werden soll. Abbildung 57 zeigt das Schema der Optimierung und den groben Ablauf der Schritte bis zur Ausgabe am taktilem Hyperglobus. Der Vorteil der Optimierungsvariante ist, dass die Verschneidung zwischen Vektordaten und Rasterdaten automatisch mittels Python-Skripts abläuft, das heißt der Nutzer/die Nutzerin braucht nur das Python-Skript starten und eine lokale Webseite aufrufen, wo eine Flugnummer eingegeben werden muss und das fertige Textfile mit den aktuellen Koordinaten des Luftfahrzeuges wird gespeichert. Durch Aufrufen eines weiteren Python-Skripts mit dem Namen „*schleife.py*“ wird über ein Layer-File, ein PNG-Bild (Hintergrund und die Vektordaten als punktförmige Signatur) automatisch in den Zielordner gespeichert und muss nur mehr in die Software OmniSuite integriert werden. Den genauen und detaillierten Ablauf erfahren Sie auf den nachfolgenden Seiten.

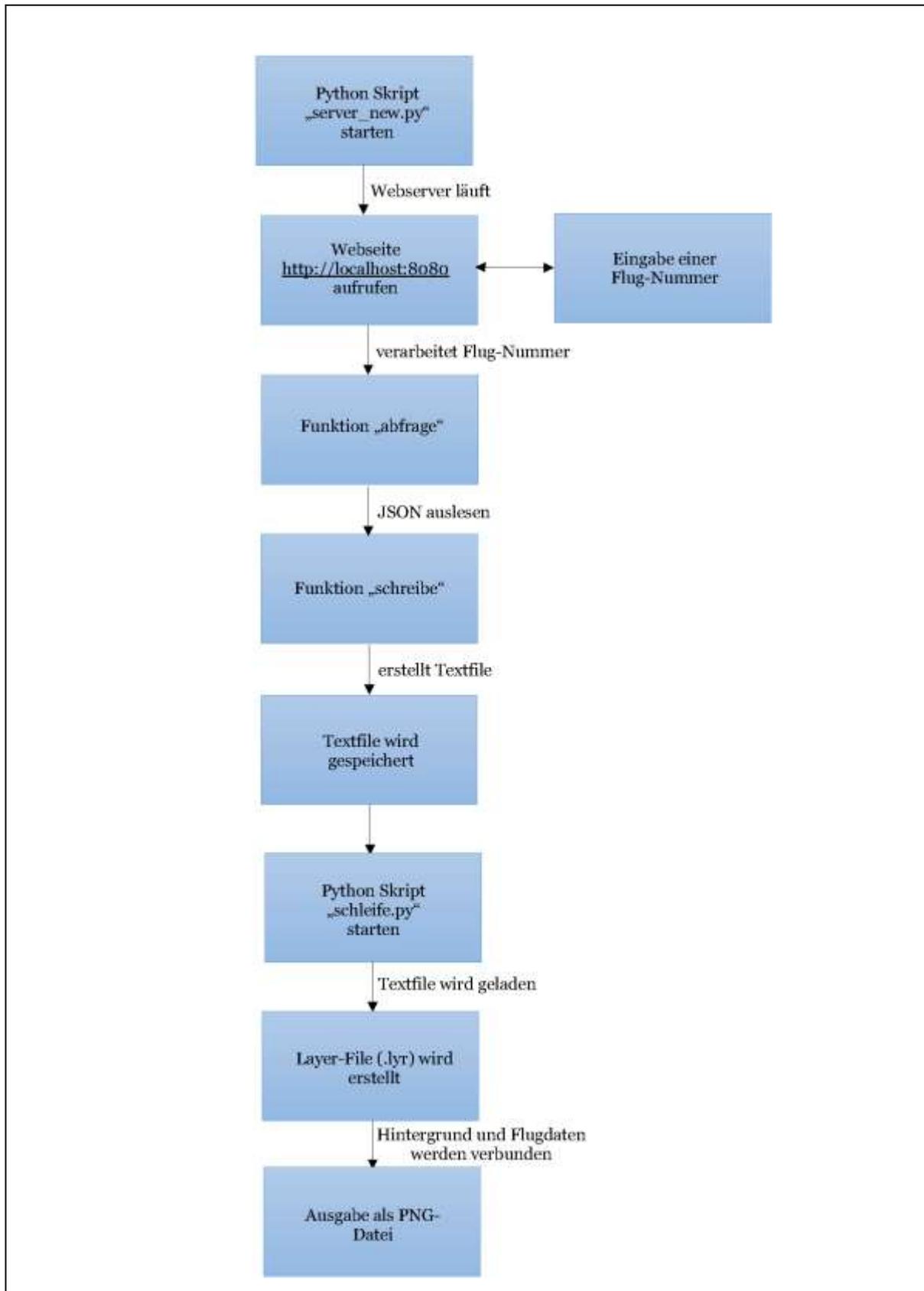


Abbildung 57 - Optimierung der Vektorenintegration mittels Python-Skripts, Quelle: Eigene Darstellung

Der erste Schritt ist das Starten des Python-Skripts mit dem Namen „*server_new*“. Dieses Skript beinhaltet einen mittels Programmiersprache Python erstellten Webserver, welcher nach dem Starten des Skripts gestartet wird, sodass der Nutzer/die Nutzerin über einen Internetbrowser die Webseite (<http://localhost:8080>) aufrufen kann. Dieser Webserver wird benötigt, um über einen Link einer Webseite das Auslesen von Daten zu ermöglichen, da von Seiten von FlightRadar24 keine API (Application Programming Interface) zur Verfügung gestellt wird.

In der internen Struktur der Webseite von FlightRadar24.com besteht folgender allgemeiner Link:

<https://www.flightradar24.com/v1/search/web/find?query=OS81&limit=1>

Wenn dieser Link in ein Browserfenster eingeben wird, bekommt der Nutzer folgende Ansicht der Daten im JSON (JavaScript Object Notation)-Format⁴ zurück (siehe Abb. 58):

```
{ "results": [ { "id": "d1cf27a", "label": "OS81 / AUA81 / B772 (OE-LPA)", "detail": { "lat": 69.1, "lon": -17.9, "schd_from": "VIE", "schd_to": "LAX", "ac_type": "B772", "route": "Vienna (VIE) → Los Angeles (LAX)", "logo": "fr24:AUA_logo0.png", "reg": "OE-LPA", "callsign": "AUA81", "flight": "OS81", "operator": "AUA", "type": "live", "match": "begins" }, "stats": { "total": { "all": 22, "airport": 0, "operator": 0, "live": 1, "schedule": 21, "aircraft": 0 }, "count": { "airport": 0, "operator": 0, "live": 1, "schedule": 0, "aircraft": 0 } } } ] }
```

Abbildung 58 - JSON-Ausgabe, Quelle: Eigene Darstellung

⁴ JSON ist ein Datenaustauschformat, dass nicht nur für den Menschen, sondern auch für Maschinen leichter lesbar ist (siehe Abb. 58).

Nach Seip, Korduan und Zehner [SEI-17] gelten für den Aufbau folgende Regeln:

- *JSON-Objekte beginnen mit einer öffnenden geschweiften Klammer: {*
- *JSON-Objekte enden mit einer schließenden geschweiften Klammer: }*
- *Zwischen den geschweiften Klammern sind keine (null) oder mehr Schlüssel/Wert-Paare (key-value pair), die Mitglieder (members) genannt werden*
- *Die Mitglieder werden durch Kommata getrennt*
- *Der Schlüssel und der Wert jedes Mitglieds werden durch einen Doppelpunkt getrennt*
- *Der Schlüssel ist eine Zeichenfolge und muss in doppelte Anführungszeichen gesetzt werden*
- *Das Format der Werte hängt von ihrem Datentyp ab*

Quelle: [SEI-17]

Wichtig hier zu erwähnen ist, dass die Flug-Nummer den wichtigsten Teil dieses Links ausmacht, da über diese eindeutige Nummer die Daten von der Datenbank von FlightRadar24 abgerufen werden. Sollte keine Flugnummer bekannt sein, kann dieser Link auch mittels Kennzeichen des Luftfahrzeuges (zum Beispiel: OE-LPA) verwendet werden.

Nach dem Öffnen der lokalen Webseite (siehe Abb. 59) bietet sich dem Nutzer/der Nutzerin folgendes Erscheinungsbild und eine Flugnummer muss im leeren Feld eingegeben werden.



Abbildung 59 - lokale Webseite, Quelle: Eigene Darstellung

Anschließend wird der oben angeführte Link über eine Funktion (*abfrage*) mittels der Bibliothek „*urllib2*“ ausgelesen. Mit Hilfe der Bibliothek „*json*“ wird der Datensatz im JSON-Format ausgelesen, um die benötigten Variablen, welche sich im Dictionary-Format⁵ befinden (*latitude*, *longitude*, *origin*, *destination*, *registration* and *flightnumber*) auszugeben (siehe Abb. 60).

⁵ “A Dictionary (or “dict”) is a way to store data just like a [list], but instead of using only numbers to get the data, you can use almost anything. This lets you treat a [dict] like it’s a database for storing and organizing data.” Quelle: <https://learnpythonthehardway.org/book/ex39.html>, abgerufen am 15.09.2017

```

def abfrage(self, flieger): #Funktion definieren
    print(flieger)

link='https://www.flightradar24.com/v1/search/web/find?query={fliegerreg}&limit=1'
    htmlheaders = { 'User-Agent' : 'Mozilla/5.0' }
    lineformat='{0} {1} {2} {3} {4} {5}\n'

    #Request des Links + auslesen
    req = urllib2.Request(link.format(fliegerreg=flieger), None,
htmlheaders)
    html = urllib2.urlopen(req).read()

    #JSON auslesen und LAT und LON ausgeben
    myjson = json.loads(html)
    flightid=myjson['results'][0]['id']
    number=myjson['results'][0]['label']

    lat=None
    lon=None
    origin=None
    destination=None
    reg=None
    flightnumber=None

    if flightid not in number:
        lat= myjson['results'][0]['detail']['lat']
        lon= myjson['results'][0]['detail']['lon']
        try:
            origin= myjson['results'][0]['detail']['schd_from']
            destination= myjson['results'][0]['detail']['schd_to']
        except:
            pass
        reg= myjson['results'][0]['detail']['reg']
        flightnumber=myjson['results'][0]['detail']['flight']#greifen
auf Daten zu um diese ins Txt zu schreiben (ob vorhanden ist)
        newline=lineformat.format(lat,lon,origin,destination,reg,flightnumber)
#name vom neuen String

    return newline

```

Abbildung 60 - Funktion Abfrage, Quelle: Eigene Darstellung

Als nächsten Schritt wird in der Funktion (*schreibe*) ein Textfile als Output erstellt, in dem sich ein Zeitstempel im Dateinamen befindet. Weiters werden die in das Textfile zu schreibenden Informationen in der Überschrift des Textfiles definiert (siehe Abb. 61).

```
def schreibe(self, liste):
    timeformat='{:%Y_%m_%d_%H_%M_%S}'

    fileheader='lat long origin destination reg flightnumber\n'
    textfile='output/{0}.txt'

    #erstellen TXT-File, oeffnen es, schreiben lat, lon hinein und
    #schliessen es
    timestamp=timeformat.format(datetime.datetime.now()) #zeitstempel
    filename=textfile.format(timestamp)
    f=open(filename, 'w+')
    f.write(fileheader)
    f.writelines(liste)
    f.close()

    return filename
```

Abbildung 61 - Funktion Schreibe, Quelle: Eigene Darstellung

Um diese Schritte für den Nutzer zu automatisieren, wurden keine Registrierungen (Luftfahrzeugkennzeichen) oder Flugnummern aus einem Flugplan von Luftfahrzeugen im Skript vordefiniert, sondern mit Hilfe der Bibliotheken „BaseHTTPServer“, „BaseHTTPRequestHandler“, „HTTPServer“ ein HTTP-Server über den Port 8080 definiert. Somit ist es jedem Nutzer möglich, eine Flugnummer oder Registrierung eines Luftfahrzeuges über eine definierte Box in einem HTML-File einzugeben (siehe Abb. 59). Durch einen GET und POST Request werden die eingegebenen Daten (z. B. OS81) über das Skript verarbeitet und ein Textfile in den zuvor erstellten Output-Ordner geschrieben (siehe Abb. 62).

```
#!/usr/bin/python
from BaseHTTPServer import BaseHTTPRequestHandler,HTTPServer
from os import curdir, sep
import cgi
import urllib2
import json
import datetime

PORT_NUMBER = 8080

#This class will handle any incoming request from the browser
class myHandler(BaseHTTPRequestHandler):
```

Abbildung 62 - Bibliotheken und Definition des Ports des HTTP-Servers, Quelle: Eigene Darstellung

```
#Handler for the GET requests
def do_GET(self):
    if self.path==" / ":
        self.path="/index.html"

    try:
        #Check the file extension required and set the right mime type

        sendReply = True
        mimetype='text/html'

        if sendReply == True:
            #Open the static file requested and send it
            f = open(curdir + sep + self.path)
            self.send_response(200)
            self.send_header('Content-type',mimetype)
            self.end_headers()
            self.wfile.write(f.read())
            f.close()
        return

    except IOError:
        self.send_error(404,'File Not Found: %s' % self.path)
```

Abbildung 63 - GET-Request, Quelle: Eigene Darstellung

```
#Handler for the POST requests
def do_POST(self):
    if self.path==" / send ":
        form = cgi.FieldStorage(
            fp=self.rfile,
            headers=self.headers,
            environ={'REQUEST_METHOD':'POST',
                    'CONTENT_TYPE':self.headers['Content-Type'],
                })

        flugnummer = form["txt"].value
        out_txt = self.abfrage(flugnummer)
        filename = self.schreibe(out_txt)

        print "Your name is: %s" % flugnummer
        self.send_response(200)
        self.end_headers()
        self.wfile.write("In Ihrem Output-Ordner befinden sich die
Daten für %s !" % flugnummer)
        return
```

Abbildung 64 - POST-Request, Quelle: Eigene Darstellung

```
try:
    #Create a web server and define the handler to manage the incoming
    request
    server = HTTPServer(('', PORT_NUMBER), myHandler)
    print 'Started httpserver on port ' , PORT_NUMBER

    #Wait forever for incoming http requests
    server.serve_forever()

except KeyboardInterrupt:
    print '^C received, shutting down the web server'
    server.socket.close()
```

Abbildung 65 - WebServer definieren und Eingabe verarbeiten, Quelle: Eigene Darstellung

In den Abbildungen 63 bis 65 wird der GET und POST Request des Webservers ausgeführt und die Eingabe des Nutzers verarbeitet.

Nach einer erfolgreichen Eingabe bekommt der/die Anwender/Anwenderin von der Webseite folgende Meldung retour: „In Ihrem Output-Ordner befinden sich die Daten für OS81!“ Im dafür vorgesehenen Output-Ordner befindet sich ein Textfile mit der aktuellen Position des Luftfahrzeugs. In unserem Fall handelt es sich hier um den Austrian Airlines Flug (OS81) von Wien-Schwechat (VIE) nach Los Angeles (LAX) (siehe Abb. 66).

```
lat long origin destination reg flightnumber
63.7 -15.9 VIE LAX OE-LPC OS81
```

Abbildung 66 - Inhalt des Textfiles nach Aufrufen des Webservers, Quelle: Eigene Darstellung

Als nächster Schritt wird ein weiteres Python-Skript mit dem Namen „schleife.py“ aufgerufen, welches das zuvor entstandene Textfile einliest und den Inhalt in ein ArcGIS LayerFile (.lyr) umbaut und eine Punktsignatur entstehen lässt.

Abbildung 67 zeigt die ersten Zeilen des Skripts „schleife.py“, wo diverse Module und Variablen definiert werden, um problemlos arbeiten zu können.

```
# Import system modules

import arcpy, os
arcpy.env.overwriteOutput = True

# Set local variables
output_path = r"C:\Users\austr\Dropbox\Masterarbeit\Pythonsachen\output"
path_pythonsachen = r"C:\Users\austr\Dropbox\Masterarbeit\Pythonsachen"
path_mxd =
r"C:\Users\austr\Dropbox\Masterarbeit\Pythonsachen\test\flug_new.mxd"
symbolology = path_pythonsachen + os.sep + r"test\symbology.lyr"
out_png_path = path_pythonsachen + os.sep + r"test\png"

arcpy.env.workspace = output_path
```

Abbildung 67 - Setzen von Modulen und Variablen, Quelle: Eigene Darstellung

Abbildung 68 zeigt eine while-Schleife, wo zuerst im Output-Ordner gesucht wird, ob dort ein Textfile beziehungsweise ein neues Textfile vorhanden ist, welches zur weiteren Bearbeitung herangezogen werden kann. Wenn ein Textfile vorhanden ist, wird der Name des Textfiles, in unserem Fall ein Zeitstempel im Format (2017_06_27_13_37_33 oder allgemein (Jahr_Monat_Tag_Stunde_Minute_Sekunde) angeschrieben und die Zahl 1 läuft aufgrund der Schleife durch. Sobald ein neues Textfile dem Output-Ordner hinzugefügt wird, wird wiederum der Name des Textfiles angeschrieben und die Zahl 2 läuft durch bis ein weiteres Textfile dazukommt. Bei jedem Hinzufügen eines Textfiles wird automatisch ein (.lyr)-File erstellt, wo die aktuelle Position des Luftfahrzeuges als Punkt dargestellt wird.

```
n=1
i=0

while n==1:
    try:
        dataset = arcpy.ListFiles()
        if dataset[i]:
            print dataset[i] + str(i)

            arcpy.env.workspace = output_path
            #dataset = arcpy.ListFiles() # remove

            output_txt = output_path + os.sep + dataset[i]

            fieldName0 = "lat"
            fieldName1 = "long"
            fieldName2 = "origin"
            fieldName3 = "destination"
            fieldName4 = "reg"
            fieldName5 = "flightnumber"

            # Set the local variables
            x_coords = fieldName1
            y_coords = fieldName0

            #z_coords = "POINT_Z"
            out_Layer = "flug"
            saved_Layer = path_pythonsachen + os.sep + r"test\flug" +
str(i) + ".lyr"

            # Set the spatial reference
            spRef = path_pythonsachen + os.sep + r"test\welt_prj1.shp"

            # Make the XY event layer...
            arcpy.MakeXYEventLayer_management(output_txt, x_coords,
y_coords, out_Layer, spRef)

            # Save to a layer file
            arcpy.SaveToLayerFile_management(out_Layer, saved_Layer)
```

Abbildung 68 - Schleife und Ausgabe als (.lyr)-File, Quelle: Eigene Darstellung

Der nächste Schritt ist die Erstellung eines ArcMap Dokuments mit einer quadratischen Platkarte, bevor das Skript ausgeführt wird. Es werden die Ländergrenzen der Welt und das entstandene (.lyr)-File in das ArcMap Dokument hineingeladen und die entstandene Karte im Format 4096 x 2048 Pixel exportiert (siehe Abb. 69). Dieses entstandene Rasterbild im Format PNG kann nun in das OmniSuite integriert werden und die aktuelle Position des Luftfahrzeuges wird dargestellt. Wenn von einem Flug mehrere Bilder vorhanden sind, kann die zeitliche Entwicklung des Fluges aufbereitet werden und ein Real Time-Eindruck beziehungsweise eine Bewegung entsteht für die BetrachterInnen des taktilem Hyperglobus.

```
# Add layer to mxd
mxd = arcpy.mapping.MapDocument(path_mxd)
dataFrame = arcpy.mapping.ListDataFrames(mxd, "*")[0]
addLayer = arcpy.mapping.Layer(saved_Layer)
arcpy.mapping.AddLayer(dataFrame, addLayer)

# Apply Symbology
lyr = arcpy.mapping.ListLayers(mxd)[0]
arcpy.ApplySymbologyFromLayer_management(lyr, symbology)

# Save as png
out_png = out_png_path + os.sep + "flug" + str(i) + ".png"
arcpy.mapping.ExportToPNG(mxd, out_png, dataFrame,
                          df_export_width=4096,
                          df_export_height=2048,
                          world_file=True,
                          transparent_color="255, 255, 255")

        i=i+1
except:
print i
```

Abbildung 69 - Ausgabe als PNG-File, Quelle: Eigene Darstellung

Von Vorteile ist, wie schon am Anfang dieses Kapitels erwähnt, dass ausschließlich Python-Skripte gestartet werden müssen und nur einmal eine Flugnummer eingegeben werden muss, um eine geeignete Kombination von Vektor- und Rasterdaten automatisch zu generieren. Außerdem können bei dieser Methode kaum Fehler auftreten, da das verwendete Koordinatensystem am Anfang definiert wird und das gleiche Koordinatensystem von Anfang bis Ende verwendet wird. Nachteil dieser Methode ist, wenn im Textfile keine Daten vorhanden sind, da der Flug sich derzeit nicht in der Luft befindet oder weil es keine Daten zum Flug gibt, dann startet das zweite Skript trotzdem und es wird nur die Weltkarte ohne punktförmige Signatur dargestellt. Ein weiteres Problem ist, dass eine Internetverbindung vorhanden sein muss, um den Webserver über den Port 8080 zu starten und die aktuellen Positionsdaten herunterladen zu können. Außerdem muss erwähnt werden, dass eine komplette Automatisierung des Skriptes aufgrund des Webserver nur schwer möglich ist und deshalb nicht umgesetzt wurde, aber in Kapitel 8.3 wird eine automatische Variante vorgestellt und näher erläutert.

7.3.2 Raster- und Vektordaten intern verschneiden

Abbildung 70 zeigt schematisch die Datenintegration von Vektor- und Rasterdaten ohne vorherige Verschneidung mit anderen Softwarepaketen. Hierbei werden Vektordaten und Rasterdaten gleichzeitig in OmniSuite eingelesen. Intern werden die Ebenen kombiniert, sodass der Nutzer eine richtige Darstellung der Daten erhält (vgl. [LIE-12], S. 45-46). Diese

Art der Datenintegration ist die zweite Methode, um Real Time-Flugdaten am Globus darzustellen. Anschließend wird genau erläutert, wie solche Textfiles für die Verwendung in OmniSuite erstellt werden müssen. Die rote Markierung kennzeichnet wiederum den Teil, welcher mittels eines Skripts automatisiert wurde.

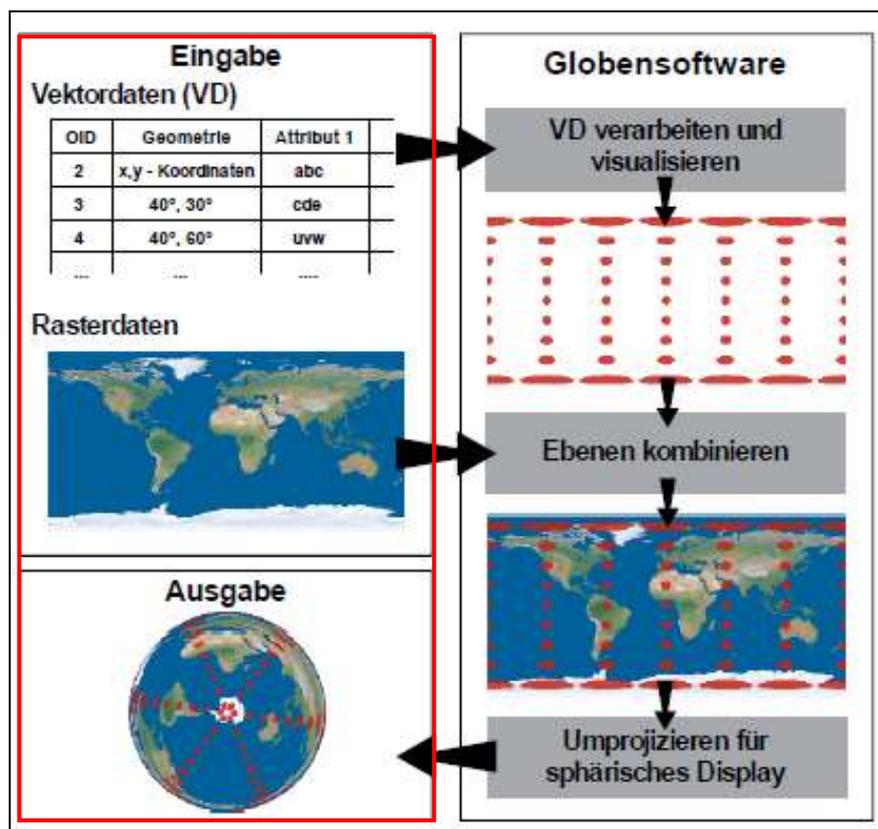
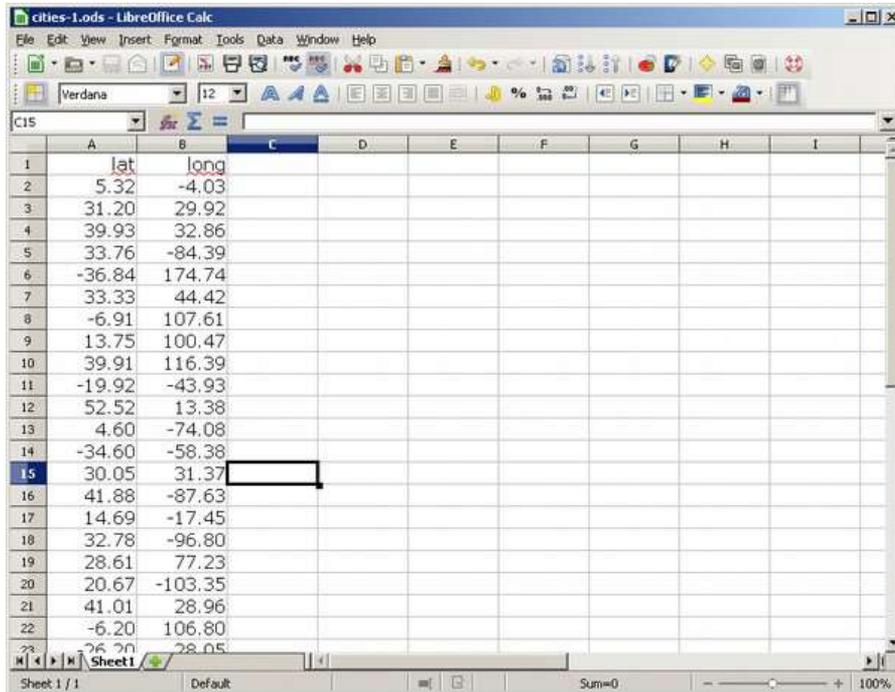


Abbildung 70 - Schema der internen Vektordatenintegration, Quelle: [LIE-12], S. 46

Die manuelle Erstellung von Punktdatensätzen ist seit der OminiSuite Version 4.14 möglich und diese Daten können als Textfiles eingelesen werden. Dadurch wurde auch im Python-Skript (siehe Kap. 7.3.1) das Output-Textfile so definiert, dass dieses auch für die interne Methode verwendet werden kann.

Um einen Punktdatensatz richtig zu laden, müssen folgende Schritte einhalten und folgende Einstellungen beachtet werden:

In einem Rechen- und Kalkulationsprogramm, zum Beispiel Microsoft Excel, werden in der ersten Spalte die Koordinaten der geographischen Breite (latitude) eingetragen und in der zweiten Spalte die geographische Länge (longitude) (siehe Abb. 71). Beide Spalten müssen das Datenformat Gleitkommazahl (= float) und Dezimalgrad aufweisen, da sonst die Zahlen nicht richtig interpretiert werden. Bevor Sie die Datei exportieren können, ist es zwingend notwendig, dass das Trennzeichen ein Leerzeichen sein muss.



	A	B	C	D	E	F	G	H	I
1		lat	long						
2		5.32	-4.03						
3		31.20	29.92						
4		39.93	32.86						
5		33.76	-84.39						
6		-36.84	174.74						
7		33.33	44.42						
8		-6.91	107.61						
9		13.75	100.47						
10		39.91	116.39						
11		-19.92	-43.93						
12		52.52	13.38						
13		4.60	-74.08						
14		-34.60	-58.38						
15		30.05	31.37						
16		41.88	-87.63						
17		14.69	-17.45						
18		32.78	-96.80						
19		28.61	77.23						
20		20.67	-103.35						
21		41.01	28.96						
22		-6.20	106.80						
23		76.20	28.05						

Abbildung 71 - Erstellung Textfile, Quelle: [GLO-14], S. 91

Als nächster Schritt erfolgt die Erstellung eines Cascading Style Sheets (CSS), um graphische Punkte für die Koordinaten anzeigen zu können. Davor sollten aber Gestaltungstests am Hyperglobus durchgeführt werden, damit die richtige Größe für eine optimale Darstellung der Punktsignaturen sichergestellt werden kann. Abbildung 72 zeigt eine Übersicht von unterschiedlich großen Punktsignaturen. Allgemein kann gesagt werden, dass Pixelgrößen von unter 7 Pixel und über 20 Pixel sich nicht für eine einfache punkthafte Darstellung einer aktuellen Position am Hyperglobus eignen, da diese einerseits zu klein und andererseits zu groß sind, um noch die aktuelle Position am Hyperglobus zu erkennen. Anzumerken ist, dass die richtige Signaturengröße vom Thema bestimmt wird. Bei Thematiken wo der Durchmesser des Kreises abhängig ist von einem absoluten Wert wird die Größe die 20 Pixel überschreiten.

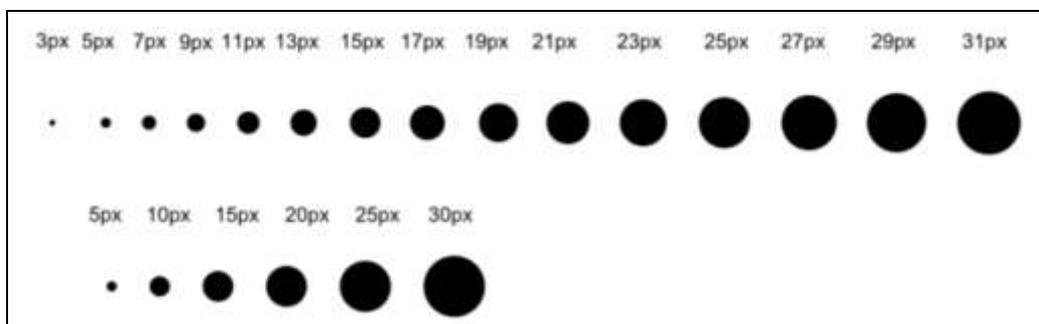


Abbildung 72 - Variation von Punktgrößen, Quelle: Eigene Darstellung

Standardmäßig wird nun folgendes CSS erstellt, welches Kreise mit 20 Pixel Durchmesser in der Farbe Rot und einer schwarzen Begrenzung erstellt.

```
.category1 {  
  
    marker-symbol: ellipsoid;  
  
    marker-width: 20;  
  
    marker-height: 20;  
  
    marker-color: rgba(255, 0, 0, 255);  
  
    marker-outline-style: solid;  
  
    marker-outline-width: 2px;  
  
    marker-outline-color: rgba(0, 0, 0, 255);  
  
}
```

Abbildung 73 zeigt die Darstellung nach dem Importieren der x/y-Koordinaten mittels des Textfiles.

Hier sollte nochmal darauf hingewiesen werden, wenn Textfiles mittels OmniSuite direkt eingelesen werden, dass die aus Kapitel 5.3. verwendete Formel der Tissot'schen Indikatrix im Hintergrund angewandt wird und die von der geographischen Breite ausgehende Verzerrung herausgerechnet wird.



Abbildung 73 - Darstellung der Punkte, Quelle: [GLO-14], S. 95

Diese CSS-Datei muss selbstständig in das Medienverzeichnis von OmniSuite in den Ordner „materials/styles“ kopiert und beim Erstellen der Materialien im Material Editor ausgewählt werden.

Eine zusätzliche Erweiterung ist das Hinzufügen von Namen der Städte, um eine Beschriftung bei den einzelnen Städten anzuzeigen (siehe Abb. 74) (vgl. [GLO-14], S. 96).

	A	B	C	D	E	F	G
1	lat	long	name				
2	5.32	-4.03	Abidjan				
3	31.20	29.92	Alexandria				
4	39.93	32.86	Ankara				
5	33.76	-84.39	Atlanta				
6	-36.84	174.74	Auckland				
7	33.33	44.42	Baghdad				
8	-6.91	107.61	Bandung				
9	13.75	100.47	Bangkok				
10	39.91	116.39	Beijing				
11	-19.92	-43.93	Belo Horizonte				
12	52.52	13.38	Berlin				

Abbildung 74 – Städtenamen für Beschriftung der Punkte, Quelle: [GLO-14], S. 96

Das passende CSS-Style Sheet für das Hinzufügen von Beschriftungen ist wie folgt definiert (vgl. [GLO-14], S. 96):

```
.category1 {  
  marker-symbol: ellipsoid;  
  marker-width: 15;  
  marker-height: 15;  
  marker-color: rgba(200, 155, 255, 100);  
  marker-outline-style: solid;  
  marker-outline-width: 2px;  
  marker-outline-color: rgba(0, 0, 0, 255);  
  text-name: [name];  
  text-dy: 5;  
  text-placement: S;  
}
```

Abbildung 75 zeigt das Ergebnis nach dem Hinzufügen der Städtenamen.

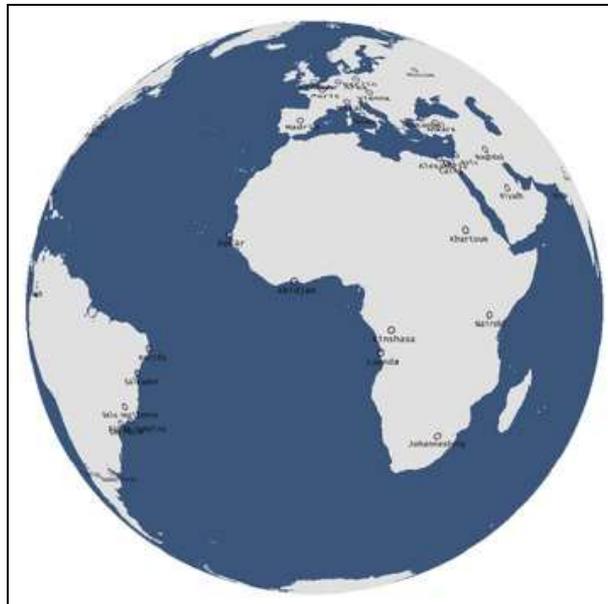
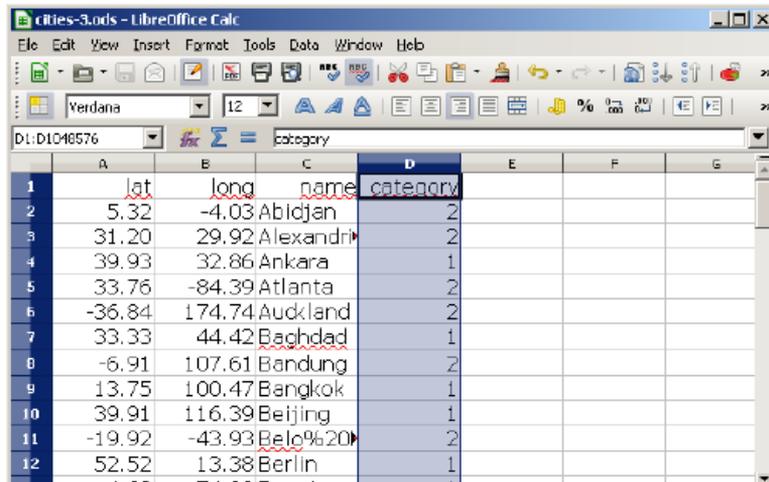


Abbildung 75 - Darstellung der Punkte und Namen, Quelle: [GLO-14], S. 97

Sollte es aufgrund der Thematik notwendig sein, können Kategorien erstellt werden. Hier wird eine weitere Spalte dem Textfile hinzugefügt und bei zum Beispiel zwei Kategorien die Zahlen 1 und 2 für die einzelnen Städte vergeben (vgl. [GLO-14], S. 98) (siehe Abb. 76).



	A	B	C	D	E	F	G
	lat	long	name	category			
1	5.32	-4.03	Abidjan	2			
2	31.20	29.92	Alexandria	2			
3	39.93	32.86	Ankara	1			
4	33.76	-84.39	Atlanta	2			
5	-36.84	174.74	Auckland	2			
6	33.33	44.42	Baghdad	1			
7	-6.91	107.61	Bandung	2			
8	13.75	100.47	Bangkok	1			
9	39.91	116.39	Beijing	1			
10	-19.92	-43.93	Belo Horizonte	2			
11	52.52	13.38	Berlin	1			
12	4.60	74.00	Buenos Aires	2			

Abbildung 76 - Kategorisierung der Punkte, Quelle: [GLO-14], S. 98

Weil die Punkte nun kategorisiert sind, ist es im weiteren Verlauf möglich, im CSS-File pro Kategorie eine andere Art der Darstellung zu wählen. Mögliche Änderungen sind verschiedene Farben und/oder Größen, wie im nachfolgenden CSS-File ersichtlich ist (vgl. [GLO-14], S. 99).

```
.category1 {
    marker-symbol: ellipsoid;
    marker-width: 15;
    marker-height: 15;
    marker-color: rgba(200, 155, 255, 100);
    marker-outline-style: solid;
    marker-outline-width: 2px;
    marker-outline-color: rgba(0, 0, 0, 255);
    text-name: [name];
    text-dy: 5;
    text-placement: S;
.category2 {
    marker-symbol: ellipsoid;
    marker-width: 15;
    marker-height: 15;
    marker-color: rgba(125, 155, 55, 100);
    marker-outline-style: solid;
    marker-outline-width: 2px;
    marker-outline-color: rgba(0, 0, 0, 255);
    text-name: [name];
    text-dy: 5;
    text-placement: S;
}
```

Die farbliche Kategorisierung von zuvor, eignet sich sehr gut für die Darstellung von Bevölkerungsdaten. Es wird eine weitere Spalte „value“ im Excel hinzugefügt und Bevölkerungsdaten von Weltstädten hinzugefügt. Um unterschiedliche Kreissignaturen zu bekommen werden über ein CSS-File unterschiedliche Farben für Kategorien gewählt, in unserem Fall grün und violett. Die Datenpunkte werden 2-fach überhöht dargestellt, um eine bessere Abgrenzung vornehmen zu können (siehe Abb. 77).

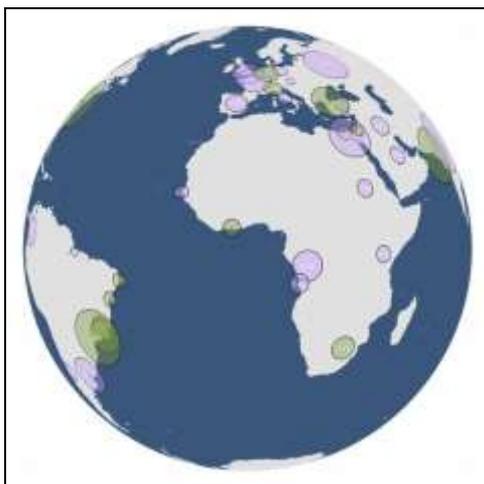


Abbildung 77 - Kategorisierung mit Bevölkerungsdaten, Quelle: [GLO-14]

Möchte man einen animierten Datensatz aufbereiten, dass einzelne Signaturen nur zu einem bestimmten Zeitintervall sichtbar sind, dann eignet sich das Beispiel der Erdbebenstandorte (siehe Abb. 78) sehr gut dafür. Dazu werden neben den x/y-Koordinaten auch das Datum mit Zeit und die Stärke des Erdbebens in einem Textfile festgehalten. Weiters muss eine Start- und Dauerspalte eingefügt werden. Über die Funktion „Tage“ in Microsoft Excel berechnet man die Anzahl der Tage zwischen zwei Daten. Durch die Dauer wird definiert, wie lange die einzelnen Punkte sichtbar sind (vgl. [GLO-14], S. 103).

	A	B	C	D	E	F
1	lat	long	value	start	time	duration
2	55,394	-134,65	7,5	0,11983109	05.01.13 08:58	10
3	-62,571	-161,432	6,1		15.01.13 16:09	
4	4,966	95,856	6,1		21.01.13 22:22	
5	42,605	79,708	6,1		28.01.13 16:38	
6	-28,08	-70,621	6,8		30.01.13 20:15	
7	-10,635	166,371	6,1		30.01.13 23:03	
8	-10,628	166,382	6,1		31.01.13 03:33	
9	-11,104	165,532	6		01.02.13 05:36	
10	-10,896	165,379	6,3		01.02.13 22:16	
11	-11,12	165,378	6,4		01.02.13 22:18	
12	42,758	143,106	6,9		02.02.13 14:17	
13	-10,938	165,255	6		02.02.13 18:58	
14	-10,865	165,248	6		06.02.13 00:07	

Abbildung 78 - Erstellung animierter Layer, Quelle: [GLO-14], S. 103

Das passende CSS-File für diese Art der Visualisierung ändert sich gegenüber den anderen CSS-Files der Beispiele von zuvor, da mit einem vorgefertigten PNG-Bild als Signatur gearbeitet wird, welches in das Medienverzeichnis gespeichert werden muss. Hierbei wird im Hintergrund des Programms die Verzerrung herausgerechnet (vgl. [GLO-14], S. 104).

```
.category1 {
    marker-symbol: image;
    marker-image:
    url('24\earthquakes\earthquakes_signatures\earthquake_signature.png');
    marker-width: 64;
    marker-height: 64;
    marker-start-transition: alpha 1.0 1.5;
    marker-start-transition: scale 2.0 0.25;
}
```

Auch für die zweite Methode der internen Vektordatenintegration (siehe Abb. 70) wurde vom Autor dieser Masterarbeit eine Optimierungsvariante erarbeitet, mit der eine automatisierte Erstellung von Textfiles notwendig ist. Diese Punktdatensätze werden nach Anlehnung an das Python-Skript „*server_new*“ aus Abbildung 67-69 mittels der Skriptsprache PHP (Hypertext Preprocessor) automatisch erstellt.

Die Webseite ist unter folgendem Link aufrufbar:
<http://www.unet.univie.ac.at/~a1167667/masterarbeit/index.php>⁶

Abbildung 79 zeigt ein Schema zum Ablauf dieser Erstellung.

⁶ Um Missbrauch zu verhindern, werden die erzeugten Textfiles derzeit in einem Ordner des Autors dieser Masterarbeit gespeichert!

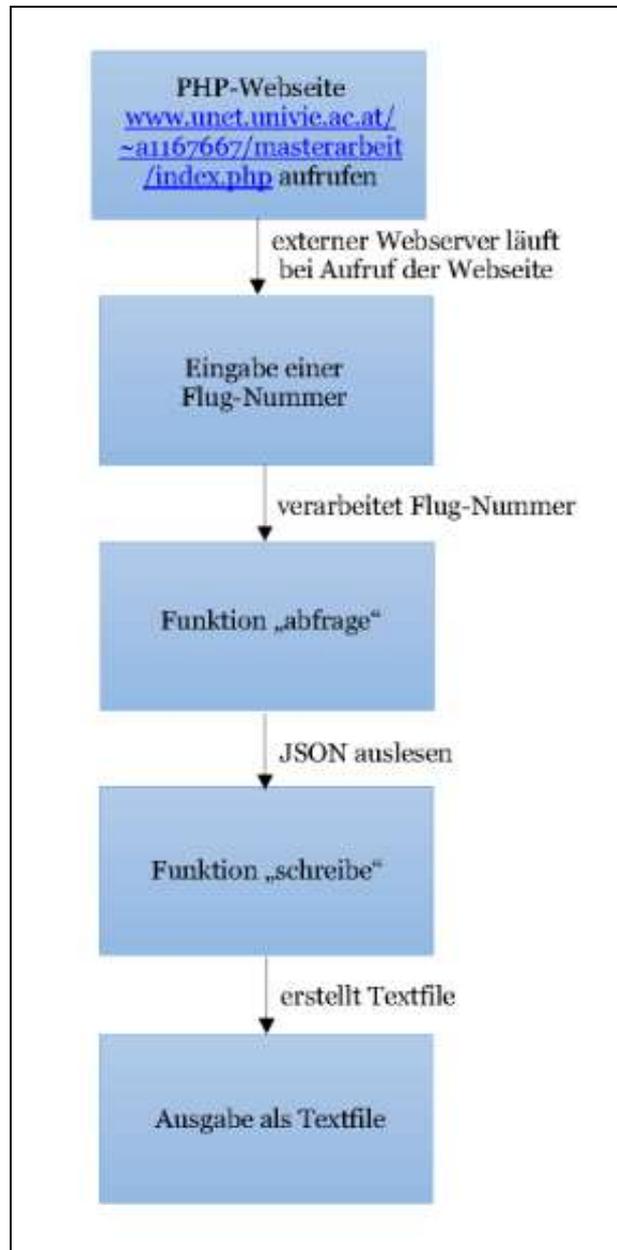


Abbildung 79 - Optimierung von Variante 2, Quelle: Eigene Darstellung

Es wurde ein PHP-Skript erstellt, sodass die beiden Funktionen „*schreibe*“ und „*abfrage*“ wiederum vorhanden sind (siehe Abb. 80/81). Das Umschreiben wurde notwendig, da ein Python-Skript nicht in eine dynamische Webseite eingebettet werden kann.

```
<?php
function schreibe($txt) {
    $timestamp = date("Y_m_d_H_i_s");
    $myfile = fopen("output/". $timestamp. ".txt", "w+") or die("Unable to
open file! ". $timestamp);
    $fileheader = 'lat long origin destination reg flightnumber\n';
    fwrite($myfile, $fileheader);
    fwrite($myfile, $txt);
    fclose($myfile);
}
function abfrage($flieger)
```

Abbildung 80 - Funktion "schreibe", Quelle: Eigene Darstellung

```
    $url =
'https://www.flightradar24.com/v1/search/web/find?limit=1&query=' .
$flieger;
    $json = file_get_contents($url);
    $myjson = json_decode($json);
    $flightid = $myjson['results'][0]['id'];
    $number = $myjson['results'][0]['label'];
    $lat = null;
    $lon = null;
    $origin = null;
    $destination = null;
    $reg = null;
    $flightnumber = null;
    $lat = $myjson['results'][0]['detail']['lat'];
    $lon = $myjson['results'][0]['detail']['lon'];
    try {
        $origin = $myjson['results'][0]['detail']['schd_from'];
        $destination = $myjson['results'][0]['detail']['schd_to'];
    }
    catch (Exception $ignore) {}
    $reg = $myjson['results'][0]['detail']['reg'];
    $flightnumber = $myjson['results'][0]['detail']['flight']; // #greifen
auf Daten zu um diese ins Txt zu schreiben (ob vorhanden ist)
    $line = "$lat $lon $origin $destination $reg $flightnumber\n";
    return $line;
}
if(isset($_GET["txt"])) { // wenn eine Flugnummer im Formular gesendet
wurde
    $flugnummer = $_GET["txt"];
    $out_txt = abfrage($flugnummer);
    $filename = schreibe($out_txt);
    echo "In Ihrem Output-Ordner befinden sich die Daten für $flugnummer
!<br/><br/><br/><hr/>";
}
?>
```

Abbildung 81 – Funktionen „abfrage“, Quelle: Eigene Darstellung

Der gesamte Mittelteil des Skriptes wird hier bewusst ausgelassen, da dieser nur die Gestaltung der eigentlich angezeigten Webseite ausmacht.

Abbildung 82 zeigt einen Auszug des Skriptes, welches die Daten des gesuchten Fluges auf der Webseite auflistet. Als Zusatz wird auch die aktuelle Position des Luftfahrzeuges bereits auf einer Google-Karte angezeigt (siehe Abb. 83).

```

<script id="flug_tmp" type="text/x-handlebars-template">
  {{#if results}}
    {{#each results}}
      <div class="6u$ 12u$(medium) ">
        <ul class="alt">
          <li><b>{{label}}</b></li>
          <li>Operator: {{detail.operator}}</li>
          <li>Flight: {{detail.flight}}</li>
          <li>Route: {{detail.route}}</li>
          <li>Lat: {{detail.lat}}</li>
          <li>Lon: {{detail.lon}}</li>
          <li>From: {{detail.schd_from}}</li>
          <li>To: {{detail.schd_to}}</li>
          <li>Reg: {{detail.reg}}</li>
        </ul>
        </div>
        <div class="5u$ 12u$(medium) " style="clear:none"><iframe
src="https://www.google.com/maps/embed/v1/place?key=AIzaSyAbrh-
uRnfyqZpOguHcUJnmeKwSVXx1zPc&q={{detail.lat}},{{detail.lon}}&zoom=4"
width="600" height="450" frameborder="0" style="border:0"
allowfullscreen></iframe></div>
          {{/each}}
        {{else}}
          <p>kein Ergebnis</p>
        {{/if}}
      </script>

```

Abbildung 82 – Skript zur Anzeige der Flugdaten mit Karte, Quelle: Eigene Darstellung

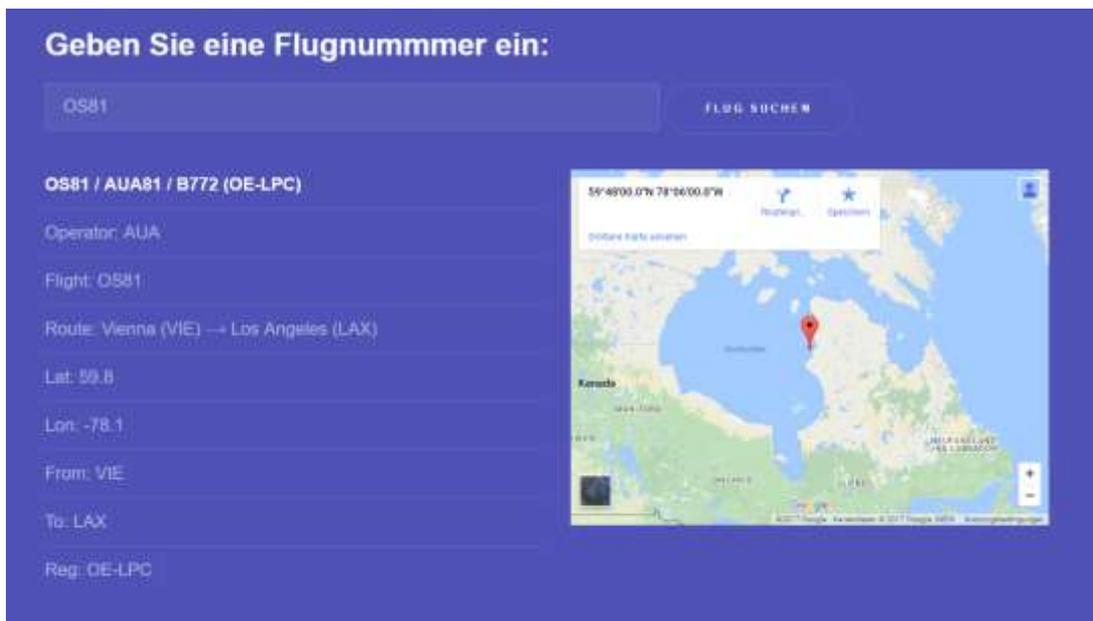


Abbildung 83 – Webseite, Quelle: Eigene Darstellung

Um eine Anzeige der Daten als Text und als Graphik auf der Homepage zu ermöglichen, wurde im unteren Teil des PHP-Skripts eine JavaScript Funktion eingebettet (siehe Abb. 84). Dadurch, dass die Daten im JSON-Format geliefert werden (siehe Abb. 58) und das Auflösen über einen Webserver erfolgen muss, aber durch eine Beschränkung der Webserver der Univerisät Wien nicht verwendet werden konnte musst auf einen Kollegen mit einer

Informatikfirma „Firma Expert Solutions Strahlhofer“ ausgewichen werden. Über den Webserver dieser Firma wird ein Service für die oben angeführte Webseite bereitgestellt, dass die JSON-Daten getrennt werden können und die benötigten Daten in das Textfile geschrieben werden können.

```
<script type="text/javascript" src="http://code.jquery.com/jquery-
latest.min.js"></script>
<script src="https://code.jquery.com/ui/1.12.1/jquery-ui.js"></script>
<script src="js/bootstrap.min.js"></script>
<script
src="https://cdnjs.cloudflare.com/ajax/libs/handlebars.js/4.0.6/handlebars.
min.js"></script>

<script>
$( document ).ready(function() {
// Suchen
$('#search').click(function (thing_id) {
    var flightId = $('#flug').val();

    //url =
'http://www.strahlhofer.com/flugsuche/proxy.php?curl=https://www.flightrad
ar24.com/v1/search/web/find?query=' + flightId;

$.ajax({
    type: 'POST',
    url: 'flugsuche/proxy.php?flight=' + flightId,
    //dataType: 'json',

    // jsonp : "callback",
    // jsonpCallback: "jsonpcallback",

    success: function (response) {
Handlebars.registerHelper('formatKomma', function(value) {
    //return value.toString().replace(".",",");
    return ((Math.round(value * 10) / 10).toString()).replace(".",",");
});

var source = $("#flug_tmp").html();
var template = Handlebars.compile(source );
$('#flug_end').html(template(response));
console.log(response)
    }

});
});
});
</script>
</body>
</html>
```

Abbildung 84 – JavaScript Funktion, Quelle: Eigene Darstellung

Da die meisten Anwenderinnen und Anwender keine Flugnummern auswendig wissen, gibt es auf der Homepage eine Auswahl von möglichen Langstreckenverbindungen von Austrian Airlines (siehe Tab. 11).

Flug-Nr.	Von	Nach	Abflugzeit [local time]	Ankunftszeit [local time]
OS17	Wien	Mauritius	1740	0655+1
OS18	Mauritius	Wien	0825	1610
OS21	Wien	Seychellen	2005	0800+1
OS22	Seychellen	Wien	1010	1630
OS23	Wien	Havanna	1015	1625
OS24	Havanna	Wien	1810	1050+1
OS25	Wien	Bangkok	2320	1520+1
OS26	Bangkok	Wien	2355	0525+1
OS31	Wien	Kapstadt	1015	2240
OS32	Kapstadt	Wien	0020	1050
OS45	Wien	Male	2005	0905+1
OS46	Male	Wien	1025	1630
OS47	Wien	Colombo	1750	0725+1
OS48	Colombo	Wien	1005	1600
OS51	Wien	Tokio	1755	1205+1
OS52	Tokio	Wien	1400	1900
OS57	Wien	Hongkong	1340	0805+1
OS58	Hongkong	Wien	1100	1630
OS63	Wien	Peking	1745	1020+1
OS64	Peking	Wien	1220	1600
OS65	Wien	Chicago	1010	1355
OS66	Chicago	Wien	1620	0835+1
OS71	Wien	Toronto	1025	1425
OS72	Toronto	Wien	1810	0845+1
OS75	Wien	Shanghai	1320	0650+1
OS76	Shanghai	Wien	1115	1600
OS81	Wien	Los Angeles	1000	1330
OS82	Los Angeles	Wien	1505	1200+1
OS87	Wien	New York	1045	1415
OS88	New York	Wien	1740	0820+1
OS89	Wien	Newark	1015	1350
OS90	Newark	Wien	1750	0825+1
OS93	Wien	Washington	1030	1440
OS94	Washington	Wien	1705	0820+1
OS97	Wien	Miami	0955	1515
OS98	Miami	Wien	1645	0900+1

Tabelle 11 – Langstreckenverbindungen von Austrian Airlines, Quelle: Austrian Airlines Timetable
(+1 = Landung am nächsten Kalendertag)

Die Vorteile der internen Methode sind eine rasche und einfache Erstellung der Daten, da x/y-Koordinaten im WGS-84 Format notwendig sind und diese sehr einfach für diverse Geodaten oder Ereignisse zu erhalten sind. Außerdem ist eine neue Form der Datengewinnung möglich, da animierte Global Stories erstellt werden können. Außerdem bietet diese Methode bereits eine Datenvisualisierung auf einer Webseite, bevor die Daten am Hyperglobus visualisiert werden können, um eventuelle Fehler zu minimieren.

Ein Nachteil dieser Methode besteht darin, dass ohne bestehender Internetverbindung kein Abruf von Daten möglich ist. Außerdem ist es nicht möglich, mehrere verschiedene Flüge gleichzeitig abzufragen, sondern es ist nur möglich, für einen Flug pro Abfrage Daten zu laden.

Im nächsten Kapitel erfahren Sie mehr über die neuesten Erweiterungen der direkten Visualisierung von Vektordaten am Hyperglobus. Außerdem bekommen Sie einen Überblick über die neuen interaktiven Anwendungen, die entstanden sind.

8 Optimierungsvorschläge zur Generierung von Vektordaten

Dieses Kapitel beschäftigt sich mit möglichen und bereits im Rahmen von Projekten gemeinsam mit Herrn Mag. Jürgen Kristen umgesetzten neuen Methoden zur direkten Implementierung von Vektordaten. Außerdem sollen die über das Werkzeug der Punkt-Datensätze möglichen Interaktionsmöglichkeiten besprochen werden, da erst mit Beginn dieser Masterarbeit, Projekte im Bereich der Optimierung mittels Vektordaten umgesetzt worden sind.

8.1 Interaktionsmöglichkeiten mit Punkt-Datensatz

In diesem Kapitel sollen Beispiele erläutert werden, welche während des Verfassens dieser Arbeit entstanden sind und auf dem Prinzip des Punktdatensatzes und der Interaktionserweiterung basieren.

Seit der Version 4.14 besteht auch die Möglichkeit über eine HTML-Seite mit Java-Script Funktionen die Interaktivität von Stories zu steuern. Eine Auswahl von möglichen Funktionen findet man unterhalb:

- *openCollection(string Collectionname)*
- *play()*
- *pause()*
- *resetStoryRotation()*
- *gotoSecond(int second)*
- *setLanguage(string languageID)*
- *addStoryRotation(float ggLongitude, float ggLatitude, float Inclination)*
- *gotoBookmark(string bookmarkname)*
- *setMaterialLayerTransparency(string layername, float factor)*
- *openStory(string storyname)*
- *setHTMLparam1(float value)*
- *startCapturing(string left, string top, string width, string height, int mode, bool useRelativePosition, bool hideController)*

- *endCapturing()*
- *setTimeSpan(string startTime, string endTime, int fps)*
- *getCurrentOrientaion()*
- *highlightPoint(string name)*
- *setExaggeration(float value)*
- *reloadCurrentMaterials()*
- *setCurrentSecond(int second)*
- *setDuration(int duration)*
- *setStoryname(string storyname)*
- *clearBookmarkList()*
- *addBookmark(string bookmarkname)*
- *selectBookmark(int bookmarkindex)*
- *clearLayerList()*
- *addLayer(string layername)*
- *OrientationRecieved(string orientaion)*
- *searchStory(string searchstring)*
- *hideCalendar()*
- *showCalendar(string timespan)*

8.1.1 Beispiel Städte-Quiz

Das Städte-Quiz ist die erste fertige Umsetzung, mit der das Werkzeug des Punktdatensatzes aus dem Manual 4.14 mit den möglichen Interaktionsmöglichkeiten (siehe Kap. 8.1.) umgesetzt wurde. Nutzen des Quiz ist es, dass Personen ihr räumliches und geographisches Wissen testen können, um die richtige Lage des abgefragten Ortes zu finden.

Ausgehend von einem Textfile mit Koordinaten für Punkte von Städten der Welt wird ein CSS-File genommen, in dem alle Punkte mit der gleichen Größe und ohne Namen in OmniSuite dargestellt werden. Nach den ganzen Voreinstellungen wird eine Story erstellt, in der keine automatische Rotation eingestellt wird, aber das selbstständige Rotieren des Anwenders über das Interface des OmniSuite Controllers möglich ist. Dies wird mit der

Funktion (*AllowRotationManipulation*) aufgerufen. Auf der zweiten Globuskugel definiert man ein zweites Material, in diesem Fall eine Hand (siehe Abb. 85), die als Zeiger für den richtig ausgewählten Punkt der Stadt dient.



Abbildung 85 - Interface des Städte Quiz, Quelle: Eigene Darstellung

Der HTML-Code der Startseite für die Webseite (siehe Abb. 86), die durch Java-Script Funktionen erweitert und mit einer Global Story verbunden ist, schaut wie folgt aus:

```
<span class="de">
<p class="bodyText">
Ein Beispiel für ein Städte Quiz auf einer Stummen Karte: Zeige mir
Istanbul.
</p>
<p class="bodyText">
1. Rotiere den Globus zu der gesuchten Stadt:
</p>
<p class="bodyText">
2. Wenn fertig, drücke den Button darunter:
</p>
<p style="background-image:url(./Cities_Quiz/button.png); width: 200px;
height: 60px; text-align: center; vertical-align: middle; line-height:
60px; margin-left: 9%" onClick="window.external.getCurrentOrientation()">
Zeige Ergebnis
</p>
<p class="bodyText" id="result">
</p>
</span>
```

Abbildung 86 - HTML Startseite Städte Quiz, Quelle: Eigene Darstellung

In der Funktion wird zunächst eine Variable definiert, welche die Toleranz bei der Berechnung des Ergebnisses beinhaltet. Die Toleranz könnte angepasst werden, indem der

User unter verschiedenen Schwierigkeitsstufen auswählen kann (Normal, Fortgeschritten, Experte oder Altersstufen).

```
var toleranz = 2;
```

Danach wird die Variable (*orientation*), welche von OmniSuite zurückgegeben wird, an den Positionen der Leerzeichen zerlegt und im Array (*values*) gespeichert. Die Variable (*orientation*) beinhaltet die geographische Breite der Story, die geographische Breite der Benutzerrotationen, die geographische Länge der Story sowie die geographische Länge der Benutzerrotation – getrennt durch Leerzeichen.

Aus den Werten von (*values*) wird dann die aktuelle geographische Länge und Breite der vom Benutzer gewählten Orientierung berechnet, indem die jeweiligen Werte von Story und Benutzerrotation addiert werden.

```
var values = orientation.split(" ");  
var latitude = Number(values[1]) + Number(values[3]);  
var longitude = Number(values[0]) + Number(values[2]);
```

Danach erfolgt die Überprüfung des Ergebnisses. Die geographische Länge und Breite der Stadt Istanbul sind noch direkt in der Überprüfung enthalten. Geprüft wird, ob die oben von der aus OmniSuite erhaltenen Orientierung mit der tatsächlichen Orientierung und dem Toleranzbereich übereinstimmt.

```
if(latitude > 41-toleranz && latitude < 41+toleranz && longitude <  
29+toleranz && longitude > 29-toleranz)
```

Je nach Ergebnis wird dann der Inhalt des HTML-Elementes (*result*) verändert, zum Beispiel für das korrekte Ergebnis:

```
document.getElementById("result").innerHTML = "<span  
class=\"de\"><strong>Richtig!</strong></span><span  
class=\"en\"><strong>Well done!</strong></span>";
```

Als Nächstes soll dem Benutzer auch am Globus das Ergebnis angezeigt werden. Dazu soll erstens der gesuchte Punkt hervorgehoben werden und zweitens auch der Globus so orientiert werden, dass der Benutzer diesen sehen kann, falls dieser völlig falsch gelegen ist.

Das Hervorheben des Punktes erfolgt über den Aufruf von:

```
window.external.highlightPoint("Istanbul");
```

Dies weist OmniSuite an, einen Punkt mit dem Namen (Spalte im Textfile) Istanbul hervorzuheben. Dazu ist auch ein weiterer Stil mit Namen (*categoryHighlight*) hinzuzufügen, der die Formatierung des hervorgehobenen Punktes definiert.

Anschließend wird die Differenz zwischen aktueller Orientierung am Globus und tatsächlicher Position der Stadt berechnet:

```
var diffLat = latitude-41+3;  
var diffLong = longitude-29;
```

Die Orientierung des Globus soll um diese Differenz rotiert werden, sodass die gesuchte Stadt genau unter dem Zeiger des Benutzers positioniert wird.

```
window.external.addStoryRotation(-diffLong,diffLat,0);
```

Genau genommen werden 3° bei der Berechnung der Differenz der geographischen Länge addiert, damit die Stadt nicht unter dem Zeiger positioniert wird.

Weitere Verbesserungsmöglichkeiten könnten wie folgt aussehen:

Dem Benutzer könnte angeboten werden, zuerst die Städte auf einer vollständigen Darstellung ansehen zu können. Der folgende Code ruft eine weitere Story auf, welche diese vollständige Darstellung beinhaltet:

```
<a href="#"  
onclick="window.external.openStory('Urban_Areas_by_Population')">here</a>  
  
var cities = [  
{"name":"Jakarta", "latitude":-6.20, "longitude":106.80},  
{"name":"Bogota", "latitude":4.60, "longitude":-74.08},  
{"name":"Kinshasa", "latitude":-4.33, "longitude":15.32},  
{"name":"Abidjan", "latitude":5.32, "longitude":-4.03},  
{"name":"Nairobi", "latitude":-1.28, "longitude":36.82},];  
function CityChanged() {  
    window.external.highlightPoint("");>  
    document.getElementById("result").innerHTML = "";  
}
```

Abbildung 87 zeigt das fertige Interface-Design der Webseite, in der eine Stadt ausgewählt werden kann und der Anwender/die Anwenderin den Globus selbstständig zur gesuchten Stadt rotieren kann.



Abbildung 87 - Interface der Webseite Städte-Quiz, Quelle: Eigene Darstellung

8.2 Integration von Linien

Der Autor dieser Arbeit hat sich aufgrund der Thematik Luftfahrt eine weitere Methode überlegt, abgesehen von Punktdaten auch linienhafte Elemente darzustellen. Naheliegend ist bei der Luftfahrtthematik neben der aktuellen Position des ausgewählten Luftfahrzeuges die Orthodrome zwischen dem Start- und Zielflughafen zu visualisieren. Dieses Kapitel gibt einen Überblick eines Python-Skriptes, welches ausgehend von einem erstellten Textfile mit Flugdaten eine Orthodrome in eine Plattkarte zeichnet und diese als PNG-Datei speichert.

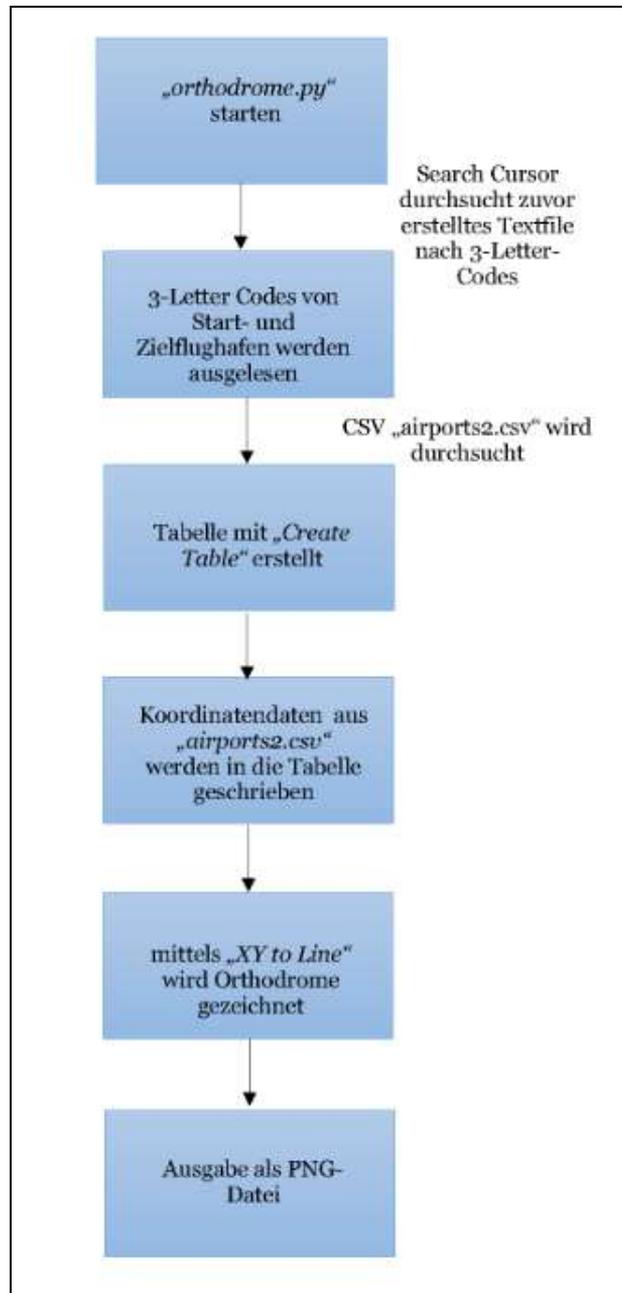


Abbildung 88 – Orthodromendarstellung als PNG mittels Python-Skript, Quelle: Eigene Darstellung

Abbildung 88 zeigt eine neue Methode mittels eines Python-Skripts mit dem Namen „*orthodrome.py*“. Ausgehend von den heruntergeladenen Daten, werden die Buchstabenkombinationen (3-Letter-Codes)⁷ der Flughäfen herangezogen, um eine Orthodrome mittels einer ArcGIS Funktion „*XY to Line*“ von einem Startflughafen zu einem Zielflughafen zu erstellen und diese als PNG anzugeben. Am Beginn wurde von der Firma

⁷ z.B. VIE = Flughafen Wien; LAX = Los Angeles

ESRI ein Layer Package⁸ mit allen in der Welt vorkommenden Flughäfen gedownloadet und die Daten gesichtet und aktualisiert, da einige Flughäfen, unter anderem Belgrad, Klagenfurt, Basel, Dnipropetrowsk, Erbil, Genf, Lyon, Nizza, Teheran und Varna in dieser Datei fehlten. Weiters wurden für jeden Flughafen mittels der Funktion „Calculate Geometry“ die X- und Y-Koordinaten hinzugefügt und anschließend die Tabelle in eine CSV-Datei (= Comma-separated values) mit dem Namen „airpots2.csv“ exportiert. Anschließend wurden Daten ohne Koordinaten gelöscht und die Daten neu transponiert, sodass die erste Zeile die 3-Letter-Codes der Flughäfen beinhalten und die Zeilen 2 und 3 die Koordinaten. Diese Transponierung ist wichtig, um im Python-Skript die Abfrage der Daten einfacher zu gestalten (siehe Abb. 89).

	A	B	C	D		A	B	C	D	E	F	G	H	I
1	NAME	IATA	POINT_X	POINT_Y	1	AAE	AAI	AAJ	AAK	AAL	AAM	AAN	AAO	AAQ
2	Inuvik Mike	YEV	-133,48	68,30	2	7.81	9.85	55.61	37.35	10.62	52.17	-1.86	48.23	-75.44
3	Resolute Bay	YRB	-94.97	74.72	3	36.82	57.05	24.26	43	56.3	16.19	38.95	30.37	40.65
4	Provideniya	PVS	-173,24	64,38	4									
5	Nome	OME	-165,45	64,51	5									
6	Dillingham	DLG	-158,50	59,05	6									
7	Fairbanks Int	FAI	-147,86	64,82	7									
8	McGrath		-155,61	62,95	8									
9	Talkeetna		-150,09	62,32	9									
10	King Salmon	AKN	-156,65	58,68	10									
11	Kodiak	ADQ	-152,49	57,75	11									
12	Yellowknife	YZF	-114,44	62,46	12									
13	Watson Lake	YQH	-128,82	60,12	13									
14	Fort Smith	YSM	-111,96	60,02	14									
15	Fort McMurr	YMM	-111,22	56,65	15									
16	Terrace	YXT	-128,38	54,47	16									
17	Smithers	YYD	-127,18	54,82	17									

Abbildung 89 – Daten der Flughäfen der Welt, links: ohne Transponieren, rechts: mit Transponierung, Quelle: Eigene Darstellung

Abbildung 90 zeigt die allgemeinen Einstellungen des Python-Skripts und die Definierung des Workspace und das Importieren diverser Funktionen.

⁸ Quelle: <http://www.arcgis.com/home/item.html?id=dccf673a74194790af4479d0d332f8df>, abgerufen am 12.10.2017

```

1  # Import system modules
2  import arcpy, os, sys
3  from arcpy import env
4
5  arcpy.env.overwriteOutput = True
6
7  # Define input_table_airport
8  ws = r"C:\Users\austr\Dropbox\Masterarbeit\Pythonsachen"
9  arcpy.env.workspace = ws + os.sep + "output"
10 datasets = arcpy.ListFiles()
11 for dataset in datasets:
12     print dataset
13
14 # Set local variables
15 input_table_airport = ws + os.sep + "output" + os.sep + dataset
16 input_table_coord = ws + os.sep + "airports2.csv"
17 out_table_path = ws + os.sep + r"test\flug.gdb"
18 out_name = "coordinates"
19 out_table = ws + os.sep + r"test\flug.gdb" + os.sep + out_name
20 out_lines = ws + os.sep + r"test\flug.gdb\flug"
21 saved_Layer = ws + os.sep + r"test\orthodrome.lyr"
22 path_mxd = ws + os.sep + r"test\flug.mxd"
23 out_png = ws + os.sep + r"test\png\orthodrome.png"
24

```

Abbildung 90 - Allgemeine Einstellungen, Quelle: Eigene Darstellung

```

25 #-----Get Coordinates-----#
26
27 # Get origin and destination airport
28 SC = arcpy.SearchCursor(input_table_airport)
29 origin_field = "origin"
30 destination_field = "destination"
31 for row in SC:
32     origin = row.getValue(origin_field)
33     destination = row.getValue(destination_field)
34
35     print origin
36     print destination
37
38 # Get coordinates
39 fc = input_table_coord
40 fields = origin, destination
41 values_orig = [row[0] for row in arcpy.da.SearchCursor(fc, fields)]
42 values_dest = [row[1] for row in arcpy.da.SearchCursor(fc, fields)]
43 origin_x = values_orig[0]
44 origin_y = values_orig[1]
45 dest_x = values_dest[0]
46 dest_y = values_dest[1]
47

```

Abbildung 91 - Koordinaten auslesen, Quelle: Eigene Darstellung

In Abbildung 91 werden mittels eines Search Cursors die Daten durchsucht und die 3-Letter Codes vom Start- und Zielflughafen ausgelesen. In einer Tabelle mit dem Namen „airports2.csv“ werden alle internationalen Flughäfen mit den diversen 3-Letter Codes aufgelistet. Diese Datei wird beim nächsten Schritt in Abbildung 91 herangezogen. Am Anfang wird eine leere Tabelle mit der Funktion „CreateTable“ erzeugt und eine leere Zeile eingefügt und die Daten, ausgehend vom Search Cursor und der Tabelle, mit den gesamten Koordinaten der Flughäfen abgesucht und die benötigten Daten in die neue Tabelle mittels

der Funktion „AddField“ geschrieben und eine Orthodrome mit dem Werkzeug „XY to Line“ gezeichnet.

```

48 #-----Write coordinates into table-----#
49
50 # Create table
51 arcpy.CreateTable_management(out_table_path, out_name)
52 arcpy.AddField_management(out_table, "startx_field", "DOUBLE")
53 arcpy.AddField_management(out_table, "starty_field", "DOUBLE")
54 arcpy.AddField_management(out_table, "endx_field", "DOUBLE")
55 arcpy.AddField_management(out_table, "endy_field", "DOUBLE")
56
57 # Insert row into table
58 rows = arcpy.InsertCursor(out_table)
59 row = rows.newRow()
60 rows.insertRow(row)
61
62 # Delete cursor and row objects to remove locks on the data
63 del row
64 del rows
65
66 # Write coordinates into table
67 arcpy.CalculateField_management(out_table, "startx_field", "{}".format(origin_x))
68 arcpy.CalculateField_management(out_table, "starty_field", "{}".format(origin_y))
69 arcpy.CalculateField_management(out_table, "endx_field", "{}".format(dest_x))
70 arcpy.CalculateField_management(out_table, "endy_field", "{}".format(dest_y))
71
72 #-----Create orthodrome-----#
73
74 #XY To Line
75 arcpy.XYToLine_management(out_table,out_lines,
76     "startx_field","starty_field","endx_field",
77     "endy_field","GREAT_CIRCLE")
78

```

Abbildung 92 - Koordinaten in Tabelle schreiben und Orthodrome erstellen, Quelle: Eigene Darstellung

```

79 #-----Save as png-----#
80
81 # Make feature layer
82 arcpy.MakeFeatureLayer_management (out_lines, "orthodrome")
83
84 # Save orthodrome to layer file
85 arcpy.SaveToLayerFile_management("orthodrome", saved_Layer)
86
87 # Add layer to mxd
88 mxd = arcpy.mapping.MapDocument(path_mxd)
89 dataframe = arcpy.mapping.ListDataFrames(mxd, "**")[0]
90 addLayer = arcpy.mapping.Layer(saved_Layer)
91 arcpy.mapping.AddLayer(dataFrame, addLayer)
92
93 # Save as png
94 arcpy.mapping.ExportToPNG(mxd, out_png, dataframe,
95     df_export_width=5000,
96     df_export_height=2200,
97     world_file=True,
98     transparent_color="255, 255, 255")
99

```

Abbildung 93 - Speichern als PNG, Quelle: Eigene Darstellung

Der letzte Schritt (siehe Abb. 93) ist das Speichern als PNG-Datei, um diese für den Globus als zusätzliches Material einfügen zu können. Das Ergebnis des PNG-Bildes ist in Abbildung 94 zu sehen, in diesem Fall wird ein Flug von Dubai nach New York visualisiert.



Abbildung 94 - Ergebnis Orthodrome, Quelle: Eigene Darstellung

Eine weitere Methode wurde ebenso umgesetzt, indem nach einer Formel gesucht wurde, mit der die Zwischenpunkte entlang der Orthodrome von einem Fixpunkt zu einem anderen Fixpunkt berechnet werden.

Folgende Beschreibung und Formel wurde gefunden und zur Umsetzung verwendet:

„In previous sections we have found intermediate points on a great circle given either the crossing latitude or longitude. Here we find points (lat, lon) a given fraction of the distance (d) between them. Suppose the starting point is (lat1,lon1) and the final point (lat2,lon2) and we want the point a fraction f along the great circle route. f=0 is point 1. f=1 is point 2. The two points cannot be antipodal (i.e. lat1+lat2=0 and abs(lon1-lon2)=pi) because then the route is undefined. The intermediate latitude and longitude is then given by”⁹:

```
A=sin((1-f)*d)/sin(d)
B=sin(f*d)/sin(d)
x = A*cos(lat1)*cos(lon1) + B*cos(lat2)*cos(lon2)
y = A*cos(lat1)*sin(lon1) + B*cos(lat2)*sin(lon2)
z = A*sin(lat1)      + B*sin(lat2)
lat=atan2(z, sqrt(x2+y2))
lon=atan2(y, x)
```

Ausgehend von dieser Formel wurde ein Werkzeug mittels der Programmiersprache C++ erstellt, welches die Orthodrome auf einer Plattkarte darstellt und als Rasterbild im PNG-

⁹ <http://www.edwilliams.org/avform.htm#Intermediate> und <https://stackoverflow.com/questions/1868115/calculating-shortest-path-between-2-points-on-a-flat-map-of-the-earth>, abgerufen am 28.06.2017

Format speichert. Über die folgenden Parameter kann das Aussehen der dargestellten Linie festgelegt werden.

Parameter:

- Line-Color A,R,G,B
- Point-Color A,R,G,B
- output-directory (full path or relative path)
- width of image
- height of image
- duration (in seconds; 0 = static image)
- latitude point1
- longitude point1
- latitude point2
- longitude point2
- draw background-image (boolean); background-image has to be named basemap.png in the same directory as OrthodromeGenerator.exe
- point-size (on the last position of the orthodrome, a point will be drawn if the size > 0)
- line-width (if width == 0, no line will be drawn)
- draw point every x positions
- Anschließend findet man hier drei Aufrufe, um ein Rasterbild im PNG-Format zu erzeugen (siehe Abb. 95).
- first draw "establishing connection"
 - example animation VIE - NY wo basemap / 3px single point / 1px line
 - OrthodromeGenerator.exe 255,100,204,255 255,240,245,245 .\out 4096 2048 5 48.20 16.366667 40.63975 -73.778925 false 3 1 0
- second draw finished connection line
 - example image VIE - NY wo basemap / no point / 1px line
 - OrthodromeGenerator.exe 255,100,204,255 255,240,245,245 .\out 4096 2048 0 48.20 16.366667 40.63975 -73.778925 false 0 1 0
- third rename resulting image of step 2 into "basemap.png" and draw the "packets" above

- example animation VIE - NY wo basemap / 5px repeating point / no line
- OrthodromeGenerator.exe 255,100,204,255 255,240,245,245 .\out 4096 2048 5 48.20 16.366667 40.63975 -73.778925 true 5 0 10



Abbildung 95 - Orthodrome nach Ablauf des Werkzeugs, Quelle: Eigene Darstellung

Ausgehend vom umzusetzenden Beispiel mit den Real Time-Flugdaten wird das entstandene Rasterbild nun als zweiter Layer in OmniSuite integriert und mit der aktuellen Position des Luftfahrzeuges kombiniert. Somit wird nach einem bestimmten Zeitintervall (44 Sekunden für 1 Pixel) (siehe Berechnung in Kap. 3.2) die Position des Luftfahrzeuges aktualisiert und die Daten wiederum neu in OmniSuite integriert. Natürlich kann das Flugzeug, aufgrund diverser Einflüsse, auch nicht entlang der Orthodrome fliegen. Ursachen sind unter anderem das Wetter (Gewitterzellen, etc.), rechtliche Luftraumsperrungen (Militärgebiete, etc.) und fehlende Überfluggenehmigungen abhängig der Airlines (z.B. taiwanische Airlines dürfen nicht über chinesisches Staatsgebiet fliegen).

Als nächster Bearbeitungsschritt sollen nun die vom Autor dieser Arbeit erstellten Python-Skripte ineinandergreifen und bei jedem Upload der Daten eines bestimmten Luftfahrzeuges bzw. einer bestimmten Flugnummer automatisch die Daten zur Orthodrome als PNG-Datei mitgeliefert werden.

8.3 Interner Aufruf von Daten

Dieses Kapitel gibt einen Überblick über eine Neuerung in OmniSuite. Die Flugdaten werden in einem Textfile in einen Systemordner von OmniSuite gespeichert und sobald OmniSuite eine Datei in diesem Ordner entdeckt, wird dieses Textfile automatisch geladen und die Koordinaten der aktuelle Position des abgefragten Luftfahrzeuges werden visualisiert.

Bevor die Daten automatisiert abgerufen werden können, muss eine Story in OmniSuite, die ein fertiges Textfile verwendet, erstellt werden. Das verwendete Textfile ist im Medienverzeichnis gespeichert. Aus dem Pfad des Medienverzeichnisses und dem Pfad der Text-Datei im Materialskript ergibt sich der Dateiname, dessen Inhalt überschrieben wird (oder die komplette Datei ersetzt). In der automatisch erstellten HTML-Datei zur Story wird ein Button mit der JavaScript-Funktion *reloadCurrentMaterials()* zuvor hinzugefügt (siehe Abb. 96). Dieser Button wird nach dem Aufruf der Story im OmniSuite Controller geklickt, wodurch das neueste Textfile wird geladen und ein Unterschied der Position am Hyperglobus sichtbar wird.

```
<div class="mybutton" onClick="window.external.reloadCurrentMaterials()" >Refresh Position</div>
```

Abbildung 96 - Button zur Aktualisierung im OmniSuite Controller, Quelle: Eigene Darstellung

Im Rahmen von Tests hat sich wiederum eine Verbesserung der oben beschriebenen Überwachung des Ziel-Ordners ergeben. Derzeit ist es möglich, durch einen automatisierten Abruf einer Flugnummer über das Python-Skript mit dem Namen „flugsuche2.py“ und die Speicherung des Textfiles immer unter dem gleichen Namen im vorgesehenen Material-Ordner (*[medienverzeichnis]/materials/textures/84/tests/punktdaten/xyz.txt*) von OmniSuite. Im HTML-Interface des OmniSuite Controller wird ein JavaScript-Timer (JS-Timer) hinzugefügt, die Funktion „*reloadCurrentMaterials*“ wird automatisch aufgerufen und der Flugpunkt bewegt sich dann alle 44 Sekunden weiter (Berechnung siehe Kap. 3.2). Der JS-Timer wird dem Button hinzugefügt (siehe Abb. 97), um beim erstmaligen Klicken des Buttons diesen Timer zu starten.

```
<div class="mybutton" >Start Timer</div>  
onClick="setInterval(function(){window.external.reloadCurrentMaterials();}, 44000)"
```

Abbildung 97 - Timer zur Aktualisierung in OmniSuite Controller, Quelle: Eigene Darstellung

Durch regelmäßiges Überschreiben der Textdatei und dem einmaligen Drücken des Start-Timer Buttons beim ersten Aufruf des OmniSuite Controllers ändert sich die aktuelle Position des Luftfahrzeuges automatisch nach dem eingestellten Zeitintervall. Beim Testen hat sich herausgestellt, dass eine Aktualisierungsrate alle 44 Sekunden (siehe Berechnung in Kap. 3.2) als zu kurz angesehen werden kann. Die Datenaktualisierungsrate von Seiten FlightRadar erfolgt zwar nach 44 Sekunden, nur ist die Änderung der Daten so Minimal, dass eine Weiterbewegung nicht erkennbar ist. Bei einer eingestellten Aktualisierungsrate von einer Minute ist eine Bewegung am sphärischen Display ohne Probleme sichtbar.

Zusätzlich kann in der HTML-Datei noch eine Zoom-Leiste mit einem Maximum von 400 % eingeführt werden (siehe Abb. 98).

```
<div>
  <p class="subHeader">Zoom Level:</p>
  <p class="bodyText">
    100%
    <input type="range" name="zoomFactor_de" min="100" max="400"
step="10" value="100"
onchange="window.external.setZoomLevel(zoomFactor_de.value, 90)">
    400%
  </p>
</div>
```

Abbildung 98 – Zoom-Toolbar, Quelle: Eigene Darstellung

In der Box unterhalb sieht man die Berechnung für die Weiterbewegung um 1 Pixel bei 400 %-Zoom. Die Ausbreitung von 1 Pixel am Äquator beträgt 2,4 km. Wenn die durchschnittliche Geschwindigkeit eines Langstreckenflugzeuges mit einbezogen wird, beträgt die Bewegung um 1 Pixel auf der Bildgröße 4096 x 2048 Pixel 11 Sekunden. Diese Aktualisierungsrate muss auf 20 Sekunden erhöht werden, um eine Weiterbewegung zu sehen.

Berechnung des Zeitintervalls bei 400%-Zoom:

Bildgröße: 4096 x 2048 Pixel

Äquatorumfang: $40075,017 / 4 = 10018,75$ km

Ausbreitung von 1 Pixel am Äquator: $10018,75 / 4096 = 2,4$ km

Ø Geschwindigkeit eines Langstreckenflugzeuges: 800 km/h = 13 km/Minute

Bewegung um 1 Pixel: 11 Sekunden

Das Ergebnis nach dem Uploaden von verschiedenen Flugnummern oder Flugzeugregistrierungen sieht am Hyperglobus wie folgt aus:



**Abbildung 99 - Darstellung der fliegenden Langstreckenflotte von Austrian Airlines,
Quelle: Eigene Darstellung**

Abbildung 99 zeigt den Upload der aktuellen Position von sieben verschiedenen Langstreckenflugzeugen (Boeing 767 und Boeing 777) von Austrian Airlines. Eine weitere Verbesserung wäre, die gesamte Flotte einer Airline anzuzeigen, um zu sehen, wo sich die Flugzeuge derzeit auf der Welt befinden. Um die diversen Flugzeugtypen einer Airline besser auseinander zu halten, kann beim Stylesheet mit unterschiedlichen Kategorien (Farben, Bilder, etc.) gearbeitet werden. Außerdem könnte man sich auf einen Flugzeugtyp beschränken, aber verschiedene Airlines mit unterschiedlichen Farben darstellen. Derzeit in Planung ist eine unsichtbare Box zu erstellen, in der oberhalb des Flugzeugsymbols Daten zum visualisierten Flug angezeigt werden, dies könnte neben der Flugnummer der Start- und Endflughafen sein. Leider ist es bei dieser Art von Datendownload in ein Textfile nicht möglich, Daten zur Geschwindigkeit und des Steuerkurses (heading) zu bekommen, diese Daten gibt es nur beim Upload eines fertig abgeschlossenen Fluges über das CSV- oder KML-Format (vgl. Abb. 53). Wenn der Steuerkurs vorhanden wäre, könnte man ebenfalls Kategorien mit Flugzeugsymbolen in den unterschiedlichen Himmelsrichtungen machen, um dem Nutzer eine weitere Information zu geben, in welche Himmelsrichtung das Luftfahrzeug fliegt.

9 Zusammenfassung und Ausblick

Im Rahmen dieser Arbeit wurden einige Methoden verbessert und viele neue Methoden entwickelt, um Real Time-Flugdaten nicht händisch erstellen zu müssen, sondern sie über Python-Skripte beziehungsweise über eine Abfrage auf einer Webseite als Textfile auszugeben und diese Textfiles über eine Ordnerstruktur von OmniSuite direkt zu laden.

Die Forschungsfrage „Wie lässt sich die Vektordatenintegration am sphärischen Display optimieren bzw. automatisieren?“ kann folgendermaßen beantwortet werden: Es ist möglich die Integration von Vektordaten zu optimieren beziehungsweise zu automatisieren. Ausgehend von der Diplomarbeit von Johannes Liem wurden die derzeit bestehenden Methoden der Vektordatenintegration evaluiert und mittels diverser Python-Skripte optimiert und teilweise automatisiert (siehe ab Kap. 7.3 ff). Dies führt dazu, dass Vektordaten schneller und effizienter verwendet werden können, um am sphärischen Display abgebildet zu werden. Außerdem schafft es die Möglichkeit Real Time-Daten, in meinem Fall Flugdaten, heranzuziehen und diese in Echtzeit darzustellen (siehe ab Kap. 6 ff). Außerdem kann darauf hingewiesen werden, dass auch einige Neuerungen hinsichtlich der Interaktivität neu umgesetzt wurden, welches wiederum zu einer Optimierung der Vektordatenintegration führt (siehe ab Kap.8).

Die beiden gestellten Arbeitsfragen „Wie unterscheiden sich die Methoden der Datengewinnung sowie -verarbeitung hinsichtlich des Zieldatenformats (Vektor versus Raster)?“ und „Wie können Verzerrungen bei der Visualisierung (Punkte, Linien) vermieden werden?“ können beide wie folgt beantwortet werden: Die erste Arbeitsfrage kann so erwidert werden, dass die Methoden der Datengewinnung und -verarbeitung bei Rasterdaten einfacher und rascher von statten gehen, da hier nur das anzuzeigende Rasterbild vorhanden sein muss, um am sphärischen Display abgebildet werden zu können. Bei den Vektordaten benötigt es durchaus komplexere Methoden, da hier das Rasterbild nur die Kartengrundlage darstellt und die Vektordaten lagerichtig auf dieser Grundlage verortet werden müssen, um korrekt am sphärischen Display angezeigt werden zu können (siehe ab Kap. 7 ff). Bei der zweiten Arbeitsfrage kann auf die mathematische Formel der Tissot'schen Indikatrix (siehe Kap. 5.3) verwiesen werden. Anhand dieser Formel kann der Verzerrungsfaktor gemäß der geographischen Breite berechnet werden, um wie viel die Ausgangssignatur in der Breite verzerrt werden muss, um korrekt am Hyperglobus abgebildet zu werden.

Anschließend möchte ich einen kurzen Überblick in die Zukunft geben, welches Potential im Bereich der Visualisierung am sphärischen Display vorhanden ist.

In Zukunft könnten Luftfahrtkarten als Hintergrund (Raster) dienen, um Luftfahrtstraßen zu sehen, dass mögliche direktere Flugrouten, durch Sperren von Lufträumen (military areas), eventuell nicht möglich sind und dies besser nachvollziehen zu können. Des Weiteren könnte eine Wetteranimation der Wolkenbedeckung mit spezieller Ausweisung von Gewitterbereichen und/oder Niederschlagsbereichen am Hyperglobus angezeigt werden, um notwendige Umwege des Luftfahrzeuges dem Anwender verständlich zu machen.

Nutzer solcher Hypergloben mit der Darstellung von Flugzeugpositionen könnten Airlines auf der ganzen Welt sein, die diese Globen als zusätzliches System in ihrem Hauptbüro verwenden, um zu sehen, wo sich die Flugzeuge der eigenen Flotte derzeit auf der Erde befinden. Hierbei müsste man sich ein Farbschema oder eine Darstellung der Flugnummern bei der Punktsignatur überlegen, da sonst nicht unterschieden werden kann, welcher Flug von A nach B geht (siehe Kap. 8.3.). Weiters könnte es möglich sein, haptisch auf einzelne Punktsignaturen mit der Hand zu klicken, um weitere Details zum Flug aufzurufen.

Eine weitere Möglichkeit für die Nutzung von Hypergloben wären die Flughäfen dieser Welt. Hier könnten alle ankommenden und abfliegenden Luftfahrzeuge des Tages animiert dargestellt werden. Hierbei würde es sich eher um ein Entertainmentprodukt handeln.

Flightradar24 hat bereits auf deren Homepage und Apps die zuvor erwähnten Erweiterungen (Wolkenbedeckung, Wetter, Niederschlag, Gewitter, etc.) integriert. Es könnte somit geprüft werden, ob Flightradar24 nicht mit einer Firma, welche Datenmaterialien für Hypergloben erstellt, zusammenarbeitet, um folglich eine neue Marktnische für den Bereich Luftfahrt besetzen zu können. Daraus würden sich nach Meinung des Verfassers einige Synergien ergeben, da die Datenaktualisierung und -aktualität noch schneller und besser funktionieren würde.

10 Literatur

- [AUS-16] AUSTROCONTROL (2016): Surveillance, Quelle: <https://www.austrocontrol.at/flugsicherung/technik/radarsurveillance>, abgerufen am 25.10.2016
- [BIL-96] BILL R. (1996): Grundlagen der Geo-Informationssysteme, Band 2, Wichmann Verlag, Heidelberg
- [BRA-o.J.] BRAUN S. (o.J.): Ausarbeitung zeit- und ereignisgesteuerte Echtzeitsysteme, Projektgruppe AUTOLAB der Universität Dortmund, Quelle: <https://ess.cs.tu-dortmund.de/Teaching/PGs/autolab/ausarbeitungen/Braun-Zeit-und-ereignisgesteuerte-Echtzeitsysteme-Ausarbeitung.pdf>, abgerufen am 28.06.2017
- [BRO-13] BROWN P., STEWART I., HAEMEL N., POOLEY A., RASMUS A. und SHAH M. (2013): EXT_texture_compression_s3tc, Quelle: https://www.khronos.org/registry/OpenGL/extensions/EXT/EXT_texture_compression_s3tc.txt, abgerufen am 20.09.2017
- [COZ-11] COZZI P. und RING K. (2011): 3D Engine Design for Virtual Globes, CRC Press, Taylor & Francis Group, Boca Raton
- [FLI-16a] FLIGHTRADAR24.COM (2016): About Flightradar24.com, Quelle: <https://www.flightradar24.com/about>, abgerufen am 25.10.2016
- [FLI-16b] FLIGHTRADAR24.COM (2016): How it works, Quelle: <https://www.flightradar24.com/how-it-works>, abgerufen am 25.10.2016
- [FLI-16c] FLIGHTRADAR24.COM (2016): Premium subscription plans, Quelle: <https://www.flightradar24.com/premium/?change=true>, abgerufen am 25.10.2016
- [FLI-16d] FLIGHTRADAR24.COM (2016): Playback of Austrian Airlines flight OS87, Quelle: <https://www.flightradar24.com/data/flights/os87/#b6a05da>, abgerufen am 25.10.2016
- [GEO-11] GI GEOINFORMATION GmbH (2011): ArcGIS 10 – das deutschsprachige Handbuch für ArcView und ArcEditor, Wichmann Verlag, Berlin und Offenbach
- [GLA-12] GLASER S. (2012): Der taktile Hyperglobus und seine Wirkung auf den Lernzuwachs am Beispiel von Schülerinnen und Schülern der Oberstufe - Diplomarbeit, Universität Wien, Wien
- [GLO-14-17] GLOBOCCCESS (2014-2017): OmniSuite 4.14-4.19 Manual
- [GLO-17a] GLOBOCCCESS (2017): <http://globoccccess.at/showroom>, abgerufen am 26.06.2017
- [HAK-o2] HAKE G., GRÜNREICH D. und MENG L. (2002): Kartographie – Visualisierung raum-zeitlicher Informationen, 8. Auflage, Walter de Gruyter, Berlin

-
- [HAR-09] HARCKE R. (2009): Geoinformationssysteme – Geodaten [Kapitel 4], Quelle: <http://romanharcke.de/geoinformationssysteme-geodaten-kapitel-4/>, abgerufen am 10.05.2017
- [HOF-14] HOFER N.I. (2014): Real Time-Visualisierung seismologischer Geodaten: Kartographische Methoden und Ansätze gezeigt auf einem taktilen Hyperglobus, Bachelorarbeit (nicht veröffentlicht), Universität Wien, Wien
- [HOL-12] HOLZAPFEL M: (2012): Interaktion mit Geodaten an einem multitouchfähigen taktilen Hyperglobus - Diplomarbeit, Universität Wien, Wien
- [IBM-o.J.] IBM (o.J.): Geografisches Koordinatensystem, Quelle: https://www.ibm.com/support/knowledgecenter/de/SSEPGG_9.5.0/com.ibm.db2.luw.spatial.topics.doc/doc/csb3022a.html, abgerufen am 20.09.2017
- [KAI-12] KAINZ W. (2012): Unterlagen zur Lehrveranstaltung „Räumliche Bezugssysteme“, Universität Wien
- [KOP-11] KOPETZ H. (2011): Real Time-Systems – Design Principles for Distributed Embedded Applications, 2. Auflage, Springer Verlag, New York, Dordrecht, Heidelberg, London
- [KRE-70] KRETSCHMER I. (1970): Zur Wahl der Netzentwürfe in der thematischen Kartographie-In: ARNBERGER E. (Hrsg.): Grundsatzfragen der Kartographie, Wien, Österreichischen Geographischen Gesellschaft (ÖGG), S. 150-168
- [KRI-12] KRISTEN J. (2012): Entwicklung von interaktiven kartographischen 3D-Applikationen am Beispiel eines taktilen Hyperglobus – Diplomarbeit, Universität Wien, Wien
- [LAM-11] LAMPERT A. (2011): Herausforderungen und Lösungsvarianten bei der Aufbereitung von Globenbildern als Basis für Texturen am taktilen Hyperglobus, Diplomarbeit, Universität Wien, Wien
- [LAN-13] DE LANGE N. (2013): Geoinformatik in Theorie und Praxis, Springer Spektrum, 3. Auflage, Berlin Heidelberg
- [LIE-12] LIEM J. (2012): Vektordaten am taktilen Hyperglobus – über Verfahren zur Aufbereitung und Visualisierung geographischer Vektordatensätze am sphärischen Display, Diplomarbeit, Universität Wien, Wien
- [MAY-07] MAYER M. (2007): Die Visualisierung von globalen Echtzeit-Geodaten auf einem taktilen Hyperglobus, gezeigt anhand der Thematik der Wetterdynamik – Diplomarbeit, Universität Wien, Wien
- [MOK-05] MOKRE J. (2005): Das Globenmuseum der Österreichischen Nationalbibliothek, Wien, Bibliophile Edition
-

-
- [NOA-17] NOAA (2017): <https://sos.noaa.gov/Datasets/live-programs.php>, abgerufen am 26.06.2017
- [RIC-10] RICHARDS, W. R.; O'BRIEN, K.; MILLER, D. C. (2010): New Air Traffic Surveillance Technology-In: Aero Quarterly 02/10, Quelle: http://www.boeing.com/commercial/aeromagazine/articles/qtr_02_10/pdfs/AERO_Q2-10_article02.pdf, abgerufen am 25.10.2016
- [RIE-00] RIEDL A. (2000): Virtuelle Globen in der Geovisualisierung – Untersuchungen zum Einsatz von Multimediatechniken in der Geopräsentation – Dissertation, Universität Wien, Wien
- [RIE-08] RIEDL A. (2008): Entwicklung und Perspektiven von taktilen Hypergloben – In: Mitteilungen der Österreichischen Geographischen Gesellschaft, Bd. 150. Wien, 2008, S. 339-356.
- [RIE-09] RIEDL A. (2009): Taktile Hypergloben – die nächste Stufe in der Globenevolution? – In: KRIZ K., KAINZ W. und RIEDL A. (Hrsg.): Geokommunikation im Umfeld der Geographie – Tagungsband zum Deutschen Geographentag 2009 in Wien, Wiener Schriften zur Geographie und Kartographie, Band 19, S. 176-183
- [RIE-10] RIEDL A. und KRISTEN J. (2010): Der Einsatz sphärischer Displays zur Visualisierung globaler Phänomene – In: Kartographische Nachrichten, 60. Jg., Heft 3/2010, S. 129-137.
- [RIE-11a] RIEDL A. (2011): Entwicklungsgeschichte digitaler Globen – In: Der Globusfreund 57/58, Wien, Internat. Coronelli-Ges. für Globen- u. Instrumentenkunde 2011, S. 153-166.
- [RIE-11b] RIEDL A. (2011): Der Globus ist tot, es lebe der Globus! – In: Kriz K., Kainz W., Riedl A. (Hrsg.): 50 Jahre Österreichische Kartographische Kommission. Wien, Institut für Geographie der Universität Wien, 2011 (=Wiener Schriften zur Geographie und Kartographie, Band 20) S. 79 - 87.
- [RIE-12] RIEDL A. (2012): Skriptum zur Einführung in die Geoinformation (EGI), Universität Wien, Sommersemester 2012
- [SCH-17] SCHUPPAR B. (2017): Geometrie auf der Kugel - Alltägliche Phänomene rund um Erde und Himmel, Springer Verlag, Berlin, Heidelberg
- [SCH-00] SCHUMANN H. und MÜLLER W. (2000): Visualisierung – Grundlagen und allgemeine Methoden, Springer Verlag, Berlin, Heidelberg, New York
- [SEI-17] SEIP C., KORDUAN P. und ZEHNER M.L. (2017): Web-GIS – Grundlagen, Anwendungen und Implementierungsbeispiele, Wichmann Verlag, Berlin, Offenbach
- [SNY-93] SNYDER J.P. (1993): Flattening the Earth. Two Thousand Years of Map Projections, The University of Chicago Press, Chicago

[SVA-16] SVATEK P. (2016): Unterlagen zur LV „Geschichte der Kartographie“, Universität Wien, WS 2016/2017

[WIN-12] WINTNER S. (2012): Digital Storytelling angewandt auf einen taktilen Hyperglobus - Diplomarbeit, Universität Wien, Wien

11 Anhang

11.1 server_new.py

```
#!/usr/bin/python
from BaseHTTPServer import BaseHTTPRequestHandler,HTTPServer
from os import curdir, sep
import cgi
import urllib2
import json
import datetime

PORT_NUMBER = 8080

#This class will handles any incoming request from the browser
class myHandler(BaseHTTPRequestHandler):

    def abfrage(self, flieger): #Funktion definieren
        print(flieger)

link='https://www.flightradar24.com/v1/search/web/find?query={fliegerreg}&l
imit=1'
    htmlheaders = { 'User-Agent' : 'Mozilla/5.0' }
    lineformat='{0} {1} {2} {3} {4} {5}\n'

    #Request des Links + auslesen
    req = urllib2.Request(link.format(fliegerreg=flieger), None,
htmlheaders)
    html = urllib2.urlopen(req).read()

    #JSON auslesen und LAT und LON ausgeben
    myjson = json.loads(html)
    flightid=myjson['results'][0]['id']
    number=myjson['results'][0]['label']

    lat=None
    lon=None
    origin=None
    destination=None
    reg=None
    flightnumber=None

    if flightid not in number:
        lat= myjson['results'][0]['detail']['lat']
        lon= myjson['results'][0]['detail']['lon']
        try:
            origin= myjson['results'][0]['detail']['schd_from']
            destination= myjson['results'][0]['detail']['schd_to']
        except:
            pass
        reg= myjson['results'][0]['detail']['reg']
        flightnumber=myjson['results'][0]['detail']['flight']#greifen
auf Daten zu um diese ins Txt zu schreiben (ob vorhanden ist)
```

```

newline=lineformat.format(lat,lon,origin,destination,reg,flightnumber)
#name vom neuen String

    return newline

def schreibe(self, liste):
    timeformat='{:Y_m_d_H_M_S}'

    fileheader='lat long origin destination reg flightnumber\n'
    textfile='output/{0}.txt'

    #erstellen TXT-File, oeffnen es, schreiben lat, lon hinein und
    schliessen es
    timestamp=timeformat.format(datetime.datetime.now()) #zeitstempel
    filename=textfile.format(timestamp)
    f=open(filename, 'w+')
    f.write(fileheader)
    f.writelines(liste)
    f.close()

    return filename

#Handler for the GET requests
def do_GET(self):
    if self.path=="":
        self.path="/index.html"

    try:
        #Check the file extension required and set the right mime type

        sendReply = True
        mimetype='text/html'

        if sendReply == True:
            #Open the static file requested and send it
            f = open(curdir + sep + self.path)
            self.send_response(200)
            self.send_header('Content-type',mimetype)
            self.end_headers()
            self.wfile.write(f.read())
            f.close()
        return

    except IOError:
        self.send_error(404,'File Not Found: %s' % self.path)

#Handler for the POST requests
def do_POST(self):
    if self.path=="/send":
        form = cgi.FieldStorage(
            fp=self.rfile,
            headers=self.headers,
            environ={'REQUEST_METHOD':'POST',
                    'CONTENT_TYPE':self.headers['Content-Type'],
                })

        flugnummer = form["txt"].value
        out_txt = self.abfrage(flugnummer)
        filename = self.schreibe(out_txt)

```

```
        print "Your name is: %s" % flugnummer
        self.send_response(200)
        self.end_headers()
        self.wfile.write("In Ihrem Output-Ordner befinden sich die
Daten für %s !" % flugnummer)
        return

try:
    #Create a web server and define the handler to manage the incoming
request
    server = HTTPServer(('', PORT_NUMBER), myHandler)
    print 'Started httpserver on port ' , PORT_NUMBER

    #Wait forever for incoming http requests
    server.serve_forever()

except KeyboardInterrupt:
    print '^C received, shutting down the web server'
    server.socket.close()
```

11.2 schleife.py

```

# Import system modules
import arcpy, os
arcpy.env.overwriteOutput = True

# Set local variables
output_path = r"C:\Users\austr\Dropbox\Masterarbeit\Pythonsachen\output"
path_pythonsachen = r"C:\Users\austr\Dropbox\Masterarbeit\Pythonsachen"
path_mxd =
r"C:\Users\austr\Dropbox\Masterarbeit\Pythonsachen\test\flug_new.mxd"
symbology = path_pythonsachen + os.sep + r"test\symbology.lyr"
out_png_path = path_pythonsachen + os.sep + r"test\png"

arcpy.env.workspace = output_path

n=1
i=0

while n==1:
    try:
        dataset = arcpy.ListFiles()
        if dataset[i]:
            print dataset[i] + str(i)

            arcpy.env.workspace = output_path
            #dataset = arcpy.ListFiles() # remove

            output_txt = output_path + os.sep + dataset[i]

            fieldName0 = "lat"
            fieldName1 = "long"
            fieldName2 = "origin"
            fieldName3 = "destination"
            fieldName4 = "reg"
            fieldName5 = "flightnumber"

            # Set the local variables
            x_coords = fieldName1
            y_coords = fieldName0

            #z_coords = "POINT_Z"
            out_Layer = "flug"
            saved_Layer = path_pythonsachen + os.sep + r"test\flug" +
str(i) + ".lyr"

            # Set the spatial reference
            spRef = path_pythonsachen + os.sep + r"test\welt_prj1.shp"

            # Make the XY event layer...
            arcpy.MakeXYEventLayer_management(output_txt, x_coords,
y_coords, out_Layer, spRef)

            # Save to a layer file
            arcpy.SaveToLayerFile_management(out_Layer, saved_Layer)

            # Add layer to mxd
            mxd = arcpy.mapping.MapDocument(path_mxd)
            dataFrame = arcpy.mapping.ListDataFrames(mxd, "*")[0]
            addLayer = arcpy.mapping.Layer(saved_Layer)

```

```
arcpy.mapping.AddLayer(dataFrame, addLayer)

# Apply Symbology
lyr = arcpy.mapping.ListLayers(mxd)[0]
arcpy.ApplySymbologyFromLayer_management(lyr, symbology)

# Save as png
out_png = out_png_path + os.sep + "flug" + str(i) + ".png"
arcpy.mapping.ExportToPNG(mxd, out_png, dataFrame,
                          df_export_width=4096,
                          df_export_height=2048,
                          world_file=True,
                          transparent_color="255, 255, 255")

    i=i+1
except:
    print i
```

11.3 flugsuche2.py

```

import sched, time
import urllib2
import json
import datetime

#Variablen definieren
link='https://www.flightradar24.com/v1/search/web/find?query={fliegerreg}&limit=1'
htmlheaders = { 'User-Agent' : 'Mozilla/5.0' }
lineformat='{0} {1} {2}'
fileheader='lat long flightnumber\n'
textfile=
r"C:\Users\austr\Dropbox\Masterarbeit\OmniSuite\media_punktdate_n_aktualisieren\materials\textures\84\tests\punktdate_n\{0}.txt"
fliegerreg=['ACA844']oder diverse Flugzeug-Registrierungen (OE-LPA)

s = sched.scheduler(time.time, time.sleep)
def repeat_code(sc):

    def abfrage(flieger): #Funktion definieren
        print(flieger)

        #Request des Links + auslesen
        req = urllib2.Request(link.format(fliegerreg=flieger), None,
htmlheaders)
        html = urllib2.urlopen(req).read()

        #JSON auslesen und LAT und LON ausgeben
        myjson = json.loads(html)
        flightid=myjson['results'][0]['id']
        number=myjson['results'][0]['label']

        lat=None
        lon=None
        flightnumber=None

        if flightid not in number:

            lat= myjson['results'][0]['detail']['lat']
            lon= myjson['results'][0]['detail']['lon']
            flightnumber=myjson['results'][0]['detail']['flight'] #greifen
auf Daten zu um diese ins Txt zu schreiben (ob vorhanden ist)

        newline=lineformat.format(lat,lon, flightnumber) #name vom neuen
String

        #erstellen TXT-File, oeffnen es, schreiben lat, lon hinein und
schliessen es
        filename=textfile.format(flieger)
        f=open(filename, 'w+')
        f.write(fileheader)
        f.write(newline)
        f.close()

        #print newline
        #print filename

```

```
    return True

for i in fliegerreg:
    abfrage(i) #Funktion ausgeben

    s.enter(300, 1, repeat_code, (sc,)) # here enter interval in seconds
    (currently: every 44 seconds)

s.enter(1, 1, repeat_code, (s,))
s.run()
```

11.4 orthodrome.py

```

# Import system modules
import arcpy, os, sys
from arcpy import env

arcpy.env.overwriteOutput = True

# Define input_table_airport
ws = r"C:\Users\austr\Dropbox\Masterarbeit\Pythonsachen"
arcpy.env.workspace = ws + os.sep + "output"
datasets = arcpy.ListFiles()
for dataset in datasets:
    print dataset

# Set local variables
input_table_airport = ws + os.sep + "output" + os.sep + dataset
input_table_coord = ws + os.sep + "airports2.csv"
out_table_path = ws + os.sep + r"test\flug.gdb"
out_name = "coordinates"
out_table = ws + os.sep + r"test\flug.gdb" + os.sep + out_name
out_lines = ws + os.sep + r"test\flug.gdb\flug"
saved_Layer = ws + os.sep + r"test\orthodrome.lyr"
path_mxd = ws + os.sep + r"test\flug.mxd"
out_png = ws + os.sep + r"test\png\orthodrome.png"

#-----Get Coordinates-----
-#

# Get origin and destination airport
SC = arcpy.SearchCursor(input_table_airport)
origin_field = "origin"
destination_field = "destination"
for row in SC:
    origin = row.getValue(origin_field)
    destination = row.getValue(destination_field)

print origin
print destination

# Get coordinates
fc = input_table_coord
fields = origin, destination
values_orig = [row[0] for row in arcpy.da.SearchCursor(fc, fields)]
values_dest = [row[1] for row in arcpy.da.SearchCursor(fc, fields)]
origin_x = values_orig[0]
origin_y = values_orig[1]
dest_x = values_dest[0]
dest_y = values_dest[1]

#-----Write coordinates into table-----
-#

# Create table
arcpy.CreateTable_management(out_table_path, out_name)
arcpy.AddField_management(out_table, "startx_field", "DOUBLE")
arcpy.AddField_management(out_table, "starty_field", "DOUBLE")
arcpy.AddField_management(out_table, "endx_field", "DOUBLE")
arcpy.AddField_management(out_table, "endy_field", "DOUBLE")

```

```
# Insert row into table
rows = arcpy.InsertCursor(out_table)
row = rows.newRow()
rows.insertRow(row)

# Delete cursor and row objects to remove locks on the data
del row
del rows

# Write coordinates into table
arcpy.CalculateField_management(out_table, "startx_field",
"{0}".format(origin_x))
arcpy.CalculateField_management(out_table, "starty_field",
"{0}".format(origin_y))
arcpy.CalculateField_management(out_table, "endx_field",
"{0}".format(dest_x))
arcpy.CalculateField_management(out_table, "endy_field",
"{0}".format(dest_y))

#-----Create orthodrome-----
-#

#XY To Line
arcpy.XYToLine_management(out_table,out_lines,
                          "startx_field","starty_field","endx_field",
                          "endy_field","GREAT_CIRCLE")

#-----Save as png-----
-#

# Make feature layer
arcpy.MakeFeatureLayer_management (out_lines, "orthodrome")

# Save orthodrome to layer file
arcpy.SaveToLayerFile_management ("orthodrome", saved_Layer)

# Add layer to mxd
mxd = arcpy.mapping.MapDocument (path_mxd)
dataFrame = arcpy.mapping.ListDataFrames (mxd, "*") [0]
addLayer = arcpy.mapping.Layer (saved_Layer)
arcpy.mapping.AddLayer (dataFrame, addLayer)

# Save as png
arcpy.mapping.ExportToPNG (mxd, out_png, dataFrame,
                           df_export_width=5000,
                           df_export_height=2200,
                           world_file=True,
                           transparent_color="255, 255, 255")
```