universität wien

MASTERARBEIT / MASTER'S THESIS

Titel der Masterarbeit / Title of the Master's Thesis

„Geovisualisierung von Partikelsysteme auf Basis von Vektorfeldern am sphärischen Display veranschaulicht am Beispiel der globalen Windverhältnisse“

verfasst von / submitted by

Christian Rauh

angestrebter akademischer Grad / in partial fulfilment of the requirements for the degree of

Master of Science (MSc)

Wien, 2021 / Vienna 2021

Studienkennzahl lt. Studienblatt / degree programme code as it appears on the student record sheet:	UA 066 856
Studienrichtung lt. Studienblatt / degree programme as it appears on the student record sheet:	Masterstudium Kartographie und Geoinformation
Betreut von / Supervisor:	Ass.-Prof. Mag. Dr. Andreas Riedl

Inhaltsverzeichnis

Inhaltsverzeichnis ii

Abbildungsverzeichnis iv

Tabellenverzeichnis v

Abstract vi

1. Einleitung 1

1.1 Problemstellung und Zielsetzung 2

1.3 Wissenschaftliche Fragestellung 3

1.2 Aufbau der Arbeit 4

2. Globen 5

2.1 Der Begriff Globus 5

2.2 Analoge Globen 7

2.2.1 Definition 7

2.2.2 Historischer Abriss und Bedeutung 8

2.3 Hypergloben 9

2.3.1 Virtueller Hypergloben 9

2.3.2 Taktile Hypergloben 11

2.3.2.1 Entwicklung von taktilen Hypergloben 12

2.3.2.2 Projektionsverfahren 14

2.3.3 Hologloben 16

3. (Geo)visualisierung 17

3.1 Visualisierung 17

3.1.1 Der Begriff Visualisierung 18

3.1.2 Bedeutung und Nutzen 19

3.1.3 Historischer Blick auf Visualisierung 20

3.2 Geovisualisierung 22

3.2.1 Beschreibung und Überblick 22

3.2.2 Bedeutung von Geovisualisierung 25

3.2.3 Historische Entwicklung 27

3.2.4 Statische und Dynamische Geovisualisierung 28

3.3 Zwei und Dreidimensionale Visualisierung 34

4. Vektorfelder 36

4.1 Definition 36

4.2 Darstellung von Vektorfeldern 38

4.3 Vektorfelder zur Geovisualisierung 41

5. Partikelsysteme 44

5.1 Definition und Hintergrund 44
5.2 Funktionsweise 46
5.3 Bestandteile von Partikelsystemen 49
5.3.1 Emitter 49
5.3.2 Generation Shape 50
5.3.3 Modifikatoren 50
5.4 Partikel 51
5.4.1 Position 51
5.4.2 Geschwindigkeit und Richtung 52
5.4.3 visuelle Erscheinung 52
5.4.4 Lebenszeit 53
5.5 Partikelsysteme zur Geovisualisierung 53
6.Umsetzung von Partikelsystemen auf Globen 56
6.1 Ausgangsdaten 56
6.1.1 Globenwürdigkeit 56
6.1.2 Herkunft der Daten 57
6.1.3 Beschreibung der Daten 58
6.1.4 Bearbeitung der Daten 59
6.2 Allgemeine Methodik 61
6.3 Taktiler Globus 64
6.3.1 Ein und Ausgabe am taktilen Globus 64
6.3.2 Dynamische Darstellung auf dem taktilen Globus 67
6.3.3 Anforderungen an ein Partikelsystem am taktilen Globus 68
6.3.4 Implementierung 72
6.3.4.1 Allgemeiner Aufbau des Partikelsystems 73
6.3.4.2 Der Emitter 75
6.3.4.3 Die Klasse Partikel 76
6.3.4.4 Initialisierung der Visualisierung 82
6.3.4.5 Ablauf der Animation – die Funktion animate 83
6.3.5 Ergebnisse 85
6.4 virtueller Hyperglobus 88
6.4.1 Anforderungen an ein Partikelsystem 88
6.4.2 Möglichkeiten zur Implementierung 91
6.4.3 Implementierung 92
6.4.3.1 Aufbau der Website 92
6.4.3.2 Initialisierung der Szene 93
6.4.3.4 Koordinatentransformation 96
6.4.3.5 Partikel 98
6.4.3.6 Emitter 98
6.4.3.7 Die init Funktion – Vorbereitung der Visualisierung 100
6.4.3.8 Ablauf der Visualisierung 100
6.4.4 Ergebnisse 103
7 Zusammenfassung und Diskussion der Ergebnisse 106
8 Ausblick 109
Literatur 111
Onlinequellen 116
Anhang 117

Abbildungsverzeichnis

Abbildung 1 Wind dargestellt mittels eines Partikelsystems. Quelle: www.windy.com 2
Abbildung 2 Innenprojektionen. links: Fischaugensystem, rechts: Spiegelsystem. Quelle: Riedl, 2008 15
Abbildung 3 Der Map-Use Cube veranschaulicht die Ziele der Geovisualisierung Quelle: MacEachren 1994 23
Abbildung 4 Linking und Brushing zwischen einem Scatter-Plot einer Karte und einem Parallelen Koordinatenplot. Quelle: Nöllenburg 2006 24
Abbildung 5 "Cholera Map" von John Snow aus dem Jahr 1854. Quelle: Nöllenburg 2006 26
Abbildung 6 Entwicklung der Disziplinen im Umfeld der Geovisualisierung. Quelle: Kraak 2008 28
Abbildung 7 graphische Variablen nach Bertin. Eingeteilt nach Verwendungsmöglichkeit der Skalenniveaus. Quelle: MacEachren 1995 30
Abbildung 8 Minards Visualisierung über den Russlandfeldzug 1812. Dargestellt sind Position und Anzahl der Truppen sowie die Temperaturen. Quelle: commons.wikimedia.org 30
Abbildung 9 Dynamische Variablen. Basierend auf MacEachren, 1995 33
Abbildung 10 Zwei- und Dreidimensionale Darstellung eines Erdrutsches. Quelle: Wahyudi et al. 2020 35
Abbildung 11 Beispielhafte Darstellung eines zweidimensionalen Vektorfeldes. Eigene Darstellung 36
Abbildung 12 Kritische Punkte in einem zweidimensionalen Vektorfeld. Quelle: nach Helmann&Hesselink, 1989 37
Abbildung 13 Unterschiedliche Möglichkeiten Vektorfelder statisch zu visualisieren. Nach Laidlaw et al., 2005 .38
Abbildung 14 Vergleich zwischen LIC (links) und Spot Noise (rechts). Quelle: De Leeuw&Van Liere,1998 39
Abbildung 15 Verbesserte Darstellung eines Vektorfeldes durch Strömungslinienoptimierung. Quelle: Turk&Banks, 1996 40
Abbildung 16 Vektorfeld, welches die Windströmungen über Australien anzeigt. Quelle: Australian Bureau of Meteorology 41
Abbildung 17 Das gleiche Vektorfeld aus Abbildung 10 symbolisiert mit Barben. Quelle: Australian Bureau of Meteorology 42
Abbildung 18 Explosion visualisiert durch ein Partikelsystem, Quelle: Reeves 1983 44
Abbildung 19 Waldszene generiert mit einem stochastischen Partikelsystem. Quelle: Reeves&Blau, 1985 48
Abbildung 20 animiertes Partikelsystem. Rechts: statisches Partikelsystem. Quelle: wikimedia.org 48
Abbildung 21 Oben: Attraktor - Die Partikel werden in das Zentrum des Quaders gelenkt. Unten: Turbolenzelement - Partikel werden von ihrer Bahn abgelenkt. Quelle. Ferschin, 1999 51
Abbildung 22 Partikel mit einem zufällig bestimmten Austrittswinkel entlassen. Quelle: Reeves,1983 52
Abbildung 23 Vulkanausbruch visualisiert mit einem Partikelsystem. Quelle: Huber 2012 53
Abbildung 24 Darstellung atmosphärische Strömungen durch Partikel auf derm Projekt earch von Baccario. Quelle: earth.nullschool.net 55
Abbildung 25 Die horizontalen Geschwindigkeiten des Windes dargestellt in graustufen. Positive Werte beziehen sich auf Windrichtung West nach Ost. Quelle: eigene Abbildung, Datenquelle: National Center Of Environmental Information 59
Abbildung 26 Darstellung einer Grib Datei welche u- und v- Komponenten des Windes beinhaltet. Quelle: Darstellungssoftware: XyGrib (opengribs.com), Datenquelle: National Center Of Environmental Information 60
Abbildung 27 Ausschnitt der Eingabedatei der Partikelsysteme am Globus. Quelle: eigene Abbildung 61
Abbildung 28 Schematische Darstellung einer Flugbahn eines Partikels, dass den Nordpol überfliegt. Quelle: eigene Abbildung 62
Abbildung 29 Schematische Bewegung eines Partikels durch ein Vektorfeld. Quelle: eigene Abbildung 63
Abbildung 30 Benutzeroberfläche für ein Städtequiz mit dem taktilen Globus. Quelle: Smutny 2017 66
Abbildung 31 unbearbeitete Punktsignaturen auf einer Plattkarte und als Globenansicht. Quelle: Liem, 2012 ...69
Abbildung 32 Punktsignaturen mit Verzerrung nach tissotsche Indikatrix. Quelle: Liem, 2012 70
Abbildung 33 Die Klasse Emitter. Hier werden Partikel erstellt und gelöscht. Quelle: eigene Abbildung 75
Abbildung 34 Der Konstruktor der Klasse ‚Particle'.Quelle: eigene Abbildung 77
Abbildung 35 Die Methode getNewPosition. Sie berechnet die neue Position der Partikel. Dabei wird die aktuelle Position mit dem Vektor an dieser Stelle verrechnet. Quelle: eigene Abbildung 78
Abbildung 36 Die Funktion edges. Hier wird der Übertritt der Partikel über die Datumsgrenze und über die Pole überprüft. Quelle: eigene Abbildung 80
Abbildung 37 Die Methode draw, hier werden die Partikel auf dem Canvas gezeichnet. Quelle: eigene Abbildung 81

Abbildung 38 Die Funktion 'init'. Hier wird die Animation gestartet. Quelle: eigene Abbildung.....82
Abbildung 39 Der Kern der Funktion 'animate'. Hier wird jedes Partikel durchlaufen und die Lebenszeit geprüft um es anschließend an einer neuen Position zu zeichnen. Quelle: eigene Abbildung 83
Abbildung 40 Ablaufplan der Visualisierung. Quelle: eigene Darstellung 84
Abbildung 41 Die Visualisierung eines Partikelsystems angepasst an ein sphärisches Display. Partikel werden ihrer Lage auf der Plattkarte entsprechend verzerrt. Quelle: eigene Abbildung 85
Abbildung 42 Gegenüberstellung des Partikelsystems auf einer Kugel und auf einer flachen Karte. Quelle: eigene Abbildung 86
Abbildung 43 Übertritt der Partikel über den 180. Längengrad. Quelle: eigene Abbildung.....86
Abbildung 44 Vergleich der Anwendung (links) mit der Visualisierung von windy (rechts) der atmosphärischen Strömungen am 29.06.2021. Quelle: links: eigene Abbildung, rechts: windy.com 87
Abbildung 45 Gegenüberstellung der Visualisierung (links) mit dem Projekt "Earth" (rechts). Quelle: links: eigene Abbildung, rechts: earth.nullschool.net 88
Abbildung 46 Übersicht über Kugel- und Kartesische Koordinaten. 90
Abbildung 47 Einstellung der Szene, des Lichts, der Kamera und des Renderers. Quelle: eigene Abbildung.....93
Abbildung 48 Parameter der Kamera. Quelle: eigene Darstellung 94
Abbildung 49 Verschiedene Möglichkeiten von Lichtquellen. Quelle: Intexsoft (https://ichi.pro/de/eine-einfuhrung-in-die-3d-webentwicklung-217424846026555) 94
Abbildung 50 Die Erschaffung des Globus. Quelle: eigene Darstellung 95
Abbildung 51 Geometrie und Material der Partikel. Quelle: eigene Abbildung 95
Abbildung 52 Die Funktion XYZ_to_LL berechnet Breiten- und Längengrad ausgehend von einer dreidimensionalen Kartesischen Koordinate. Quelle: eigene Abbildung 97
Abbildung 53 Berechnung der kartesischen Koordinaten aus Breiten- und Längengrad. Quelle: eigene Abbildung 97
Abbildung 54 Prinzip der Speicherung von Eigenschaften der Partikel in zwei verschieden Arrays. Quelle: eigene Abbildung 98
Abbildung 55 die Methode initialiseParticles. Hier werden die Partikel des Partikelsystems zufällig auf dem Globus vereilt und ihnen ihre Eigenschaften (Lebenszeit und Position) zugeschrieben. Quelle: eigene Abbildung 99
Abbildung 56 Die Funktion animte. Sie steuert den Ablauf der Animation. Quelle: eigene Abbildung.....100
Abbildung 57 Abziehen der Lebenszeit und Ermittlung der Position auf dem Globus. Anschließend werden die Windkomponenten neu berechnet. Abbildung: eigene Abbildung.....102
Abbildung 58 Screenshot des virtuelllen Hyperglobus. Quelle: eigene Abbildung 103
Abbildung 59 Vergleich der Partikelsysteme des Projektes „Earth“ (links) und dem Partikelsystems des virtuellen Hyperglobes (rechts) am 25.06.2021 um 06:00 Uhr. Quelle: links: earth.nullschool.net, rechts: eigene Darstellung.....104
Abbildung 60 Partikelanhäufung in Strömungskonvergenzen. Quelle: eigene Abbildung 105

Tabellenverzeichnis

Tabelle 1 Kategorien von Hypergloben (nach Riedl, 2010) 9

Kurzfassung

Die vorliegende Arbeit befasst sich mit der Problematik der dynamischen Darstellung von Informationen an einem taktilen und einem virtuellen Hyperglobus. Speziell wurden Partikelsysteme entwickelt, welche auf Basis eines Vektorfeldes globale Winde darstellen. Die Visualisierungen müssen bestimmte Voraussetzungen erfüllen, damit eine unverzerrte interaktive Darstellung der Partikelsysteme auf den Globen möglich ist. Diese wurden durch Literaturrecherche vorangegangener Untersuchungen an Globen zusammengetragen und letztendlich bei der Programmierung der Systeme berücksichtigt. Der praktischen Umsetzung dieser Arbeit ist ein theoretischer Teil vorrausgegangen, der sich mit den Grundlagen von Globen, Visualisierung, Vektorfeldern und Partikelsystemen auseinandersetzt. Neben allgemeinen Beschreibungen und kurzen Blicken auf die Entwicklung, werden dem Leser die Bedeutungen der Teilgebiete dargelegt. Die Ergebnisse der Arbeit stellen zwei Webseiten dar, auf welchen jeweils ein Partikelsystem für den taktilen beziehungsweise für den virtuellen Hyperglobus zu sehen ist. Mit den Visualisierungen kann während der Laufzeit durch Anwender interagiert werden.

Abstract

The present work dealt with the problem of the dynamic representation of information on a tactile and a virtual hyperglobe. In particular, particle systems were developed which represent global winds on the basis of a vector field. The visualizations had to fulfill certain requirements, in order to allow an undistorted interactive representation of the particle systems on the globes. These requirements were collected based on literature research of previous investigations on globes and eventually taken into account in the programming of the systems. The practical implementation of this work is based on a theoretical part, which deals with the basics of globes, visualization, vector fields and particle systems. In addition to general descriptions and insights into the evolution of these fields, the importance of each field is explained to the reader. The results of the work are two web pages, each are showing a particle system for the tactile and the virtual hyperglobe, respectively. The visualizations can be interacted with by users during runtime.

1. Einleitung

Visualisierungen bieten Möglichkeiten Daten optisch erfassbar zu machen. Abstrakte Sachverhalte und komplexe Zusammenhänge erschließen sich uns einfacher, wenn sie durch Abbildungen oder Graphiken visuell nachvollziehbar aufbereitet werden. Wenn es dabei um globale Geschehnisse geht, werden uns zumeist Karten präsentiert, auf denen die Inhalte gezeigt werden. Doch sind nur Globen in der Lage, die Zusammenhänge tatsächlich unverzerrt darzustellen. Die Visualisierung von Informationen auf Globen ist daher ein bedeutendes Thema, welches in der Arbeitsgruppe *Kartographie und Geoinformation* der *Universität Wien* kontinuierlich bearbeitet wird.

Auch diese Arbeit wird sich mit Visualisierung an Globen beschäftigen. Im Detail sollen Partikelsysteme erstellt werden, die globale Windverhältnisse auf Hypergloben zeigen. Partikelsysteme, die vor allem aus der Film- und Videospielindustrie bekannt sind, um dort Explosionen, Feuer und andere Objekte ohne Scharfe Grenzen darzustellen, haben vielfältige Einsatzmöglichkeiten. Neben der Animation zur Unterhaltung dienen sie beispielsweise zur Simulation des Luftverkehrs zur Detektion von möglichen Kollisionen oder der Überprüfung von Luftströmungen in unterirdischen Bergwerksstollen (*vgl. Kapitel 5 Partikelsysteme*). In der vorliegenden Arbeit dienen Partikelsysteme zur Visualisierung eines Vektorfeldes, das die horizontalen Winde in der Atmosphäre beschreibt. Darunter ist ein zweidimensionales Raster zu verstehen, welches pro Zelle Richtung und Geschwindigkeit der lokalen Strömung angibt.

Globen haben aufgrund ihrer sphärischen Oberfläche spezielle Anforderungen an Visualisierungen. Daher stützt sich diese Arbeit zum Teil auf bereits vorrangegangene Untersuchungen zur Darstellung von Inhalten am taktilen Globus, welche ebenfalls an der *Universität Wien* veröffentlich wurden. Dazu gehören vor allem die Arbeiten von *Liem* (2012), *Smutny* (2017) und *Kordasch* (2019). Hauptsächlich wurde dabei die Darstellung von Vektordaten am sphärischen Display thematisiert. Auch wenn in der vorliegenden Arbeit zum großen Teil mit der Visualisierung von rasterbasierten Medien gearbeitet wurde, konnten aus vorherigen Untersuchungen Anforderungen an das sphärische Display zusammengetragen werden. Neben dem taktilen wird zudem ein Partikelsystem für den virtuellen Globus erarbeitet. Auch eine Visualisierung an diesem Globus muss bestimmte Voraussetzungen erfüllen, welche zum gegebenen Zeitpunkt thematisiert werden.

1.1 Problemstellung und Zielsetzung

Ziel der Arbeit soll es sein, atmosphärische Strömungen mithilfe eines Partikelsystems auf einem virtuellen und einem taktilen Hyperglobus dazustellen. Als Grundlage für die Darstellung der Strömungen soll ein Vektorfeld dienen, welches die Richtungen und Geschwindigkeiten der globalen Windverhältnisse beinhaltet. Globale Darstellungen der atmosphärischen Luftströmung ist dabei keine Neuheit. Verschiedene Webseiten für Wettervorhersagen bieten bereits Tools an, die den Wind einer bestimmten Region mittels eines Partikelsystems darstellen. Dies geschieht jedoch auf einer planaren Karte. Abbildung 1 zeigt ein Beispiel der Website windy.com. Der Wind wird auf einer animierten Karte durch Partikel, hier in Weiß zu erkennen, symbolisiert.

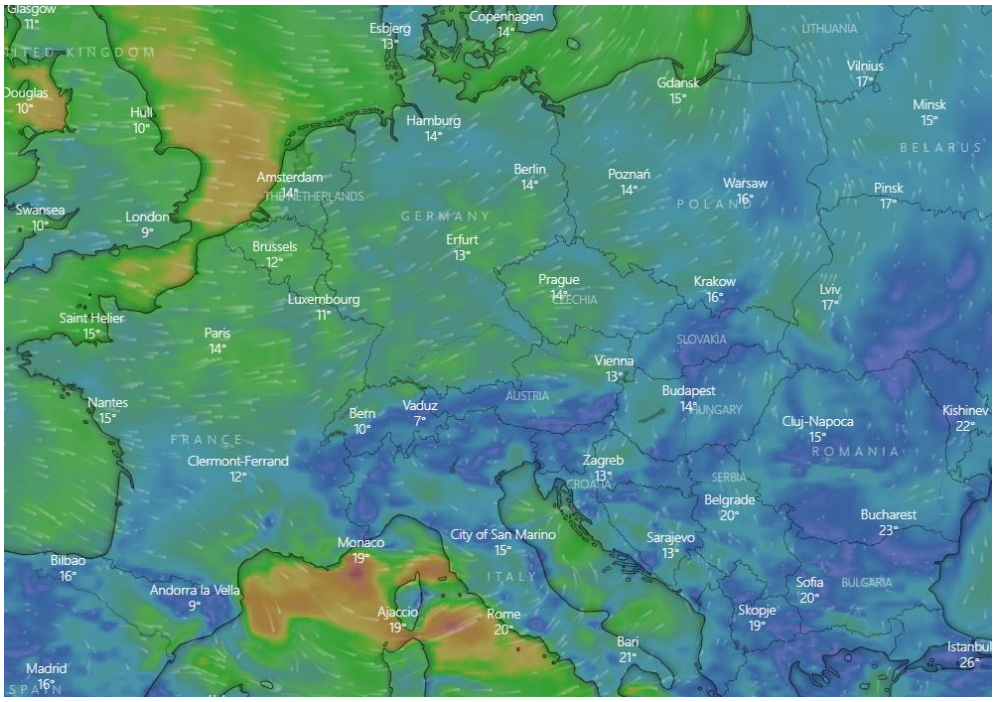

Abbildung 1 Wind dargestellt mittels eines Partikelsystems. Quelle: www.windy.com

Ein weiteres Projekt, dass bereits Partikelsysteme zur Darstellung globaler Windströme nutzt ist *Cameron Baccarios* Projekt *earth* (earth.nullschool.net). Auf einer Darstellung, welche unter earth.nullschool.net zu sehen ist, werden die atmosphärischen Strömungen aus verschiedenen Höhen, wie es zunächst scheint, auf einem Globus dargestellt. Bei genauerer Betrachtung fällt jedoch auf, dass es sich um eine orthographische Projektion der Erde handelt, die sich durch Interkation mit der Karte ganz ähnlich zu einem virtuellen Globus verhält. Es existiert dabei jedoch kein 3D Raum, was bedeutet, dass sich Partikel dementsprechend nur im sichtbaren Bereich der Erde bewegen. Nach der Interaktion mit

der Karte lädt das System zunächst neu, was keine kontinuierliche globale Visualisierung darstellt.

Diese Arbeit möchte nun, angelehnt an die eben genannten Beispiele, zeigen wie sich die Darstellung von Partikelsystemen von Karten auf Globen übertragen lässt. Probleme die es dabei zu lösen gibt, liegen vor allem an der sphärischen Form des Darstellungsmediums. Insbesondere die Darstellung auf einem taktilen Globus stellt besondere Ansprüche an eine Visualisierung. Genaue Anforderungen an die Darstellung werden in den jeweiligen Abschnitten zu den Globen in Kapitel 6 beleuchtet.

Zudem soll es möglich sein die Visualisierung interaktiv zu beeinflussen. Einerseits sollen die Partikel des Systems durch den Benutzer in ihrer Erscheinung konfiguriert werden können. Dazu gehören zum Beispiel die Anzahl und die Größe der darzustellenden Partikel. Andererseits soll auch das Vektorfeld durch den Benutzer ausgetauscht werden können. Für jenen Zweck wird im Laufe der Arbeit beantwortet, wie dieses Feld aufgebaut und welche Parameter es enthalten muss.

1.2 Wissenschaftliche Fragestellung

Im Rahmen dieser Masterarbeit sollen mehrerer Forschungsfragen beantwortet werden. Diese sollen Fragen zur Beschaffenheit von zu erstellenden Partikelsystemen sowie zu den Anforderungen an die Daten beantworten. Die Forschungsfragen lauten:

- Wie müssen Partikelsysteme gestaltet sein, um den Anforderungen zur Visualisierung am Hyperglobus gerecht zu werden?
- Welchen Anforderungen müssen die Daten entsprechen, um mit Partikelsystemen visualisiert zu werden?

1.3 Aufbau der Arbeit

Zielstellung der Arbeit ist es mit einem Partikelsystem globale atmosphärische Strömungen auf Hypergloben darzustellen. Dazu werden zunächst verschiedene wesentliche theoretische Grundlagen erläutert. Zunächst wird sich die Arbeit genauer mit Globen befassen, indem ein Überblick über Globen im Allgemeinen gegeben wird. Anschließend wird erläutert was unter Hypergloben speziell zu verstehen ist. Für diesen Zweck werden die Funktionsweisen von Hypergloben genauer betrachtet und erklärt. Ein weiteres Kapitel wird sich mit Visualisierungen beschäftigten. Hier wird zunächst betrachtet was unter einer Visualisierung zu verstehen ist und wie die Wissenschaft Visualisierungen nutzen kann. Die Einordnung verschiedener Systematiken, wie wissenschaftliche Visualisierung, Informationsvisualisierung oder Datenvisualisierung wird vorgestellt bevor beleuchtet wird, wie Visualisierung zur Informationsvermittlung genutzt werden kann. Da die Arbeit Geodaten visualisiert, wird im speziellen auf die Geovisualisierung eingegangen.

Im Weiteren werden Vektorfelder vorgestellt. Da die Arbeit diese als Grundlage nimmt, um Wind mit Partikel zu Visualisierungen, sollen Strömungsfelder genauer beleuchtet werden. Neben genereller Beschreibung dieser, fällt ein genauerer Blick auf ihre Bedeutung für die Geoinformation.

Der letzte große Theorieteil dieser Arbeit widmet sich Partikelsystemen. Neben der historischen Entwicklung und dem aktuellen Stand der Forschung wird genauer auf die Funktionsweise dieser Systeme eingegangen. Dabei sollen vor allem aktuelle Anwendungsbeispiele beleuchtet werden, welche Geodaten visualisieren beziehungsweise im Rahmen der Geoinformation angewendet werden.

Letztendlich wird vorgestellt, wie Partikelsysteme auf Globen visualisiert werden sollen. Nachdem die Auswahl der Daten sowie das Format dieser aufgearbeitet wurde, sollen zunächst Möglichkeiten vorgestellt werden, die es erlauben Partikelsysteme zu erstellen. Nach Auswahl der geeigneten Instrumente, werden mit diesen entsprechende Partikelsysteme erstellt und erläutert. Der zugrundeliegende Quellcode wird dafür Schritt für Schritt betrachtet. Anschließend werden die Ergebnisse vorgestellt und Diskutiert.

Anschließend werden die wissenschaftlichen Fragestellungen, welche im nächsten Kapitel gestellt werden, beantwortet und eine Zusammenfassung der Arbeit gegeben. Ein letztes

Kapitel beschäftigt sich mit Fragestellungen, welche während der Arbeit aufgekommen sind, und die es sich lohnt in zukünftigen Untersuchungen zu erforschen.

2. Globen

Zentrales Thema dieser Arbeit ist, wie Partikelsysteme auf sphärischen Oberflächen dargestellt werden können. Daher wird sich das erste große Kapitel des Theorieteils mit Globen beschäftigen. Neben ihren analogen Vertreter wird der Fokus hauptsächlich auf Hypergloben gelegt. Analoge Globen werden, obwohl sie die Geschichte des Globus dominieren, nur kurz angesprochen. Dies geschieht da zum einen ihre Geschichte und ihr Nutzen bereits in zahlreichen anderen Arbeiten dargelegt wurde (beispielsweise *Smutny* 2017), zum anderen, weil sie sich aufgrund ihres Wesens schlicht nicht für die Darstellung von dynamischen Darstellungen eignen. Dennoch soll ein kurzer Überblick gegeben werden, da sie maßgeblich an der Entwicklungsgeschichte von Hypergloben beteiligt sind. Auch Hologloben werden nur im Ansatz besprochen, da sie im Gegensatz zu analogen Globe, die ihre Bedeutung vor allem in der Vergangenheit hatten, erst in Zukunft eine Rolle spielen werden. Denn die aktuellen technischen Möglichkeiten lassen es noch nicht zu, dass Hologloben eine alternative zu virtuellen oder taktilen Hypergloben wären. Mit letzteren wird sich dieses Kapitel dagegen ausführlicher beschäftigen. Virtuelle und taktile Hypergloben belebten das Medium „Globus“ wieder. Statt hauptsächlich als Dekorationszweck zu dienen, hauchen die neuesten Entwicklungen Globen wieder neue Bedeutung ein (*Riedl*, 2011).

2.1 Der Begriff Globus

Unter Globen versteht man Modelle der realen Welt. Sie repräsentieren dabei meistens die Erde aber auch andere Himmelskörper können durch Globen dargestellt werden. Sie sind die einzige kartographische Möglichkeit, Himmelskörper in ihrer originären unverzerrten Dreidimensionalität zu präsentieren (*Riedl*, 2000).

Die Definition des Begriffs Globus wurde in den vergangenen Jahrzehnten mehrfach überarbeitet. Eine ausführliche Begriffsdiskussion findet sich in *Hrubys* Dissertation „Karten vs. Globen“ aus dem Jahr 2013. Dort wird gezeigt, dass sich der Begriff zunächst

nur auf ein Abbild der Erde bezog. Später wurde die Definition deutlich differenzierter. Er führt die Definition von *Imhof* aus dem Jahr 1972 auf, die zwar noch immer die Erde als einziges Modell des Globus annimmt, dagegen aber schon die Möglichkeit von Reliefgloben anerkennt. Erst im Jahr 1994 wurde in der Begriffsdefinition von *Hake et al.* erwähnt, dass Globen neben der Erde auch Nachbildungen anderer Himmelskörper darstellen können:

„*Globen sind Nachbildungen der Erde, eines anderen Weltkörpers oder der scheinbaren Himmelkugel. Sie bestehen aus Holz, Pappe, Blech, Glas oder Kunststoff und weisen meist Durchmesser von rund 25 bis 50 cm auf, was bei Erdgloben Maßstäben von 1:50 Mio. bis 1:25 Mio. entspricht.*“ *(Hake et al. 1994)*

Riedl kritisiert diese Definition jedoch aus mehreren Gründen. Zum einen gibt es keinen Hinweis darauf, dass Globen auch in virtuellen Versionen bereits existierten (siehe Kapitel 2.3.1 „virtuelle Globen“). Zum zweiten erscheint es überflüssig, die Materialien aus den Globen bestehen können aufzuzählen. Da dies bei anderen kartographischen Produkten ebenfalls nicht getan wird und die Materialien keine Aussage zum Inhalt oder der Qualität des Globus beitragen, könne man diesen Hinweis aus der Definition entfernen. (*Riedl*, 2000). Weiter definiert *Riedl* Globen wie folgt:

„*Ein Globus präsentiert ein maßgebundenes und strukturiertes Modell eines Himmelskörpers (bzw. der scheinbaren Himmelskugel) in seiner unverzerrten dreidimensionalen Ganzheit.*“ (*Riedl*, 2000)

Der Hinweis auf die Materialien wurde demnach entfernt. Es wurde auch der Bezug zur Erde nicht mehr verwendet, da diese ohnehin zu den erwähnten „Himmelskörpern“ zählt. *Riedl* merkt weiterhin an, dass Globen nicht zwangsweise rund sein müssen, da auch Asteroiden in ihrer unregelmäßigen sphärischen Gestalt mittels Globen dargestellt werden könnten. *Hruby* (2013) äußert in seiner Untersuchung „Globen vs. Karten“ allerdings Kritik an diesem Standpunkt. Denn seiner Meinung nach würde dies des allgemeinen Verständnisses über Globen wiedersprechen. Zudem stellt sich die Frage ob neben Asteroiden, deren Größe zwischen 1000 Meter und 1 Meter definiert ist, auch ähnlich große irdische Wackelsteine mittels Globen repräsentiert werden könnten, beziehungsweise deren Modelle als Globus bezeichnet werden könne.

2.2 Analoge Globen

Wie eingangs des Kapitels erwähnt sollen analoge Globen in dieser Arbeit nicht die gleiche Aufmerksamkeit wie nicht-analoge Globen erlangen. Auch wenn analoge Globen nicht zur Präsentation von dynamischen Darstellungen geeignet sind, erscheint die Thematik des Globus unvollständig. Daher wird sich das Kapitel neben der Definition auch mit einem kurzen historischen Abriss und der Bedeutung von analogen Globen beschäftigen.

2.2.1 Definition

Dass es eine eigene Definition für analoge-Globen bedarf, ist den technischen Errungenschaften des 20. Jahrhunderts zu verdanken, denn erst seit den 1990er Jahren ist diese Unterscheidung mit dem Aufkommen virtueller Globen nötig. Vorher waren analoge Globen, sei es aus Stein, Holz, Pappe oder aufblasbar aus Kunststoff und in allen anderen erdenklichen Möglichkeiten, das alleinige Mitglied in der heutigen vielfältigen Globenfamilie. Ihre Bedeutung hat sich, wie sich noch zeigen wird, im Laufe der Jahrhunderte immer wieder verändert. Wie zuvor kurz genannt, werden analoge Globen in vielgestaltiger Weise produziert. Was jedoch alle Vertreter gemeinsam haben sind analoge Globuskörper und Globenkarten. *Hruby* definierte den Begriff „analoger Globus“ 2013 wie folgt:

„*Analoge Globen sind Repräsentamen eines kartographischen Zeichens, dessen Objekt Himmelskörper und astronomische Satelliten (im Sinne der IAU) sein können. Wesentliche semiotische Merkmale dieses Zeichens sind, dass die Relation zwischen Repräsentamen und Interpretant durch eine Wahrnehmbarkeit eben dieser Objektform durch die Nutzer/innen bestimmt sind. Wesentliches technologisches Merkmal ist der analoge Charakter, der sowohl Globuskarte, als auch Globuskörper zu eigen sind*.“ (*Hruby, 2013*)

Da analoge Globen in dieser Arbeit sekundäre Bedeutung haben, soll es bei dieser Definition belassen werden und keine weitere Diskussion stattfinden. Es sei dabei auf die Arbeit “Globen vs. Karten” von *Hruby* verwiesen, in der eine ausführliche Besprechung des Begriffs stattfindet und welche bei weiterem Interesse nachgelesen werden kann.

2.2.2 Historischer Abriss und Bedeutung

Die Bedeutung von analogen Globen hat sich über die letzten Jahrhunderte stark verändert. Waren erste Globen bereits aus der Spätantike bekannt, so erlangten sie erst ab dem 15. Jahrhundert an Bedeutung. Vor allem für die großen Entdeckungsfahrten leisteten sie Orientierungshilfe (*Muris&Saarmann*, 1961). Darüber hinaus dienten sie als wissenschaftliches Instrument zur Beantwortung astronomischer und geographischer Fragestellungen, wie Ermittlungen des Sonnenstandes oder Bestimmung von Tageszyklen (*Riedl*, 2000). Jedoch änderte sich die Bedeutung von Globen zugunsten von Karten in der zweiten Hälfte des 16. Jahrhunderts:

„*Hatte noch die kolumbianische Zeit im Globus das eigentliche Orientierungsmittel gesehen, so beginnt bei Überhang der extensiven Erdforschung zur intensiven der Globus mehr und mehr zum Schaustück abzusenken. Während die Karte das ausschließliche Orientierungsmittel wird*" (*Muris & Saarmann*, 1969)

Ein Zeuge dieses Wandels ist auch eine Einzelanfertigung, die Mercator im Auftrag von Kaiser Karl V anfertigte. Sie bestand in einem Himmelsglobus, angefertigt aus Glas. Die Sternenbilder wurden dabei mit einem Diamanten eingraviert. Ein weiteres Beispiel, das den Zeitgeist über Globen repräsentiert ist Philipp Apians (1551 - 1589) Globus, welcher für den damaligen bayrischen Landesherren angefertigt wurde. Dass dieser nicht vorrangig zur Orientierung angefertigt wurde, zeigen zahlreiche Malereien, die über den gesamten Globus verteilt sind. Ziel war es keinen möglichst genauen Globus anzufertigen, sondern ein Prunkstück zu schaffen, das fortan als Dekoration dienen sollte. Globen dienten nicht mehr dazu, Weltbilder zu vermitteln oder der Orientierung auf See zu leisten, sondern waren zur Dekoration verkommen. *Sumira&Schneider* (2016) ergänzen, dass Globen Symbole für Macht darstellten, mit denen Würdenträger ihren Besitz und Autorität präsentierten-

Erst zum Ende des 18. Jahrhunderts, als in Europa die Zahl der Schüler aus allen Bevölkerungsschichten stieg, wurde der Globus wieder als Wissensvermittler in Schulen eingesetzt. Für diesen Zweck wurden auch neuartige Globen wie etwa Spiel- und Puzzlegloben entwickelt um Schülern das Bild der Erde zu vermitteln (*Sumira&Schneider*, 2016). Globen waren bis zur Mitte des 20. Jahrhundert Teil der Didaktik im Geographieunterricht (Riedl, 2011). Heute dienen analoge Globen wieder vorwiegend zur Dekoration und Einrichtungsaccessoire. Als Wissensvermittler wurden sie derweil von

Hypergloben abgelöst. Was darunter zu verstehen ist, und welche Möglichkeiten diese bieten wird in den nächsten Kapiteln ausgeführt.

2.3 Hypergloben

Hypergloben stellen eine Weiterentwicklung analoger Globen dar. Der Fortschritt ist vor allem durch digitale Inhalte, Interaktivität und flexible Themenvielfalt geprägt. Die Bezeichnung *Hyper* soll einen Bezug zu Daten der realen Welt herstellen und diese in einem anderen Maßstab darstellen. Der Begriff ist angelehnt an *Hyperlinks,* welche ebenfalls als Verbindung zu Informationen agieren.

Gemeinsam haben alle Formen von Hypergloben, dass sie ein digitales Globenbild zeigen. Die Beschaffenheit des Globenkörpers differenziert Hypergloben in folgende Kategorien:

- virtueller Hyperglobus
- taktiler Hyperglobus
- holographischer Hyperglobus

In Tabelle 1 werden die drei Arten der Hypergloben gegenübergestellt. Neben der bereits angesprochenen Art der Darstellung des Globenbildes sowie die Beschaffenheit des Globenkörpers wird zudem unterschieden in welcher Umgebung die Globen erlebbar sind.

Tabelle 1 Kategorien von Hypergloben (nach Riedl, 2010)

	Abbild	Raum	Globenkörper
analoger Globus	analog	analog	analog
virtueller Hyperglobus	digital	virtuell	virtuell
taktiler Hyperglobus	digital	real	materiell
Hologlobus	digital	real	virtuell

2.3.1 Virtueller Hypergloben

Virtuelle Globen stellen die größte und am weitesten verbreitete Gruppe der Hypergloben dar (*Hruby*, 2013). Bereits mit Beginn der 1990er Jahre wurden verschiedene virtuelle Globen präsentiert. Anlässlich seines 500. Jahrestages wurde 1992 eine digitale Version des Behaim-Globus vorgestellt. Neben der reinen Betrachtung des Globus erlaubte die

digitale Version darüber hinaus Vermessungen auf dem virtuellen Globuskörper durchzuführen (*Dorffner*, 2010).

Im Jahr 1993 stellte die Firma *Art+Com* aus Berlin einen virtuellen Globus namens *TerraVision* vor. Dieser bestand aus Kacheln aus Luft- und Satellitenbildern. Je nach Zoomstufe wurden die Bilder in unterschiedlichen Auflösungen dargestellt. Es war zudem möglich durch 3D-Stadtmodelle (Berliner Innenstadt und Manhatten) zu „fliegen". Darüber hinaus konnte der Globus verschiedene Layer zu den Themen Klima, Ozonschicht und Magnetfeld anzeigen. Der Globus diente dem Auftraggeber „Telekom" zunächst als Werbeobjekt auf Messen. Er war zuletzt auf der Expo 2000 in Hannover zu sehen (*Schweikart* et al., 2009).

Ein weiterer früher Vertreter ist der, später in *EarthBrowser* umbenannte, *Planet Earth* Hyperglobus. Mit diesem konnten Wolken in Echtzeit global dargestellt werden (*Riedl, 2010*). Entwickelt wurde dieser an der Universität Oregon im Jahr 1996. Eine zweite Version wurde im Jahr 2003 veröffentlicht. Diesmal konnten mit dem Globus weitere Echtzeit-Datensätze angezeigt werden. Des Weiteren verfügte er über mehrere Webcams, die sich ansteuern ließen sowie das Anzeigen von Wetterberichten ermöglichte (*Schweikart et al.* 2009).

Größere Bekanntheit erreichten virtuelle Globen durch die Veröffentlichung von *Google Earth* im Jahr 2005. Das Projekt startete zunächst unter dem damaligen Namen *Keyhole* und wurde zuerst von der gleichnamigen Firma entwickelt. Es erreichte auf der 5. Afrikanischen GIS-Konferenz größerer Aufmerksamkeit mit der Folge, dass *Google* das Projekt aufkaufte und unter dem Namen *Google Earth* weiterentwickelte. Bereits in den ersten zwei Jahren wurde die Software über 250 Millionen Mal heruntergeladen (*Schweikart et al.*, 2009). Weiterhin veröffentlichten auch *Microsoft* mit *Bing Maps 3D* (vormalig *Visual Earth*) und *NASA* mit dem Projekt *World Wind* virtuelle Globen. Riedl bemerkt jedoch an, dass diese Globen zwar als solche vermarktet werden, die Software jedoch nicht das Ziel hat den Inhalt als Ganzheit zu präsentieren (*Riedl*, 2010). Denn laut Definition bilden Globen ein unverzerrtes Abbild von Himmelskörpern in ihrer Ganzheit ab (siehe *Kapitel 2.1 der Begriff Globus*). Genutzt werden eben genannte Globen jedoch hauptsächlich für großmaßstäbige Zwecke, wie *Schöning et al.* 2008 berichten. Dort heißt es, dass über die Hälfte der Nutzer die Software dafür nutzen, um ihre eigenen Häuser oder andere individuelle Orte zu erkunden. Als zweithäufigster Grund für die Nutzung wurde Navigation angegeben.

Hruby et al (2009) untersuchten daher, das Verständnis, als welches Nutzer das Produkt *Google Earth* sehen. Demnach würden Nutzer Google Earth als Globus anerkennen, aber nur wenn die gesamte Erde zu sehen ist. Gleichzeitig wird Google Earth als Karte akzeptiert, nämlich dann, wenn nur großmaßstäbige Ausschnitte des Globus zu sehen sind. Weiter heißt es Google Earth sei also ein Globus, wird aber nicht als solcher genutzt, was sich mit den Aussagen von *Schöning et al.* deckt. Zusammenfassend stellen Hruby et al. fest:

„GE [Google Earth] software is an *earthbrowser* that allows, among many other spatial perspectives, to visualize and perceive the entire Earth in its spherical form in terms of a *virtual hyperglobe*. (*Hruby et al*, 2009).

Google Earth sei demnach ein *Earthbrowser* (Erderkunder), der in der Form eines virtuellen Hyperglobes dem Nutzer räumliche Perspektiven ermöglicht.

2.3.2 Taktile Hypergloben

Ein weiterer Vertreter der Globenfamilie sind taktile Hypergloben. Da hier der Globenkörper im realen Raum existiert, können taktile Globen als Nachfolger beziehungsweise Weiterentwicklung der analogen Globen angesehen werden. Letztere konnten nur wenige Thematiken gleichzeitig abbilden. Dagegen ist es dem taktilen Hyperglobus möglich, eine praktisch unzählige Fülle an Themen zu visualisieren. Der Begriff taktil stammt aus der Idee, dass der Globus grundsätzlich taktil zu erkunden ist, also schlicht gesagt, dass man ihn anfassen kann. Der Begriff wirft laut *Hruby* (2013) Probleme auf. Denn „taktil“ wurde in der Kartographie bisher für kartographische Produkte, sei es Karten, Globen oder kartenähnliche Darstellungen, genutzt, um zu verdeutlichen, dass dieses Produkt auch für sehbehinderte Menschen zugänglich und ertastbar sind. Durch entsprechende Erhebungen, Reliefe, differenzierte Oberflächenbeschaffenheiten (glatt, rau, geriffelt etc.) oder durch Brailleschrift können Inhalte barrierefrei dargestellt werden. Da taktile Globenprodukte bis dato jedoch keine der eben aufgezählten Eigenschaften aufweisen, könnte es rein begrifflich zur Verwirrung anstiften. Ein weiterer Kritikpunkt *Hrubys* ist, dass taktile Hypergloben zwar angefasst, sie aber auch ohne Touch-sensitive Interaktion als solche, also „taktile“ Produkte, definiert werden können. Auch wenn erste Vertreter mit

berührungsintensiven Oberflächen bereits auf dem Markt sind, würde es bei den meisten Modellen, die zurzeit im Umlauf sind, keinen Unterschied machen, ob man den Globuskörper berührt oder nicht. *Hruby* schlägt daher den Begriff „materieller Hyperglobus“ vor. Dieser würde sich zum einen genug von analogen Globen abgrenzen, da sich „Hyper“ auf die digitalen Globenkarten bezieht und zum anderen der Begriff „materiell“ den kartographisch bereits belegten Begriff „taktil“ ersetzen würde und dennoch die Besonderheit des Hyperglobus herausstellt.

2.3.2.1 Entwicklung von taktilen Hypergloben

Die Geschichte der taktilen Hypergloben begann Anfang der 90er Jahre in Brasilien. Dort wurde der „GeoSphereGlobe“ vom „National Institute for Space Research“ in Sao Paulo entwickelt. Aufgrund begrenzter Laufleistung der Projektoren in jener Zeit wurden auf die Oberfläche, welche aus transparentem Acrylglas bestand, halbtransparente Satellitenbilder aufgebracht, um auch mit ausgeschalteten Projektoren eine Darstellung zu ermöglichen. Die Projektoren selber waren im inneren der Globuskugel angebracht und ermöglichten die Abbildung verschiedener Themen auf der Oberfläche des Globus (*Riedl*, 2004).

Ein weiterer taktiler Globus, mit einer gänzlich anderen Herangehensweise wurde im Jahr 2000 in Bad Homburg von der Firma Amadeus Germany in Deutschland entwickelt. Dieser Globus (genauer gesagt war es nur ein halber Globus), bestand aus 110 000 Glasfasern, die gebündelt zu einer Halbkugel geformt wurden. Die Enden der Fasern ergaben dabei die Projektionsfläche. Die Projektion selber wurde zunächst in einem separaten Raum auf eine Leinwand abgespielt. Von dort aus transportierten die Glasfaser das Abbild auf die Globusoberfläche, also auf das andere Ende der Fasern (*Riedl*, 2004).

Waren die genannten Beispiele bisher noch Unikate wurden ab dem Jahr 2002 schon einige taktile Globen in Serie produziert. *Riedl* (2010) bezeichnet den Zeitpunkt daher als „Wendepunkt der taktilen Hypergloben“. Er berichtete von 3 Varianten die in diesem Jahr produziert wurden. Die Firma ARC Science Simulation in den USA entwickelte beispielsweise den „OmniGlobe“. Ein Globus aus transparentem Acrylglas, dessen Abbildungsverfahren der Themen durch Innenprojektion mit einem konvexen Spiegel realisiert wird. Die Globendurchmesser waren dabei 0.8, 1.5 oder 2 Meter. Ein weiterer

Vertreter wurde von „Global Imagination“ in Campbell (Kalifornien, USA) vorgestellt. Der Globus arbeitet mit einer linsenbasierten Innenprojektion. Die Globen konnten in verschieden Größenstufen zwischen 0.41 und 1.8 Metern erworben werden. Eine dritte Variante, die 2002 auf den Markt kam, wurde von der NOAA (National Oceanic and Athmospheric Administraction) ebenfalls in den USA (Boulder, Colorado) entwickelt. Diese Version eines taktilen Globus funktionierte, anders als die vorigen genannten, durch Außenprojektion. Auch dieser Globus wurde in drei Größen (1.5, 2 und 3 Metern) hergestellt (*Riedl*, 2010). Mit *Globoccess* (Hamburg, Deutschland) und *Pufferfish* (Edinburgh, Großbritannien) kamen in den Folgejahren weitere Firmen hinzu, die taktile Globen in Serie produzierten und verkauften (*Riedl*, 2011). Letztere entwickelten eine Version eines taktilen Globus, die rein der Semantik des Begriffs „taktil“ bisher wohl am nahesten kommt. Der *Puffertouch Range*, der spätestens ab 2016 vorgestellt wurde, besitzt eine berührungssensitive Oberfläche, die das Erkunden des dargestellten Inhalts ermöglicht (pufferfishdisplays.com). Da es bereits weitere angemeldete Patente zu berührungsintensiven sphärischen Oberflächen gibt, könnten taktile Globen mit interaktiver Oberfläche in Zukunft zum Standardmodell werden (*Riedl*, 2011). Diese Entwicklung würde auch die Kritik an dem Begriff „taktiler Globus“ obsolet werden lassen.

Anfang der 2010er Jahre wurde davon ausgegangen, dass sich die Technologie der LED Globen mit direkter Projektion im Laufe der Dekade weiterentwickeln würde. Dies ist bis jetzt allerdings nicht geschehen und weitere Neuerungen sind nicht bekannt.

Dagegen hat sich in den letzten Jahren ein anderer Trend aufgetan. Seit Mitte des Jahrzehnts sind Touch-Sensitive Globen auf den Markt gekommen, welche mittlerweile auch in Serie hergestellt werden. Die Firma *Pufferfish* aus Edinburgh stellte den Globus *Range touch* vor. Es handelt sich dabei um einen aus Acrylglas hergestellten Globus, welcher mittels Innenproduktion (Fischaugentechnik) das Globusbild an den Körper projiziert. Mit welcher Technik, der Globus Berührungen am Globuskörper registriert, wurde bisher nicht von der Firma kommuniziert (*Crespel et. al* 2017).

Holzapfel stellte 2012 jedoch mehrere Möglichkeiten vor, mit denen es generell möglich ist, berührungssensitive Globen zu realisieren. Zu den Techniken, die eine Touch-Funktion erlauben gehören:

- elektronische Sensorik – eine Berührung der Oberfläche verändert den Stromfluss in einer elektrisch leitenden Schicht, welche auf dem Medium angebracht ist. Elektronische Sensoren können so die Position der Berührung ausfindig machen.
- akustische Sensorik – die Oberfläche des Mediums wird durch Schallwellen in Bewegung gebracht. Eine Berührung verändert so die Ausbreitungsmuster und - Intensität der Wellen. Spezielle Sensoren messend die Variation der Wellen und können den Berührungspunkt ermitteln.
- optische Sensorik - bei diesen Systemen werden Infrarot-Wellen ausgesandt welche durch spezielle Sensoren wieder eingefangen werden. Durch Berührung der Oberfläche wird die Strahlungsintensität der Wellen verändert und der Sensor erkennt so, an welcher Stelle die Oberfläche berührt wurde.

Vertiefende Informationen zu den verschiedenen Sensortechniken finden sich bei *Holzapfel* (2012).

2.3.2.2 Projektionsverfahren

Die Bespielung taktiler Globen mit Inhalten lässt sich durch verschiedene Methoden realisieren. Grundsätzlich stehen drei Möglichkeiten zur Verfügung:

- Innenprojektion
- Außenprojektion
- Direkte Projektion

Im Folgenden sollen alle drei Möglichkeiten näher erläutert werden.

I Innenprojektion

Wie dem Namen dieser Variante bereits zu entnehmen ist, befindet sich die Abspielfläche der Projektion auf der Innenseite des Globenkörpers. Prinzipiell stehen dabei zwei Möglichkeiten zur Verfügung: Fischaugenbasierte Systeme und

Spiegelsysteme (siehe Abbildung 2). Beide gemein haben, dass sich der Projektor dabei im Regelfall unter der Kugel befindet und von dort durch ein Loch in das Innere des Globenkörpers strahlt. Einerseits wird bei fischaugenbasierten Systemen der Lichtstrahl von einem Weitwinkelobjektiv so gebrochen, dass der Lichtkegel die Innenseite des Globus bespielt. Andererseits kann durch einen konvexen Spiegel, der sich an der gegenüberliegenden Seite des Eintrittsloches befindet, der Lichtkegel auf die Globusinnenseite geleitet werden. Durch diese spiegelbasierten Systeme könne zwar höhere Auflösungen erreicht werden (durch die Möglichkeit des Einsatzes eines zweiten Beamers), wodurch großmaßstäbige Abbildungen realisiert werden können, jedoch haben sie im Gegensatz zu fischaugenbasierten Systemen zwei statt nur einen Blindspot. Diese befinden sich nahe der Eintrittsstelle des Lichtstrahls sowie auf er gegenüberliegenden Seite hinter dem konvex gebogenen Spiegel (*Riedl,* 2009).

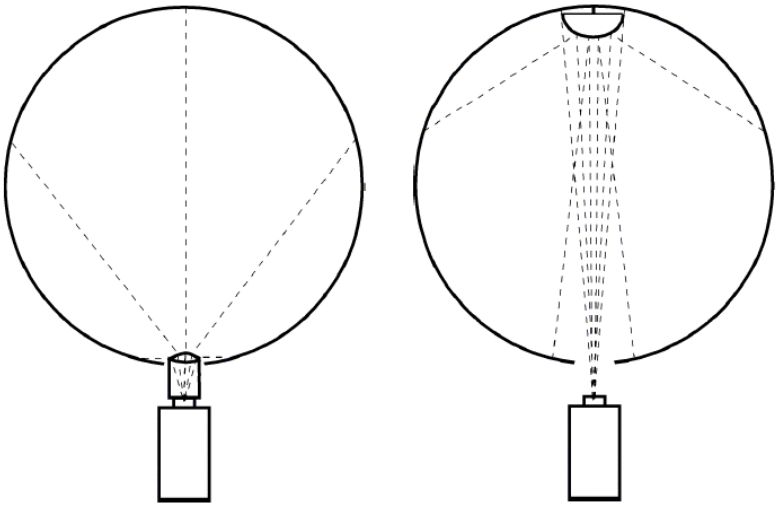

Abbildung 2 Innenprojektionen. links: Fischaugensystem, rechts: Spiegelsystem. Quelle: Riedl, 2008

II Außenprojektion

Bei dieser Möglichkeit strahlen mindestens vier Beamer das Globenbild auf den Globenkörper. Diese befinden sich dabei gleichmäßig verteilt um den Globus herum. Um auch die Pole abzubilden, werden zwei weitere Beamer, jeweils lotrecht zum Nord- beziehungsweise Südpol positioniert, benötigt. Durch den Einsatz mehrere Projektoren kommt es zwangsweise zu Überschneidungen des projizierten Abbildes auf dem Globenkörper. Durch Edge-Blending können Überlappungsbereiche graphisch akzeptabel dargestellt werden. Die Vereinheitlichung des Globenbildes ist jedoch sehr aufwendig. Jeder Beamer benötigt mindestens einen Rechner. Ein

weiterer ist zusätzlich von Nöten um alle Rechner zu synchronisieren. Gegenüber der Innenprojektion kann bei dieser Methode, aufgrund mehrerer Beamer eine sehr hohe Auflösung erreicht werden (Riedl, 2008). Weiterhin hat die Methode, zumindest in der Theorie, den Vorteil den Globus lückenlos zu bespielen. In der Praxis ist dies aber nur schwer umzusetzen, da hier der Betrachter zu jeder Zeit hinter den Beamern stehen muss, um den Lichtstrahl zwischen Projektor und Globusoberfläche nicht zu beeinflussen. Zusätzlich muss der Globus an einer Stelle des Globenkörpers befestigt sein, was eine Bespielung an dieser Stelle ebenfalls unmöglich macht. Nach *Riedl* (2008), eignet sich demnach die Außenprojektion vor allem bei großen Globen (ab 1,5 Meter Durchmesser).

III Direkte Projektion

Bei der direkten Projektion bildet der Globenkörper gleichzeitig den Projektor sowie die Projektionsfläche (*Riedl, 2008*). Da keine Zwischengeschalteten Systeme existieren, ergeben sich daraus mehrere Vorteile, die von *Riedl* 2008 erläutert wurden. Zum einen existiert kein Blindspot und keine Abschattung, die bei der Innen- und Außenprojektion unumgänglich sind. Zum anderen kommt es nicht zu Verzerrungen der Pixel, da diese direkt die sphärische Oberfläche bilden.

Bereits existierende Globen mit direkter Projektion bestehen aus LED-Paneelen oder aus Glasfaserleitungen. Ein Beispiel des letztgenannte wurde bereits im Kapitel 2.3.2.1 beschrieben. Ein Beispiel für einen Globus der mit LED-Paneelen ausgestatten ist, ist der *GeoCosmos II*, welcher in Tokyo ausgestellt ist. Dieser Globus mit einem Durchmesser von 6 Metern und 10 MPixel ist jedoch immer noch nur eines von wenigen Beispielen dieser Art. Obwohl *Riedl* 2011 prognostizierte, dass OLED-basierte Globen am Ende der Dekade ernsthafte Konkurrenz für die mittlerweile in Serie hergestellten Projektor-basierten taktilen Globen darstellen könnten, sind bisher nur wenige derartiger Globen bekannt.

2.3.3 Hologloben

Unter Hologloben versteht man Globen, welche auf einem holographischen Körper digitale Inhalte im realen Raum anzeigen. Dies bedeutet, dass kein materieller Körper

existiert, sondern ein Bild des Globus in der Luft schwebt. Durch aktuelle begrenzte Möglichkeiten der Technik, existiert ein derartiger Globus jedoch noch nicht.

Der Fortschritt zu holographischen Darstellungen beschränkt sich zumeist auf optische Täuschungen beziehungsweise die Simulation eines Holographs. Die Firme *Looking Glass Factory* (lookingglassfactory.com) wirbt beispielsweise mit holographischen Displays. Dabei wird jedoch kein immaterielles Hologramm im realen Raum erzeugt, sondern vielmehr ein Bild innerhalb eines Glaskörpers, welches von verschiedenen Blickwinkeln betrachtet werden kann und so die Illusion eines Hologramms erzeug.

Eine weitere Möglichkeit wurde 2019 von *Hirayama et al.* vorgestellt. Ein sehr kleines Kunststoffpartikel wird durch Ultraschallwellen im Raum bewegt und von Lichtquellen beschienen. Durch eine sehr hohe Geschwindigkeit des Partikels erzeugt diese Technik den Anschein eines volumetrischen Körpers im realen Raum. Dennoch handelt es sich auch hierbei nicht um ein echtes Hologramm, da letztendlich die Lichtstrahlen auf ein Material treffen, um ein Bild zu erzeugen. Es scheint also nicht absehbar, dass tatsächlich ein holographisch erzeugter Globus in naher Zukunft existieren wird.

3. (Geo)visualisierung

Ziel dieser Arbeit ist es, Geodaten zu visualisieren. Daher wird nun die Visualisierung und anschließend die Geovisualisierung im speziellen näher beleuchtet. Zunächst erfolgt jeweils eine Begriffserläuterung sowie ein Blick auf die Geschichte und neusten Entwicklungen. Auch der Nutzen und Bedeutung der Visualisierung und Geovisualisierung werden näher beleuchtet. Um letztendlich eine dynamische Visualisierung zu erarbeiten, werden zudem in diesem Kapitel theoretische Grundlagen der statischen und dynamischen sowie zwei- und dreidimensionalen Visualisierung gegeben.

3.1 Visualisierung

Der Blick richtet sich zunächst auf Visualisierung im Allgemeinen. Hier soll zunächst genauer auf den Begriff eingegangen werden, um zu verstehen, wie sich die Geovisualisierung im Anschluss einordnet.

3.1.1 Der Begriff Visualisierung

In den meisten Sprachen ist „sehen“ gleichbedeutend mit „verstehen“ (*Cauvin* et al. 2010). Daher sind Visualisierungen ein weit verbreitetes Mittel, um Informationen mit bildhaften Mitteln darzustellen. Denn Sie machen sich, wie Dodge et al. bemerken, die Fähigkeit des Geistes zu Nutze, Zusammenhänge schneller mit Hilfe von Bildern aufzunehmen und zu verstehen (*Dodge* et al., 2008). Die bildhaften Darstellungen von Informationen, kann demnach Schwererklärbares, Verborgenes oder Unsichtbares aus Texten oder Tabellen zum Vorschein bringen (*Leggewie*, 2009).

Ursprünglich verstand man unter dem Begriff „Visualisierung“, das mentale Zeichnen eines Bildes vor dem geistigen Auge. Heute steht er für die Darstellung von Information, welche mithilfe von graphischen Mitteln überbracht wird (*Cauvin et al.* 2010). Darüber hinaus verstehen *Schumann &Müller* unter dem Begriff einen kreativen Prozess, der Strukturen und Zusammenhänge aufdecken soll. Grundlage für jede Visualisierung seien dabei darzustellende Informationen. Die Visualisierungen selber können ein einzelnes Bild oder eine Bildsequenz beinhalten. Die Darstellung kann einfache geometrische Grundformen umfassen oder komplexe dreidimensionale Animationen zeigen (*Schumann&Müller*, 2000).

Der Begriff „Visualisierung“ kann in verschiedene Bereiche aufgegliedert werden. „Visualisierung“ ist oftmals nur ein Sammelbegriff für viele verschiedene Untergliederungen, die einerseits von unterschiedlichen Autoren verschieden definiert und andererseits alle mit dem einfachen Term „Visualisierung“ abgekürzt werden. Allesamt verfolgen sie dennoch die gleichen Ziele, welche später in diesem Kapitel erläutert werden. Zunächst soll jedoch der Begriff Visualisierung näher beleuchtet werden.

Friendly (2018) unterscheidet beispielsweise zwischen *data* und *information Visualization.* Unter beiden Termini versteht er die Präsentation qualitativer oder quantitativer Informationen, um den Betrachter Muster, Trends, Anomalien, Beständigkeit oder Variationen zu zeigen. Dabei werden Mittel angewendet, die über die bloße Präsentation der Daten als Tabelle oder Text hinausgehen. Jedoch sei laut *Friendly* *information Visualization* ein weitgreifender Begriff, dem alle anderen Felder der Visualisierung untergeordnet sind. Denn auch die ältesten Eingravierungen von Karten auf Steinen oder Tontafeln stellen eine Art der Informationsvisualisierung dar. Enger definiert sei dagegen die *data Visualization*. Daten sind laut Friendly Informationen, welche schematisch zu Variablen und Attributen abstrahiert wurden. Die Datenvisualisierung nutzt hauptsächlich

statistische Grafiken zur Repräsentation. Auch seien laut *Friendly* (2009) thematische kartographische Visualisierungen Datenvisualisierungen.

Card (2008) unterscheidet zwischen *information Visualization* und *scientific Visualization.* Ersteres wird dabei definiert als der Nutzen von computerbasierten interaktiven Repräsentationen von abstrakten Daten, um den Erkenntnisprozess zu stimulieren. Letzteres sei demnach ähnlich jedoch spezifisch auf wissenschaftliche Daten zugeschnitten. Der Term *information Visualization* beschränkt sich hier also auf Computergestützte Methoden anders als bei *Friendly*, welcher lediglich meint, dass der Term *information Visualization* gegenwärtig die Verwendung von Computern zumindest andeutet. *Cauvin et al*. (2008) stellen diesbezüglich fest, dass schon allein der Begriff „Visualisierung" heutzutage computergestützte Methoden impliziert. Weiterhin heißt es bei *Cauvin et al*. aber auch, dass *Visualization* nur die Abkürzung für *scientific Visualization* sei und alle anderen Formen (einschließlich die hier bereits besprochenen) dem untergestellt seien. Es scheint also es gäbe keine allgemeingültige Definition des Begriffs geschweige denn der Bereiche in die sich „Visualisierung" aufteilt.

3.1.2 Bedeutung und Nutzen

Mehr Einigkeit herrscht über die Ziele und den Nutzen von Visualisierungen. Die Informationen hinter den Daten sollen mithilfe von Graphiken preisgegeben werden. Dies gelingt laut *Card* (2008), da Visualisierungen dem Geist eine weitere Ressource im Erkenntnisprozess beifügen. So werden dem Betrachter Bearbeitungsprozesse und die Suche nach Informationen abgenommen und zusammengefasst bildhaft präsentiert. Der Kern der Visualisierung liegt also in der Reduzierung der Datenmenge und in dem Aufzeigen von Beziehungen. So lassen sich auch in große Datensätzen Muster und Trends erkennen, die in Text- oder Tabellenform wohlmöglich verborgen geblieben wären. Laut Dodge et al. sind demnach die besten Visualisierungen die, welche neue Einblicke eröffnen, welche bisher nicht offensichtlich waren (*Dodge et al*., 2008).

Der Nutzen liegt in einem besseren Verständnis von Phänomenen und komplexen Zusammenhängen. Mithilfe von Visualisierungen und den Erkenntnissen, die aus diesen geschlossen werden können, können Schlussfolgerungen und Entscheidungen getroffen werden. Visualisierungen sind also ein gutes Mittel Informationen zu kommunizieren. Allerdings stellen *Unwin et al*. fest, dass auch das Gegenteil zutrifft:

„Graphic displays are often very effective in communicating information. They are also often not effective at communicating information "(Unwin et al., 2008)

Graphiken seien also sehr gut und gleichzeitig sehr schlecht zur Kommunikation von Informationen geeignet. Als Grund dafür wird genannt, dass es heute praktisch jedem möglich ist Informationen zu visualisieren. Wie kann nun gemessen werden ob eine Visualisierung ihr Ziel erreicht? *Schumann&Müller* bewerten, ob eine Visualisierung geeignet für ihren Zweck ist, mit der Qualität dieser. Die Qualität einer Visualisierung beschreibt, wie gut die bereits besprochenen Ziele erreicht werden konnten. *Schumann&Müller* definieren die Qualität einer Visualisierung unter anderem folgendermaßen:

„[Die Qualität] läßt [sic] sich als das Verhältnis von der dem Betrachter in einem Zeitraum wahrgenommenen Informationen zu der im gleichen Zeitraum zu vermittelnden Informationen beschreiben. „(*Schumann&Müller*, 2000)

Kann der Betrachter den Kontext der realen Welt aus der zum Teil stark abstrahierten Darstellung erkennen, gilt die Visualisierung als gelungen (*Schumann&Müller*, 2000). Um falsche Interpretationen oder Verständnisprobleme bei dem Betrachter zu vermeiden, zeigen *Schumann&Müller* Faktoren, die die Qualität der Visualisierung beeinflussen. Dazu gehören unter anderem das Vorwissen und die visuellen Fähigkeiten des Betrachters, übliche Metaphern oder Konventionen des Fachgebietes und das Bearbeitungsziel der der Abbildung.

3.1.3 Historischer Blick auf Visualisierung

Die Visualisierung von Informationen ist keine Erfindung der Moderne. Bereits mit den frühesten Kartendarstellungen wurden Daten visualisiert. Auch in der Forschung wurde die Visualisierung in verschiedenen Fachbereichen untersucht. Wie *Schumann&Müller* im Jahr 2000 anmerken wurde beispielsweise in der Kunst untersucht wie Informationen am besten abgebildet werden. Technische Wissenschaften präsentieren ihre Ergebnisse ebenfalls schon lange mit Visualisierungen unterschiedlichster Ausprägungen.

Friendly listet in seinem Projekt „Milestones in the history of thematic cartography, statistical graphics, and data visualization" aus dem Jahr 2009 alle bekannten nennenswerten Meilensteine der Datenvisualisierung auf. Bei dem ältesten Eintrag

handelt es sich um eine Karte der Stadt Kony in der Türkei aus der Zeit 6200 vor Christi. Friendly nennt als eine der ältesten tatsächlichen Datenvisualisierungen geometrische Diagramme, die die Positionen von Sternen anzeigen und mit deren Hilfe bereits 200 vor Christi Karten zur Orientierung hergestellt wurden. Eine weitere bemerkenswerte Darstellung stammt aus dem 10. Jahrhundert. Ein anonymer Autor visualisierte die sieben zu dem Zeitpunkt markantesten Himmelskörper auf einem Kartesischem Koordinatensystem. Darauf war auf der vertikalen Achse die Bahnneigung der Planeten zu sehen und auf der horizontalen Achse war die Zeit in 30 Intervallen eingezeichnet. Bemerkenswert war diese Darstellung nicht zuletzt, da das gerasterte Koordinatensystem erst im 17. Jahrhundert vollständig ausgereift war und es sich hierbei um die erste bekannte Darstellung temporaler Daten handelte (*Friendly*, 2009).

Noch einmal 200 Jahr später begann, wie *Friendly* es 2009 taufte, das goldene Zeitalter der Visualisierung. Zahlreiche Methoden zur statistischen Darstellung wurden zum Wechsel vom 18. in das 19. Jahrhundert bekannt. Zum ersten Mal tauchten beispielweise Kreisdiagram und Balkendiagram auf. Beide sind auf William Playfair (1759 bis 1823) zurückzuführen.

Im Letzen Jahrhundert hat sich die Visualisierung zu multidisziplinärem Fachgebiet entwickelt. Aufgrund neuer Entwicklungen in theoretischer und technischer Infrastruktur konnten neue Errungenschaften in der Visualisierung von Informationen vorangetrieben werden. Dazu zählen die Anwendung von Visualisierungsmethoden auf immer größer werdende Datenstrukturen. Auch neue Methoden zur Visualisierung wurden entwickelt, wie beispielsweise höherdimensionale Scatterplot Matrizen, deren Erstellung nur durch die Anwendung von Computern möglich ist (*Friendly*, 2006). Spätestens seit den 1970er Jahren wird Visualisierung daher immer mit Computern in Verbindung gebracht. Wenn von Visualisierung gesprochen wird, ist der Hinweis auf Computertechnik bereits impliziert (*Dodge et al.,* 2008). Neben neuen technischen Errungenschaften erhöhte sich die Aufmerksamkeit auf die kognitiven und wahrnehmungsbezogenen Aspekte von Visualisierungen (*Friendly*, 2006). So beschäftigte man sich zum Beispiel in der Kunst damit, wie Informationen am besten durch Symbole repräsentiert werden könnten (*Schumann&Müller*, 2000).

3.2 Geovisualisierung

Nachdem Visualisierung erläutert wurde, wird sich die Arbeit in den folgenden Kapiteln speziell auf das Teilgebiet der Geovisualisierung konzentrieren.

3.2.1 Beschreibung und Überblick

Die Geovisualisierung, kurz für geographische Visualisierung, ist ein Teilgebiet der Visualisierung. Nach *Cauvin et al*. (2010) ist sie genau genommen der ‚scientific Visualization' untergeordnet'. Sie beschäftigt sich im speziellen mit der Darstellung von Geodaten, also Datensätzen, die georeferenziert wurden und damit einen räumlichen Bezug zur realen Welt aufweisen. Die Geovisualisierung bedient sich an Techniken und Methoden aus der traditionellen Kartographie (Farben, Symbole), wissenschaftlichen Visualisierung (Diagramme) und der Computer Technik (interaktive Animationen, 3D Umgebungen), um Daten zu präsentieren. Auch *MacEachren&Kraak* (2001) sehen die Geovisualisierung als eine Komposition aus verwandten Fachgebieten:

„Geovisualization integrates approaches from visualization in scientific computing (ViSC), cartography, image analysis, information visualization, exploratory data analysis (EDA), and geographic information systems (GISystems) to provide theory, methods, and tools for visual exploration, analysis, synthesis, and presentation of geospatial data […]" *(MacEachren&Kraak,* 2001)

Diese Definition adressiert zugleich die Ziele der Geovisualisierung. Sie soll Theorien, Methoden und Werkzeuge zur Verfügung stellen um Daten visuell erkunden, analysieren und präsentieren zu können. Eine verkürzte Definition liefern Dodge et al., welche jede Abbildung, die es ermöglicht ein räumliches Verständnis von Inhalten zu bekommen, als eine Form der Geovisualisierung sehen:

„ […] we see geographic visualization as the application of any graphic designed to facilitate a spatial understanding of things, concepts, conditions, processes or events in the human world" (*Dodge et al*. 2008).

Zu den bereits angesprochenen Zielen ergänzt *Nöllenburg* (2006), dass Geovisualisierung die menschlichen visuellen und kreativen Fähigkeiten sowie das allgemeinen Wissen mit der modernen Computerleistung verknüpft, um große georeferenzierte Datenmengen zu

erkunden. Es bestehen mehrere Ansätze, Geovisualisierungen nach ihren Zielen einzuteilen. *Longley* (2005) unterscheiden zwischen den folgenden vier Kategorien:

- Exploration (Erforschen)
- Analysis (Auswerten)
- Synthesis (Zusammenfassen)
- Presentation (Präsentieren)

Diese können mit dem von *MacEachren* bereits 1994 vorgestellten *Map-Use Cube* (siehe Abbildung 3) veranschaulicht werden. Anhand eines Würfels werden drei Dimensionen präsentiert, die unterschiedliche Einsatzgebiete von Visualisierungen abbilden. Die drei Achsen stehen für:

- Anwender: reicht von privaten bis öffentliche Nutzer
- Grad der Interaktion: von hoch bis kaum/nicht interaktiv
- Aufgabe der Visualisierung: von Wissen präsentieren bis Schaffung von Wissen

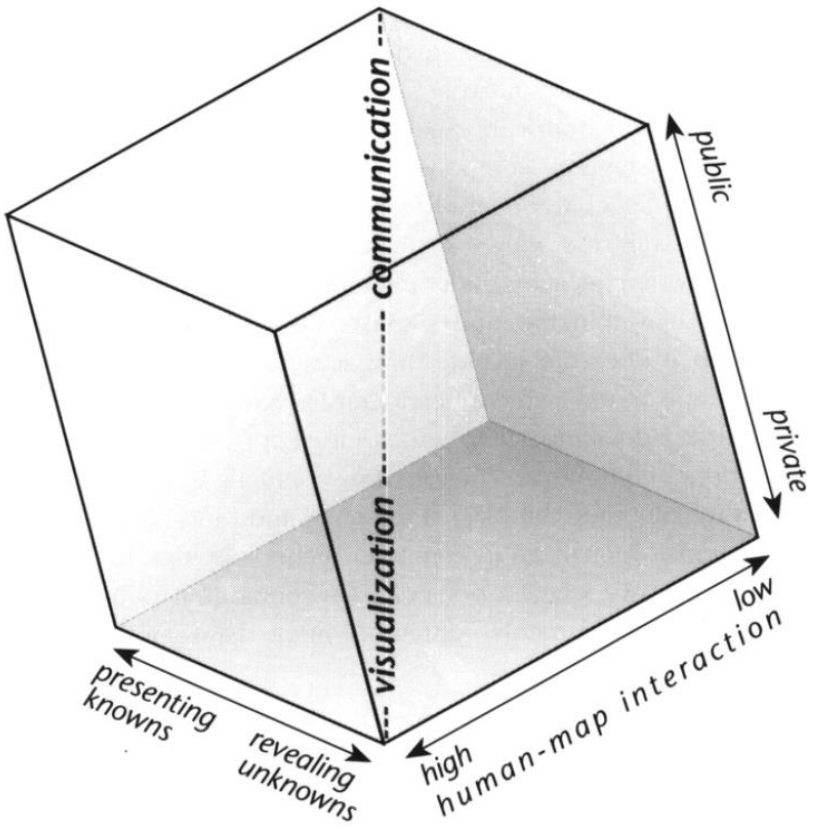

Abbildung 3 Der Map-Use Cube veranschaulicht die Ziele der Geovisualisierung Quelle: MacEachren 1994

Die von *Longley* erwähnten vier Ziele lassen sich laut Nöllenburg am *Map-Use Cube* diagonal von einem extrem (hohe Interaktivität, privater Nutzer, Inhalte erkunden), dem explorativen Charakter, bis an das andere Ende des Spektrums also keine oder wenig Interaktion, weitumfassende öffentliche Zielgruppe und bekannte Inhalte, der Präsentation, abbilden (*Nöllenburg*, 2006).

Dodge et al. (2008) teilen Geovisualisierungen dagegen in drei Erkenntnisgewinnenden Klassen ein:

1. „Looking“ (Suchen): meist zweidimensionale zum Teil interaktive Darstellungen, in denen der Benutzer navigieren und einfache Abfragen durch Interaktion mit den Inhalten durchführen kann.
2. „Querying“ (Abfragen): visuelle Schnittstellen zu Datenbanken. Ermöglichen dem Benutzer durch komplexe Informationsräume zu navigieren.
3. „Questioning“ (Hinterfragen): visuelle Erkundungsstrukturen, die tiefere Einblicke in die Daten ermöglichen sollen. Meist besteht die Möglichkeit durch Interaktionen die Anzeigen zu manipulieren. Der Benutzer kann beispielsweise Projektionen und verknüpfte Grafiken verändern. Derartige Visualisierungen sollen dem Benutzer die Möglichkeit bieten nicht nur „Was“- Fragen, sondern auch „Warum“-Fragen beantworten zu können.

Auch hier reicht die Bandbreite von Visualisierungen, die nur zur Präsentation von Daten geschaffen werden bis zu hoch interaktiven Anwendungen, die dem Benutzer ermöglichen

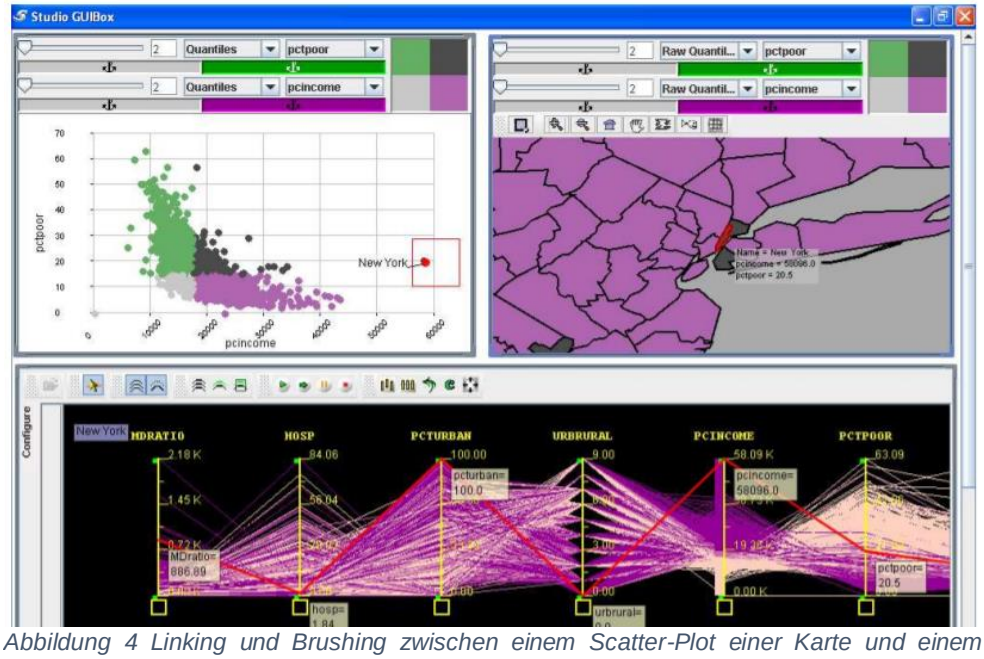

Abbildung 4 Linking und Brushing zwischen einem Scatter-Plot einer Karte und einem Parallelen Koordinatenplot. Quelle: Nöllenburg 2006

nach bestimmten Daten zu suchen, diese zu verknüpfen und so neue Erkenntnisse zu gewinnen. Zwei der wichtigsten Methoden Daten in Visualisierungen so zu verbinden sind *Linking* und *Brushing*. Unter Ersterem versteht man die gleichzeitige Darstellung der Daten in verschiedenen Ansichten. Durch *Brushing* werden auf eben diesen Ansichten gleiche Datenpunkte gekennzeichnet. Abbildung 4 zeigt ein solches Beispiel. Es sind drei Ansichten zu sehen, welche den gleichen Datensatz mit verschiedenen Darstellungsmethoden zeigen. Auf allein drei Visualisierungen ist der gleiche Datenpunkt gekennzeichnet. Durch die kombinierte Darstellung eröffnen sich neue Blickwickel, die der Interpretation dienen und die Analyse der Daten unterstützen (*Nöllenburg*, 2006).

3.2.2 Bedeutung von Geovisualisierung

Geovisualisierungen ermöglichen ein räumliches Verständnis von Konzepten, Zuständen, Prozessen oder Ereignissen der realen Welt zu erhalten. Auch Muster und räumliche Zusammenhänge können durch Darstellungen auf Karten entdeckt werden. Ein berühmtes Beispiel hierfür ist die „Cholera Map“ (Abbildung 5) von *John Snow* (1813 - 1858) aus dem Jahr 1854. Er zeichnete aufgetretenen Cholera Fälle und Wasserpumpen auf einer Karte ein. Das Verständnis für den Zusammenhang zwischen Orten des Auftretens der Krankheit und dem Ort des Auslösers konnte so hergestellt werden, da sich Krankheitsfälle, um eine bestimmte Pumpe sammelten. Diese recht einfache Geovisualisierung zeigt, wie Zusammenhänge erst mit Hilfe von räumlichen Eigenschaften erkannt werden konnten.

Abbildung 5 "Cholera Map" von John Snow aus dem Jahr 1854. Quelle: Nöllenburg 2006

Geovisualisierung ist nicht nur für die Forschung von großer Bedeutung. Sie hilft uns ein mentales Bild der realen Welt in unseren Köpfen zu formen. Die Bedeutung der Geovisualisierung soll ein Zitat von Dodge et al. verdeutlichen:

> „Geovisualization makes our world“ (*Dodge et al.*, 2008)

Denn geographische Visualisierungen bestimmen wie wir uns den realen Raum vorstellen. Es kann dadurch aber auch ein verzerrtes Bild der Wirklichkeit entstehen. Als Beispiel nennen Dodge et al. die Londoner „Tube Map“. Dabei handelt es sich um ein Topogramm, dass die Strecken der Londoner U-Bahn zeigt. Die stark vereinfachte schematische Darstellung, die zum Vorbild für Netzpläne auf der ganzen Welt wurde, formte in den Köpfen der Londoner ein falsches Bild des ‚echten‘ Londons, da Stationen nicht an ihren ‚realen‘ Positionen eingezeichnet waren (*Dodge et al.*, 2008). Ersteller von Karten oder Geovisualisierungen sollten sich demnach den Einfluss ihrer Darstellungen bewusst sein.

3.2.3 Historische Entwicklung

Die Entwicklung der Geovisualisierung ist eng mit der Geschichte der Menschheit und insbesondere mit der Sesshaftwerdung derer verbunden. Karten als Mittel der Kommunikation sind mindestens so alt wie die Sprache selbst (*Dodge et. al*, 2008). Die Geschichte der Kartographie zeigt den vielfältigen Nutzen der Visualisierung von Geodaten. Die Organisation räumlichen Wissens, Verdeutlichung territorialer Ansprüche oder die Erleichterung der Navigation sind einige von ihren Funktionen.

Historisch gesehen, beschränkte sich die Visualisierung von Geodaten zumeist auf planare Darstellungen. Inzwischen ist es jedoch nicht mehr möglich die Darstellungsform so eindimensional zusammenzufassen, da sich durch die technologische Entwicklung die Möglichkeiten der Datenpräsentation geändert hat. Auch die Medien, auf denen Daten dargestellt werden, haben sich im Laufe der Zeit weiterentwickelt. Wurden in den vergangenen Jahrhunderten handgezeichnete oder bedrucke Karten und Globen zur Darstellung oder zur puren analogen Datenspeicherung von Geodaten genutzt, ist die Bandbreite der Präsentationsmöglichkeiten heute weitaus ausgedehnter (*Dodge et. al*, 2008).

Die Weiterentwicklung der Computertechnik wirkt sich auf alle Bereiche der Geovisualisierung aus. Datenerfassung, Produktion und Publikation haben sich durch den raschen technologischen Fortschritt stark verändert. Auch der Zugang zu Geodaten und Geovisualisierungen hat sich durch das Internet vereinfacht. Es ist heute praktisch jedem möglich, Geodaten über Portale wie beispielsweise *Openstreetmap.com* oder *Google* Maps abzurufen. Die Produktion von geographischen Visualisierungen ist heute häufig automatisch generiert und nur kurzweilig. Online Portale wie *MapQuest* produzieren mehr digitale kartographische Produkte als jeder Kartograph es je könnte (*Dodge* et al. 2008 nach *Peterson* 2001). Durch Augmented Reality können geographische Visualisierungen zudem in Echtzeit erstellt werden und haben eine Lebensdauer, die der Länge der Betrachtungszeit auf dem Bildschirm entspricht.

Auch die pure Präsenz von Geovisualisierungen, die uns heute umgibt, ist um ein vielfältiges höher als noch wenige Jahrzehnte zuvor. Grund dafür ist auch die enorm gestiegene Menge an verfügbaren Geodaten. Im Jahr 2001 schätzen *MacEachren&Kraak,* dass circa 80% aller erhobenen Daten einen räumlichen Bezug aufweisen. Daher scheint es nicht verwunderlich, wenn Wood konstatiert:

„Today we live in a map saturated World“ (*Dodge et al.* 2008 nach *Wood* 1992).

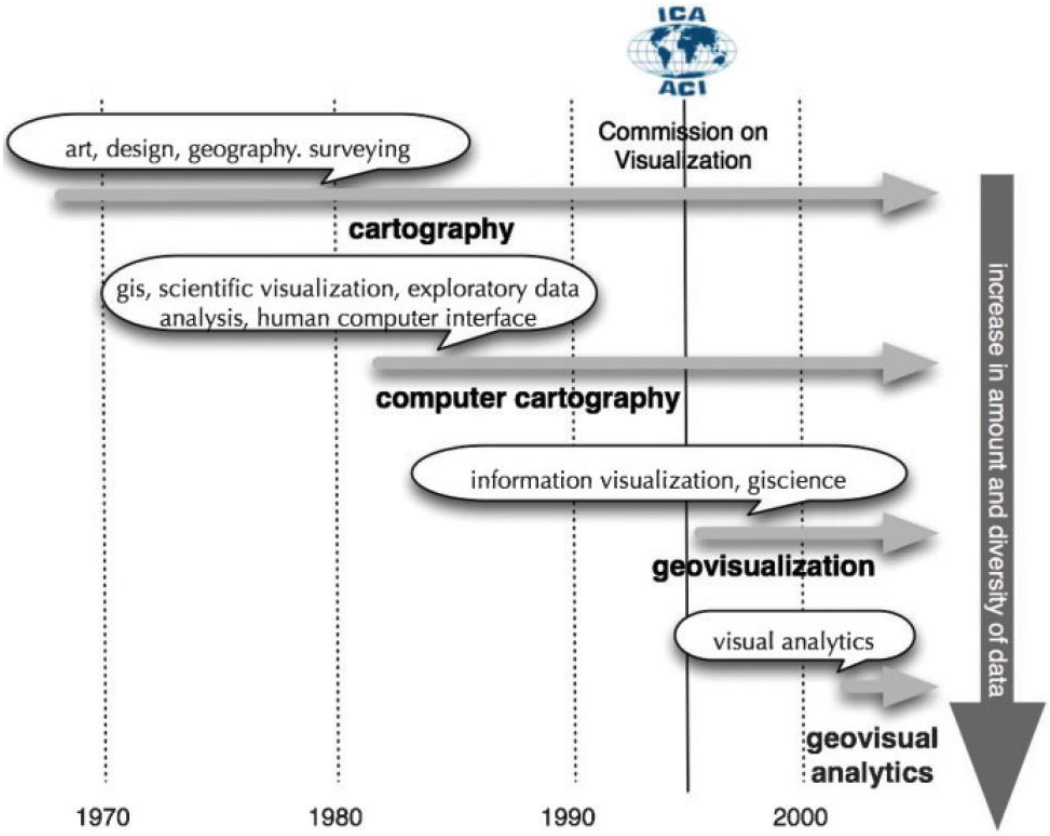

Abbildung 6 Entwicklung der Disziplinen im Umfeld der Geovisualisierung. Quelle: Kraak 2008

Die Entwicklung der Geovisualisierung und Popularität führt laut *Kraak* (2008) aber auch dazu, dass der Term zunehmend falsch benutzt wird. Er bemängelt, dass er als Synonym für „mapping“ eingesetzt wird und als Folge seine Bedeutung verliert. *Kraak* schlägt daher vor Anwendungen, die sich mit dem Map Use Cube zuordnen lassen, mit „Geovisual Analytics“ zu beschreiben. Außerdem müsste in Zukunft eine genauere Definition von „Geovisualisierung“ geschaffen werden, um die Fachbereiche (siehe Abbildung 6) klar zu trennen (*Smith et al.* 2013).

3.2.4 Statische und Dynamische Geovisualisierung

Dieses Kapitel soll einen Überblick auf die Unterschiede statischer und dynamischer Visualisierung legen. Es soll sich dabei vor allem von der Visualisierung im Kontext der Geoinformation handeln. Generell gelten alle Prinzipien jedoch für Visualisierungen in allen Fachbereichen.

Es gibt grundsätzlich zwei Möglichkeiten Informationen mittels Visualisierungen darzustellen. Es stehen statische oder dynamische Visualisierungen zur Verfügung. Unter

statischen Visualisierungen können Darstellungen verstanden werden, welche eine einzelne Abbildung auf analogen oder digitalen Displays zeigen. Bewegungen innerhalb der Abbildung und Interaktion mit der Darstellung sind nicht möglichen. Dynamische Visualisierungen dagegen zeichnen sich durch Animationen innerhalb der Darstellung aus. Daher können diese auch nur auf digitalen Bildschirmen realisiert werden.

Statische Geovisualisierungen greifen oft auf graphische Variablen zurück, welche auch in der Kartographie geläufig sind. Diese wurden von *Bertin* (1918 – 2010) erstmals formuliert. Es handelt sich dabei um Mittel zur Gestaltung für geometrische Grundformen, also Punkte, Linien und Polygone. Es gibt sieben Variablen, die auf diese Elemente angewendet werden können (*MacEachren, 1995*). Die Variablen lauten:

- Position
- Größe
- Farbe
- Helligkeit
- Textur
- Ausrichtung
- Form

Mit diesen Variablen, soll es möglich sein einen Bezug zwischen Erscheinungsform von Kartenelementen und Datenwerten herzustellen. Dabei kann jedoch nicht jede Variable auf jede Grundform angewendet werden. Bertin gibt daher Empfehlungen aus, wie diese Variablen angewendet werden sollen. Zudem stellte er Regeln für Variablen auf, nach denen ersichtlich ist, für welches Skalenniveau sie genutzt werden können (siehe Abbildung 7).

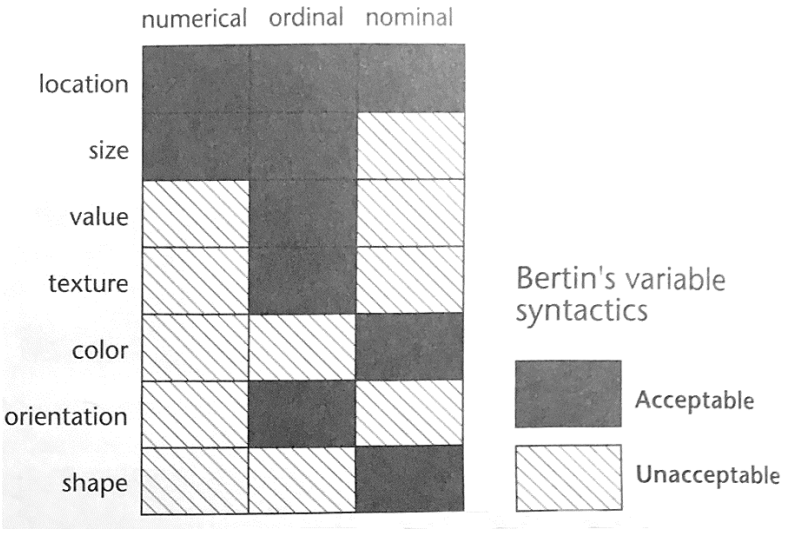

Abbildung 7 graphische Variablen nach Bertin. Eingeteilt nach Verwendungsmöglichkeit der Skalenniveaus. Quelle: MacEachren 1995

Weiterführende ausführliche Informationen zu Bertins graphische Variablen findet sich unter anderem in *MacEachrens* Veröffentlichung „How Maps Work“.

Eine besondere Herausforderung für (Geo)Visualisierungen ist die Darstellung der temporalen Dimension. Ein bekanntes Beispiel, dass auch statische Visualisierungen in der Lage sind zeitbezogene Daten abzubilden ist Minards Karte vom französischen Russlandfeldzug, angeführt von Napoleon im Jahr 1812 (siehe Abbildung 8).

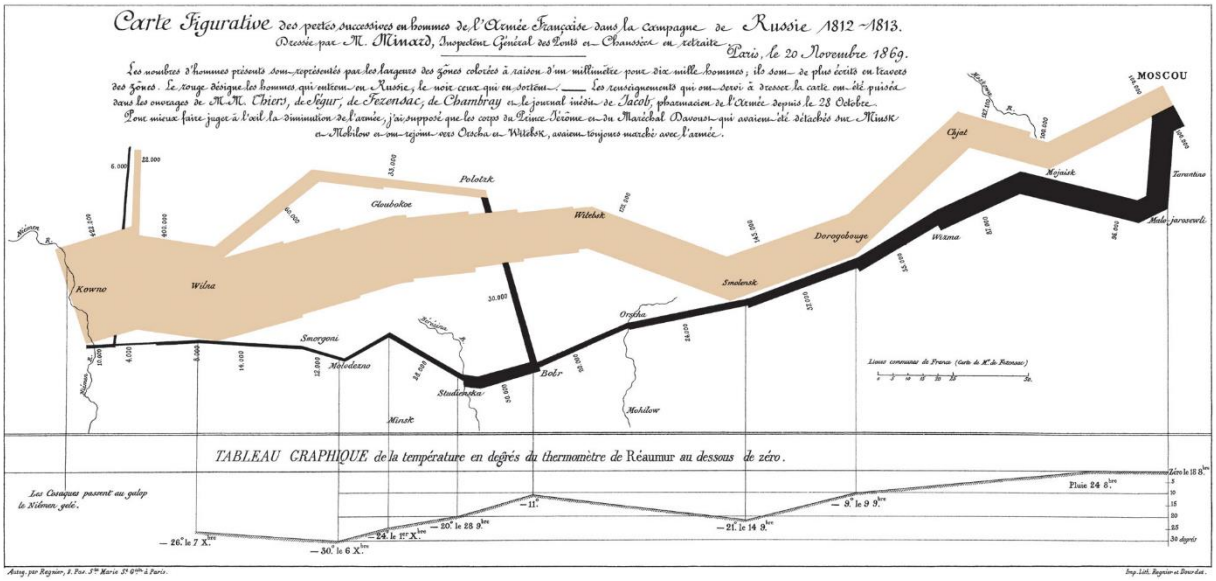

Abbildung 8 Minards Visualisierung über den Russlandfeldzug 1812. Dargestellt sind Position und Anzahl der Truppen sowie die Temperaturen. Quelle: commons.wikimedia.org

Es handelt sich dabei um eine multivariate Darstellung, welche die Route und die Verluste der Truppen sowie die Temperatur über einen Zeitraum von mehreren Monaten während des Marsches durch Russland zeigt. Die temporale Komponente wird durch Daten angegeben, die an verschiedenen Punkten der Linie angebracht ist, welche die Truppenstärke vermitteln soll. Durch die mit der Zeit abnehmende Anzahl an Soldaten wird zudem eine Entwicklung über einen bestimmten Zeitraum deutlich.

Neben der Datumsangabe können statische Bilder auch durch angedeutete Bewegungen mental dynamischer Bilder bei dem Betrachter erzeugen (*Ploetzer et al*., 2008). *Ploetzer et al.* führen dazu das Beispiel eines Bildes an, auf dem ein Kind einem Ball auf einer Straße hinterherrennt. Ein im Hintergrund ankommendes Auto erzeugt im Geist der Betrachter den Schluss eines eventuell kommenden Aufpralls. Ebenso können Pfeile auf einer Abbildung Bewegungen imitieren oder eine zeitliche Abfolge von Ereignissen anzeigen.

Dennoch müssen derartige Darstellungen Kompromisse eingehen. *Nöllenburg* bemerkt dazu, dass zweidimensionale statische Darstellungen zwar in der Lage sind, Verläufe weniger Objekte darzustellen, jedoch könnten Geschwindigkeiten und Treffpunkte zweier oder mehrerer Objekte wiederum nicht richtig abgebildet werden (*Nöllenburg*, 2006). Daher bieten sich weitere Darstellungsmöglichkeiten an, um derartige Inhalte zu visualisieren. Neben statischen Karten gibt es laut *Kraak* (2002) zwei weitere Möglichkeiten zeitbezogene Ereignisse in einer Visualisierung darzustellen. Einerseits würde sich eine Serie von statischen Abbildungen anbieten, die nebeneinander aufgereiht Veränderungen anzeigen können. Zum anderen drängen sich animierte Darstellungen auf, wenn es darum geht Bewegungen darzustellen oder die temporale Dimension einzubeziehen.

Längst sind dynamische Darstellungen und Interaktivität ein gängiges Mittel in der Geoinformation. Mit steigender Popularität von Computern wurden nach und nach immer mehr zwei- und dreidimensionale kartographische Animationen erstellt, so dass real-time Landschaftsszenen oder interaktive Kartenanimationen im Internet heute gang und gäbe sind (*Fabrikant* 2005). Technisch gesehen handelt es sich bei dynamischen Visualisierungen um Animationen, also eine Aneinanderreihung einzelner statischer Abbildungen beziehungsweise Zustände.

Neben der Darstellung von Zeitabhängigen Phänomenen, Ereignissen oder Konzepten können Animationen den Fokus auf bestimmte Elemente der Visualisierung lenken. Auch nichtdynamische Daten könne in animierter Form dargestellt werden, in dem zwei oder mehrere Zustände der Abbildung nacheinander eingeblendet werden. Dieses Verfahren

ist auch unter dem Begriff *blinking* bekannt, und wird hauptsächlich dazu genutzt, um verschiedene statische Darstellungen vergleichend darzustellen (*Fabrikant*, 2005).

Um animierte Geovisualisierungen zu realisieren, können nach *MacEachren* (1995), die soeben vorgestellten graphischen Variablen von *Bertin*, um sechs weitere ergänzt werden:

- Dauer: Dauer zwischen zwei Zuständen eines Elementes
- Änderungsrate: Größenordnung der Veränderung eines Elements pro Frame/Szene oder Zeiteinheit
- Reihenfolge: Abfolge der Ereignisse; zum Beispiel chronologisch oder aufsteigend nach einem quantitativen Wert
- Anzeigezeitpunkt: chronologische Position, an dem sich ein Element innerhalb der Animation verändert
- Frequenz: Anzahl der identifizierbaren Zustände pro Frame/Zeiteinheit
- Synchronisation: Zeitpunkte verschiedener Starts von Animationen

Auch hier wurden Regeln aufgestellt, die festlegen, welches Skalenniveau durch die Variablen dargestellt werden können. Auf Abbildung 9 sind die Empfehlungen von *MacEachren* zu sehen, zu welchen der Skalenniveaus die neuen Variablen eingesetzt werden können.

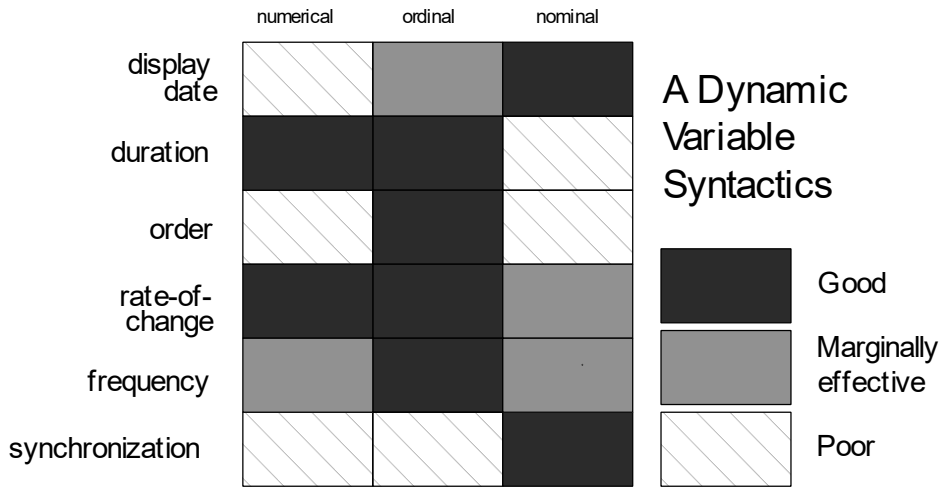

Abbildung 9 Dynamische Variablen. Basierend auf MacEachren, 1995

Neben diesen graphischen Variablen können Animationen aber auch Abbilder realer Prozesse zeigen, die weniger Abstrakt eher einem Film gleichen. Im Umfeld der Geoinformation könnten dies zum Beispiel auseinanderdriftende Platten sein, um die Plattentektonik zu erklären. Des Weiteren gibt es die Möglichkeit animierter Graphen, welche sich im Laufe der Zeit ändern mit Visualisierungen zu kombinieren (*Schnotz&Lowe,* 2008).

Je mehr Veränderungen, die Visualisierung jedoch beinhaltet, desto größere Mengen an Informationen muss der Betrachter gegebenenfalls im Gedächtnis speichern und verarbeiten. Zudem können sich Elemente auf dynamischen Visualisierungen zur gleichen Zeit verändern. Auch hier ist der Betrachter gefordert, den Überblick über die gesamte Darstellung zu behalten und Informationen simultan zu verarbeiten (*Windhager*, 2011). Dies steht in gewisser Weise im Gegensatz zu einem der großen Vorteile von statischer Visualisierung, welche Informationen komprimiert präsentieren und so Speicher- und Verarbeitungsressourcen für Betrachter auf die Visualisierung umlegen (*Card*, 2008). Interaktionen mit der Darstellung können diesen Problemen entgegenwirken. Denn durch das Erlauben des Anhaltens der Animation oder des Zulassens von manuellen Tempoanpassungen oder gar Zeitsprüngen, könnten Benutzer bestimmte Szenen erneut betrachten (*Nöllenburg*, 2006). Dennoch betonen *Schnotz&Lowe* (2008), dass dynamische Animationen psychologisch gesehen, von dem gleichen wahrnehmenden und kognitiven System des Menschen bearbeitet wird. Animationen sind daher als Erweiterung von statischen Visualisierungen zu sehen. Es müssten keine neuen Regeln erfunden werden, um mit dynamischen Visualisierungen Lernenden Prozesse zu erklären.

Trotzdem gibt *Nöllenburg* zu bedenken, dass mit Animationen behutsam umgegangen werden möge und sich der Verfasser stets die Frage stellen sollte: „Warum sollte ich diese Daten animieren?“. Das Mittel der Animation sollte demnach nur dann eingesetzt werden, wenn es auch tatsächlich Sinn macht.

3.3 Zwei und Dreidimensionale Visualisierung

Um räumliche Daten zu präsentieren können entweder zwei- oder dreidimensionale Darstellungen erstellt werden. Je nach Anwendungsfall ist abzuwägen, wie die Informationen am besten zu übermitteln sind. Nach *Tory et al.* (2006) eignen sich 2D Darstellungen beispielsweise, um konkrete Zusammenhänge zu veranschaulichen. Des Weiteren wird erwähnt, dass zweidimensionalen Visualisierungen für präzise Navigation und Entfernungsmessungen geeignet seien. 3D Darstellungen könnten hingegen zur Vermittlung dreidimensionaler Räume und Raumeindrücke genutzt werden.

Neben dem Anwendungsbereich ist auch die Datenkomplexität für die optimale Darstellung der Informationen entscheidend. *Dübel et al.* (2014) weisen darauf hin, dass Zusammenhänge zwischen einer hohen Anzahl an Objekten besser auf dreidimensionalen Visualisierungen wahrgenommen werden. Auch *Carvalho* (2011) merkt an, dass auf zweidimensionalen Visualisierungen Informationen verloren gehen können, da nur ein Blickwinkel auf die Daten dargestellt werden kann. Durch Einführung einer weiteren räumlichen Dimension, könnte man diesem Problem entgegenwirken. Aber auch im dreidimensionalen Anzeigebereich gibt es verschiedene Anzeigemöglichkeiten. *Herman et al. (2018)* weisen auf den Unterschied zwischen pseudo und realen 3D-Visualisierungen hin. Bei pseudo-3D-Darstellungen handelt es sich dabei um dreidimensionale Abbildungen auf flachen Bildschirmen. Hier werden Tiefeneindrücke nur durch monokulare Tiefeneindrücke erzeugt. Dies führt dazu, dass Betrachter Distanzen auf der z-Achse nur schwer einschätzen können. Währenddessen sind Entfernungen und Muster auf den orthogonalen x- und y- Achsen problemlos zu erkennen. Dies ist auch der Grund warum zweidimensionale Darstellungen besser für präzise Messungen geeignet sind (*Nöllenburg*, 2006).

Reale 3D-Darstellungen beziehen die Stereoskopie mit ein, welche jedoch zumeist nicht ohne Weiteres auf Bildschirmen dargestellt werden kann. Um den dreidimensionalen

Raumeindruck zu erzeugen sind weitere Hilfsmittel, wie bestimmte Brillen, notwendig. Diese Technologie ist aufgrund der zusätzlich erforderlichen Peripheriegeräte weniger weit verbreitet. *Herman et al.* (2018) berichten weiter, dass die Art der 3D Darstellung jedoch weniger Einwirkung auf die Verständlichkeit des Inhaltes hat als das Level an Interaktivität mit der der Benutzer mit der Visualisierung kommuniziert. Dabei sei es wichtig, dass verschiedene Möglichkeiten zur Orientierung angeboten werden. Im Falle dreidimensionaler Darstellungen auf flachen Bildschirmen seien dies vor allem Zoomen und Möglichkeiten, die 3D Umgebung zu überfliegen. In immersiven realen dreidimensionalen Anwendungen müsste dagegen zusätzlich die Möglichkeit gegeben werden durch die virtuelle Welt laufen zu können. Generell sind langsame Fortbewegungsmöglichkeiten sinnvoll, da schnelle Modi, wie beispielsweise Teleportation, Orientierungsverluste beim Betrachter nach sich ziehen könnten (*Nöllenburg*, 2006).

Zur Verständlichkeit der Inhalte einer statischen Geovisualisierung verglichen *Wahyudi et al.* (2020) zwei Abbildungen eines Erdrutsches, welche einmal in 2D und einmal in 3D dargestellt wurde (siehe Abbildung 10).

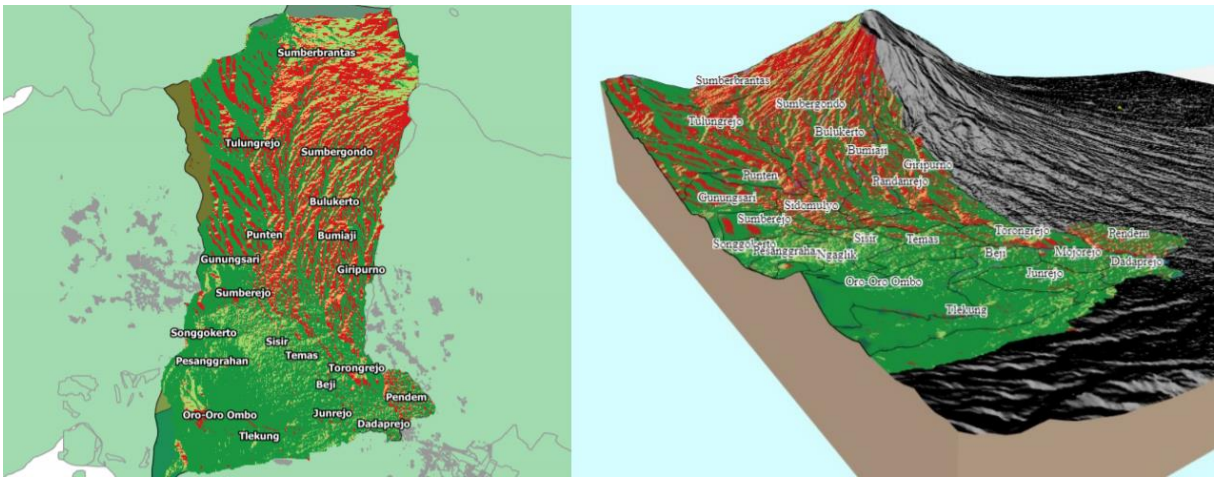

Abbildung 10 Zwei- und Dreidimensionale Darstellung eines Erdrutsches. Quelle: Wahyudi et al. 2020

Bei der Befragung von Teilnehmern der Studie zeigte sich, dass die dreidimensionale Visualisierung die dargestellten Informationen besser vermitteln konnte. Vor allem die Ursache des Erdrutsches, also das Höhenprofil, konnte von den meisten Teilnehmern der Studie besser erkannt werden. Es wurde aber zusätzlich festgestellt, dass mit zunehmendem Alter und Bildungsgrad auch alle Informationen der zweidimensionalen Abbildung erfasst wurden konnte (*Wayhudi et al.*, 2020). Das bedeutet, dass neben den bereits angeführten Entscheidungskriterien auch das Zielpublikum dafür ausschlaggebend ist ob eine 2D oder 3D Visualisierungen angefertigt werden soll.

4. Vektorfelder

In dieser Arbeit dienen Vektorfelder als Ausgangspunkt zur Darstellung atmosphärischer Strömungen. Das folgende Kapitel wird sich daher mit der Thematik dieser Felder beschäftigen. Zunächst wird erläutert was genau unter einem Vektorfeld zu verstehen ist. Anschließend wird der Stand der Forschung sowie Anwendungsmöglichkeiten aufgezeigt. Zuletzt wird in diesem Kapitel beschrieben, wo Vektorfelder in der Geoinformation zu finden sind.

4.1 Definition

Unter einem Vektorfeld ist ein Raum zu verstehen, in dem jeder Punkt durch eine gerichtete Größe (Vektor) beschrieben ist. Ein Vektor wird dabei entweder durch seine horizontalen und vertikalen Komponenten, die meist als u und v gekennzeichnet sind, oder durch seine Größe und Richtung beschrieben.

Vektorfelder, welche auch Strömungsfelder genannt werden, können dazu genutzt werden, Strömungsverhältnisse zu beschreiben. Die Visualisierung eines Vektorfeldes kann die Richtung und Geschwindigkeit des Flusses an den abgebildeten Punkten darstellen. Abbildung 11 zeigt ein Beispiel eines Feldes, in dem die Richtung der Strömung durch Pfeile signalisiert wird.

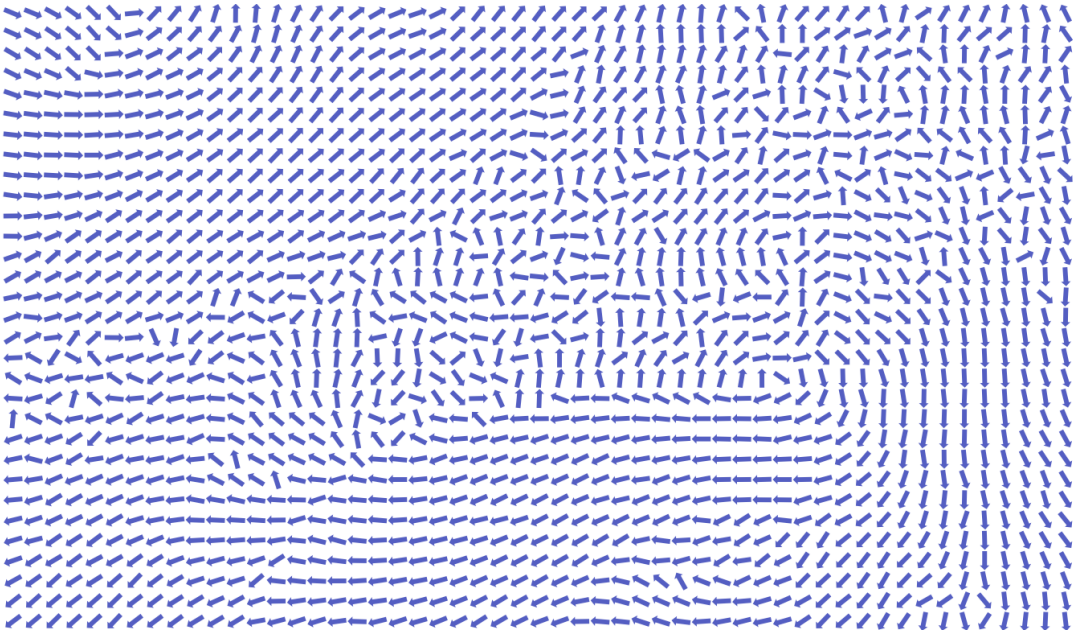

Abbildung 11 Beispielhafte Darstellung eines zweidimensionalen Vektorfeldes. Eigene Darstellung

Die Visualisierung von Vektorfeldern ist bereits seit einigen Jahrzehnten Thema in unterschiedlichen wissenschaftlichen Disziplinen. Die Darstellung von Strömungen in der Luft oder in Flüssigkeiten ist für viele Fachbereiche von großer Bedeutung. Beispielsweise beschäftigt sich die Mathematik und Geometrie, Physik (insbesondere Strömungsmechanik, Elektrodynamik, Aeronautik), Meteorologie oder auch die digitale Bildbearbeitung mit dieser Thematik (*Hellmann&Hessellink* 1991, *Max&Crawfis* 1992, *Gelder&Wilhelm* 1992*, Crawfis&Max* 1992, *Stalling* 1998). Vor allem am Anfang der 1990er Jahre wurden Arbeiten veröffentlicht, die sich mit Visualisierungsmöglichkeiten von Vektorfeldern beschäftigten. Zentrales Thema dabei war es, die Topologie dieser Felder möglichst gut für das menschliche Auge verständlich darzustellen. Vor allem sind es die sogenannten kritischen Punkte in Vektorfeldern, die ersichtlich gemacht werden sollten. Als kritische Punkte werden jene Stellen bezeichnet, an denen die Größe eines Vektors gleich null ist. Abbildung 12 zeigt die sechs verschiedenen Möglichkeiten dieser Punkte in einem zweidimensionalen Strömungsfeld. Kritische Punkte sind von großer Bedeutung, da sie den globalen Fluss der Strömung in einem Vektorfeld bestimmen (*Helmann&Hessellink*, 1991).

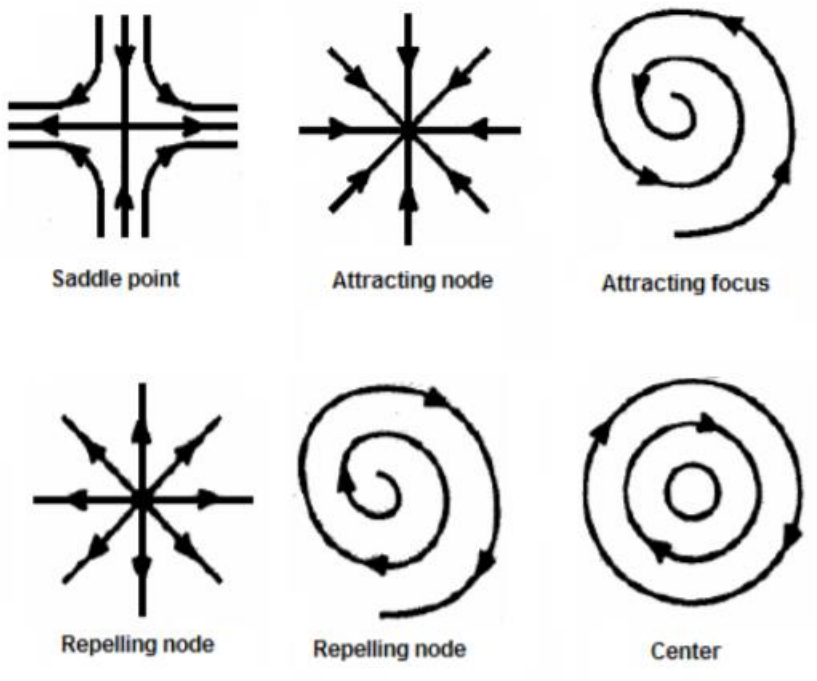

Abbildung 12 Kritische Punkte in einem zweidimensionalen Vektorfeld. Quelle: nach Helmann&Hesselink, 1989

4.2 Darstellung von Vektorfeldern

Es gibt vielfältige Darstellungsmöglichkeiten von Vektorfeldern. Nach *Cuntz* (2009) kann dabei zwischen drei Varianten der Visualisierung unterschieden werden:

- Direkte Visualisierung
- Texturbasierte Visualisierung
- Geometrische Visualisierung

Unter **direkter Visualisierung** werden meist statische Abbildungen verstanden, welche den Fluss des Feldes durch Symbole wiedergeben. Bei dieser einfachsten aller Möglichkeiten ein Vektorfeld abzubilden, gilt es zu beachten, dass lediglich ein bestimmter Zeitpunkt eines möglicherweise dynamischen Strömungsfeldes abgebildet werden kann. Abbildung 13 zeigt verschiedene Methoden, Vektorfelder statisch und direkt zu visualisieren. Abgebildet sind drei Möglichkeiten:

- Grid: Symbole auf einem regelmäßigen Raster
- JIT (Jittered/Verwackelt): Symbole auf einem unregelmäßigen Raster
- LIT (Line Integral Tensor/ Linienintegral Tensor): unregelmäßige Darstellung von Symbolen. Eine spezielle Darstellungsform, die sich Konzepten aus der Ölmalerei bedient

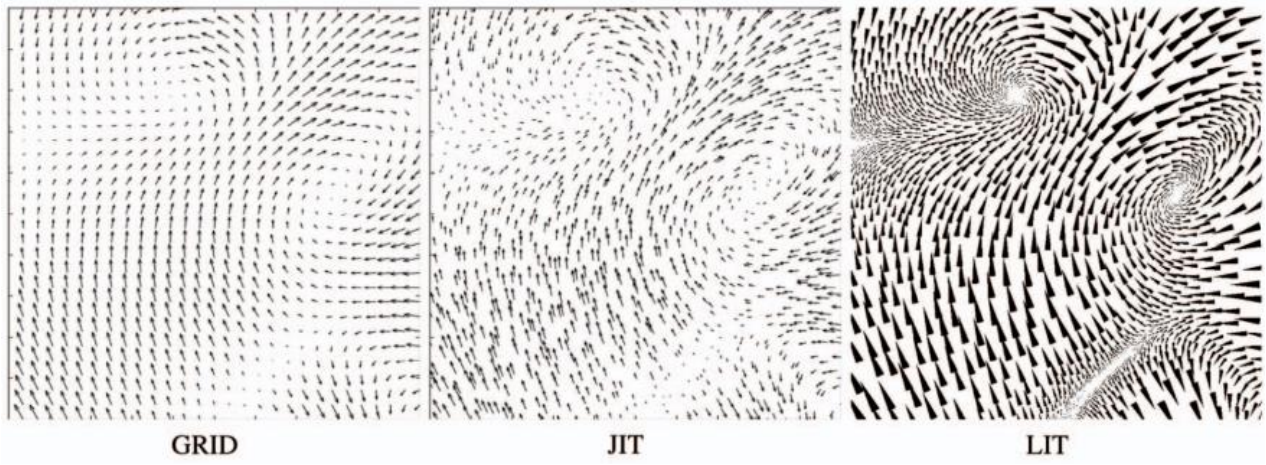

Abbildung 13 Unterschiedliche Möglichkeiten Vektorfelder statisch zu visualisieren. Nach Laidlaw et al., 2005

Auf der Abbildung ist zu erkennen, dass die Informationen des zugrunde liegenden Vektors durch Winkel und Größe des Pfeiles, welcher die Strömung an der jeweiligen Stelle wiedergibt, repräsentiert wird. Hier liegt auch der Nachteil der direkten

Visualisierungen. Den einzelnen Vektoren steht nur ein begrenzter Raum zur Verfügung. Anderenfalls kommt es zu Überlappungen mit benachbarten Symbolen. Durch die Darstellung dreidimensionaler Vektorfelder durch Pfeilsymbole kann dieser Effekt verstärkt werden und Informationen können aufgrund chaotischer Darstellungen verloren gehen.

Um den verfügbaren Platz bestmöglich auszuschöpfen wurden **texturbasierte Methoden** entwickelt (*Stalling*, 1998). Dabei handelt es sich um engmaschige Muster, welche so auf das Vektorfeld moduliert werden, dass die Strömungen aus der Visualisierung abgelesen werden können. Bekannte Beispiele sind das Spot-Noise verfahren von *Van Wijk* aus dem Jahr 1991 und LIC-Verfahren (Line-Integral-Convolution), was zuerst von *Cabral&Leeodom* im Jahr 1993 vorgestellt wurde (siehe Abbildung 14).

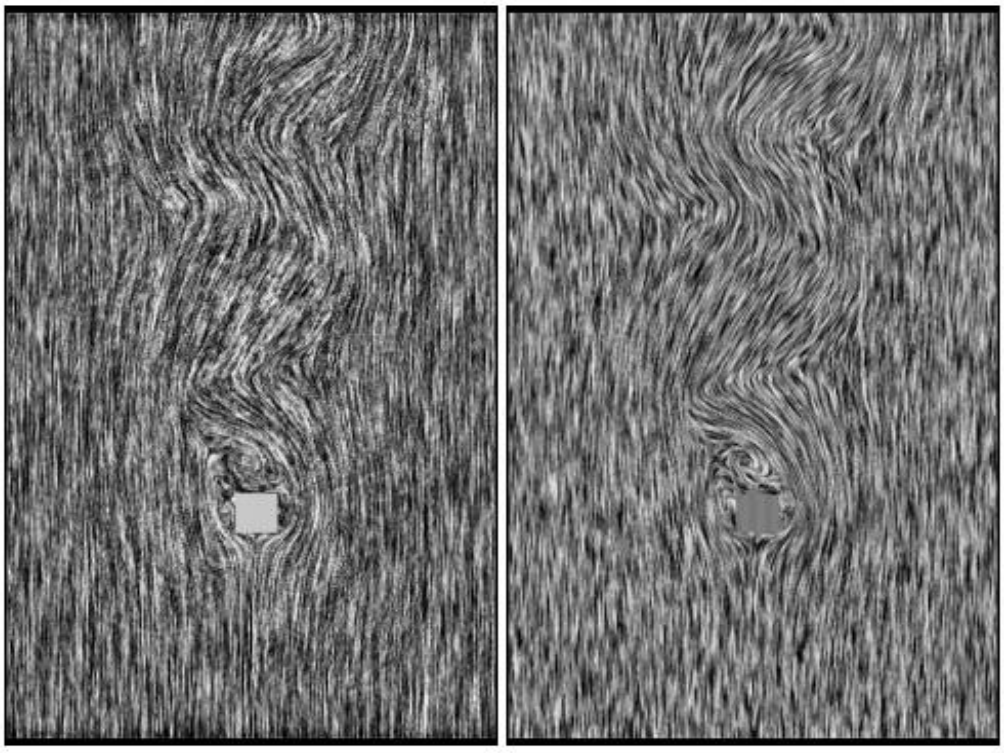

Abbildung 14 Vergleich zwischen LIC (links) und Spot Noise (rechts). Quelle: De Leeuw&Van Liere,1998

Die dritte Möglichkeit Vektorfelder zu symbolisieren, sind **geometrische Verfahren**. Dazu werden geometrische Objekte wie Linien, Punkte oder Oberflächen genutzt, um Strömungen wiederzugeben. Bekannte Linienverfahren sind Streamlines („Stromlinien“), Pathlines („Pfadlinien“) und Streaklines. Unter Stromlinien versteht man Linien, die tangential zu den Vektoren eines Strömungsfeldes verlaufen. Sie zeichnen den Verlauf

eines imaginären masselosen Partikels nach, das zu einem bestimmten Zeitpunkt auf das Vektorfeld gelegt wurde und sich entsprechend zu den Vektoren durch das Feld bewegt. Streamlines können zufällig verteilt, oder optimal auf der zur Verfügung stehenden Fläche verteilt werden (*Turk&Banks*, 1996). Abbildung 15 zeigt beide Möglichkeiten.

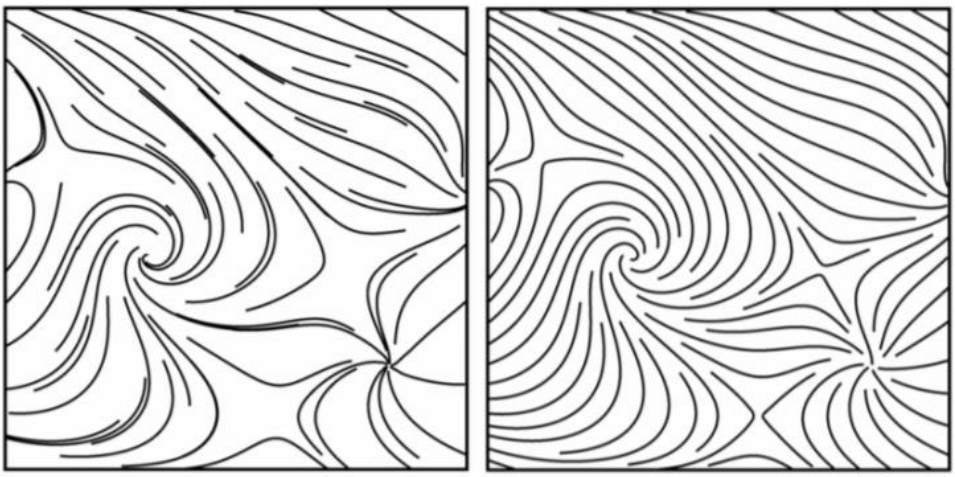

Abbildung 15 Verbesserte Darstellung eines Vektorfeldes durch Strömungslinienoptimierung. Quelle: Turk&Banks, 1996

Eine Pfadlinie verhält sich ähnlich zur Stromlinie, mit der Ausnahme, dass kein bestimmter Zeitpunkt innerhalb eines dynamischen Vektorfeldes gewählt wurde. Die Pfadlinie beschreibt demnach ein Partikel welches sich in einem sich verändertem Strömungsfeld bewegt. Ähnliche verhält es sich mit Streaklines. Dabei handelt es sich um Linien, die die Pfade mehrere Teilchen, welche nacheinander am gleichen Punkt freigelassen wurden, nachzeichnen. Handelt es sich bei dem benutzen Strömungsfeld um ein statisches Vektorfeld, so gleichen die Streaklines den Strömungslinien (*Cuntz*, 2005).

Eine weitere Möglichkeit Vektorfelder geometrisch zu visualisieren ist der Einsatz von Partikelsystemen. Dabei werden geometrische Objekte, meist Punkte, welche als Partikel bezeichnet werden, auf dem Vektorfeld verteilt. Die Partikel verhalten sich nun so, wie es die Strömung an ihrer Position vorgibt. Während der Animation werden fortlaufend neue Partikel auf dem Vektorfeld verteilt während andere wieder aus dem System entfernt werden. So entsteht der Eindruck einer stetigen Strömung. Der Einsatz von dynamischen Partikelvisualisierungen von Vektorfeldern reicht dabei von simpler zweidimensionaler Animation bis hin zu dreidimensionalen virtuellen Umgebungen (Kuester et al., 2001). Partikelsysteme werden im nächsten Kapitel ausführlich beschrieben. Zuvor soll noch ein Blick auf Vektorfelder in der Geoinformation geworfen werden.

4.3 Vektorfelder zur Geovisualisierung

Anwendungen und Untersuchungen zu Vektorfeldern im Rahmen der Geoinformation verteilen sich auf vielfältige Forschungsfelder. *Li&Hodgson* (2004) geben eine Übersicht zu welchen Themen Vektoren und speziell Vektorfelder zum Tragen kommen. Dazu zählen zum Beispiel Untersuchungen zu Bewegungsmustern, digitale Oberflächenmodellierung, Modellierungen zur Sonneneinstrahlung oder die Evaluation von Sanddünen.

Ein Feld, in dem schon seit einigen Jahrzehnten Untersuchungen angestellt werden, ist die Darstellung von Wetterkarten sowie Analysen von Vektorfelder, die Windrichtung und -geschwindigkeit darstellen. Abbildung 16 zeigt eine typische Abbildung, auf der atmosphärische Strömungen durch Pfeile gekennzeichnet sind.

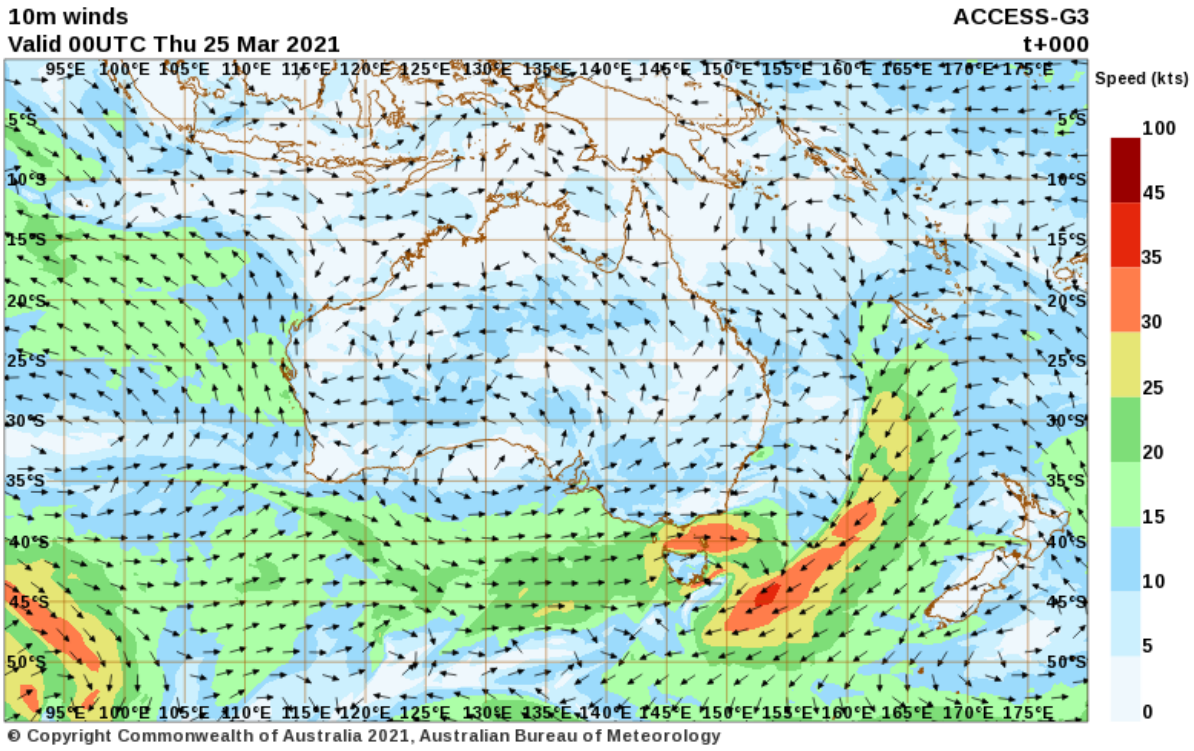

Abbildung 16 Vektorfeld, welches die Windströmungen über Australien anzeigt. Quelle: Australian Bureau of Meteorology

Wetterkarten greifen darüber hinaus auf eine eigene Symbolik für die Strömung zurück. Durch spezielle Windpfeile können Richtung und Geschwindigkeit durch gleichmäßige Symbole dargestellt werden (siehe Abbildung 17). Dadurch können weiter oben angesprochene Überlappungen vermieden werden, da die Symbolgröße nicht zur Darstellung eines Wertes genutzt werden muss. Dass Geschwindigkeit und Richtung zugleich auf Vektorfeldern dargestellt werden sollten, stellten *Klink&Willmott* 1989 fest. Andernfalls könnten Betrachter die Windkarte fehlinterpretieren.

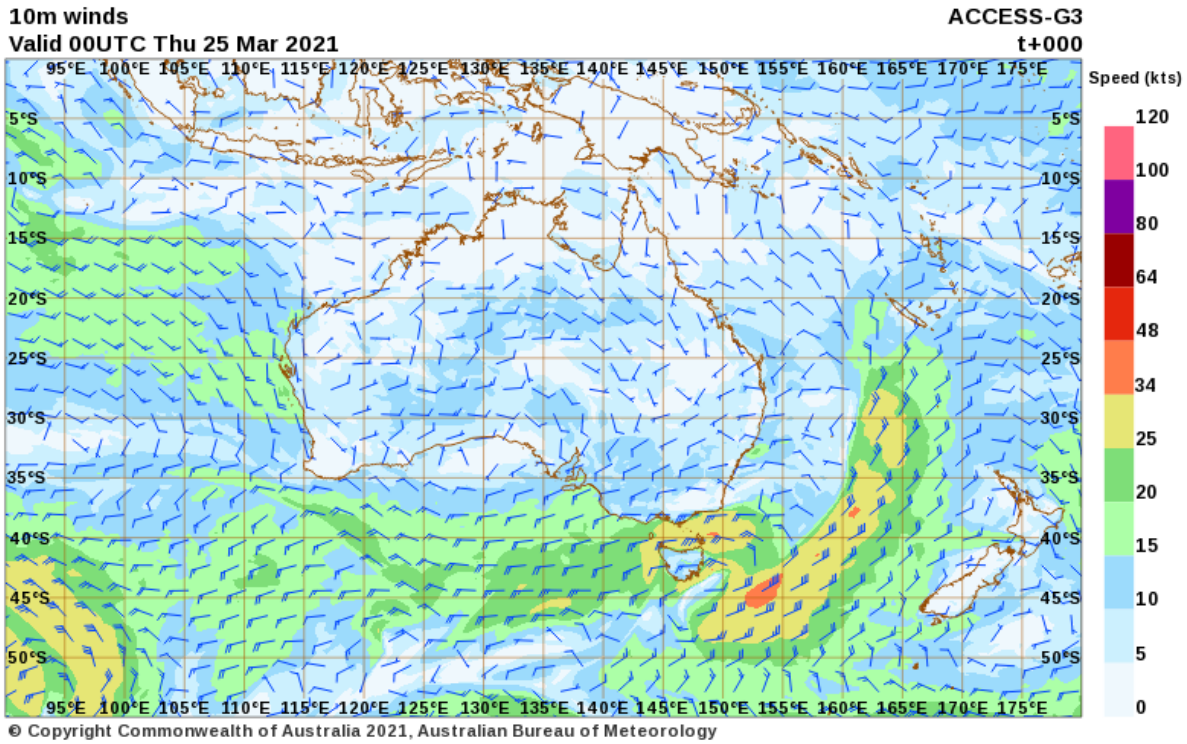

Abbildung 17 Das gleiche Vektorfeld aus Abbildung 10 symbolisiert mit Barben. Quelle: Australian Bureau of Meteorology

Die dynamische Visualisierung von Wind auf der Grundlage von Vektorfeldern wurde bereits 1992 von *Max et al.* umgesetzt. Mithilfe von sich bewegenden Wolkentexturen wurden die Windrichtungen und Geschwindigkeiten simuliert.

Jedoch fehlt bisher ein Datentyp um Vektorfelder für geographische Analysen effizienter zu gestalten. Bisher müssen Anwender mit Skalarfeldern arbeiten, welche in einem mehrbandigen Rasterformat wie etwa TIFF oder GRIB zusammengefasst sind. Jedes Skalarfeld speichert dabei nur ein Element des Vektors. Es müssen daher bisweilen mindestens zwei Raster vorhanden sein um die u und v Komponenten der Vektoren getrennt zu speichern. Demnach würden in einem neuen Rasterformat Zellwerte Vektoren darstellen, anstelle von skalaren Werten. Im Jahr 2005 stellten *Wang&Pullar* ein

Datenmodell vor, mit dem Geoinformationssysteme getrennte Skalarwerte als Vektoren lesen und verarbeiten könnte. In modernen Geoinformationssystemen wurden derartige Datenmodelle bisher jedoch nicht implementiert. Vektoroperationen wie „VectorField“ oder „Partikelverfolgung“ der Software ArcGis, benötigen beispielsweise für die Komponenten der Vektoren getrennte Eingaberaster (ArcGis, 2021).

5. Partikelsysteme

Dieses Kapitel widmet sich dem Thema der Partikelsysteme und all ihren Erscheinungsformen und Bestandteilen.

5.1 Definition und Hintergrund

Als einer der ersten hat sich *Reeves* 1985 mit Partikelsystemen beschäftigt. Für eine möglichst realistische Darstellung von Flammen entwarf er 1983 eins der ersten Partikelsysteme. Seine Definition von Partikelsystemen lautet:

„A particle system is a collection of many minute particles that together represent a fuzzy object“ (*Reeves*, 1983)

Partikelsysteme waren demnach zunächst eine Möglichkeit unscharfe Körper digital darzustellen. Es handelt sich dabei um Systeme, die aus vielen kleinen Objekten, den Partikeln, bestehen, welche gemeinsam einen unscharfen Körper darstellen. Jedes Partikel hat eigens zugewiesene Eigenschaften wie Größe, Farbe, Geschwindigkeit und Richtung und bewegt sich meist unabhängig zu anderen Partikeln nach vorgegebenen Regeln. Zusammen bilden sie das Volumen des darzustellenden Körpers ohne dabei feste Grenzen zu bilden. So konnte *Reeves* in seiner Animation Feuer und Explosionen abbilden. Weitere unscharfe Objekte die so animiert werden können sind beispielsweise Rauch, Nebel oder Wolken. Partikelsysteme wurden zunächst in der Unterhaltungsindustrie eingesetzt, um eben jenen Effekt zu erzeugen (siehe Abbildung 18). Nicht nur in Filmen wurde diese neue Technik genutzt, sondern auch Computerspiele konnten davon Gebrauch machen um beispielsweise Explosionen in Echtzeit zu animieren.

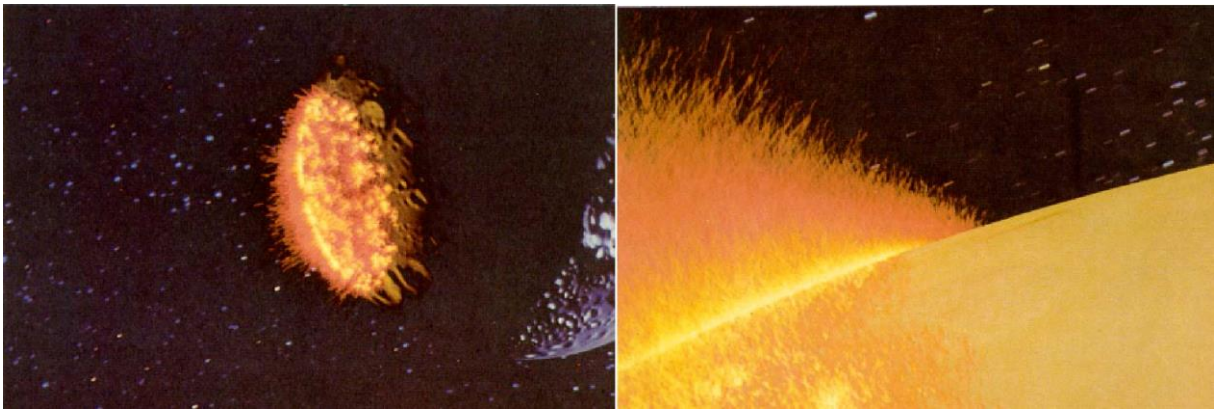

Abbildung 18 Explosion visualisiert durch ein Partikelsystem, Quelle: Reeves 1983

Es zeigte sich jedoch wenig später, dass Partikelsysteme nicht nur zur Darstellung von unscharfen Körpern genutzt werden konnten. Denn im Jahr 1992 stellten *Szeliski&Tonnesen* ein Partikelsystem vor, bei dem sich die einzelnen Partikel voneinander abhängig bewegen. Zudem führten Sie ausgerichtete Partikel ein, die in der Form von kleinen Plättchen Oberflächen formen konnten. Mit diesen „Surfel" getauften Partikeln konnten nun auch feste Oberflächen mit Partikelsystemen gebildet werden (*Szeliski&Tonnesen,* 1992). Da sich einzelne Elemente des Systems zueinander abhängig bewegen, können verformbare Körper wie Gewebe oder elastisches Plastik in Echtzeit modelliert werden.

Auch *Reeves* stellte wenige Zeit später ein weiteres Partikelsystem vor, welches einige grundsätzliche Unterschiede zu seinem ursprünglichen System aufwies. Im Jahr 1995 entwickelte er zusammen mit *Blau* ein System, bei dem Partikel statisch und voneinander abhängig generiert wurden. Auch mit dieser Technik ließen sich Objekte mit ‚echten' Grenzen darstellen. Dieses System wurde als strukturiertes Partikelsystem bekannt und eignete sich Wald- und Graslandschaften zu konstruieren (*Reeves&Blau*, 1995). In *Kapitel 5.2* wird noch einmal auf den Unterschied zwischen strukturierten und stochastischen Partikelsystemen eingegangen.

Weitere Arbeiten über Partikelsysteme beschäftigten sich mit der Implementierung verschiedener Naturkräfte wie Gravitation oder Wind. 1986 stellten *Yeager&Upson* ein System vor welches in Ansätzen auf *Reeves* Partikelsystem von 1983 beruhte. Es sollte die Oberfläche des Jupiters für den Film „2010" darstellen. Um die Wirbelstürme abbilden zu können, basierte das Partikelsystem auf einem Vektorfeld, welches die Flugbahnen der Partikel bestimmt. Auch die Corioliskraft wurde beachtet und die Trajektorie der Partikel entsprechend berechnet. Weiterhin beschreiben *Dorsey et al.* (1996) ein Partikelsystem, mit welchem Vergitterungen auf Oberflächen simuliert werden können. Partikel werden dabei von Gravitation, Wind, Reibung, Rauheit und weiteren Parametern kontrolliert und bewegen sich entlang der Oberfläche unregelmäßiger Körper. Die Arbeit zeigt, dass Partikelsysteme nicht nur für Animationen, sondern auch für Simulationen und Vorhersagen genutzt werden können.

Heute nutzen nahezu allen modernen Videospielen und Animationsfilme Partikelsysteme, um volumetrische Objekte oder Effekte darzustellen. Dazu gehören zum Beispiel Explosionen, Funken, Wolken/Nebel oder Wasser (*Hastings et al.,* 2009). Mit speziellen Regeln, wie sich Partikel untereinander verhalten sollen, zeigte *Reynolds* 1987 darüber hinaus, wie mit einem Partikelsystem Herden- beziehungsweise Schwarmverhalten von

Tieren nachempfunden werden kann. Auch dies kann in Animationen eingesetzt werden um realistischere Umgebungen zu gestalten.

Partikelsysteme sind jedoch nicht nur der Unterhaltungsindustrie von großem Nutzen. Auch andere Bereiche, können diese Systeme zur Simulation und damit zur Vorhersage nutzen. *Huang* zeigten 2009, wie mit Hilfe eines Partikelsystems die Luftzirkulation eines unterirdischen Bergwerkes simuliert werden kann. Dies kann bei der Planung zukünftiger Anlagen genutzt werden zur Arbeitssicherheit die Frischluftzufuhr zu garantieren.

5.2 Funktionsweise

Partikel werden in das System geboren und bewegen sich entsprechend der ihnen auferlegten Regeln bis die Dauer ihrer Lebenszeit aufgebraucht ist. Danach werden sie aus dem System gelöscht und neue Partikel entstehen. Wie genau die Erscheinung und das Verhalten der Partikel definiert sind, legen verschiedene Konfigurationen fest, welche während der Erstellung des Partikelsystems zu berücksichtigen sind. Eine der wichtigsten Entscheidungen beim Aufbau eines Partikelsystems ist, ob dieses **zustandserhaltend** oder **zustandslos** implementiert werden soll.

Partikel von **zustandslosen Partikelsystemen** reagieren nicht auf Einflüsse nachdem sie in das System „geboren“ werden. Ihre Position errechnet sich aus einer Funktion, die die Startposition und die vergangene Zeit beinhaltet. Diese könnte wie folgt aussehen:

$$p_t = p_0 + v_0 + a_t$$

Dabei wird die Position zum Zeitpunkt t p_t aus der initialen Position p_0 sowie der initialen Geschwindigkeit v_0 und der Beschleunigung zum Zeitpunkt t a_t berechnet. Derartige Systeme benötigen weniger Rechenkraft als zustandserhaltende Partikelsysteme, da die errechneten Positionen und andere Parameter wie Farbe oder Größe nicht zwischengespeichert werden müssen. Ein weiterer Vorteil ist, dass Partikelpositionen zu jedem Zeitpunkt aus den initialen Eigenschaften mit der gezeigten Formel berechnet werden können, da Partikel nicht von ihren Bahnen abweichen können. Allerdings sind zustandslose Systeme aus genau diesem Grund sehr unflexibel. Sie werden aufgrund dessen für kleinerer Effekte zur Visualisierung von Explosionen oder ähnlichen in Videospielen eingesetzt.

In **zustandserhaltende Partikelsystemen** hingegen, berechnen sich Partikelpositionen aus Parametern, die im vorangegangenen Frame berechnet wurden. Zustände werden also nach jedem Durchgang gespeichert und wiederverwendet. Dadurch kann das System flexibel auf äußere Einflüsse reagieren. Auch Partikelinteraktionen untereinander sind so möglich. Eine Funktion, die Positionen der Partikel zum Zeitpunkt t berechnet könnte wie folgt lauten:

$$p_t = p_{t-1} + \overrightarrow{v_t}$$

Dabei wird die aktuelle Position p_t aus der Position des vorigen Frames p_{t-1}und dem aktuellen Geschwindigkeitsvektor $\overrightarrow{v_t}$ berechnet. Letzterer ergibt sich aus der Situation an der Position p_{t-1}. Durch vorherrschende Kräfte, die durch Modifikatoren (siehe *Kapitel 5.3.3 Modifikatoren)* beeinflusst werden, wird die Geschwindigkeit beziehungsweise Beschleunigung bestimmt. Auch ein Vektorfeld kann die Gegebenheiten an einer bestimmten Position diktieren. Zustandserhaltene Partikelsysteme eignen sich für fortschrittlichere Animationen und Simulationen, benötigen aber dementsprechend mehr Rechenkapazität (*Kolb et* al*., 2004*).

Neben der Differenzierung der Zustandserhaltung können Partikelsysteme weitere grundlegende Konfigurationen annehmen, die ihre Erscheinung ausschlaggebend unterscheiden. Ein bedeutender Unterscheid besteht beispielsweise zwischen **stochastischen** und **strukturieren** Partikelsystemen.

Stochastische Partikelsysteme werden eingesetzt, um volumetrische Körper zu visualisieren. Die Partikel bewegen sich meist unabhängig voneinander in einem festgelegten Bereich, sind aber abhängig von vordefinierten Kräften wie Gravitation oder äußeren Einflüssen wie Turbolenzen. Die genaue Form des Objektes, welches die Partikel in ihrer Summe symbolisieren, kann vor der Simulation, bis auf die maximalen Ausmaße nicht genau spezifiziert werden. Diese Systeme eignen sich daher um unscharfe Körper wie Feuer oder Rauch zu generieren.

Strukturierte Partikelsysteme dagegen werden zur Darstellung von zwei- oder dreidimensionalen Körpern genutzt, welche im Gegensatz den eben genannten Beispielen feste Grenzen aufweisen. Durch diese Methode können beispielsweise Bäume oder Sträucher dargestellt werden. Reeves&Blau beschreiben 1985 eine Methode, mit der sie ganze Wälder und Graslandschaften mittels stochastischer Partikelsysteme digital erschufen. Partikel erschienen in diesem System als Linien oder Kreise, welche Äste

beziehungsweise Stämme und Blätter symbolisieren. Jedes Partikel ist abhängig dem vorangegangenen Partikel. Ast-Partikel stehen beispielsweise in Beziehung zu Stamm-Partikel und befinden sich mit einem bestimmten Winkel abstehend zum Stamm. Blatt-Partikel wiederum orientieren sich an zuvor berechneten Ast-Partikeln. Auf Abbildung 19 ist eine Szene zu sehen, die *Reeves&Blau* mit stochastischen Partikelsystemen geschafft haben.

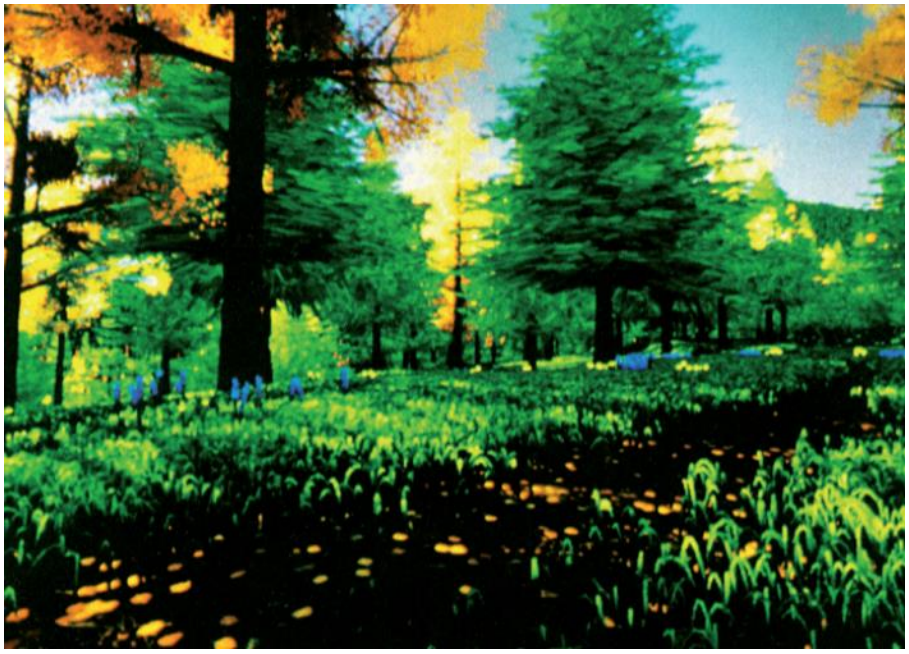

Abbildung 19 Waldszene generiert mit einem stochastischen Partikelsystem. Quelle: Reeves&Blau, 1985

Eine weitere Unterscheidung besteht zwischen **animierten** und **statischen** Systemen. In **statischen Partikelsystemen** wird der gesamte Lebenszyklus eines Partikels als ein Status dargestellt. Oft ist daher nur das Ergebnis des Systems zu sehen. Auf Abbildung

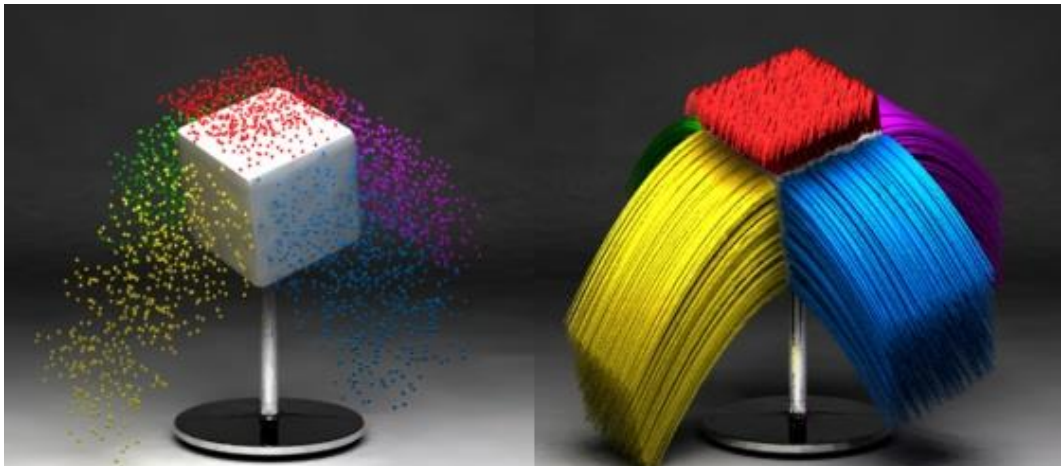

Abbildung 20 animiertes Partikelsystem. Rechts: statisches Partikelsystem. Quelle: wikimedia.org

20 ist rechts ein solches System abgebildet. Diese Methode ist aufgrund der visuellen Ähnlichkeit zu Haaren als „Hair-System“ bekannt. In **statischen Systemen** können die Fluglinien der Partikel besonders anschaulich verdeutlicht werden.

In **animierten Systemen** wird dagegen jeder Frame der Animation neu gerendert. Diese Methode wird genutzt um unscharfe Objekte zu animieren. Abbildung 20 verdeutlicht den Unterschied beider Systeme.

Weiterhin ist zu unterscheiden ob Kollisionen zwischen Partikeln registriert werden sollen. Wenn nicht, spricht man von einem **freien System**. Es gibt jedoch auch die Möglichkeit Interaktionen zwischen Partikeln zuzulassen und ihre Flugbahnen entsprechend anzupassen. Diese Systeme werden als **gebundene Partikelsysteme** bezeichnet. Die Kollisionsdetektion benötigt jedoch erheblich mehr Rechenkapazität. Diese erhöht sich, wenn Partikel durch komplexe Geometrien repräsentiert sind (*Kolb et al.*, 2004).

Neben der Konfiguration spielen vor allem die Partikel eine maßgebliche Rolle zur Erscheinung des gesamten Systems. Dabei sind die Dichte, Lebenszeit und Bewegungseigenschaften der Partikel von Bedeutung. Meist werden Partikel als einzelne Pixel oder Punkte dargestellt, es ist aber auch möglich, dass sie komplexe geometrische Formen annehmen oder durch Grafiken repräsentiert werden. Des Weiteren sind rekursive Systeme möglich, in denen jedes Partikel ein weiteres Partikelsystem verkörpern (*Reeves, 1983*). Weitere Informationen zu den Partikeln werden in Kapitel 5.4 (*Partikel*) erläutert.

5.3 Bestandteile von Partikelsystemen

Das folgende Kapitel listet Elemente auf, welche typischerweise in Partikelsystemen vorkommen. Dazu werden eine Erläuterung und praktische Beispiele genannt.

5.3.1 Emitter

Emitter (zu Deutsch: Aussender) dienen dazu, Partikel an einer bestimmten Stelle, oder innerhalb eines abgegrenzten Raumes entstehen zu lassen. Ein Partikelsystem kann dabei auch mehrere Emitter besitzen. Neben der initialen Position werden den Partikeln im Emitter weitere anfängliche Eigenschaften zugeschrieben. Meist sind dies die Größe, Farbe, Geschwindigkeit oder Richtungsvektoren. Es gibt darüber hinaus die Möglichkeit Partikelsysteme ohne Emitter zu erstellen. In diesen werden die Partikel an zufälligen Positionen des Generation Shapes „geboren“.

5.3.2 Generation Shape

Als Generation Shape taufte *Reeves* den Raum, in welchem Partikel „geboren“ werden. Üblicherweise hat das Generation Shape eine sphärische oder andere regelmäßige geometrische zwei- oder dreidimensionale Form. Aber auch komplexerer Formen, die Naturgesetzen folgen, sind möglich (*Reeves*, 1983).

5.3.3 Modifikatoren

Modifikatoren beeinflussen die Partikelbahnen. Sie können natürliche Phänomene wie Wind, Gravitation oder Magnetismus simulieren. Häufige Modifikatoren sind:

- Attraktor: simuliert eine Anziehungskraft, Partikel werden in seine Richtung gelenkt (siehe Abbildung 21)
- Reflektor: simuliert eine Barriere, an der Partikel abprallen
- Destruktor: Partikel werden aus dem System gelöscht sobald sie einen bestimmten Bereich erreichen. Je nach Einstellung kann ein gewisser Anteil der Partikel den Destruktor „überleben“. In diesem Fall dient der Destruktor dazu die Partikelanzahl lediglich zu mindern.
- Turbolenzelement: simuliert beispielsweise Wind. Partikel werden von ihrer Bahn je nach Einstellung in eine bestimmte oder zufällige Richtung abgetrieben (siehe Abbildung 21).

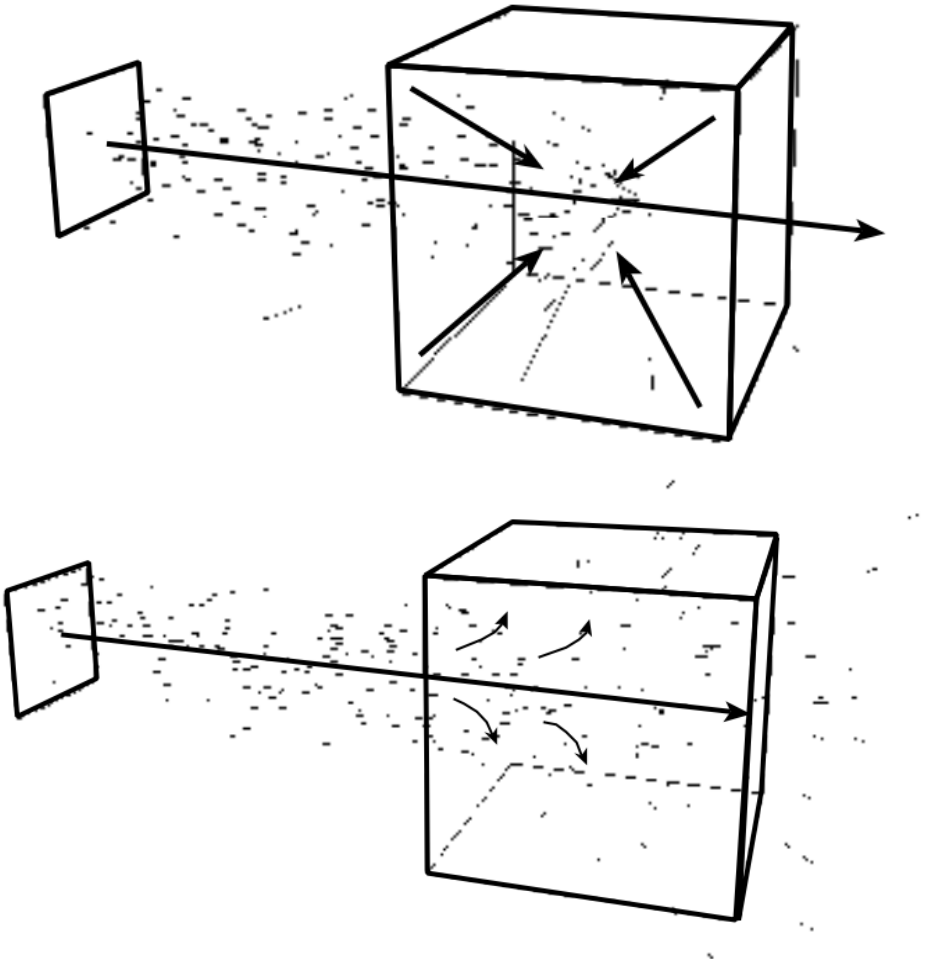

Abbildung 21 Oben: Attraktor - Die Partikel werden in das Zentrum des Quaders gelenkt. Unten: Turbolenzelement - Partikel werden von ihrer Bahn abgelenkt. Quelle. Ferschin, 1999

5.4 Partikel

Die Partikel sind der Kern eines jeden Partikelsystems. Ihr Erscheinungsbild prägt den gesamten Körper den sie gemeinsam darstellen. *Reeves* nennt dazu Eigenschaften, die jedes Partikel am Start seines Lebenszyklus zugewiesen bekommt. Diese sind Position, Geschwindigkeit und Richtung, Erscheinung (Größe, Farbe, Transparenz) und Lebenszeit (*Reeves*, 1985). Diese werden im folgenden Kapitel beschrieben.

5.4.1 Position

Die initiale Position wird üblicherweise durch den Emitter festgelegt. Der Lebenszyklus der Partikel beginnt in dem Fall in dem vom Emitter festgelegten Bereich. Darüber hinaus kann die Startposition der Partikel auch zufällig innerhalb des Generation Shapes erfolgen. Wie

sich die Positionsbestimmung im weiteren Verlauf der Animation verhält ist abhängig von der Konfiguration des Systems (siehe *Kapitel 5.2 Funktionsweise*).

5.4.2 Geschwindigkeit und Richtung

Die initiale Geschwindigkeit und Richtung geben vor, wie die Partikel aus dem Emitter austreten. Auf Abbildung 22 ist ein Beispiel zu sehen, wie Partikel mit einem zufällig bestimmten Austrittswinkel aus dem Emitter austreten. Es ergibt sich eine konische Form, welche im konkreten Beispiel eine Explosion auf einer Oberfläche darstellt.

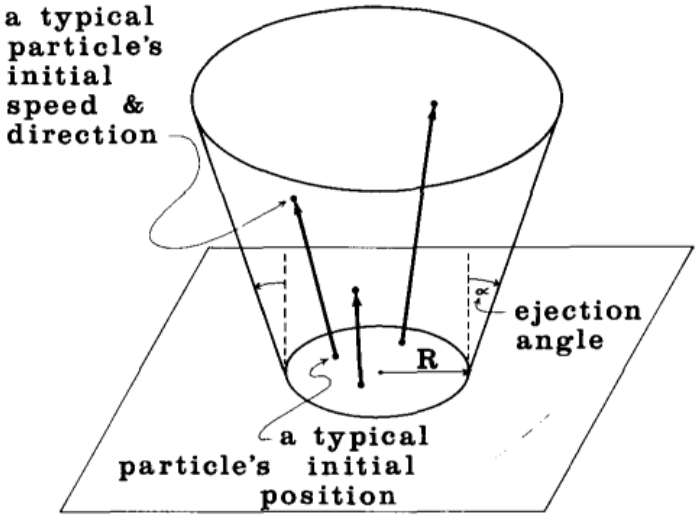

Abbildung 22 Partikel mit einem zufällig bestimmten Austrittswinkel entlassen. Quelle: Reeves,1983

5.4.3 visuelle Erscheinung

Der visuelle Charakter aller Partikel ist letztendlich maßgelblich für das Erscheinungsbild des gesamten Körpers, dass sie darstellen. Zusammen mit der Anzahl der Partikel wird so beispielsweise die Dichte des Körpers bestimmt. Die optischen Eigenschaften der Partikel beinhalten Größe, Farbe, Form und Transparenz. Diese werden den Partikeln zu Beginn zugewiesen und können sich im Laufe des Lebenszyklus ändern. Partikel erscheinen oft in der Form von primitiven geometrischen Objekten, die aus Vertices gebildet werden. Einem Vertex kann aber auch eine Textur zugewiesen werden, welche den Vertex als Ankerpunkt nutzt und an dieser Stelle dargestellt wird. Je nach Anwendungsfall können Partikel darüber hinaus auch aus komplexeren Geometrien bestehen. *Latham&Munjiza* nutzen beispielweise irreguläre dreidimensionale Formen für Partikel, um Felsstürze zu simulieren (2004).

5.4.4 Lebenszeit

Die initiale Lebenszeit des Partikels bestimmt, wie lange ein Partikel maximal Teil des Partikelsystems ist. Vorheriges Ausscheiden aus dem System sind durch Kollisionen mit anderen Partikeln beziehungsweise externe Objekte oder das Erreichen der Grenzen des Generation Shapes möglich.

5.5 Partikelsysteme zur Geovisualisierung

Auch in der Geoinformation bieten Partikelsysteme vielfältige Möglichkeiten. Nach *Kraak*, sollten Geovisualisierungen neuen Mut fassen innovative Darstellungsmöglichkeiten zu zeigen wie digitale Überflüge dreidimensionaler Landschaften (*Kraak*, 2002). In diesen Fällen könnten Partikelsysteme eingesetzt werden, um eine immersive Erfahrung zu ermöglichen, die gezeigten Welten tatsächlich realistisch aussehen zu lassen. Einige Beispiele für diesen Einsatz wurden bereits genannt. *Huber*, welcher sich in seine Diplomarbeit 2012 mit Spezialeffekten in der Geovisualisierung auseinandergesetzt hat, stellt einige Beispiele vor, wie Partikelsysteme zur Darstellung verschiedener Sachverhalte genutzt werden können. Abbildung 23 zeigt beispielsweise einen Vulkanausbruch, bei dem der Rauch durch ein Partikelsystem visualisiert wurde. Ein Emitter stößt Partikel in vertikaler Richtung aus dem Vulkan heraus. Die Partikel werden in weitere Folge durch Modifikatoren, welche Wind und Gravitation simulieren, von ihrer initialen Flugbahn abgetrieben (*Huber*, 2012).

Abbildung 23 Vulkanausbruch visualisiert mit einem Partikelsystem. Quelle: Huber 2012

Neben dem Einsatz von Partikelsystemen zur reinen Veranschaulichung von Effekten können sie auch zur Darstellung von Dynamiken eingesetzt werden. *Scheepens et al.* stellten 2016 ein System vor mit dem Verkehrsströme visualisiert wurden. Auch *Ferschin* zeigt Möglichkeiten auf mit Partikelsystemen Verkehrsströme zu visualisieren (*Ferschin*, 1999). Des Weiteren wurden Partikelsysteme bereits eingesetzt, um den Verkehr im Luftraum nachzubilden. Dabei wurden mögliche Kollisionen von Objekten berücksichtigt. *Prandini et al.* zeigen damit, wie Partikelsysteme neben der Visualisierung auch zur Simulation eingesetzt werden können (*Prandini et al,* 2006).

Für diese Arbeit besonders von Belangen sind Visualisierungen, welche Wind darstellen. Wie eingangs der Arbeit bereits angesprochen, sind bekannte Beispiele *windy.com*, *ventursky.com* und *earth.nullschool.net*. All diese Dienste visualisieren Winde als Partikel auf der Grundlage eines Vektorfeldes. Im Falle der beiden erstgenannten dient eine Karte (Mercator-Projektion) als Basis der Visualisierung. Das *earth-Projekt* wird standartmäßig in einer orthographischen Kartenprojektion präsentiert, welche einer Visualisierung eines Globus ähnelt (siehe Abbildung 24). Jedoch gibt es mehrere Unterschiede zu einem virtuellen Globus Inhalte existieren nur im sichtbaren Bereich der Karte. Partikel welche den Rand der Karte erreichen werden aus dem System gelöscht. Auf einem Globus würden Partikel ihre Bahnen auch weiterhin fortsetzten. Dies würde eine kontinuierliche Interaktion erlauben, da Partikel nach Rotation des Globus noch immer im System verbleiben. In dem Beispiel des *earth*-Projektes hingegen, lädt die Visualisierung von neuem nachdem die Karte bewegt wurde. Dies ist ein wesentlicher Unterschied, der bei der Erstellung der Visualisierung auf dem virtuellen Hyperglobe berücksichtigt werden muss um eine Weiterentwicklung darzustellen.

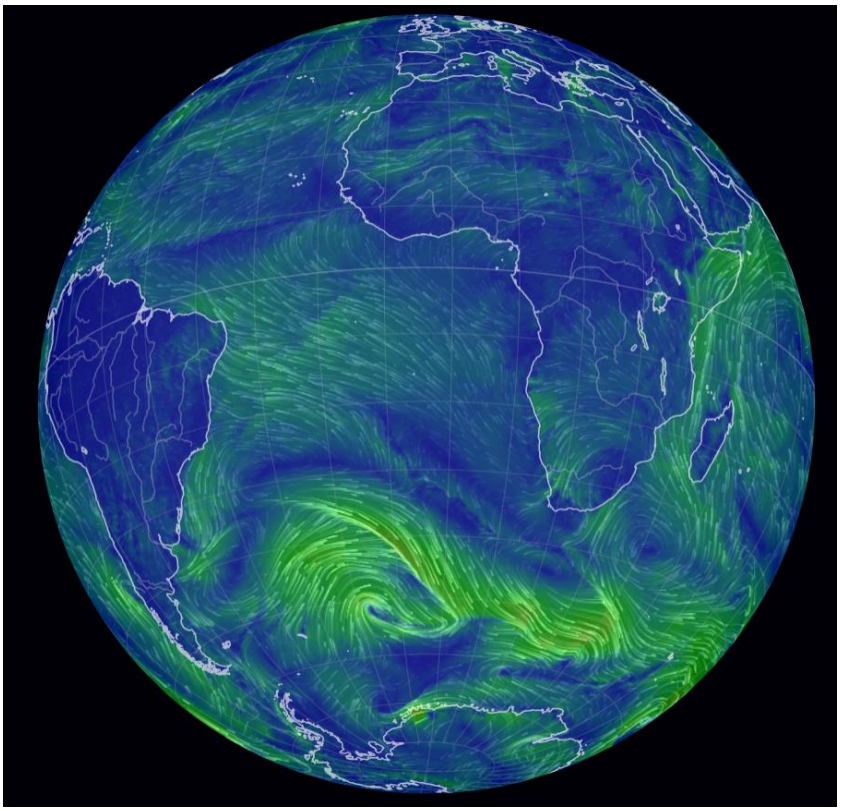

Abbildung 24 Darstellung atmosphärische Strömungen durch Partikel auf derm Projekt earch von Baccario. Quelle: earth.nullschool.net

Neben Wind können beispielsweise auch Wellenbewegungen mithilfe von Partikelsystemen dargestellt werden. *Kim* stellte 2018 dazu eine Möglichkeit vor, mit der druch HTML5 Wellen als Partikelsystem visualisiert werden.

6.Umsetzung von Partikelsystemen auf Globen

Nachdem die vorherigen Kapitel alle theoretischen Grundlagen abgedeckt haben, soll nun an der Implementierung von Partikelsystem auf Globen gearbeitet werden. Dazu müssen zunächst geeignete Daten gefunden werde, die es erlauben ein globales Partikelsystem aufzusetzen. Mit dieser Thematik wird sich das anschließende Kapitel befassen. Des Weiteren werden grundlegende Prinzipien aufgezeigt mit denen die Partikelsysteme auf dem virtuellen sowie auf dem taktilen Globus errichtet werden sollen. Wurde dies bewältigt, folgt die Beschreibungen der Implementationen der Systeme auf den jeweiligen Globen. Nachdem die speziellen Anforderungen an das entsprechende Medium dargelegt wurden, beschreibt die Arbeit detailliert den Programmcode mit welchen die Partikelsysteme umgesetzt wurden. Abschließend werden die Ergebnisse beschrieben und diskutiert.

6.1 Ausgangsdaten

Dieses Kapitel widmet sich den Daten, die zur Darstellung der globalen Windverhältnisse am Hyperglobus genutzt werden sollen. Dazu gehört zunächst die Behandlung der Globenwürdigkeit des Sachverhaltes. Auch weitere Anforderungen, wie Datenstruktur werden in diesem Kapitel behandelt

6.1.1 Globenwürdigkeit

Um Wettersysteme am Globus darstellen zu können, müssen zunächst geeignete Daten gefunden werden. Da es sich um eine Darstellung auf Globen handeln soll, müssen die Daten sowie die Thematik „globenwürdig“ sein. *Riedl* fasste die Bedeutung des Begriffs in seiner Diplomarbeit zusammen, nachdem bereits vorher vereinzelt Parameter für die Eignung von Thematiken am Globus formuliert wurden. Zusammengefasst müssen die folgenden vier Kriterien erfüllt werden (nach *Riedl*, 2000):

- Verzerrungsfreiheit
- Globalität
- Kleinmaßstäbigkeit
- Kombinationsfähigkeit

Die Verzerrungsfreiheit bezieht sich auf die optimale Darstellung eines Sachverhaltes auf einen Globus. Erst durch die globale Darstellung, können Zusammenhänge erkannt werden. Dies trifft auf globale Wettersysteme zu, da lokale Strömungen von kritischen Punkten (siehe *Kapitel 4.1 Definition von Vektorfeldern*) innerhalb des Windfeldes abhängen. Die Windverhältnisse an einem bestimmten Ort lassen sich demnach durch Verhältnisse an einem anderen Ort erklären.

Auch die Globalität ist für das vorliegende Thema zutreffend. Denn dabei handelt es sich um die globale Verfügbarkeit der Daten. Darüber hinaus sollten die Daten ein weltumspannendes Phänomen beleuchten. Beide Kriterien werden von globalen Klimavisualisierungen erfüllt.

Die Kleinmaßstäbigkeit widmet sich der Problematik der Generalisierung. Ist es sinnvoll eine Thematik trotz hoher Generalisierung zu visualisieren? Kann die Frage mit „Ja“ beantwortet werden, trifft das Kriterium Kleinmaßstäbigkeit zu. Auch im vorliegenden Fall wurden das zugrundeliegende Vektorfeld generalisiert. Wie später noch beschrieben wird, liegen die Daten zumeist in einem Raster vor, welches in 1° bis 0.25° großen Zellen aufgeteilt wurde. Trotz der Generalisierung, kann der Sachverhalt der globalen Windbewegungen sinnvoll dargestellt werden.

Das letzte Kriterium, die Kombinationsfähigkeit, bezieht sich auf Daten, die per se nicht weltumspannende Thematiken darstellen. Dazu zählen beispielsweise Sachverhalte, welche sich nur auf das Festland oder nur auf Ozeane beziehen. In diesen Fällen ist es sinnvoll das Potential der Hyperglobe auszuschöpfen, um gleichzeitig weitere Themen zu präsentieren. Die vorliegende Arbeit wird sich jedoch auf die Windbewegungen in der Atmosphäre konzentrieren, welche global vorhanden sind. Zusätzliche Darstellungen von weiteren Themen sind damit nicht notwendig.

Damit ist die Globenwürdigkeit des Themas gegeben. In weitere Folge werden nun mögliche Quellen der Daten eruiert.

6.1.2 Herkunft der Daten

Es gibt verschiedene Institutionen die weltweit verfügbaren Wettervorhersagen bereitstellen. Dazu gehören zum Beispiel das europäische Institut ECMWF (European Centre for Medium-Range Weather Forecasts) und die amerikanische Behörde NOAA (National Center For Environmental Information). Beide Institutionen bieten frei

zugängliche Daten an. Bei letztgenannter Behörde können die Vorhersagen zudem ohne jegliche Einschränkung bezogen werden. Zur vorliegenden Arbeit werden daher Daten zur Wettervorhersage der NOAA genutzt. Die Vorhersagen werden durch das Wettervorhersagemodell GFS (Globas Forecast System) produziert. Wichtig ist zu beachten, dass die Daten nur Vorhersagen beinhalten und keine tatsächlich gemessenen Werte darstellen. Die Daten des GFS werden alle 6 Stunden produziert und freigegeben. Sie beinhalten mehrere stündliche Vorhersagen ab dem Zeitpunkt der Veröffentlichung (NOAA, 2021).

6.1.3 Beschreibung der Daten

Die von dem GFS produzierten Daten beinhalten zahlreiche Variablen aus über 120 atmosphärischen Schichten. Dazu gehören zum Beispiel Temperatur, relative und tatsächliche Feuchtigkeit, Ozonmischverhältnis, Luftdruck und viele mehr. Die vorliegende Arbeit wird sich auf die Variablen der horizontalen Windgeschwindigkeit auf der Oberfläche der Erde konzentrieren. Im speziellen werden die Parameter „UGRD“ und „VGRD“ benötigt. Alternativ werden die Parameter als *u*- und *v*-Komponente des Windes bezeichnet. Positive *u*- und *v*-Werte stehen dabei für die Geschwindigkeit des horizontalen Windes von West nach Ost, respektive Süd nach Nord. Negative Werte stehen dementsprechend für Geschwindigkeiten der jeweils gegenteiligen Richtung. Standardmäßig werden die Werte dabei in Meter pro Sekunde angegeben. Die Informationen können ausschließlich im GRIB-Dateiformat bezogen werden. Dieses Format gilt in der Meteorologie als Standard und wurde von der Weltorganisation für Meteorologie definiert. Es besteht aus einem oder mehreren Bändern, wobei ein Band ein Raster beinhaltet, welches die Werte einer Variable, zum Beispiel die *u*-Komponente, enthält.

Die Raster können in verschiedener Auflösung bezogen werden. Zur Verfügung stehen 1°x1°, 0.5°x0,5° und 0.25°x0,25° großen Rasterzellen. Für den Zweck dieser Arbeit, eine globale Darstellung des Windes zu erzeugen, genügt eine Auflösung von einem Grad. Zur Veranschaulichung der Daten ist die *u*-Komponente einer solchen Datei auf Abbildung 25 zu sehen. Weiße Flächen zeigen dabei hohe Geschwindigkeiten in der West-Ost Richtung an. Dunkle dagegen in der entgegengesetzten Richtung.

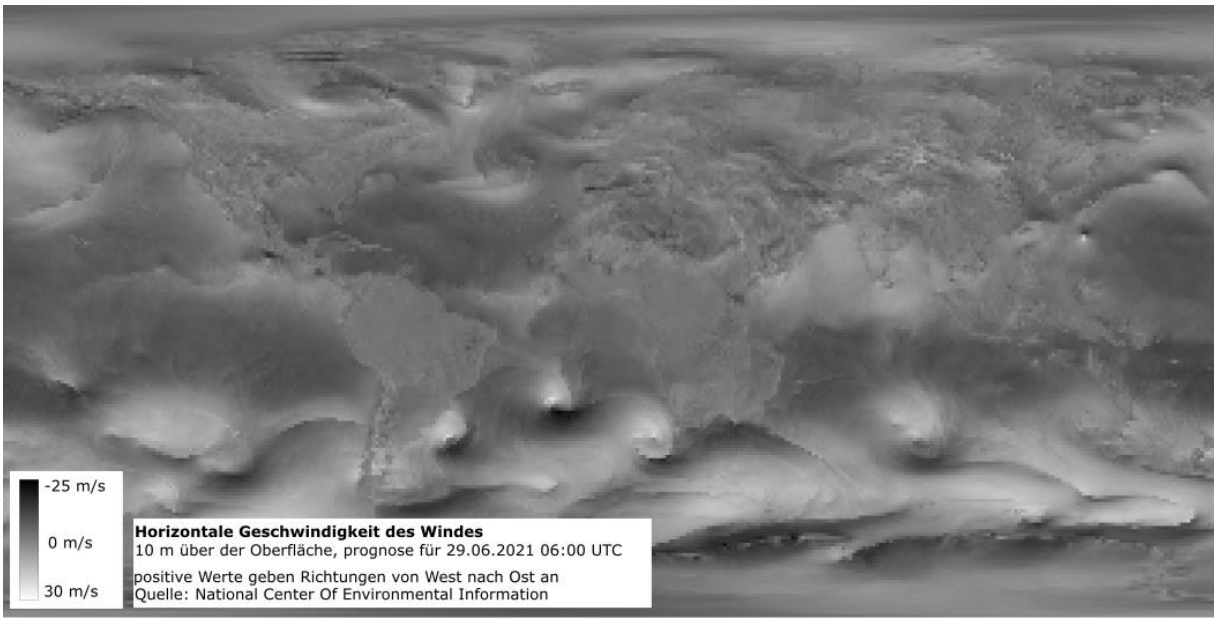

Abbildung 25 Die horizontalen Geschwindigkeiten des Windes dargestellt in graustufen. Positive Werte beziehen sich auf Windrichtung West nach Ost. Quelle: eigene Abbildung, Datenquelle: National Center Of Environmental Information

6.1.4 Bearbeitung der Daten

GRIB-Dateien können durch verschiedene Möglichkeiten visualisiert und bearbeitet werden. Zur reinen Darstellung der Daten stehen verschiedenen „Grib-Viewer“ zur Verfügung. Dabei handelt es sich um Software, welche speziell für die Visualisierung von GRIB-Dateien erstellt wurden. Einige bekannte Beispiele sind „Predict Wind“ (predictwind.com), Seaman Pro von wetterwelt.de oder *XyGrib* von OpenGribs.org. Diese Programme können die Daten jedoch nur darstellen, nicht bearbeiten. Eine beispielhafte Visualisierung des einer GRIB-Datei durch die Software *XyGrib* zeigt Abbildung 26. Geschwindigkeiten werden dabei unabhängig von der Richtung farblich dargestellt. Die Windrichtung wird durch Windbarben symbolisiert.

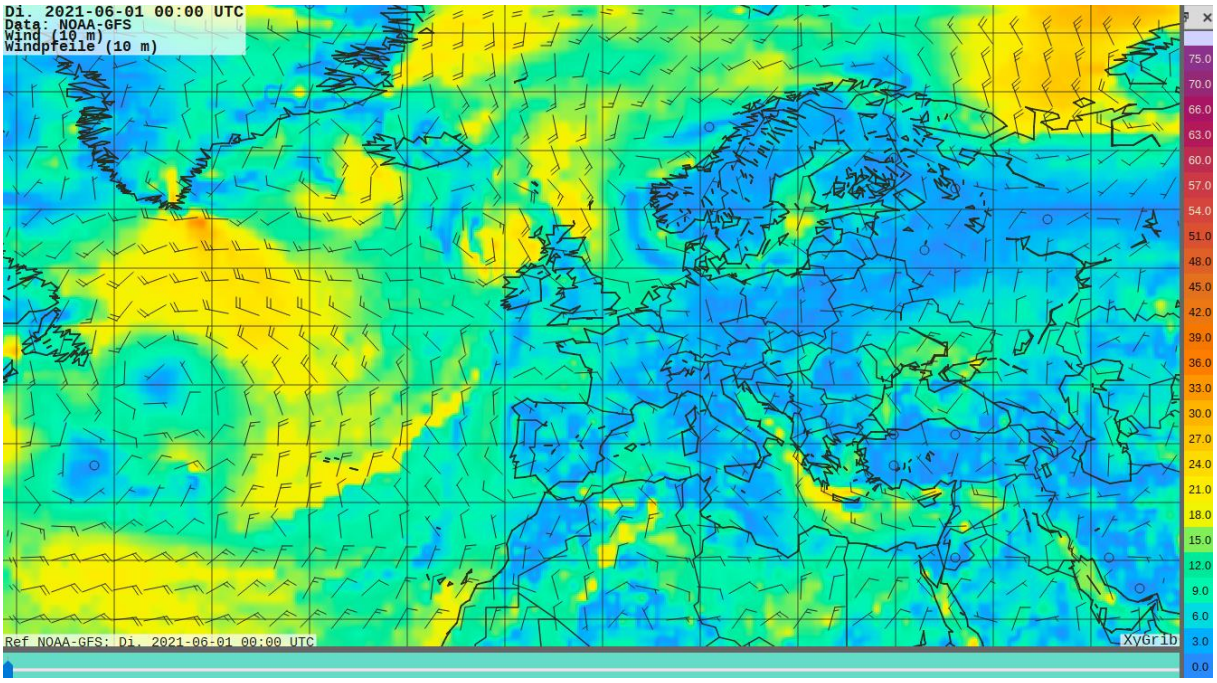

Abbildung 26 Darstellung einer Grib Datei welche u- und v- Komponenten des Windes beinhaltet. Quelle: Darstellungssoftware: XyGrib (opengribs.com), Datenquelle: National Center Of Environmental Information

Neben Software, welche speziell zur Veranschaulichung von GRIB-Dateien entworfen wurde, existieren zudem Möglichkeiten mit denen die Informationen aus den Dateien extrahiert werden können. Zu den bekannten Möglichkeiten gehört beispielsweise das Kommandozeilenprogramm *grib2json* (github.com/cambecc/grib2json). Damit können JSON-Dateien aus dem GRIB-Format gewonnen werden. Weiterhin steht das Programm *DeGrib* zur Verfügung. Auch damit können gezielt Informationen aus GRIB-Dateien extrahiert und beispielsweise im CSV oder ebenfalls im JSON Format abgespeichert werden. Letztendlich ist es zudem möglich, GRIB-Dateien mit Geoinformationssystemen zu öffnen und zu bearbeiten.

Während der Vorbereitung der Erstellung der Partikelsysteme zeigte sich, dass das GRIB-Format zum Einlesen und Verarbeiten über den Browser ungeeignet sind, da es keine Programmbibliotheken gibt, welche das Auslesen der Dateien unterstützen. Die Schnittstelle müsste demnach eigens für diese Arbeit programmiert werden. Daher wurde definiert, dass die Vektorfelder, die mittels eines Partikelsystems visualisiert werden sollen, als CSV-Datei einzugeben sind. Dabei muss die Datei zwei Spalten enthalten, die für die *u*- und *v*-Komponenten der Windvektoren stehen. Der erste Werte repräsentiert die Rasterzelle -180°W, 90°N. Es folgen die restlichen Werte des nördlichsten Breitengrades. So wird die Datei zeilenweise befüllt. Dies ist die standartmäßige Reihenfolge beim

Exportieren von gängigen GIS-Programmen sowie beim Extrahieren der Daten mittels der Software *DeGrib*. Abbildung 27 zeigt einen Ausschnitt einer solchen Datei.

```
 1  u,v
 2  -3.15865,-6.44040
 3  -3.26865,-6.38040
 4  -3.37865,-6.32040
 5  -3.48865,-6.26040
 6  -3.59865,-6.20040
 7  -3.70865,-6.14040
 8  -3.80865,-6.07040
 9  -3.91865,-6.00040
10  -4.01865,-5.94040
11  -4.12865,-5.86040
12  -4.22865,-5.79040
13  -4.32865,-5.72040
14  -4.42865,-5.64040
15  -4.52865,-5.56040
```

Abbildung 27 Ausschnitt der Eingabedatei der Partikelsysteme am Globus. Quelle: eigene Abbildung

6.2 Allgemeine Methodik

Die zu erstellende Visualisierung soll Vektorfelder mittels eines Partikelsystems visualisieren. Dazu wird nun die grundlegende Methodik vorgestellt. In den folgenden Kapiteln, wird diese weiter gezielter definiert, da der taktile und virtuelle Hyperglobus teilweise unterschiedliche Anforderungen haben.

Grundsätzlich sollen in beiden Systemen Partikel zunächst zufällig auf dem *Generation Shape* verteilt werden. Dieses stellt in beiden Fällen die Oberfläche des jeweiligen Globus dar und wird im Hintergrund beider Anwendungen durch eine Plattkarte definiert. Dabei handelt es sich um eine abstandstreue Zylinderprojektion bei der der Äquator längentreu abgebildet wird. Die Meridiane sind um die Hälfte kürzer als der Äquator, woraus sich ein Seitenverhältnis von 2:1 ergibt (*Weinhandl*, 2011).

Obwohl das Generation Shape damit klar begrenzt ist und die Form eines Rechteckes aufweist, dürfen Partikel dieses nicht verlassen. Sollte die berechnete Flugbahn das Partikel den 180. Längengrad, und damit den Rand der Karte, übertreten lassen, muss die neue Position automatisch auf die gegenüberliegende Seite der Plattkarte festgesetzt werden. Bei Übertritt des Nord- oder Südpols muss neben dem

Längengrad auch der Breitengrad automatisch angepasst werden. Der neue Längengrad ergibt sich aus dem alten Längengrad der Position im Frame vor dem Übertritt des Pols. Gleichzeitig wird der Breitengrad wieder auf den nördlichsten Punkt (bei Übertritt des Nordpols) oder südlichsten Punkt (bei Übertritt des Südpols) zurückgesetzt. Abbildung 28 verdeutlicht die Situation. Es ist der Überflug eines Partikels in einer azimutalen Projektion und einem Ausschnitt einer zylindrischen Projektion zu sehen. Während auf der erstgenannten Variante die Flugbahn ohne Unterbrechung dargestellt werden kann, muss sie auf der zylindrischen Darstellung angepasst werden.

Diese automatischen Anpassungen sind notwendig damit in der finalen Visualisierung des Partikelsystems am Globus die ursprüngliche Form der Plattkarte nicht zu erahnen ist. Details zu der Übertragung der Position finden sich in den entsprechenden Kapiteln

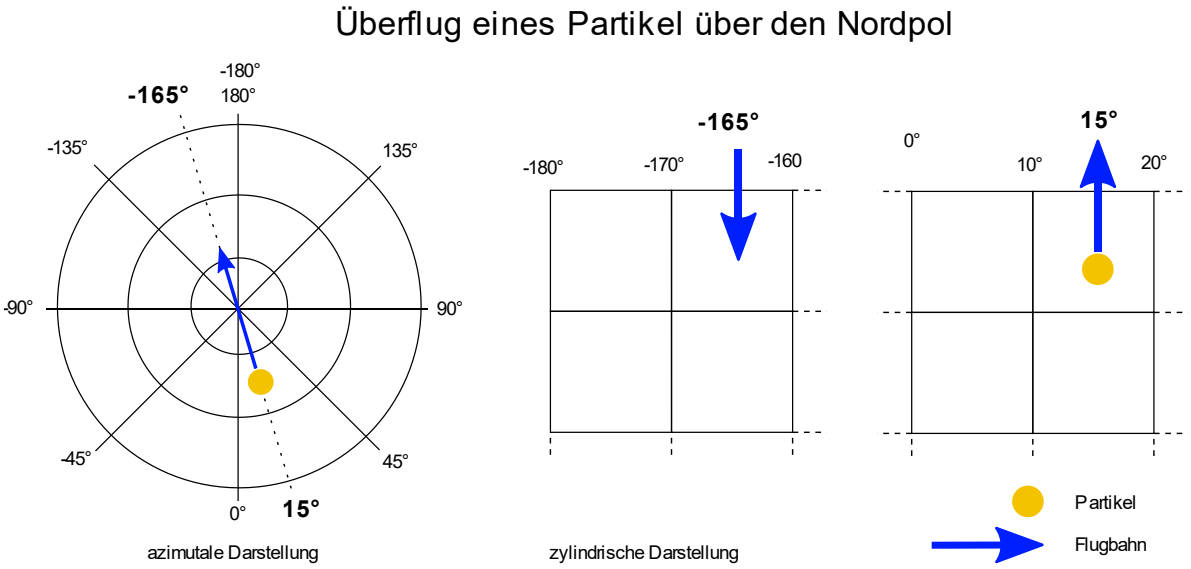

Abbildung 28 Schematische Darstellung einer Flugbahn eines Partikels, dass den Nordpol überfliegt. Quelle: eigene Abbildung

über die Anforderungen an ein Partikelsystem am Globus. Zusätzlich zu der zufälligen Position bekommt zum Start der Visualisierung jedes Partikel eine maximale Lebenszeit zugewiesen.

Nachdem Partikel im Generation Shape verteilt wurden, orientieren sich die Bewegungen, die sie in der Folge ausführen, an ihrer Umgebung. Dies bedeutet, dass nach jedem Frame die Position eines jeden Partikels bekannt sein muss. Zudem ist es notwendig die Verhältnisse an diesem Ort zu kennen. Damit sind im speziellen Fall Windvektoren gemeint, welche die Bewegung des Partikels bestimmen. Diese

Voraussetzung bedeutet, dass ein **zustandserhaltendes Partikelsystem** erstellt werden muss (vergleich *5.2 Funktionsweise von Partikelsystemen*). Weiterhin soll es sich um masselose Partikel handeln zwischen denen es zudem keine Interaktion geben wird. Dies bedeutet Partikel bewegen sich ausschließlich anhand der Vektoren, welche sich in ihrer Position befinden. Die Flugbahnen der Partikel werden damit nicht durch andere Partikel oder physikalische Größen wie Trägheit und Gravitation beeinflusst. Zum Verständnis wie sich Partikel durch ein Vektorfeld bewegen werden, zeigt *Abbildung 29* eine schematische Darstellung eines Partikels, die durch die Strömung beeinflusst werden.

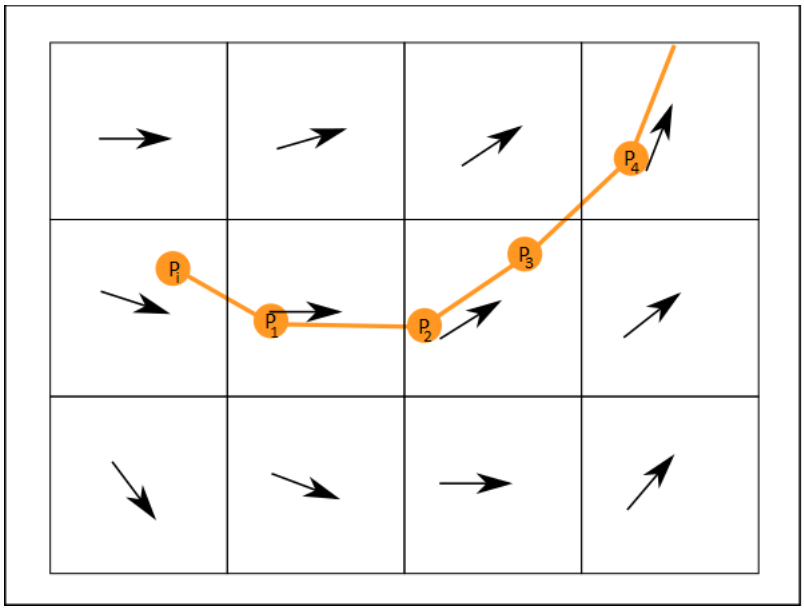

Abbildung 29 Schematische Bewegung eines Partikels durch ein Vektorfeld. Quelle: eigene Abbildung

Zu sehen ist, dass das Partikel wie beschrieben zunächst auf eine initiale zufällige Position P_i gesetzt wird. Anschließend wird es durch den Vektor an seiner aktuellen Position durch die Strömung bewegt. Die Berechnung der Position erfolgt dabei durch Vektoraddition. Die Formel für die Berechnung ist aus Kapitel 5.2 (*Funktionsweise von Partikelsystemen*) bekannt. Angepasst an das Beispiel lautet sie:

$$p_1 = \overrightarrow{p_i} + \overrightarrow{v_i}$$

Es wird im ersten Schritt des Beispiels also der Positionsvektor p_i mit dem Windvektor an der Stelle p_i addiert. Dazu wird der Rechtswert der Positionskoordinate mit der *u*-Komponente und der Hochwert mit der *v*-Komponente addiert. Aus dieser Rechnung

erfolgt ein neuer Positionsvektor (in dem Beispiel p_1), auf welchem das Partikel im folgenden Frame gesetzt wird. Zudem wird die Lebenszeit des Partikels in jedem Schritt verringert. Dieses Prinzip der Neuberechnung der Positionen wird solange wiederholt bis die Lebenszeit eines jeden Partikels aufgebraucht wird. Danach wird es gelöscht und ein neues Partikel wird erstellt und wiederum zufällig auf der Plattkarte verteilt. In den folgenden Kapiteln wird nun beschrieben, wie diese allgemeine Methodik im speziellen angepasst an die jeweilige Softwareumgebung umgesetzt wurde.

6.3 Taktiler Globus

Die folgenden Kapitel widmen sich der Erstellung eines Partikelsystems zur Darstellung auf dem taktilen Globus. Dazu wird zunächst dargelegt, wie der taktile Globus bespielt werden kann. Vor allem die Darstellung von Dynamiken wird genauer betrachtet. Anschließend wird auf die Anforderungen an ein Partikelsystem am taktilen Globus eingegangen. Letztendlich wird das erstellte System vorgestellt.

6.3.1 Ein und Ausgabe am taktilen Globus

Die vorliegende Arbeit wird sich zur Erstellung des Partikelsystems an einem taktilen Hyperglobus der *Globoccess AG* orientieren. Es handelt sich dabei um einen Globus, der durch Innenprojektion von zwei Beamern bespielt wird. Dabei ist der Globus stets mit einem peripheren Computer verbunden, mit dem sich Inhalte auf den Globuskörper übertragen lassen. Mit der Software *OmniSuite*, die von *Globoccess* in Auftrag gegeben und von einer Arbeitsgruppe der Universität Wien entwickelt wird, kann der Globus gesteuert werden. Hauptsächlich werden derartige taktile Hypergloben in Museen, Firmen oder andere Institutionen ausgestellt, um weltweite Themen zu vermitteln. Dies können wissenschaftliche Erkenntnisse sein, welche globale Zusammenhänge zeigen, oder schlicht Firmenstandorte, die auf der gesamten Welt verteilt sind. In jedem Fall dient der Globus dabei als ‚Eye-catcher' und besondere Möglichkeit Inhalte zu präsentieren. In der Themenbibliothek von *Globoccess* stehen zudem zahlreiche Storys bereit, die mithilfe von *OmniSuite* abgespielt werden können.

Bei Storys handelt es sich dabei um dynamische Visualisierungen, die oft einem Film ähnlich und inhaltlich und technisch an den Globus angepasst sind. Das präsentierte Thema legt dabei meist Augenmerk auf globale Zusammenhänge. Die Storys entspringen dabei verschiedenen Fachgebieten wie etwa Geologie, Wirtschaft, Verkehr, Geschichte, Ökologie, Botanik oder Klimatologie. Einige Beispiele, die von *Globoccess* angeboten werden, sind (globoccess.at):

- Jahreskreislauf der Biosphäre der Erde
- Veränderungen der Lichtverschmutzung 1992-2010
- Veränderung der Ozon-Konzentration 2009-2010
- Änderungen des natürlichen Bevölkerungswachstums 1950-2050
- Luftverkehr eines Tages

Neben den bereits vorhandenen Inhalten besteht die Möglichkeit eigene Themen auf dem taktilen Hyperglobus zu visualisieren. Diese können mithilfe des Programmpakets *OmniSuite* zusammengestellt werden. Die Software beinhaltet mehrere Applikationen mit dem der Globus gesteuert, sowie Stories erstellt werden können. Die wichtigsten Programme sind:

- *Material-Editor*: Graphiken, die für Storys benötigt werden, können hier importiert werden. Dazu gehören neben Grundkarten auch Legenden und Bilder.
- *Story-Editor*: Eine Benutzeroberfläche mit der Filmsequenzen erstellt werden können. Dazu werden Materialien aus dem Material-Editor zusammengefügt. Zudem kann eine Audiospur ergänzt werden.
- *Controller*: Mit dem Controller können Stories ausgewählt und abgespielt werden. Auch die Interaktion mit dem Globus wird über den Controller gesteuert.

Bei der Erstellung einer Story oder einer Visualisierung für den Globus muss darauf geachtet werden, dass die genutzten Graphiken den Anforderungen des Globus

entsprechen. Basismedien, welche den kompletten Globus umspannen sollen, müssen beispielsweise im Format 2:1 vorliegen. Zudem muss darauf geachtet werden, dass Signaturen auf einer konvexen Oberfläche verzerrt werden, sollten sie nicht an den Globus angepasst sein. Genauere Informationen zu den Anforderungen finden sich im nächsten Kapitel.

Neben dem Story Modus gibt es die Möglichkeit über den Controller mit dem Globus zu interagieren. *Smutny* zeigte 2017 zum Beispiel eine Möglichkeit auf ein Städtequiz mit Hilfe des taktilen Hyperglobus zu realisieren. Der Benutzer wird dabei auf einem sekundären Bildschirm nach der Position einer bestimmten Stadt gefragt.

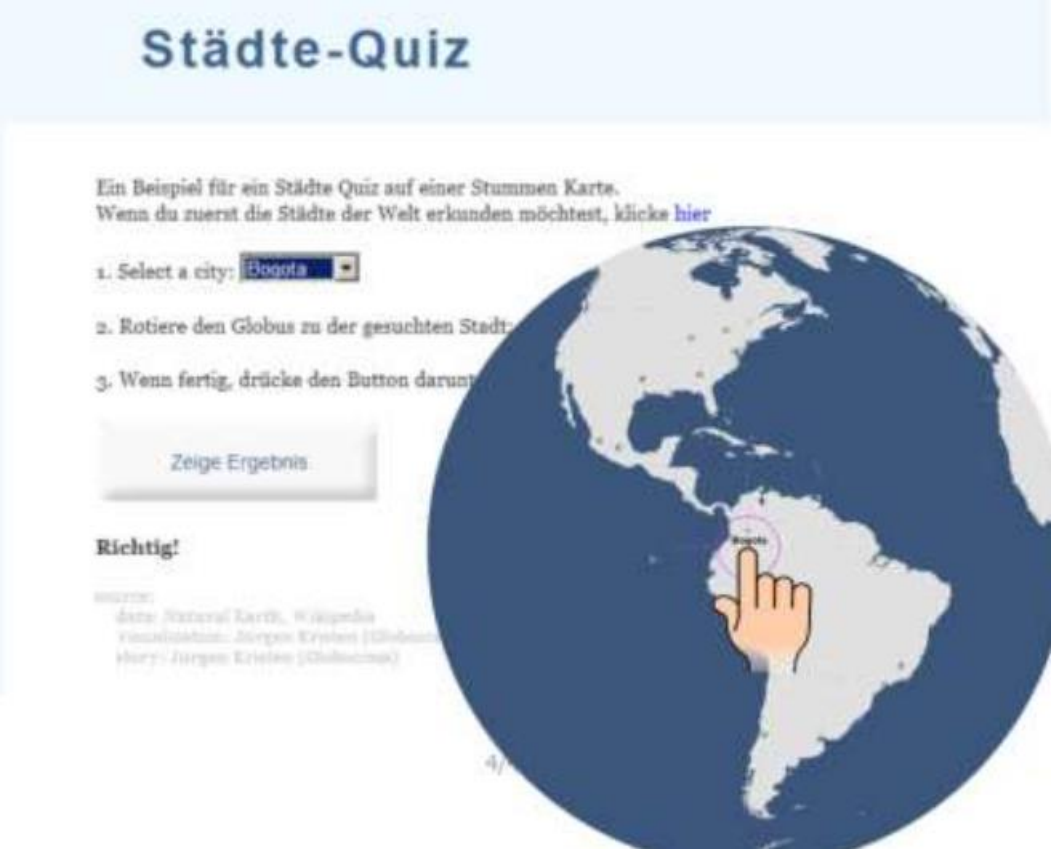

Abbildung 30 Benutzeroberfläche für ein Städtequiz mit dem taktilen Globus. Quelle: Smutny 2017

Anschließend ist es die Aufgabe des Nutzers den Globus in die richtige Position zu rotieren, so dass die gesuchte Stadt an einer bestimmten Stelle positioniert wird (siehe *Abbildung 30*). Durch derartige Anwendungen können Benutzer ihr räumliches und geographisches Wissen testen und erweitern.

Weiterhin gibt es die Möglichkeit, den Globuskörper als Display zu nutzen. Der Inhalt des Bildschirms des angeschlossenen Computers wird dabei auf die Oberfläche des Globus übertragen. In diesem Fall müssen alle Verzerrungen im Vorfeld berechnet und

schon auf dem Computerdisplay ersichtlich sein. Dies hat die Folge, dass Signaturen auf der zu übertragenden Karte auf dem flachen Bildschirm dementsprechend verzogen wirken. Vergleichende Bilder sind in Kapitel 6.3.3 (*Anforderungen an ein Partikelsystem am taktilen Globus*) auf *Abbildung 31* und *Abbildung 32* zu sehen.

6.3.2 Dynamische Darstellung auf dem taktilen Globus

Wie bereits erwähnt wurde, gibt es mehrere Möglichkeiten Inhalte dynamisch auf dem Globus zu visualisieren. Hauptsächlich steht der freie Modus sowie der Story Modus zur Verfügung. Wesentlicher Unterschied zwischen beiden Modi ist, dass eine Story bereits als Animation vorliegt und demnach nicht während der Präsentation verändert werden kann. Dynamiken werden demnach durch bereits gerenderte Rastergraphiken ermöglicht. Zusätzlich können Punktdaten über eine native Funktion der Globensoftware zu bestimmten Zeitpunkten der Animation hinzufügt werden. Dazu sind zwei Textdateien nötig, die in *OmniSuit* eingelesen werden müssen:

- Textdatei, die die Koordinaten der Punkte in tabellarischer Form enthält
- CSS-Datei, welche Eigenschaften über das Erscheinungsbild der Punkte bestimmt

Die Punktsignaturen werden durch die Software je nach Lage automatisch angepasst, so dass der Ersteller die Verzerrung auf einem sphärischen Display (Erläuterung folgt im nächsten Kapitel) nicht beachten muss. Neben den Positionen der Punkte, können in der Textdatei zusätzliche Eigenschaften deklariert werden. Zu denen gehören ein Beschriftungstext eines jeden Punktes, der auf dem Globus erscheinen kann, sowie eine Einteilung in verschiedene Kategorien. Jeder Kategorie kann in der CSS-Datei unterschiedliche Eigenschaften zugeschrieben werden. Werden beispielsweise Städte visualisiert, können diese nach der Menge der Einwohner in unterschiedliche Kategorien eingeteilt werden. In der CSS-Datei werden nun mehrere Kategorien mit verschiedenen visuellen Eigenschaften wie Farbe, Größe, Transparenz erstellt. So können letztendlich Punkte auf dem Globus durch differenzierte Darstellung unterschieden werden.

Zusätzlich können Punkte nur zu bestimmten Zeiten der Story angezeigt werden. Dazu sind in der Textdatei der Positionen der Punkte zusätzliche Spalten nötig. Diese müssen die Start- und Endzeit enthalten, während die Punkte zu sehen sein sollen. Durch geschickte Notation dieser Zeiten ist es möglich, Bewegung der Punkte zu simulieren. *Smutny* stellte hierzu 2017 in einer Diplomarbeit eine Möglichkeit vor, diese Punkte automatisch zu generieren. Ausgehend von Flugbahnen von Flugzeugen wurden automatisch Dateien erstellt, welche die Routen nachzeichnen. Das Ziel dabei war vor allem, Textdateien maschinell zu generieren, so dass Punkte nicht händisch mit sämtlichen Eigenschaften erstellt werden müssen. Die Positionseigenschaften wurden dabei automatisch von einer Website abgerufen, in eine Textdatei samt weitere Eigenschaften geschrieben und letztendlich in *OmniSuite* integriert. Trotz der automatischen Generierung von Punktdaten erscheint die integrative Methode, Vektordaten während der Laufzeit einer Global-Story mit Rasterdaten, also dem Basismedium, zu vereinen als zu komplex, als dass es möglich wäre ein Partikelsystem auf diese Weise zu realisieren. Hierfür müsste für den vorliegenden Zweck eine Textdatei erstellt werden, welche für eine bestimmte Zeitspanne alle Positionen eines jeden Partikels enthält. Zudem würde die Interaktion mit dem Partikelsystem eine weitere komplexe Herausforderung stellen.

Daher erscheint es sinnvoller, eine Visualisierung zu erstellen, welche auf dem Display des angeschlossenen Computers abläuft, und diese in Echtzeit auf den Globus zu übertragen. Für diesen Zweck kann eine Website erstellt werden, die als Grundlage für das Partikelsystem dient und die als Schnittstelle zur Interaktion zwischen Benutzer und Visualisierung verwendet wird. Auch hierbei müssen jedoch einige Herausforderungen überwunden werden. Diese werden nun im nächsten Kapitel ausführlich beschrieben.

6.3.3 Anforderungen an ein Partikelsystem am taktilen Globus

Da nun bereits dargelegt wurde, dass eine Übertragung eines Bildschirminhaltes auf den Globus die sinnvollste Möglichkeit ist (gegenüber dem Story-Modus), ein interaktives Partikelsystem zu erstellen, müssen nun die Ansprüche an eine derartige Visualisierung besprochen werden.

Ein wesentlicher Punkt, der bei der Darstellung des Partikelsystems zu beachten ist, ist die Verzerrungsproblematik. Wie bereits beschrieben geht der Darstellung auf dem sphärischen Display eine flache Abbildung voraus. Diese muss demnach schon vor Projektion auf dem Globuskörper entsprechend angepasst werden. Das heißt, dass die Grundkarte sowie die darzustellenden Elemente auf der zweidimensionalen Darstellung so angepasst werden müssen, dass diese auf dem Globus verzerrungsfrei zu sehen sind (*Kordasch*, 2019).

Die Grundkarte muss für diesen Zweck in Form einer quadratischen Plattkarte vorliegen (siehe Kapitel 6.2).

Auch für die Partikel, welche aus Kreisformen bestehen werden, müssen Verzerrungen vorgenommen werden. Das bedeutet, dass die Verzerrung der Punkte schon auf der Karte selber stattfinden muss. Würden die Kreise nicht schon auf der Plattkarte entsprechend angepasst, würden die Signaturen auf dem Globus verzerrt dargestellt sein. *Liem* (2012) erstellte zur Verdeutlichung zwei Gegenüberstellungen, die durch Darstellung der Tissotschen Indikatrix, die Problematik verdeutlichen. Die Tissotsche Indikatrix, welche auf den französischen Mathematiker *Nicolas Auguste Tissot* (1824-1897) zurückgeht, steht dabei für eine Methode, bei der durch Darstellung von Ellipsen die Verzerrungseigenschaften verschiedener Kartenprojektionen verdeutlicht werden kann.

Die erste, von *Liem* erstellte Darstellung, ist auf Abbildung *31* zu sehen. Gegenübergestellt sind unverzerrte Kreise auf einer Plattkarte sowie ein Globus, der diese Darstellung auf einer sphärischen Ansicht zeigt.

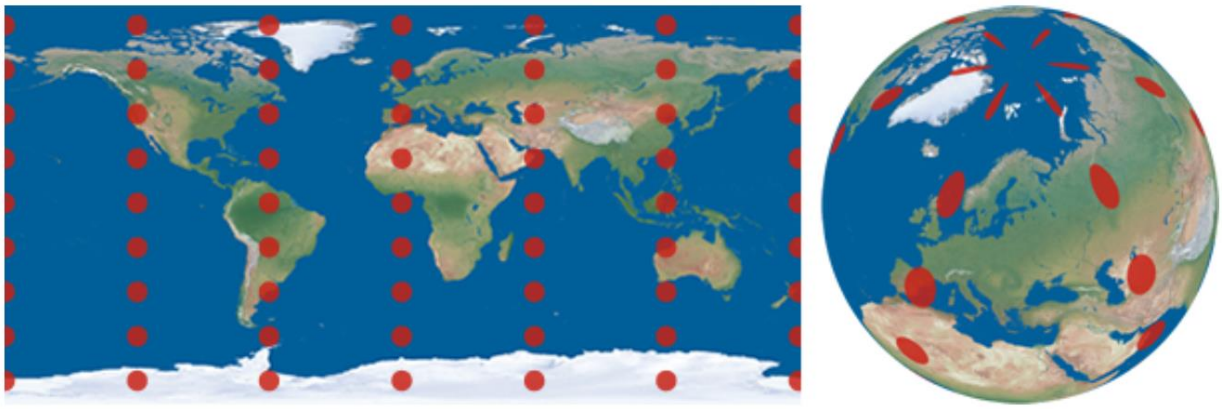

Abbildung 31 unbearbeitete Punktsignaturen auf einer Plattkarte und als Globenansicht. Quelle: Liem, 2012

Es ist deutlich zu sehen, dass kreisrunde Punkte auf einer Plattkarte keine Kreise auf einer Globenform ergeben. Um auch auf einem sphärischen Display Punkte zu erzeugen, muss die Verzerrung daher im Vorfeld berechnet werden.

Dazu wird ermittelt auf welchem Breitengrad sich der Punkt befindet. Anschließend wird folgende Formel angewandt um die Verzerrung zu ermitteln:

$$Verzerrung = \frac{1}{cos(Breite)}$$

Dabei ist zu beachten, dass die Verzerrung nur auf den horizontalen Durchmesser der Kreise angewendet wird. Dieser muss mit dem Wert der Verzerrung multipliziert werden. Die Höhe der Kreise wird dabei nicht verändert. Die bearbeiteten Punkte sind auf Abbildung 32 zu sehen ist. Die Punkte sind nun bereits auf der linken Abbildung, also auf er Plattkarte verzerrt dargestellt. Dagegen erscheinen die Signaturen auf der Globusdarstellung als unverzerrte Kreise.

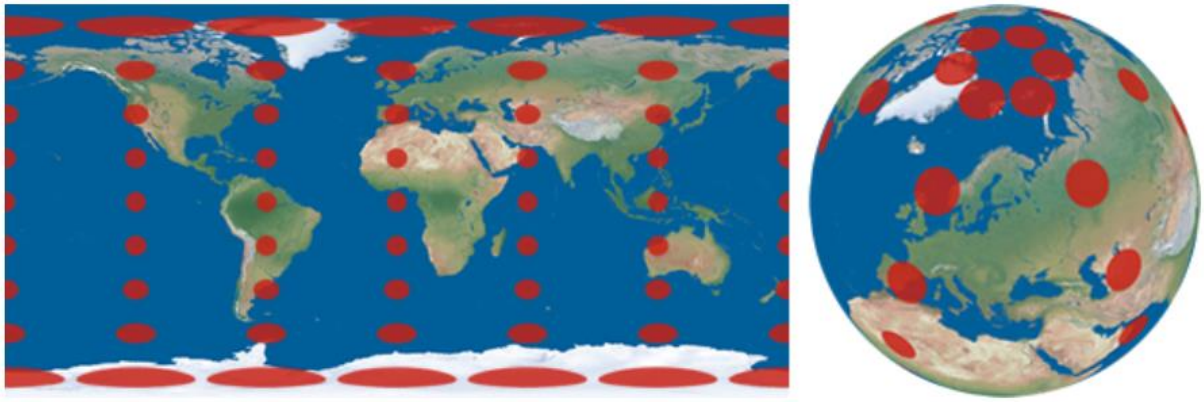

Abbildung 32 Punktsignaturen mit Verzerrung nach tissotsche Indikatrix. Quelle: Liem, 2012

Eine weitere Besonderheit, die bei der Berechnung der Laufbahnen einzelnen Partikel berücksichtigt werden muss, ist die Datumsgrenze am 180° Meridian. Auf der flachen Karte würden Partikel welche die Grenze bei 180°O respektive -180°W überschreiten aus dem System entfernt werden. Das gleiche gilt für Partikel, welche den nördlichen beziehungsweise südlichen Kartenrand erreichen. Für beide Szenarien müssen die Positionen entsprechen berechnet werden. Anstatt Partikel aus dem System zu löschen, werden sie vielmehr an ihren der Laufbahn entsprechend richtigen Punkt auf der Karte platziert.

Für den Übertritt von der westlichen zur östlichen Hemisphäre über die Datumsgrenze und vice versa ergibt sich eine einfache Formel. Wenn eine neu berechnete Position eines Partikels einen Wert annimmt, welcher geringer als -180° ist, wird die Summe aus der neuen Länge und 180 auf den Wert 180 aufaddiert. In dem anderen möglichen Fall, dass die neuberechnete Länge den Wert 180 überschreitet, wird die Differenz aus der Länge und 180 auf den Wert -180 addiert. Zusammengefasst kann der Fall folgendermaßen dargestellt werden:

$$Längengrad = \begin{cases} > 180 & Längengrad = -180 + (Längengrad - 180) \\ < -180 & Längengrad = 180 + (Längengrad + 180) \end{cases}$$

Für den Fall, dass Partikel den Nord- beziehungsweise Südrand der Karte übertreten, muss der Längen- und Breitengrad der neuen Position berechnet werden. Dazu wird zunächst der Breitengrad überprüft. Sollte dieser einen größeren Wert als der nördlichste Punkt, also 90°, sein, wird er wieder zurück auf diesen Wert gesetzt. Das Gleiche passiert bei einem Übertritt des südlichen Randes der Karte. Dementsprechend werden Werte, welche kleiner als -90° annehmen, wieder auf -90° festgelegt.

Als zweites wird der Längengrad überprüft und neu berechnet. Sollte ein Partikel den Nordpol erreichen, setzt es seinen Weg auf der gegenüberliegenden Seite fort. Dies bedeutet für eine Darstellung auf einer Plattkarte, dass das Partikel in diesem Fall einen Sprung zu der korrespondierenden Position in der anderen (östlichen respektive westlichen) Hemisphäre vollzieht. Auf dem Globus übertragen, entspricht dies einer ‚normalen' Flugbahn. Zur neuen Berechnung der Position wird zunächst geprüft auf welcher Seite des Nullmeridians sich das Partikel befindet. Ist der Wert des Längengrades positiv so ergibt sich die neue Position durch aus der Differenz aus -180 und dem alten Wert der Länge. Für den anderen Fall, dass das Partikel sich auf der westlichen Hemisphäre befindet und sich über einen der Pole bewegt, wird die neue Position gebildet, indem der alte Wert der Länge von 180 abgezogen wird.

Die beiden Fälle, die es zu prüfen gilt sobald der Breitengrad der Position eines Partikels größer als 90° respektive kleiner als -90° ist, sind in folgenden Formeln zusammengefasst:

$$\text{Breitengrad} = \begin{cases} > 90 & Breitengrad = 90 \\ < -90 & Breitengrad = -90 \end{cases}$$

$$Längengrad = \begin{cases} \geq 0 & Längengrad = -180 - Längengrad \\ < 0 & Längengrad = 180 - Längengrad \end{cases}$$

Zusammengefasst lässt sich also sagen, dass es für die Implementierung von Partikelsystemen auf taktilen Hypergloben zwei Besonderheiten gibt. Einerseits muss auf die Verzerrungsproblematik von Partikeln geachtet werden, wenn diese auf einer flachen Karte abgebildet sind, welche auf ein sphärisches Display übertragen wird.

Anderseits müssen Positionen von Partikeln, welche sich dem Rand der Plattkarte nähern, überwacht werden. Denn es ist zu beachten, dass Partikel nicht aus dem System austreten können, da auf dem Globus keine Ränder existieren. Bei dem Übertreten der Partikel von einer der Kanten der Karte muss es dementsprechend auf eine andere Position gesetzt werden, so dass auf dem Globus kein Kartenrand zu erahnen ist. Anderenfalls würde unnatürliche Barrieren erscheinen, an denen Partikel verschwinden.

6.3.4 Implementierung

Zur Implementierung des Partikelsystems auf dem taktilen Globus geht, wie bereits dargelegt, eine Übertragung einer zweidimensionalen Visualisierung auf einem Computerbildschirm voraus. Um dem Benutzer die Möglichkeit zu bieten, das Partikelsystem durch Änderung verschiedener Parameter beeinflussen zu können, soll eine Oberfläche erstellt werden, welche das Partikelsystem zeigt und zugleich als Bedienschnittstelle zwischen Benutzer und de verstellbaren Parametern dient. Zu diesem Zweck wird eine Website erstellt. Um möglichst wenige externe Frameworks zu benutzen wurde festgelegt, dass das Partikelsystem mittels HTML-Canvas sowie JavaScript erstellt wird. Durch die geringe Anzahl verwendeter Programmbibliotheken und der Verwendung nativer Funktionen von HTML und JavaScript bleibt der Programmcode übersichtlich und kann auch zu einem späteren Zeitpunkt nachvollzogen werden. Zudem zeigte *Kim* bereits 2018 in einer ähnlichen

Visualisierung, dass HTML5-Canvas und JavaScript für derartige Darstellungen geeignet sind.

Das erstellte Programm besteht aus einem Canvas Element, auf dem das Partikelsystem abgebildet wird sowie einem, durch Betätigung der Leertaste, ein- und ausblendbaren Optionsfenster. Bei dem letzteren kann das Vektorfeld gewählt, sowie verschiedene Parameter des Partikelsystems modifiziert werden. Das Canvas nimmt dabei die gesamte Fläche des Browserfensters ein. Durch die Einstellung des Vollbildmodus des Browsers wird die volle Funktionalität des Programms hergestellt und die Visualisierung kann auf den taktilen Hyperglobus übertragen werden.

Der Quellcode besteht aus einer HTML-Datei und mehreren JavaScript-Dateien. Diese sind im Anhang 1 zu finden.

Nach dem Aufruf der Website ist zunächst nur die quadratische Plattkarte zu sehen, welche den Hintergrund der Visualisierung darstellt, sowie das Optionsfenster in dem zunächst ein Vektorfeld über den Button „Datei auswählen“ gewählt werden muss. Das Vektorfeld muss wie zuvor beschrieben, aufgebaut sein. Über den gleichen Button, kann das Vektorfeld während der Laufzeit der Visualisierung ausgetauscht werden.

Das Optionsfenster beinhaltet darüber hinaus Einstellungsmöglichkeiten zur Partikelanzahl und Partikelgröße. Die Anzahl ist zunächst auf 5000 Partikel festgelegt und kann durch den Benutzer auf bis zu 20.000 Partikel erhöht werden. Die Größe der Partikel kann genau wie die Anzahl über einen Schieberegler variiert werden. Je größer die Partikel dabei sind, desto deutlicher ist die Verzerrung der Kreisformen auf der Plattkarte zu sehen. Auf dem Globenbild erscheinen die Formen dagegen zu jeder Zeit als Kreise. Zuletzt kann die relative Geschwindigkeit der Partikel eingestellt werden. Diese erhöht oder verringert die Bewegungsgeschwindigkeit aller Partikel um den eingestellten Prozentsatz.

6.3.4.1 Allgemeiner Aufbau des Partikelsystems

Im Quellcode spielen vor allem zwei Klassen eine bedeutende Rolle. Die Klassen *Emitter* und *Partikel* sind die zentralen Elemente der Visualisierung und übernehmen

die wichtigsten Operationen. Beide werden in den nächsten Kapiteln näher beleuchtet. Die Reihenfolge in welcher die einzelnen Methoden und Funktionen aufgerufen werden, bestimmt die Funktion *animate.* Alle Klassen und Funktionen werden ab dem nächsten Kapitel ausführlich besprochen.

Bevor die Visualisierung jedoch startet, wird, durch das Hochladen einer passenden Datei durch den Benutzer, ein Vektorfeld erstellt, welches die Grundlage für die Visualisierung darstellt. Der Aufbau einer solchen Datei wurde in Kapitel 6.1.4 besprochen. Ein Ausschnitt einer möglichen Datei findet sich auf Abbildung *27*.

Die Funktion *makeWindmap* erstellt aus den eingegebenen Daten ein Array, welches in Weiterer Folge als Grundlage für alle Positionsberechnungen dient und damit das grundlegende Vektorfeld der Visualisierung darstellt. Dazu werden aus den beiden Komponenten der Vektoren (*u* und *v*) zunächst die Größe beziehungsweise die Länge eines jeden Vektors bestimmt. Dies ist gleichzeitig die Geschwindigkeit des Windes an dieser Position. Die Größe eines Vektors wird mit der Wurzel der addierten Quadrate aller Komponenten des Vektors definiert. Als Formel ausgedrückt, wird die Länge *s* also bestimmt durch:

$$s = \sqrt{u^2 + v^2}$$

Des Weiteren werden für das Partikelsystem die normierten Werte der Vektoren herangezogen. Durch eine Normierung ist festgelegt, dass alle Vektoren die Länge von eins besitzen. Durch diesen Schritt ergeben sich zu einem späteren Zeitpunkt Vorteile in der Positionsberechnung der Partikel, da diese so zunächst pro Frame stets die gleiche Distanz überbrücken. Die Geschwindigkeit, welche die Distanz verlängert oder verkürzt, wird später der Formel hinzugefügt, nachdem diese wiederum mit dem Faktor der Geschwindigkeit, welche durch den Benutzer gewählt wird, verrechnet wurde. Die Normierung der Vektoren wird erreicht indem die Komponenten durch die Länge des Vektors geteilt werden. In Anhang 1.5 Ist die programmatische Umsetzung der eben beschriebenen Schritte zu sehen. Innerhalb der Funktion *makeWindmap* wird zunächst die CSV-Datei mithilfe der Bibliothek *PapaParse* in ein Array geladen, wobei jede Zeile der CSV-Datei nun durch ein eigenes Feld innerhalb des Arrays *windmap* repräsentiert wird. Zu diesem Zeitpunkt beinhaltet dieses nur die beiden Komponenten der jeweiligen Vektoren. Anschließend werden durch die Funktionen *addMagnitude*

und *normVector* die Länge des Vektors sowie die normierten *u*- und *v*-Komponenten in das entsprechende Array hinzugefügt. Letztendlich bestimmt die Funktion *getMinMax* den globalen Minimal- und Maximalwert der Länge der Vektoren. Diese werden jedoch nicht in das Array beigefügt, sondern als Variable gespeichert. Die Werte werden später benötigt um die Partikel einzufärben.

6.3.4.2 Der Emitter

Die Aufgaben des Emitters bestehen vor allem darin neue Partikel zu erstellen und sie mit einer zufälligen Position auf dem Vektorfeld zu verteilen. Dies geschieht mittels der

```
class Emitter{
    // Erstellt und löscht Partikel
    emit(){
        // Partikel mit zufälliger Startposition erstellen

        let n = Math.floor(Math.random() * width) + 1;
        let m = Math.floor(Math.random() * height) + 1;
        let particle = new Particle(n,m,2); //posX, posY, radius
        particles.push(particle);
    }

    deleteParticle(index){
         //Partikel löschen
         particles.splice(index,1);
    }

    increaseParticles(){
        //Partikel erstellen bis ParticleCount erreicht ist
        for (var i = 0; i <= ParticleCount - particles.length; i++){
            let n = Math.floor(Math.random() * width) + 1 ;
            let m = Math.floor(Math.random() * height) + 1 ;

            let particle = new Particle(n,m,2); //posX, posY, radius
            particles.push(particle);
        }
    }

    decreaseParticles(){
        //Partikel löschen bis ParticleCount erreicht ist
        while(particles.length > ParticleCount) {
            var index = Math.floor(Math.random()*particles.length );
            particles.splice(index, 1); // erstes Element des Arrays löschen
        }
    }
}
```

Abbildung 33 Die Klasse Emitter. Hier werden Partikel erstellt und gelöscht. Quelle: eigene Abbildung

Methode *emit*, welche auf Abbildung 33 zu sehen ist. Alle erstellten Partikel werden dabei in der globalen Variable *particles* gespeichert. Des Weiteren hat der Emitter die

Aufgabe Partikel, welche ihre Lebenszeit aufgebraucht haben, zu löschen. Dies geschieht in der Methode *deleteParticle.* Aufgerufen wird diese Methode in der Funktion *animate* (siehe *6.3.4.5 Ablauf der Visualisierung*) nachdem die Lebenszeit der Partikel geprüft wurde. Zudem übernimmt der Emitter zwei weitere Funktionen. Sobald im Optionsfenster der Schieberegler zur Partikelanzahl bedient wurde, verändern die Methoden *decreaseParticles* beziehungsweise *increaseParticles* die Anzahl der Elemente der Variable *particles* und damit die Anzahl der angezeigten Partikel auf dem Globus (siehe Abbildung 33).

6.3.4.3 Die Klasse Partikel

Jedes Partikel des Partikelsystems wird durch eine Instanz der Klasse *Particle* repräsentiert. Sie nimmt während der Animation eine zentrale Rolle des Partikelsystems ein. Die Methoden der Klasse übernehmen die Berechnung der Positionen sowie Überwachung der Lebenszeit der Partikel. Nur die initialen Positionen der Partikel werden im Vorfeld berechnet und der Klasse bei Initialisierung im Emitter übergeben. Der Konstruktor der Klasse besteht damit unter anderem aus den beiden initialen Positionen *x* und *y*, welche in einer Variable *Position* gespeichert werden. Die Koordinaten beziehen sich dabei auf die Position auf dem Canvas. Weiterhin repräsentiert die Variable *Speed* die Größe des Vektors und damit die absolute Geschwindigkeit des Windes an einem bestimmten Punkt der Erde. Der Variable wird zunächst der initiale Wert null zugewiesen. Im Verlauf der Visualisierung erhält sie in jedem Frame den entsprechenden Wert der Windgeschwindigkeit an der Position des Partikels.

Weitere Variablen der Klasse Partikel sind der Radius, der durch den Benutzer verändert werden kann, und die initiale Lebenszeit, die zwischen 100 und 200 Frames zufällig berechnet wird. Die Varianz der Lebenszeit soll einen stetigen Partikelfluss simulieren. Durch unterschiedlich lange Anzeigezeiten der Partikel, werden während der Animation zu keinem Zeitpunkt alle Partikel gleichzeitig gelöscht und neu generiert, was die Illusion einer beständigen Bewegung in der Atmosphäre stören würde.

Weiterhin werden die globalen Positionen der Partikel (*posX, posY*) im Konstruktor gespeichert. Dabei handelt es sich um ein Koordinatenformat, welche die globalen

Breiten- und Längengrade zwischen 0 und 180 beziehungsweise 0 und 360 Grad angibt. Dies hat gegenüber dem WGS84 (World Geodetic System 1984) und gebräuchlichen kartesischen Koordinaten den Vorteil, dass keine Minuswerte existieren. Damit kann zum einen der Index des Koordinatenpaares in einer eindimensionalen Matrix bestimmt und zum anderen die Übertragung der globalen Koordinaten auf die Pixelkoordinaten des Canvas vereinfacht werden. Abbildung 34 zeigt den Konstruktor der Klasse *Particle*.

Weiterhin beinhaltet die Klasse mehrere Methoden. Die wichtigsten sind dabei die Methoden *getNewPosition* und *draw*. Mit diesen werden in jedem Frame neue Positionen der Partikel berechnet und diese anschließend an ebenjener Stelle im Canvas gezeichnet. Die Methode *getNewPosition* ist auf Abbildung 35 zu sehen. Diese Methode übernimmt die Aufgabe „Position berechnen“ im Ablaufplan. Dazu wird zuerst die Position des Partikels im Canvas ermittelt, um anschließend die globalen

```
class Particle  {
    constructor(x,y){

        this.position = new Vector(x,y); // Position des Partikels auf dem
        Cavas
        this.r = document.getElementById("particle-size").value;
        this.life = Math.floor(Math.random() * (100 - 10 + 10) + 100);
        this.speed = 0; //magnitude des Vektors
        this.index = 0;
        this.posX = 0
        this.posY = 0;
    }
```

Abbildung 34 Der Konstruktor der Klasse ‚Particle‘.Quelle: eigene Abbildung

Koordinaten festzustellen. Hierfür wird die Pixelposition in Koordinaten umgewandelt, indem der Wert neu skaliert wird. Die ursprüngliche Position, welche Werte zwischen 0 und der Breite des Displays in Pixeln liegt, wird in einen Wert umgewandelt welcher im gleichen Verhältnis zwischen den Werten 0 und 360 liegt. Das gleiche Verfahren wird auf den Hochwert angewendet. Der ehemalige Wert, der zwischen 0 und der Höhe des Displays in Pixeln liegt, wird demnach in einen Wert zwischen 0 und 180 skaliert. Da die Auflösung der Koordinaten des Vektorfeldes nur ein Grad entspricht, kann während des Verfahrens auf ganze Zahlen gerundet werden.

Über so ermittelten Koordinaten können die Werte im Vektorfeldes ermittelt werden. Da das Vektorfeld in ein eindimensionales Array gespeichert ist und bekannt ist, dass die Koordinaten absteigend der Länge entlang aufgelistet sind, kann die Position im Vektorfeld mit einer einfachen Formel herausgefunden werden:

$$index = y * 360 + x$$

Wobei *y* den Hochwert und *x* den Rechtswert des Koordinatenpaares darstellt. Mit dem Index kann nun das entsprechende Wertepaar innerhalb der Koordinatenliste *windmap* angesprochen werden. Mit der im Vorfeld definierten Datenstruktur ist gesichert, dass sich die normalisierten Komponenten der Vektoren an vierter und fünfter Stelle befinden.

Der Indices im Array sind demnach drei und vier. Beiden Komponenten werden in einem Objekt Namens *normalized* gespeichert.

```
getNewPosition(){

    // aktuelle Position ermitteln
    this.absPosX = Math.round(this.convertRange(
      this.position.x, [ 0, width ], [ 0, 359 ]));

    this.absPosY = Math.round(this.convertRange(
       this.position.y, [ 0, height ], [ 0, 179 ]));

    // Index in eindimensionaler Matrix ermitteln
    var index = this.absPosY * 360 + this.absPosX;

    this.speed = windmap[index][2];
    let normalized = {"u":windmap[index][3],"v":windmap[index][4]}

    this.position.x += normalized.u * (this.speed * 0.25) *
       (ParticleSpeed/100);
    this.position.y -= normalized.v * (this.speed * 0.25) *
      (ParticleSpeed/100);

    //Kanten prüfen
    if (this.position.x > width) {   this.position.x = 0}
    if (this.position.x < 0)     {   this.position.x = width}
    if (this.position.y > height){   this.position.y = 0}
    if (this.position.y < 0)     {   this.position.y = height}
}
```

Abbildung 35 Die Methode getNewPosition. Sie berechnet die neue Position der Partikel. Dabei wird die aktuelle Position mit dem Vektor an dieser Stelle verrechnet. Quelle: eigene Abbildung

Im nächsten Schritt wird die neue Position des Partikels berechnet. Dazu wird die aktuelle Position im Canvas mit dem Vektor an eben jener Stelle des korrespondierenden Koordinatenpaares verrechnet. Für diesen Zweck werden die Vektoren addiert. Wie in Abbildung 35 zu sehen ist, wird *v*-Komponente dabei allerdings von der aktuellen Position abgezogen. Dies geschieht vor dem Hintergrund, dass sich der Nullpunkt des Canvas in der linken obersten Ecke befindet und hohe Werte der *y*-Achse damit Positionen am unteren Rand des Canvas-Elementes bedeuten. Eine Verringerung der *v*-Komponente bedeuten damit Bewegungen nach oben auf dem Canvas beziehungsweise Bewegungen in Richtung Norden.

Um die Bewegungen in nachvollziehbaren Geschwindigkeiten ablaufen zu lassen, wird nicht nur der normalisierte Vektor genutzt, sondern dieser weiterhin um ein Viertel gekürzt. So bewegen sich Partikel pro Frame 25% langsamer und damit für das menschliche Auge auf ersichtlichen Bahnen. Zusätzlich sollen unterschiedliche Windgeschwindigkeiten der Partikel visualisiert werden. Hierzu wird zum einen die Variable *speed* in die Formel aufgenommen. Partikel, die sich an Positionen mit hohen Windgeschwindigkeiten befinden, bewegen sich dadurch merklich schneller als Partikel an windstillen Lagen. Zum anderen werden Partikel entsprechend ihrer Geschwindigkeit eingefärbt. Dies geschieht jedoch zu einem späteren Zeitpunkt in der Methode *draw.*

Der letzte Teil der Formel bestimmt die relative Geschwindigkeit der Partikel. Der Wert *ParticleSpeed* kann durch den Benutzer über einen Schieberegler im Optionsfenster angepasst werden. Standardmäßig ist dieser Wert auf 100% gestellt. Damit wird die gesamte Formel mit eins multipliziert. Durch Verringerung des Wertes auf beispielsweise 50% wird der errechnete Vektor um die Hälfte gekürzt was zu kürzeren Sprüngen der Partikel zwischen den Frames führt und damit zu einer langsamer ablaufenden Visualisierung. Eine Erhöhung des Wertes führt dementsprechend zu schnelleren Partikelbewegungen.

Zum Schluss der Funktion *getNewPosition* wird überprüft, ob die neu-berechnete Position der Partikel die Datumsgrenze beziehungsweise die Pole überschreiten.

```
edges(x,y){

        if (x > width) {x = 0 + (x - width)}
        if (x < 0)     {x = x + width}


        if (y < 0 || y > height) {

            y < 0 ? y = 1 : y = height;

            if (x > width2){
                let diff = Math.abs(width2 - x)
                x = width2 - diff
            }
            else if (x < width2){
                x = width - x
            }
        }
        return [x,y]
    }
```

Abbildung 36 Die Funktion edges. Hier wird der Übertritt der Partikel über die Datumsgrenze und über die Pole überprüft. Quelle: eigene Abbildung

Dieser Fall wird in der Funktion *edges* überprüft. Die Funktion ist in der Abbildung 36 dargestellt. Zu sehen ist, dass zunächst die *x*-Achse überprüft wird. Wie in Kapitel 6.3.3 (*Anforderungen an ein Partikelsystem am taktilen Globus*) bereits erläutert wurde, wird das Partikel bei Übertritt der Grenze um die Differenz zwischen neuer Position und ebenjener Grenze auf „der anderen Seite“ der Plattkarte gesetzt.

Anschließend wird die Position der Partikel auf der *y*-Achse kontrolliert. Die Partikel werden an diesem Punkt auf der jeweils anderen korrespondierenden Seite der westlichen beziehungsweise östlichen Hemisphäre gesetzt. Der Wert der *y*-Achse wird auf den nördlichsten Punkt, wenn das Partikel den Nordpol erreicht, oder den südlichsten Punkt, wenn das Partikel den Südpol erreicht, zurückgesetzt.

Sind die neuen Positionen der Partikel berechnet, werden diese anschließend auf dem Canvas gezeichnet. Dies übernimmt die Methode *draw* der Klasse *Particle* (siehe Abbildung 37). Hier wird zunächst der Breitengrad ermittelt auf welchem sich das

```
draw(){
        //Breitengrad ermitteln
        let latitude;
        if (this.position.y > height2  ){
            latitude = this.convertRange(this.position.y, [0, height2],
                       [90, 0])}
        else {
            latitude = this.convertRange(this.position.y, [height2, height],
                       [0,-90])
        }

        // Verzerrung berechnen
        let verzerrung = 1 / Math.cos(latitude * (Math.PI / 180));

        // Partikel zeichnen
        ctx.beginPath();
        if (ParticleSize > 1){
            ctx.ellipse(this.position.x, this.position.y, this.r*verzerrung
            ,this.r,  0, 0, 2 * Math.PI);
        }
        else{
            ctx.rect(this.position.x, this.position.y,1,1)
        }

        //Farbwert auf Gradienten ermitteln. langsam = gelb; schnell=rot
        ctx.fillStyle = getColorRGBA(this.speed);
        ctx.fill();

    }
```

Abbildung 37 Die Methode draw, hier werden die Partikel auf dem Canvas gezeichnet. Quelle: eigene Abbildung

Partikel befindet. Dazu wird der Wert der Pixelkoordinate mit dem gleichen Verfahren wie zuvor beschrieben und auf einen Wert zwischen -90 und 90 skaliert. Mit dem Wert des Breitengrades kann anschließend die Verzerrung berechnet werden. Dies geschieht während das Partikel auf dem Canvas gezeichnet wird. Um eine Ellipse darzustellen, müssen in dem Befehl *ellipse* zunächst die Koordinaten und weiterhin die Radien der horizontalen und vertikalen Achse angeführt werden. Dabei wird die horizontale Achse mit dem Wert der Verzerrung multipliziert. Damit erscheinen die Partikel nun auf der Plattkarte in horizontaler Richtung verzerrt. Auf dem Globus ist dagegen ein Kreis zu sehen. Sollte die vom Benutzer eingestellte Partikelgröße jedoch auf nur einen Pixel eingestellt sein, wird anstelle einer Ellipse ein Rechteck mit der Größe eines Pixels gezeichnet. In diesem Fall muss keine Indikatrix berechnet werden.

Die geringe Größe und damit der verminderte Rechenaufwand bei der Darstellung des Partikels verbraucht zudem weniger Ressourcen und lässt die Animation auch bei hoher Partikelanzahl flüssig laufen.

Des Weiteren wird dem Partikel eine Farbe zugewiesen. Diese wird über die Geschwindigkeit ermittelt. Den Farbwert bestimmt die Funktion *getColorRGBA* welche im Anhang 1.5 zu sehen ist. Diese ermittelt aufgrund der Geschwindigkeit des Partikels eine Farbe auf einer Skale zwischen gelb (für geringe Geschwindigkeiten) und rot (für hohe Geschwindigkeiten). Diese Skala wird dabei für jedes Vektorfeld, welches vom Benutzer gewählt wird, neu ermittelt. Dabei orientiert sich die Funktion an der maximalen Geschwindigkeit, welche in dem Vektorfeld auftaucht.

Eine letzte wichtige Methode der Klasse *Particle* widmet sich der Überwachung der Lebenszeit. Die einzige Aufgabe der Methode namens *lifetime* ist es, die Lebenszeit in jedem Frame um eins zu verkürzen. Die Methode wird in der Funktion *animate* aufgerufen.

6.3.4.4 Initialisierung der Visualisierung

Bevor die Animation startet, wird eine festgelegte Anzahl an Partikeln erstellt. Im weiteren Verlauf der Visualisierung kann die Anzahl durch den Benutzer verändert werden. Die Erstellung wird in der Funktion *init* realisiert. Sie ist Abbildung 38 dargestellt. Nachdem die Klasse *Emitter* in der Variable *emitter* initialisiert wurde, wird die anfängliche Anzahl der

```
function init() {
    emitter = new Emitter;
    //Partikel initialisieren
    ParticleCount = document.getElementById('particle-count').value;
    for (var i = 0; i < ParticleCount; i++){
        emitter.emit()
    }
 window.requestAnimationFrame(animate)
}
```

Partikel, welche durch einen Schieberegler im Optionsfenster festgelegt wird, abgerufen. Anschließend werden entsprechend viele Partikel durch den Emitter (siehe *6.3.4.2 Der Emitter*) erstellt. Die letzte Zeile der Funktion ruft die Funktion *animate* auf und startet damit die Animation.

6.3.4.5 Ablauf der Animation – die Funktion animate

Nachdem die Visualisierung in der Funktion *init* gestartet wurde, übernimmt die Funktion *animate* die Steuerung des Partikelsystems. Im ersten Schritt der Funktion *animate* wird geprüft, ob der Variable *windmap* bereits ein Wert zugewiesen wurde und damit ob der Benutzer bereits ein Vektorfeld zur Animation ausgewählt hat. Nur in diesem Fall werden die folgenden Schritte ausgeführt.

Die weiteren Schritte sind (vgl. siehe Abbildung 40):

1. Inhalte des Canvas löschen
2. Lebenszeit der Partikel prüfen (durch *Particle*)
3. Gegebenenfalls neue Partikel erstellen lassen (durch *Emitter*)
4. Neue Positionen der Partikel berechnen lassen (durch *Emitter*)
5. Partikel und Hintergrund zeichnen (durch *Particle*)

Schritt eins wird durch einen nativen Befehl des HTML5 Canvas ausgelöst. Anschließend folgt der Kern der Funktion *animate.* Dieser besteht aus einer Schleife die das Array *particles* durchläuft und für jedes Partikel die eben beschriebenen Punkte 2 bis 5 abarbeitet. Abbildung 39 zeigt diese Schleife.

```
particles.forEach(function(particle, index, object){
      //lifespan verkürzen
      particle.lifecircle();

      // Lebenszeit des Partikels aufgebraucht?
      if (particle.life <= 0){
            emitter.deleteParticle(index)
           emitter.emit();
      }
      particle.getNewPosition(); // neue Position
      particle.draw(); // Partikel auf Canvas zeichnen
})
```

Abbildung 39 Der Kern der Funktion 'animate'. Hier wird jedes Partikel durchlaufen und die Lebenszeit geprüft um es anschließend an einer neuen Position zu zeichnen. Quelle: eigene Abbildung

Zuletzt wird in der Funktion *animate* geprüft, ob einer der Schieberegler im Optionsfester verstellt wurde. Im Falle einer Veränderung der Partikelanzahl, wird die entsprechende Methode zur Erhöhung oder Reduzierung im Emitter aufgerufen (siehe Abbildung 33).

Letztendlich ruft sich die Funktion *animate* selber auf. Damit beginnt die Visualisierung wieder mit der Bereinigung des Canvas und der erneuten Zeichnung der Partikel nachdem

die Lebenszeit geprüft und neue Positionen berechnet wurden. Der gesamte Ablauf der Animation ist auf Abbildung 40 zusammengefasst.

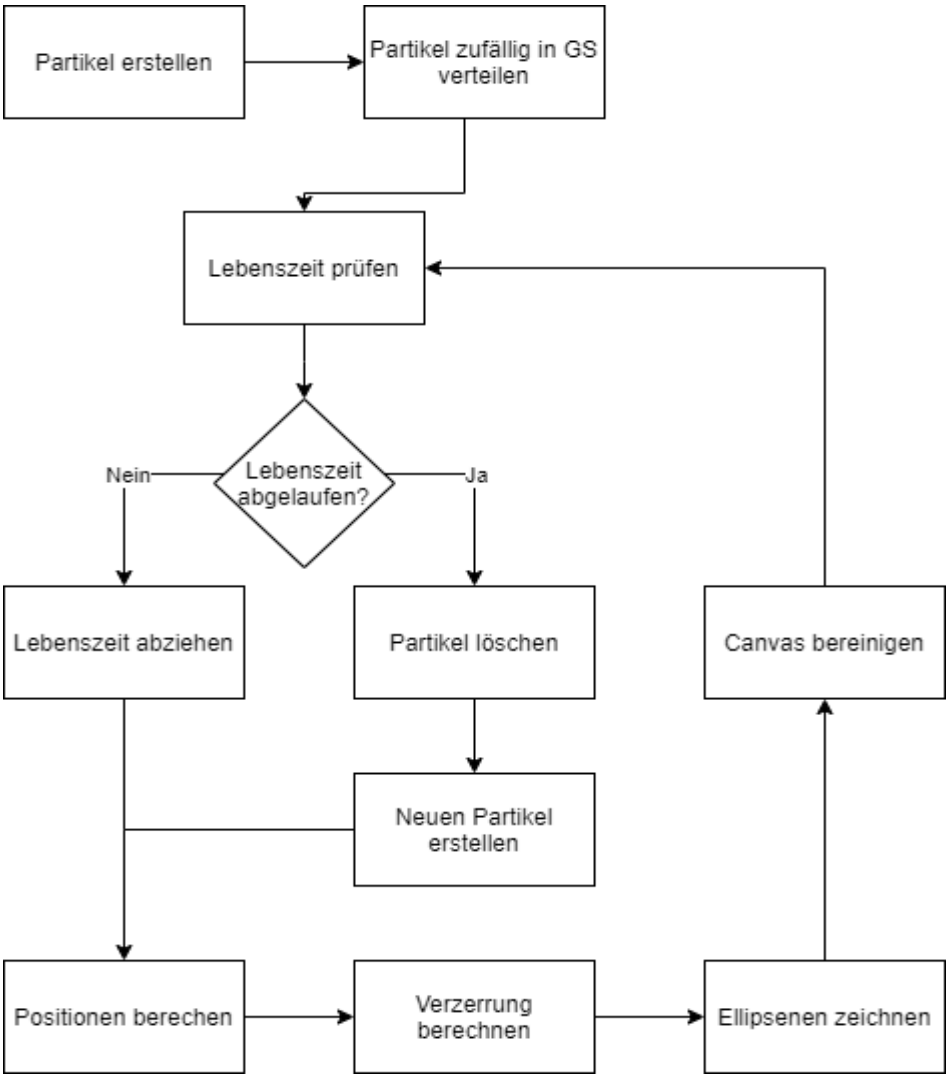

Abbildung 40 Ablaufplan der Visualisierung. Quelle: eigene Darstellung

6.3.5 Ergebnisse

Das Ergebnis der Implementation ist eine Website, auf welcher die globalen Strömungsverhältnisse angezeigt werden können. Es existiert ein Optionsfenster auf dem es möglich ist, Parameter der Partikel zu verändern. Des Weiteren kann der entsprechende Button in diesem Fenster genutzt werden, um ein Vektorfeld auszuwählen, das in der Folge visualisiert wird. Zur Visualisierung wurde die gesamte verfügbare Fläche des Browserfensters genutzt. Die volle Funktionalität ist hergestellt, sobald der Vollbildmodus im Browser eingestellt wurde. In diesem Modus kann der Bildschirm auf den taktilen Globus übertragen werden. Abbildung 41 zeigt das Ergebnis samt eingeschalteten Optionsfenster.

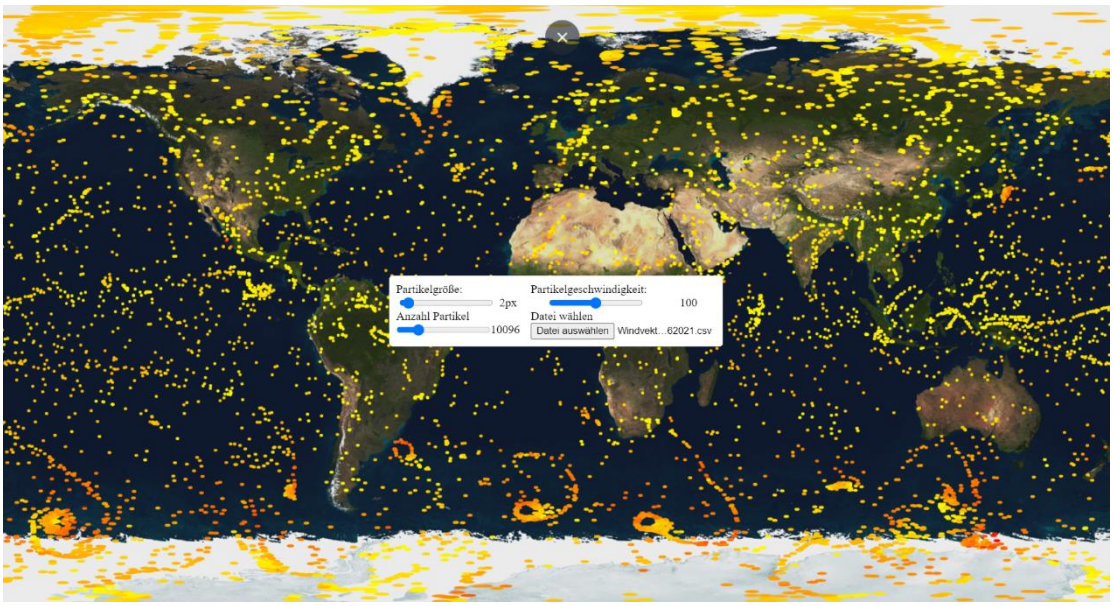

Abbildung 41 Die Visualisierung eines Partikelsystems angepasst an ein sphärisches Display. Partikel werden ihrer Lage auf der Plattkarte entsprechend verzerrt. Quelle: eigene Abbildung

Zur Beurteilung der Qualität der Visualisierung wird zunächst geprüft, ob die Herausforderungen an ein Partikelsystem am taktilen Hyperglobus überwunden wurden. Eine wichtige Anforderung an das System war die korrekte Darstellung von dynamischen Kreissignaturen auf einem sphärischen Display. Zur Gewährleistung der korrekten Darstellung von Kreisen wurden die Partikel in horizontaler Richtung verzerrt. Das Ergebnis ist auf Abbildung 41 zu sehen. Je südlicher beziehungsweise nördlicher, ausgehend vom Äquator, sich die Partikel befinden, desto größer wird der horizontalen Durchmesser des Kreises. Der vertikale Durchmesser ändert sich dabei nicht. Ein Vergleich beider Ansichten ist auf Abbildung 42 zu sehen. Beide Abbildungen zeigen den

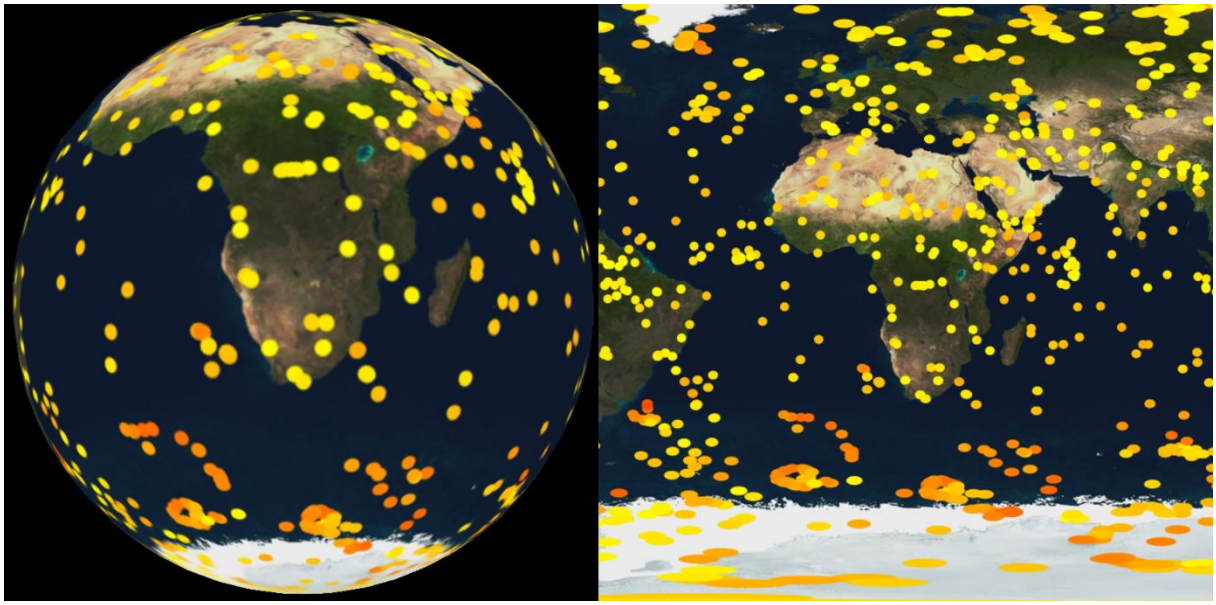

Abbildung 42 Gegenüberstellung des Partikelsystems auf einer Kugel und auf einer flachen Karte. Quelle: eigene Abbildung

gleichen Ausschnitt einer Plattkarte. Eindeutig zu sehen ist dabei, dass die Kreise auf der linken Darstellung stets die gleiche Form annehmen. Währenddessen zeigen die Partikel auf der rechten Seite markante horizontale Verzerrungen.

Eine weitere Herausforderung war die Übertragung der Flugbahnen über die Kanten der Karte hinaus. Da auf der globalen Ansicht die Grenze zwischen den Hemisphären entlang des 180. Längengrades nicht sichtbar ist, müssen auch die Flugbahnen der Partikel so angepasst werden, dass ein Übergang der Hemisphären für den Betrachter nicht anhand der Partikelbewegung nachvollziehbar ist. Softwareseitig musste zu diesem Zweck jede errechnete Position erneut geprüft werden. Mit den in Kapitel 6.3.3 (*Anforderungen an ein*

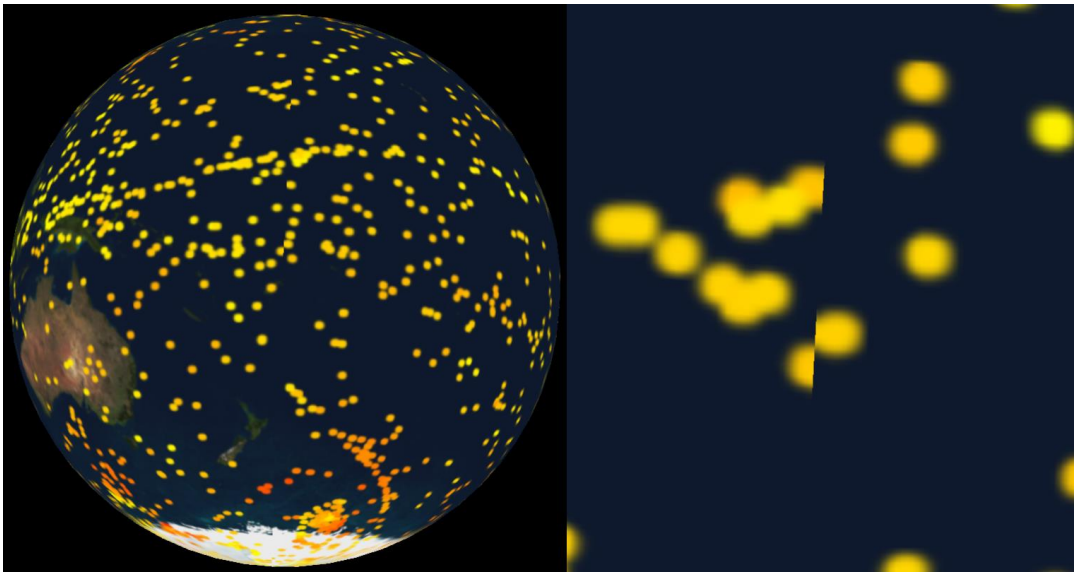

Abbildung 43 Übertritt der Partikel über den 180. Längengrad. Quelle: eigene Abbildung

Partikelsystem am taktilen Globus) vorgestellten und in Formeln ließ sich dieses Problem überwinden. Obwohl so die Bahnen von dem Übertritt der Hemisphären unberührt bleiben, offenbart Abbildung 43 ein weiteres Problem. Sobald sich ein Partikel der Grenze nähert und die Entfernung zwischen dem 180. Längengrad und dem Partikelmittelpunkt kleiner als der Radius des Partikels ist, wird nur jener Teil des Partikels gezeichnet, welcher sich zwischen seinem Mittelpunkt und dem Längengrad befindet. Dieses Problem fällt vor allem bei Partikeln mit großem Radius auf. Bei kleinen Partikelgrößen und einer hohen eingestellten Geschwindigkeit ist diese Problematik dagegen kaum zu sehen. Dennoch muss dieses Phänomen nachträglich als eine Anforderung an das Partikelsystem auf dem taktilen Globus aufgenommen werden.

Neben der Beurteilung der Funktionalität des Systems werden Vergleiche des Ergebnisses angestellt, die zeigen sollen, ob die Partikel die korrekten Strömungen wiedergeben. Dazu wurde die Anwendung mit etablierten Wettervisualisierungen gegenübergestellt. Ein Vergleich mit der Website *windy.com* ist auf Abbildung 44 zu sehen. Die Gegenüberstellung zeigt die Situation am 29.06.2021 und stellt die Winde 10m über der Oberfläche dar. Die Gegenüberstellung zeigt, dass sich die Partikel korrekt verhalten. Besonders gut zu sehen, ist dies an den Verwirbelungen südlich und

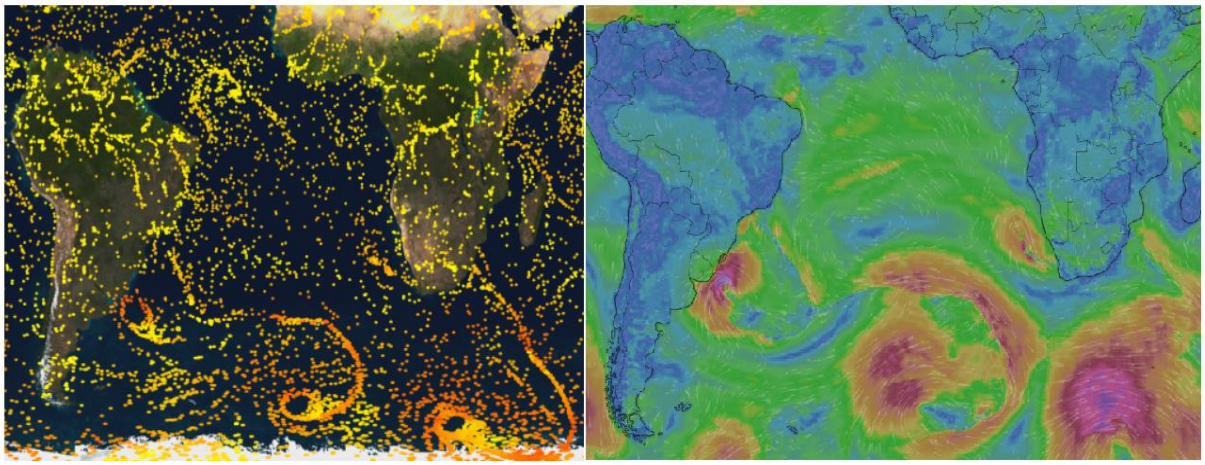

Abbildung 44 Vergleich der Anwendung (links) mit der Visualisierung von windy (rechts) der atmosphärischen Strömungen am 29.06.2021. Quelle: links: eigene Abbildung, rechts: windy.com

südwestlich von Afrika. Auch ein kleiner Wirbel östlich von Südamerika ist markant und auf beiden Abbildungen klar ersichtlich.

Eine weitere Gegenüberstellung zeigt die Situation über dem Indischen Ozean am 30.06.2021. Der Vergleich auf Abbildung 45 wird in diesem Fall mit dem *Earth* Projekt von *Baccario* angestellt. Zu sehen ist, dass sich die Partikel auf beiden Visualisierungen ähnlich Verhalten. Südwestlich von Indien ist eine Verwirbelung der Atmosphäre zu sehen.

Südlich davon bewegen sich die Partikel auf beiden Seiten der Abbildung vertikal von Ost nach West.

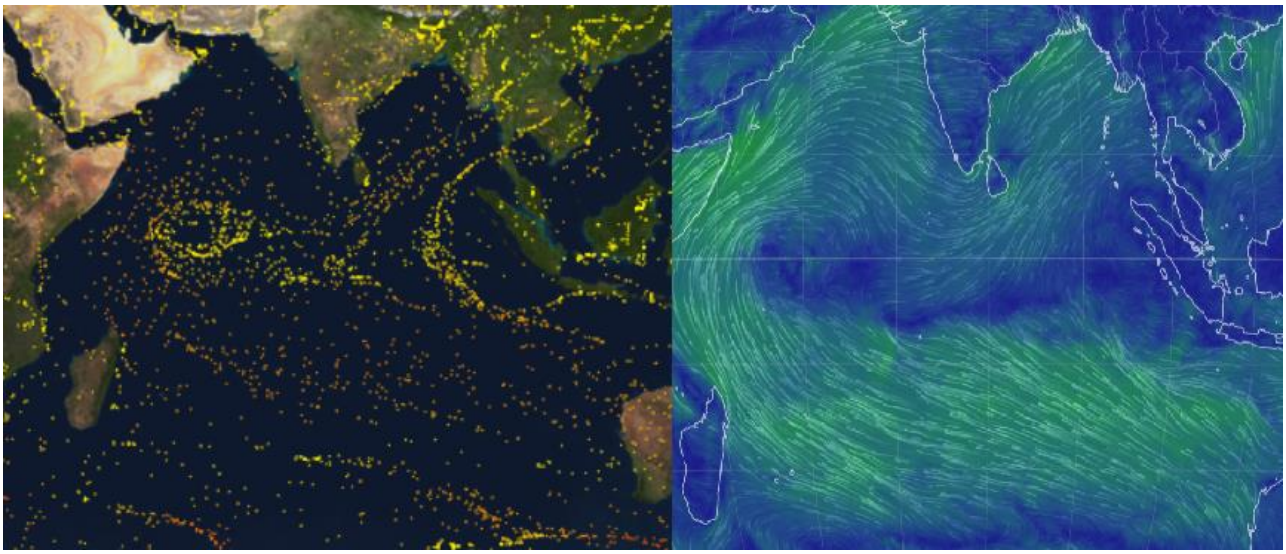

Abbildung 45 Gegenüberstellung der Visualisierung (links) mit dem Projekt "Earth" (rechts). Quelle: links: eigene Abbildung, rechts: earth.nullschool.net

Auffällig ist auf beiden Vergleichen jedoch, dass sich Partikel in der vorliegenden Visualisierung in bestimmten Gebieten anhäufen. Zumeist geschieht dies in Zonen, in denen Winde aus mehreren Richtungen aufeinandertreffen. Konvergenzzonen werden so durch die Visualisierung optisch hervorgerufen.

6.4 virtueller Hyperglobus

Dieses Kapitel beschäftigt sich mit der Implementierung eines Partikelsystems auf einem virtuellen Globus. Für diesen Zweck werden zunächst die Anforderungen an ein Partikelsystem am Globus erläutert. Anschließend werden verschiedene Möglichkeiten präsentiert, wie ein virtueller Globus erstellt werden kann. Anschließend wir die Implementierung des Globus und des Partikelsystems detailliert beschrieben

6.4.1 Anforderungen an ein Partikelsystem

Die Anforderungen an ein Partikelsystem auf dem virtuellen Globus gleichen sich in einigen Punkten mit dem zuvor betrachteten Partikelsystem. Auch in diesem Fall müssen die Grenzen zwischen -180° und 180° beachtet werden. Obwohl bei dem virtuellen Globus keine „Kanten“ zu sehen sind wie etwa bei der Plattkarte, muss bei der Positionsberechnung dennoch darauf geachtet werden, dass die Zielposition, welche sich aus aktueller Position eines Partikels sowie einem Windvektor ergibt, zwischen den

Werten -180 und 180 beziehungsweise 0° und 360° liegt. Anderenfalls kommt es zu Problemen der Koordinatentransformation sowie des Auffindens eines neuen Windvektors im Vektorfeld. Daher gelten auch bei dem Partikelsystem des virtuellen Globus die gleichen Anforderungen an die Positionsbestimmung wie zuvor in Kapitel 6.3.3 (*Anforderungen an ein Partikelsystem am taktilen Globus*) beschrieben.

Eine weitere Herausforderung, die im speziellen Fall bewältigt werden muss, ist die ständige Transformation zwischen kartesischen und sphärischen Koordinaten. Unter ersteren werden die XYZ-Koordinaten des dreidimensionalen Raumes verstanden, in welchem der virtuelle Globus implementiert wird. Letzteres sind die Längen- und Breitengrade, die benötigt werden, um die Vektoren des Windes an einer konkreten Position des Vektorfeldes zu bestimmen.

Zur Umrechnung von sphärischen Koordinaten zu kartesischen Koordinaten muss zunächst der Polarwinkel bestimmt werden. Dieser ergibt sich auf der Differenz von 90° und dem Breitengrad. In Abbildung 46 ist dieser mit θ (theta) gekennzeichnet. Des Weiteren werden der Azimutwinkel φ (phi) benötigt. Dieser muss, wenn Koordinaten zwischen -180° und 180° verwendet werden, um einen Wert von 180 erhöht werden. Zuletzt muss der Radius bekannt sein.

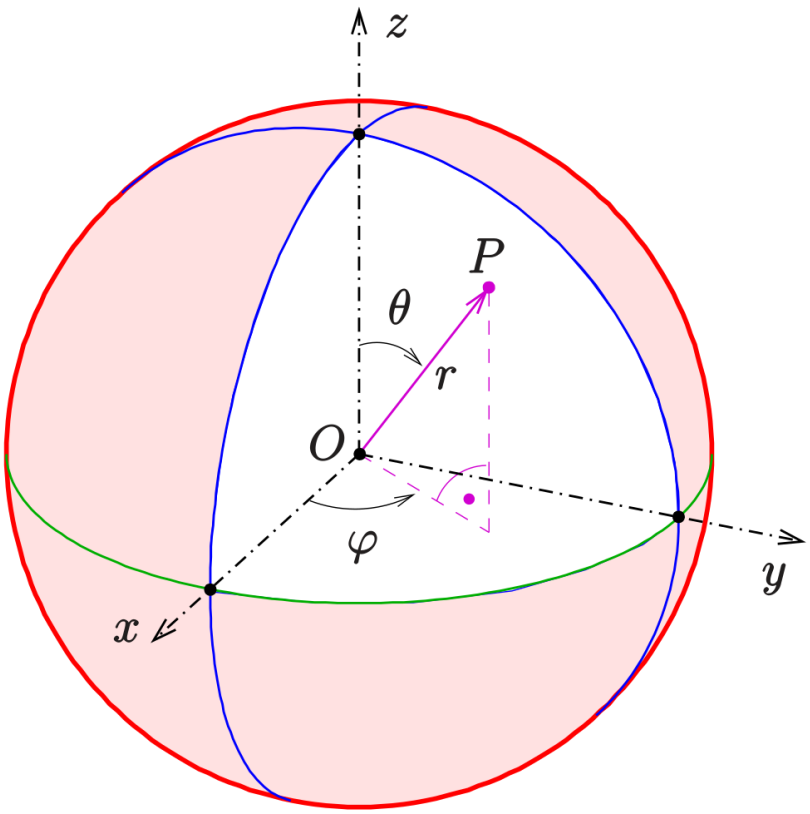

Abbildung 46 Übersicht über Kugel- und Kartesische Koordinaten.

P = zu bestimmender Punkt; r, θ, φ = sphärische Koordinaten; x,y,z = kartesische Koordinaten

Quelle: wikimedia.org

Mit diesen gegebenen Variablen können die kartesischen Koordinaten wie folgt berechnet werden:

$$X = Radius * sin(\theta) * cos(\varphi)$$

$$Y = Radius * sin(\theta) * sin(\varphi)$$

$$Z = Radius * cos(\theta)$$

Die Formel zur Transformation der sphärischen Koordinaten zu kartesischen Koordinaten lautet:

$$\theta = cos^{-1}(\frac{z}{Radius})$$

$$\varphi = atan2(x, y)$$

Zu beachten ist dabei, dass θ den Winkel zwischen Pol und den zu berechnenden Punkt angibt. Um den Breitengrad ausgehend vom Äquator zu erhalten, muss der Wert daher noch von 90 abgezogen werden. In Kapitel 6.4.3.4 (*Koordinatentransformation*) wird erläutert, wie diese Berechnung im Programmcode für den konkreten Fall des virtuellen Globus implementiert wurde.

Es wurde bereits angemerkt, dass zur Erstellung eines virtuellen Globus, Partikel auch weiterhin im System verbleiben sollen, wenn sie sich nicht im sichtbaren Bereich auf dem Bildschirm befinden. Um eine Abgrenzung und Weiterentwicklung zu bestehenden Anwendungen dieser Art zu gewährleisten, muss in diesem Fall ein 3D-Raum generiert werden, der Partikel unabhängig von der Interaktion mit dem Globus darstellt.

6.4.2 Möglichkeiten zur Implementierung

Auch das Partikelsystem auf dem virtuellen Globus soll in Echtzeit berechnet werden. Ein Rendering, welches etwa mit Software wie *Cinema4D* erstellt wurde, ist daher nicht zulässig. Vielmehr soll auch in diesem Fall eine Website erstellt werden, welche die Positionen der Partikel während der Laufzeit der Animation berechnet. Eine Möglichkeit die sich anbietet, ist die rasterbasierte Karte samt Partikeln des Partikelsystems zuvor auf einen virtuellen Globus zu modellieren. Allerdings würde das neben den Ressourcen zur Erstellung des Rasters zusätzlich für jeden Frame die Berechnung dieser Modellierung auf den Globus bedeuten. Ein vektorbasierter Ansatz scheint in diesem Fall angemessener. In diesem Fall wird ein Partikel durch einen Vertex repräsentiert, für welches die Position in einem virtuellen 3D Raum bekannt ist. In jedem Frame wird die Position eines jeden Partikels berechnet anschließend einzeln versetzt anstelle sie auf ein Raster zu zeichnen, das anschließend auf den Globus gelegt wird.

Für diesen Fall bieten sich mehrere JavaScript Frameworks an mit denen, dieses Ziel zu erreichen ist. Bibliotheken, die für diesen Zweck in Frage kommen sind unter anderem Cesium, Babylon, A-Frame und ThreeJS. Diese Frameworks nutzen die WebGL Schnittstelle (API), um Inhalte im Browser darzustellen. Durch WebGL können zwei- und dreidimensionale Objekte konstruiert und im Browser animiert werden. Dabei verwendet die API neben dem Hauptprozessor vor allem die Rechenleistung der Grafikkarte, was bedeutet, dass auch anspruchsvolle Animationen im Browser dargestellt werden können. Es liegt also im Ermessen des Erstellers, welches Framework genutzt werden kann, um

Animationen mittels WebGL zu erstellen. In diesem Fall wird aufgrund der Erfahrung mit der Bibliothek des Autors die ThreeJS Bibliothek verwendet.

6.4.3 Implementierung

Die nächsten Kapitel widmen sich der Implementierung des virtuellen Globus sowie des Partikelsystems. Dabei wird zunächst der grobe Aufbau der Website beschrieben. Im weiteren Verlauf wird genauer auf den Programmcode eingegangen.

6.4.3.1 Aufbau der Website

Bei Aufruf der Website ist zunächst nur ein Globus sowie ein Optionsfenster zu sehen. Der Globus zeigt zu diesem Zeitpunkt lediglich zufällig verteilte Partikel, welche sich noch nicht bewegen. Der Globus kann während der Visualisierung beliebig rotiert werden. Auch die Zoomstufe ist verstellbar.

Im Optionsfenster sind zum einen die Parameter zur Partikelgröße und -anzahl zu sehen. Beide sind über einen Schieberegler verstellbar. Zudem ist ein Button zu sehen, über den ein Vektorfeld angegeben werden kann. Sollte dieses in der richtigen Konfiguration vorliegen, startet die Visualisierung. Das Vektorfeld kann zu jeder Zeit über denselben Button ausgetauscht werden.

Partikel werden in diesem Partikelsystem durch Billboards repräsentiert. Die Größe der Partikel ist absolut. Das heißt, dass sich auch beim Heranzoomen an den Globus, die Größe der Partikel nicht verändert. Die Gründe hierfür werden im folgenden Kapitel erläutert.

Das Partikelsystem für den virtuellen Globus wurde mit Hilfe von *ThreeJS* verwirklicht. Die Klassen und Funktionen dieser Bibliothek nehmen daher eine zentrale Rolle in dem System ein. Analog zum Partikelsystem auf dem taktilen Globus wurde eine Klasse *Emitter* geschrieben. Diese übernimmt das Aussenden neuer Partikel, sowie die Erhöhung und Reduzierung der anzuzeigenden Partikel, bei Parameteränderung durch den Benutzer. Partikel werden in diesem Fall jedoch nicht durch eine eigene Klasse repräsentiert. Vielmehr werden sie in einem *Buffer* gesammelt, welcher durch *ThreeJS* bereitgestellt wird. Wie in dem System des taktilen Globus werden jedoch zunächst alle Elemente in einer Funktion *init* initiiert.

6.4.3.2 Initialisierung der Szene

Um mit *ThreeJS* eine dreidimensionale Szene darstellen zu können, müssen zunächst einige Grundlagen eingerichtet werden. Dazu zählt die Szene an sich, die Kamera, Licht

```
//Szene, Kamera, Licht und Renderer
   scene = new THREE.Scene();

   camera = new THREE.PerspectiveCamera( 45, window.innerWidth/window.innerHeight,
0.1, 1000 );
   camera.position.z = 3;

   var mDirectionallight = new THREE.DirectionalLight( 0xBBBBBB );
   mDirectionallight.position.y = 5;
   mDirectionallight.position.z = 13;
   scene.add( mDirectionallight);

   renderer = new THREE.WebGLRenderer();
   renderer.setSize( window.innerWidth, window.innerHeight );
   renderer.setPixelRatio( window.devicePixelRatio );
   document.body.appendChild( renderer.domElement );
```

Abbildung 47 Einstellung der Szene, des Lichts, der Kamera und des Renderers. Quelle: eigene Abbildung

sowie der Renderer. Die Erstellung dieser Elemente findet in der Funktion *init* statt (siehe Abbildung 47). Als erstes wird eine Szene erstellt. Dazu wird eine Instanz des entsprechenden Objektes von *ThreeJS* der Variable *scene* zugewiesen. Alle Elemente (Globus, Partikel et cetera), welche angezeigt werden sollen, müssen in der Folge der Szene hinzugefügt werden. Des Weiteren wird eine Kamera benötigt. Nur Elemente welche sich innerhalb des Sichtfeldes der Kamera befinden, werden auf dem Bildschirm angezeigt. Die Erstellung der Kamera benötigt mehrere Parameter. Diese sind der Bildwinkel, das Seitenverhältnis sowie die Angaben über die nahe und ferne Begrenzung des Sichtkegels. Abbildung 48 verdeutlichten die Parameter der Kamera.

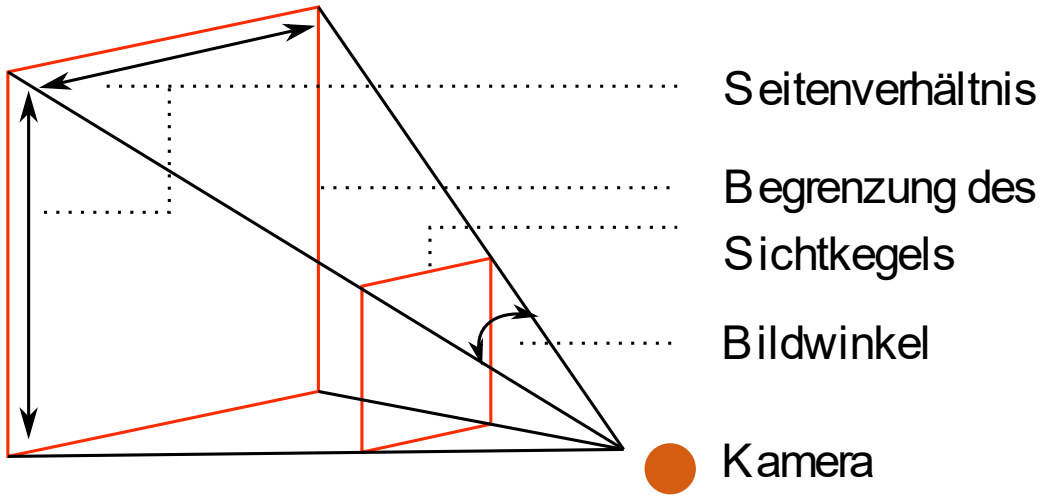

Abbildung 48 Parameter der Kamera. Quelle: eigene Darstellung

Des Weiteren muss eine Lichtquelle definiert werden. Hierfür gibt es mehrere Möglichkeiten. Auf Abbildung 49 sind die Varianten des Lichts gegenübergestellt. Da die zu erstellende Szene keinen Anspruch auf realitätsnahe Beleuchtung des Globus durch die Sonne legt, wird in diesem Fall das *Ambiente Light* (Umgebungslicht) gewählt. Dadurch wird die Szene von allen Seiten gleichermaßen beleuchtet und es erscheint kein Schatten.

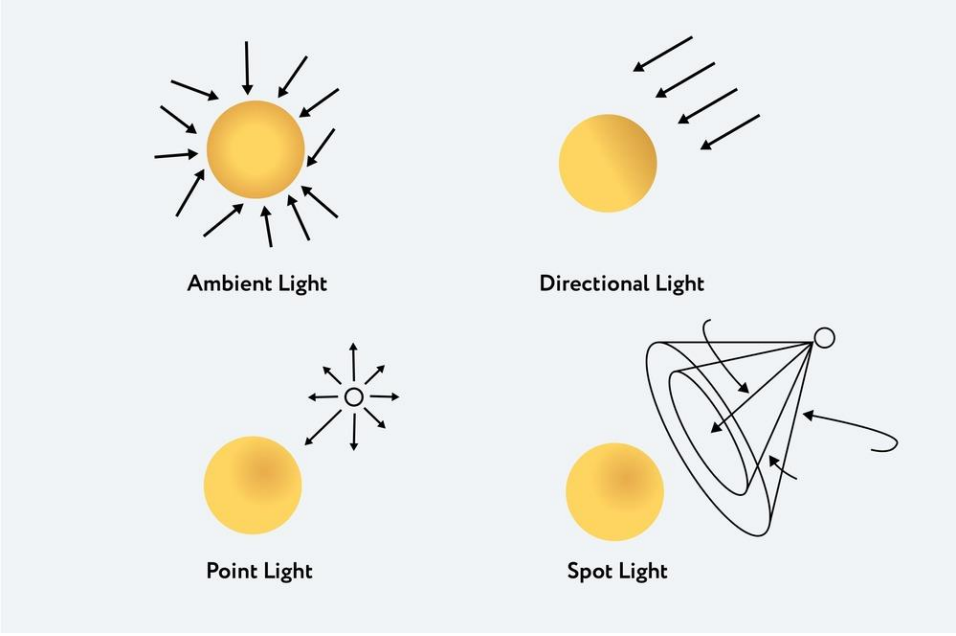

Abbildung 49 Verschiedene Möglichkeiten von Lichtquellen. Quelle: Intexsoft

Ein letztes entscheidendes Element zur Generierung einer Szene ist der Renderer. Dieser ist dafür zuständig, dass die Objekte der Szene auf dem Bildschirm gezeichnet werden.

Bei Erstellung des Renderers müssen ihm die Kamera sowie die Szene übergeben werden. Somit wird festgelegt, welche Elemente gerendert werden sollen.

Nachdem alle wesentlichen Grundlagen für eine dreidimensionale Szene festgelegt wurden, können nun Objekte modelliert werden. Im Zentrum der Szene soll ein Globus stehen auf dem das Partikelsystem zu sehen sein wird. Um ein dreidimensionales Objekt mithilfe von *ThreeJS* zu modellieren, wird eine Geometrie benötigt, die die Form bestimmt

```
    // Globus
    var texture = new THREE.TextureLoader().load( bildURL );
    var globeMaterial = new THREE.MeshBasicMaterial({
        map:texture
    });
    var globeGeometry = new THREE.SphereGeometry( radius, 32, 32 );
    globe = new THREE.Mesh(globeGeometry, globeMaterial);
    scene.add( globe );
```

Abbildung 50 Die Erschaffung des Globus. Quelle: eigene Darstellung

und ein Material welches die Erscheinung festlegt. Im vorliegenden Fall soll eine Kugel modelliert werden. Die passende Geometrie in der Bibliothek von *ThreeJS* nennt sich *SphereGeometrie*. Mit der Angabe über den Radius, sowie der Anzahl der Segmente, welche über die Auflösung der Oberfläche bestimmen, wird so ein Kugelobjekt geschaffen, so zu sehen auf Abbildung 50. Das Material besteht in diesem Fall aus einer Textur, welche eine Komposition aus Satellitenbildern der Erdoberfläche. Abgerufen wird das Bild dabei über einen Link, da lokale Bilder aus Sicherheitsgründen von vielen Webbrowsern blockiert werden. Letztendlich wird die Geometrie sowie das Material der *ThreeJS*-Klasse *Mesh* übergeben, welche ein dreidimensionaler Körper formt. Dieser ist in der Variable *globe* gespeichert. Mit dem hinzufügen des Kugelobjektes in die Szene, ist die Modellierung des Globus abgeschlossen.

Auch die Partikel werden durch eine Geometrie und einem Material repräsentiert. Dafür wird die von *ThreeJS* bereitgestellt *BufferGeometry* verwendet. Mit dieser Klasse können

```
//Partikel
particleGeometry = new THREE.BufferGeometry();
const pointsMaterial = new THREE.PointsMaterial({sizeAttenuation:false,
      size: initParticleSize, color:"lightgrey"} );
points = new THREE.Points(particleGeometry, pointsMaterial );
scene.add( points );
```

Abbildung 51 Geometrie und Material der Partikel. Quelle: eigene Abbildung

Linien-, Punkt- und Gittergeometrien konstruiert werden. Dabei lassen sich zahlreiche Objekte innerhalb einer einzigen Instanz einer *BufferGeometry* speichern. Dies bedeutet, dass ein einziger Aufruf zum rendern dieser Geometrie genügt, um tausende Objekte zur gleichen Zeit darzustellen. Das spart Rechenzeit, was für die Implementierung äußert gewinnbringend ist, da bis zu 100.000 Partikel visualisiert werden können. Das Material der Partikel wird durch die Klasse *PointsMaterial* repräsentiert. Dabei handelt es sich um die Standardform, um Punkte mit *ThreeJS* darzustellen. Die Gestalt der Partikel wird in einem späteren Kapitel beschrieben (*6.4.3.5 Partikel*). Die Geometrie sowie das Material müssen, genau wie bei dem Globus, in einer Klasse zusammengefasst werden, um sie der Szene zu übergeben (siehe Abbildung 51).

Damit ist die Szene fertig konstruiert und alle nötigen Elemente hinzugefügt. Mit dem Aufruf des Renderes durch den Befehl *renderer.render(scene, camera)* innerhalb der Funktion *animate* werden alle Objekte, die der Szene übergeben wurden, animiert. Um Bewegungen darstellen zu können, müssen zwischen dem wiederholten Aufruf des Renderers, die Positionseigenschaften der Objekte manipuliert werden. Die Berechnung der Koordinaten, welche die Grundlage für die Bewegungen der Partikel darstellt wird im anschließenden Kapitel beleuchtet.

6.4.3.4 Koordinatentransformation

Eine wesentliche Aufgabe, die das Programm zu erfüllen hat, ist die Koordinatentransformation. Ausgehend von einem Partikel, dass sich auf einem Punkt auf dem Globus befindet, soll zunächst das entsprechende sphärische Koordinatenpaar berechnet werden. Um diese zu ermitteln, werden die Formel zur Koordinatentransformation angewendet, die im Kapitel 6.4.1 (*Anforderungen an ein Partikelsystem am virtuellen Globus*) erarbeitetet wurden. Die praktische Anwendung ist auf Abbildung 52 dargestellt. Es ist zu beachten, dass die Achsen im System von *ThreeJS* nicht den Achsen auf Abbildung 46 entsprechen. Daher sind in der Anwendung die *y*- und *z*-Achsen der Formeln vertauscht. Eine weitere Besonderheit besteht in der Position des 0-Meridians der Länge. Dieser muss stets um 90° in Richtung Westen verschoben werden, um an die *ThreeJS* Koordinaten angepasst zu werden. Denn der Nullmeridian befindet sich im System von *ThreeJS* auf Höhe von -90°W, gemessen vom Meridian von Greenwich.

```
function XYZ_to_LL(x,y,z){
    //lat
    let thetaRad = Math.acos(y / Math.sqrt(x*x+y*y+z*z))
    let thetaDeg = THREE.Math.radToDeg(thetaRad);
    let lat = 90 - thetaDeg;
    //lon
    let phiRad = Math.atan2(x,z);
    let lon = THREE.Math.radToDeg(phiRad);
    lon = lon - 90;
    lon < -180 ? lon += 360: null;

    return {lat,"lon":lon}
}
```

Abbildung 52 Die Funktion XYZ_to_LL berechnet Breiten- und Längengrad ausgehend von einer dreidimensionalen Kartesischen Koordinate. Quelle: eigene Abbildung

Um die Rücktransformation im Programm Code umzusetzen, können ebenfalls die bereits vorgestellten Formeln genutzt werden. Dabei ist zu beachten, dass *ThreeJS* unterschiedliche Achsenbezeichnungen benutzt. Auf Abbildung 53 ist die Umsetzung der Formeln im Programmcode dargestellt. Zunächst wird aus der Breite der Polarwinkel berechnet, welcher *theta* bezeichnet und im Bogenmaß angegeben wird. Von der Länge werden 90° abgezogen, um sie an die Gegebenheiten von *ThreeJS* anzupassen. Auch die Länge wird anschließend in Bogenmaß umgerechnet. Anschließend erfolgt die Anwendung der drei Formeln für die entsprechenden Achsen. Die Benennung der Achsen ist zur Übersichtlichkeit als Kommentar hinzugefügt wurden (bezogen auf Abbildung 46).

```
function LL_to_XYZ(lat,lon){

    let theta = THREE.Math.degToRad(lat -90);
    let phi = THREE.Math.degToRad(lon - 90);

    let x = Math.sin(theta) * Math.sin(phi);    // X in Threejs |  Y auf Abbildung
    let y = Math.cos(theta);                    // Y in Threejs |  Z auf Abbildung
    let z = Math.sin(theta) * Math.cos(phi);    // Z in ThreeJS |  X auf Abbildung

    return {x,y,z}
}
```

Abbildung 53 Berechnung der kartesischen Koordinaten aus Breiten- und Längengrad. Quelle: eigene Abbildung

6.4.3.5 Partikel

Anders als im Partikelsystem des taktilen Globus, existiert in dieser Implementierung keine Klasse, welche mit ihren Methoden die Positionsberechnung übernimmt oder die Lebenszeit überprüft. Die Koordinatenermittlung erfolgt dagegen mittels der Funktion *moveParticle*. Gespeichert werden alle Eigenschaften der Partikel in dem von *ThreeJS* bereitgestellter Geometrietyp *BufferGeometry*. Zur Speicherung der Informationen der Partikel werden zwei eindimensionalen Arrays genutzt. In einem wird die Positionen der Partikel, angegeben in den XYZ-Koordinaten des 3D Raumes, abgespeichert. Ein weiteres beinhaltet Informationen über die Lebenszeit der Partikel sowie die korrespondierenden Positionen, angegeben im WGS84 Koordinatenformat. Auf Abbildung 54 ist das Prinzip dargestellt, wie die Zuordnung von Eigenschaften der Partikel gelingt.

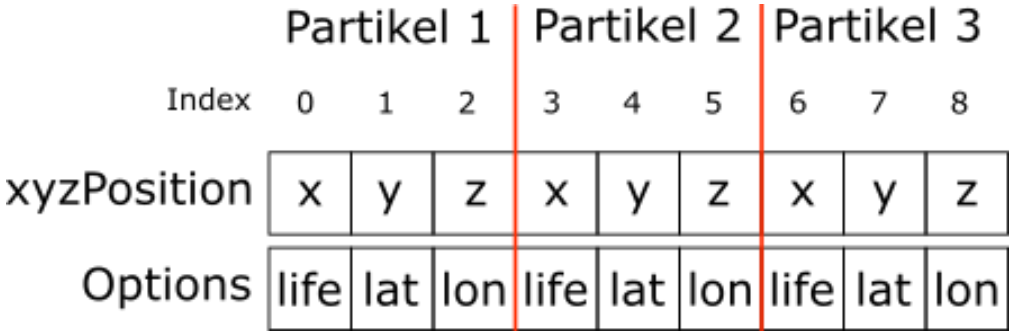

Abbildung 54 Prinzip der Speicherung von Eigenschaften der Partikel in zwei verschieden Arrays. Quelle: eigene Abbildung

Die Repräsentation der Partikel wird durch Billboards realisiert. Darunter sind zweidimensionale quadratische Plättchen zu verstehen, die zu jedem Zeitpunkt dem Benutzer zugewandt sind. Durch die geringe Größe der Partikel erscheinen diese jedoch als punkthafte Elemente. Da zu keinem Zeitpunkt eine Rotation der Plättchen berechnet werden muss, spart die Animation so an Ressourcen.

6.4.3.6 Emitter

Auch dieses Partikelsystem verfügt über einen Emitter. Die gleichnamige Klasse übernimmt die Aufgaben der Initialisierung der Partikel sowie die Erhöhung und Reduzierung der Partikelanzahl.

Noch vor Start der Animation wird in der Funktion *init* eine Instanz der Klasse Emitter in der Variable *emitter* gespeichert. Anschließend wird die Methode *initialiseParticles* aufgerufen. Diese Methode ist auf Abbildung 55 zu sehen. Zunächst werden zwei Arrays

erstellt, die zum einen die kartesischen Koordinaten und zum anderen die Lebenszeit und sphärischen Koordinaten beinhalten werden.

Im Anschluss werden die Arrays mit Inhalten gefüllt. Wie zuvor in Abbildung 54 ersichtlich, muss die Zahl der Elemente in den Arrays drei Mal so hoch sein wie die Zahl der Partikel des Systems, da jedes Partikel drei Felder des Arrays einnimmt. Zur Befüllung der Arrays wird zunächst ein zufälliger Punkt auf dem Globus ermittelt und dessen Koordinaten in das Array *position* geschrieben. Des Weiteren wird die initiale maximale Lebenszeit der Partikel ermittelt, indem ein zufälliger Wert zwischen 100 und 200 berechnet wird. Wie bei dem Partikelsystem zuvor, soll dies einen kontinuierlichen Fluss der Partikel simulieren. In dasselbe Array werden anschließend die korrespondierenden Breiten- und Längengeraden der XYZ-Position aus dem ersten Array eingetragen. Die Berechnung übernimmt die Funktion *XYZ_to_LL* (vgl. Abbildung 52*).* Zuletzt werden die beiden nun gefüllten Arrays der entsprechenden Eigenschaft der *BufferGeometry* zugeschrieben.

```
 initialiseParticles(particleCount){
        let positions = [];
        let lifetime = [];

        for (var i = 0; i <= particleCount * 3; i++){
            let XYZpos = randomSpherePoint(0,0,0,radius)
            positions.push(XYZpos[0])
            positions.push(XYZpos[1])
            positions.push(XYZpos[2])

            lifetime.push(THREE.Math.randInt(100,200))
            let initLatLon = XYZ_to_LL({x:XYZpos[0],y:XYZpos[1],z:XYZpos[2]})
            lifetime.push(initLatLon.lat);
            lifetime.push(initLatLon.lon);
        }

     particleGeometry.setAttribute('position', new THREE.Float32BufferAttribute(
            positions, 3 ) );
     particleGeometry.setAttribute('lifetime', new THREE.Float32BufferAttribute(
            lifetime, 3 ) );
}
```

Abbildung 55 die Methode initialiseParticles. Hier werden die Partikel des Partikelsystems zufällig auf dem Globus vereilt und ihnen ihre Eigenschaften (Lebenszeit und Position) zugeschrieben. Quelle: eigene Abbildung

Neben der Initialisierung erfüllt der Emitter die Aufgabe der Erhöhung und Reduzierung der Partikel. Dies findet in den entsprechenden Methoden *increaseParticles* und *decreaseParticles* statt. Der Quellcode ist in Angang 2.4 zu finden.

6.4.3.7 Die init Funktion – Vorbereitung der Visualisierung

Die Initialisierung des Partikelsystems beginnt sobald die Website aufgerufen wurde. Dies beinhaltet die Erstellung der Szene sowie die Initialisierung der Partikel. Solange über den Button „Vektorfeld laden“ noch keine passende Datei ausgewählt wurde, startet die Animation nicht. Sobald dies jedoch geschieht, wird das Vektorfeld erstellt, mit dem sich die Positionen der Partikel später errechnen. Dies geschieht auf die gleiche Weise wie es bereits bei dem taktilen Globus erläutert wurde. Nach Fertigstellung des Vektorfeldes startet die Animation in dem die Funktion *animate* aufgerufen wird.

6.4.3.8 Ablauf der Visualisierung

Nach Initialisierung der Visualisierung wird die Funktion *animate* aufgerufen (vgl. Abbildung 56). Diese stellt sicher, dass die benötigten Attribute der Geometrien stets aktualisiert werden. Am Ende der *animate*-Funktion wird sie erneut durch sich selbst aufgerufen. Dadurch entsteht eine Schleife wodurch die dynamische Visualisierung realisiert wird. In jedem Durchgang werden neue Positionen der Partikel berechnet. Dies geschieht in der Funktion *moveParticle*. Sie steuert die Bewegung der Partikel.

```
function animate(){
    points.geometry.attributes.position.needsUpdate = true;
    points.geometry.attributes.options.needsUpdate = true;
    if (windmap){
        moveParticle();
    }
    controls.update();
    renderer.render(scene, camera);
    window.requestAnimationFrame(animate);
}
```

Abbildung 56 Die Funktion animte. Sie steuert den Ablauf der Animation. Quelle: eigene Abbildung

Um die Partikel zu bewegen werden innerhalb der Funktion *moveParticle* folgende Schritte durchgeführt:

1. Lebenszeit der Partikel überprüfen
    a. Wenn Lebenszeit aufgebraucht ist neues Partikel erstellen
2. Position des Partikels ermitteln
3. Windvektor an Position ermitteln

4. Neue Position aus Windvektor und aktueller Position errechnen
5. Partikel rendern

Solange die Visualisierung nicht unterbrochen wird, fängt das Programm, durch die beabsichtigte Dauerschleife des Selbstaufrufes der Funktion *animate*, nach dem fünften Schritt wieder bei Punkt 1 an. Es werden nun die Schritte näher beleuchtet.

1- Lebenszeit der Partikel prüfen

Nach dem Aufruf der Funktion *moveParticle* wird zunächst die Lebenszeit der Partikel überprüft. Sollte die Lebenszeit des aktuellen Partikels abgelaufen sein, wird sie auf einen zufälligen Wert zwischen 100 und 200 zurückgesetzt und eine neue zufällige Position für das Partikel auf dem Globus ermittelt. Diese Funktionsweise ersetzt die übliche Vorgangsweise des Löschens und neu Erstellens von Partikeln. Dies erfolgt aufgrund der Architektur der genutzten Geometrie und bewirkt letztendlich den gleichen Effekt.

2/3 - Position des Partikels und Windvektor ermitteln

Sollte die Lebenszeit des Partikels nicht abgelaufen sein, wird zunächst die Position ermittelt. Damit ist zunächst das sphärische Koordinatenpaar in dem entsprechenden Array gemeint. Denn mit diesem lässt sich in der Folge der Windvektor im Vektorfeld ausfindig machen. Die Zuweisung der Positionen in die Variablen *particleLat* und *particleLon* erfolgt über eine Deklaration der Werte des Arrays *options* an der Stelle des aktuellen Index (siehe die schematische Darstellung auf Abbildung 54).

Anschließend werden, identisch mit dem Partikelsystem am taktilen Globus, die sphärischen Koordinaten auf Werte zwischen 0 und 180 respektive 0 und 360 skaliert. So lassen sich im nächsten Schritt die Windkomponenten *u* und *v* aus dem Array *windmap* bestimmen. Der Programmcode der beschriebenen Schritte ist auf Abbildung 57 zu sehen.

```
//lebenszeit senken
let life = points.geometry.attributes.options.array[index]- 1;
newOptions.push(life);

//sphärische Koordinaten
let particleLat = points.geometry.attributes.options.array[index+1];
let particleLon = points.geometry.attributes.options.array[index+2];

// von -90/90 -180/180 zu 0-179 0-359 für Index
let absll = ll_to_absll(Math.floor(particleLat),Math.floor(particleLon));

let fieldIndex = absll.lat * 360 + absll.lon;
let u = windmap[fieldIndex][3];
let v = windmap[fieldIndex][4];
```

Abbildung 57 Abziehen der Lebenszeit und Ermittlung der Position auf dem Globus. Anschließend werden die Windkomponenten neu berechnet. Abbildung: eigene Abbildung

4 - Neuberechnung der Position

Mit den ermittelten Komponenten des Vektors lässt sich nun die Berechnung der neuen Position durchführen. Dies geschieht indem der Vektor der Position mit dem Vektor des Windes addiert wird. Auch in diesem Fall wird, wie in dem Partikelsystem zuvor, der Windvektor verkürzt, damit eine für menschliche Augen nachvollziehbare Geschwindigkeit der Partikel gewährleistet wird. Sollten die Koordinaten der neuen Position außerhalb der zulässigen Werte für die Koordinaten liegen, werden sie im nächsten Schritt korrigiert. Dazu werden die Prüfungen angewandt, welche in zu den Anforderungen an das Partikelsystem am virtuellen Globus beschrieben wurden. Da Partikel, durch die Verkürzung der Vektoren nur sehr geringe Bewegungen durchführen, wurden die Koordinatenberechnung beim Übergang des 180. Längengrades angepasst. In der praktischen Umsetzung wird nun der Längengrad pauschal 180 gesetzt sobald ein Partikel einen geringeren Wert als -180°W, beziehungsweise auf -180, wenn ein Partikel einen Wert größer als 180°E annimmt, gesetzt. Die Berechnung der Differenz von der Position des Partikels zum Meridian entfällt entsprechend. Dies hat aber keine Auswirkungen auf die Visualisierung, da wie bereits erwähnt, die Strecken die ein Partikel zwischen zwei Frames zurücklegt sehr gering sind. Die programmatische Umsetzung ist in Angang 2.3 einzusehen.

5 – Partikel rendern

Wenn diese Schritte für alle Partikel durchgeführt wurden, werden die Partikel auf dem Globus entsprechend versetzt. Im Programmcode geschieht dies durch die Zuweisung der beiden befüllten Arrays *newPositions* und *NewOptions* zu den entsprechenden Attributen der Geometrie *position* und *options*. Ist dies vollzogen, wird die Szene durch den *Renderer* auf dem Bildschirm dargestellt. Zuletzt wird die Funktion *animate* erneut durch sich selber aufgerufen. Dadurch entsteht eine Schleife und die beschriebenen Schritte werden abermals durchgeführt.

6.4.4 Ergebnisse

Das Ergebnis der Implementierung zeigt einen Globus, auf dem die globalen Windverhältnisse zu einem bestimmten Zeitpunkt durch sich bewegende Partikel dargestellt sind. Der Globus lässt sich während der Visualisierung durch Mausinterkation rotieren, vergrößern und verkleinern. Des Weiteren wurde ein Optionsfenster eingerichtet, mit welchen sich zum einen ein Vektorfeld auswählen lässt, und zum anderen Parameter

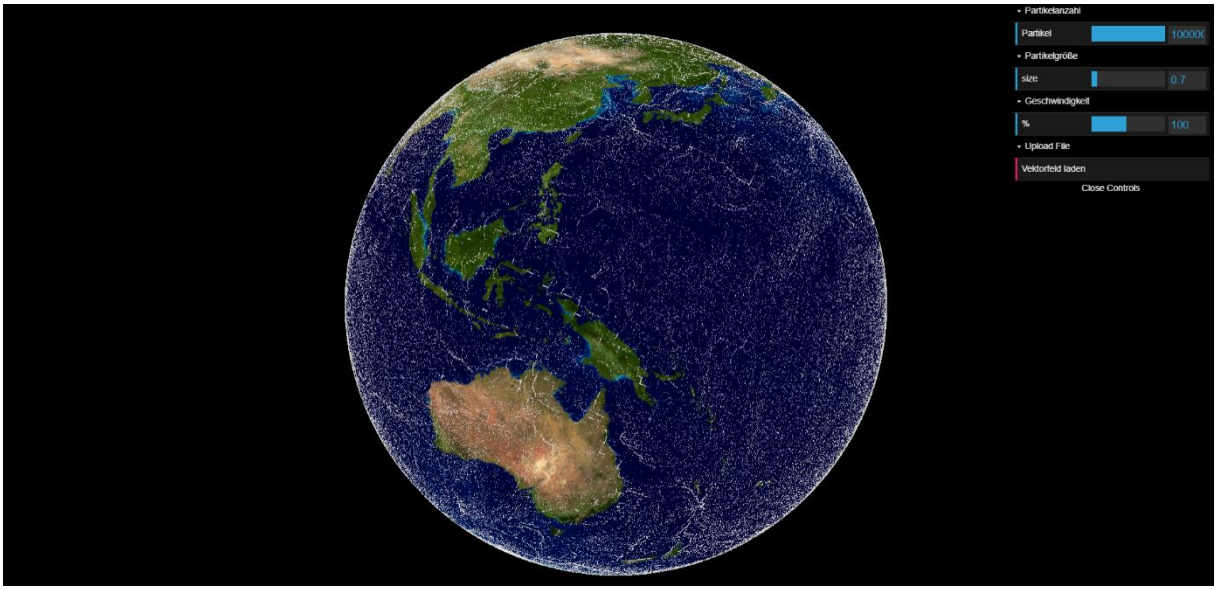

Abbildung 58 Screenshot des virtuelllen Hyperglobus. Quelle: eigene Abbildung

verstellen lassen. Dabei kann der Nutzer die Anzahl der Partikel, die Größe der Partikel sowie die Geschwindigkeit der Partikel verändern. Abbildung 58 zeigt einen Screenshot des virtuellen Hyperglobus, auf dem ein Partikelsystem visualisiert ist.

Während der Animation ist es möglich, mit dem Globus zu interagieren ohne dass die Visualisierung gestoppt wird beziehungsweise von vorne beginnt.

Zur Qualität der Umsetzung und damit zur Genauigkeit der Bewegungen der Partikel, wurde das vorliegende Partikelsystem mit etablierten Visualisierungen verglichen. Dabei zeigt sich, dass die atmosphärischen Strömungen vor allem im großen Maßstab korrekt angezeigt werden. Jedoch wurde festgestellt, dass die Bahnen der Partikel in großen Maßstäben zum Teil unnatürlich wirken.

Abbildung 59 zeigt eine Gegenüberstellung des Ergebnisses mit der Darstellung des *earth-Projektes* von *Baccario.* Beide Darstellungen zeigen die Windverhältnisse zum gleichen Zeitpunkt an der Oberfläche der Erde. Es ist zu sehen, dass auf beiden

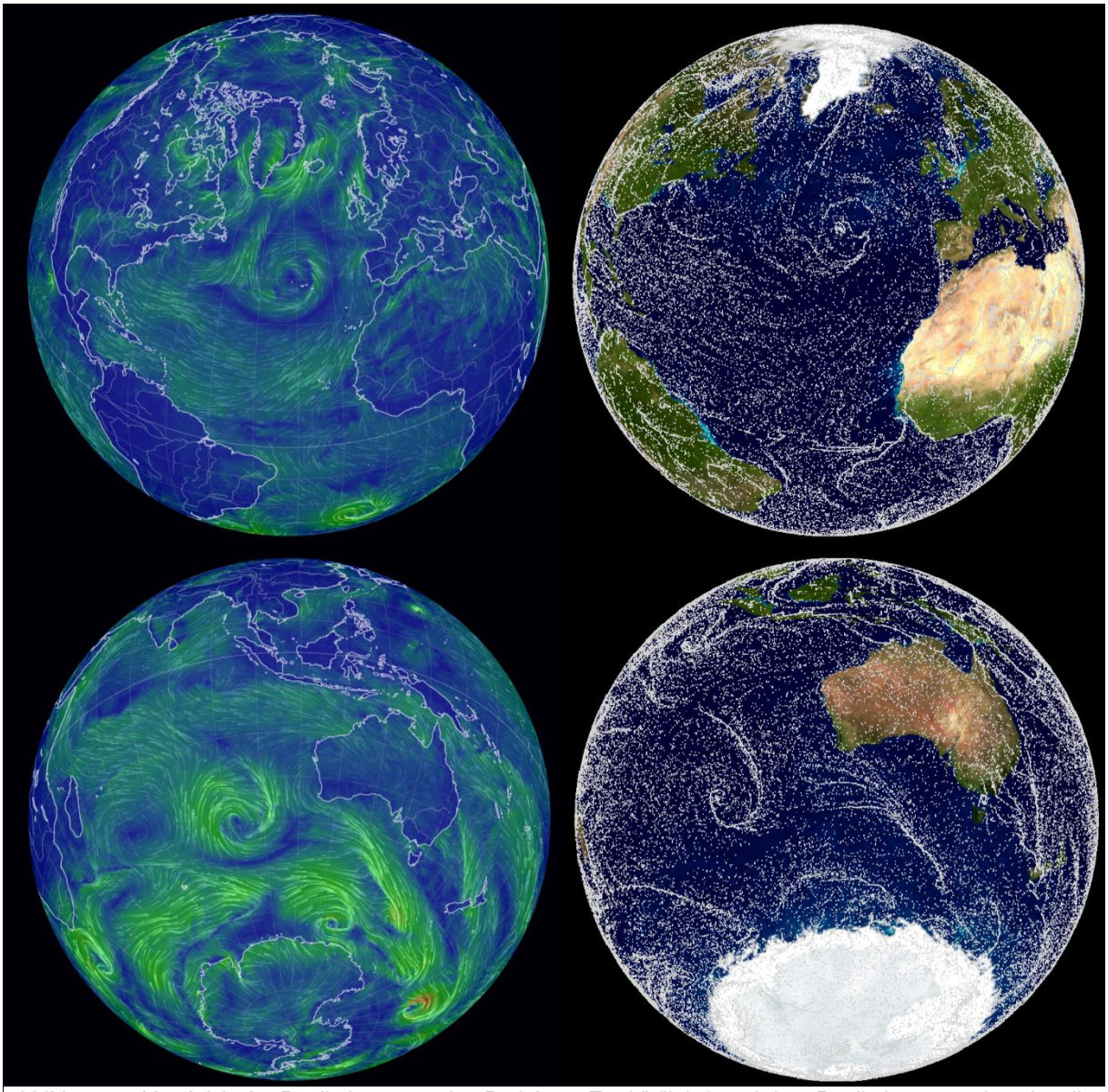

Abbildung 59 Vergleich der Partikelsysteme des Projektes „Earth“ (links) und dem Partikelsystems des virtuellen Hyperglobes (rechts) am 25.06.2021 um 06:00 Uhr. Quelle: links: earth.nullschool.net, rechts: eigene Darstellung

Darstellungen vergleichbare Muster der Partikelbahnen gezeigt werden. Auf den oberen Darstellungen ist vor allem die Verwirbelung über dem Nordatlantik charakteristisch. Deutlich ist auch die tendenzielle Ost-West Strömung zwischen Nordafrika und Südamerika. Auch der untere Vergleich zeigt deutlich Gemeinsamkeiten. Gut zu sehen ist beispielsweise die Verwirbelung südwestlich von Australien. Auch ist keine sichtbare Trennung der westlichen und östlichen Hemisphäre entlang der Datumsgrenze zu sehen. Damit zeigt sich, dass die Herausforderung, welche anfangs des Kapitels beschrieben sind, überwunden werden konnten. Die Partikel bewegen sich zwischen den Hemisphären auf kontinuierlichen Bahnen, welche durch die Umrechnung nicht beeinflusst werden.

Allerdings zeigt sich, dass einige Partikelbahnen zwar tendenziell die richtigen Verläufe abbilden, aber im Detail scheinbar unnatürliche Wege nehmen. Dieses Verhalten tritt überwiegend in Bereichen von starken Konvergenzen von Luftmassen auf. Also den Punkten, an denen Strömungen aus verschiedenen Richtungen aufeinandertreffen. Damit wirft das Partikelsystem des virtuellen Globus das gleiche Problem auf wie schon jenes am taktilen Globus. Auf Abbildung 60 ist das Problem deutlich zu sehen. Die eingezeichneten Pfeile zeigen die Richtungen der Windvektoren. Partikel, welche sich im Bereich der aufeinandertreffenden Strömungen befinden, scheinen sich zwischen den Richtungen abwechselnd zu bewegen bis ihre Lebenszeit aufgebraucht ist. Dieses Verhalten bewirkt eine Ansammlung von Partikeln inmitten der Konvergenzzone.

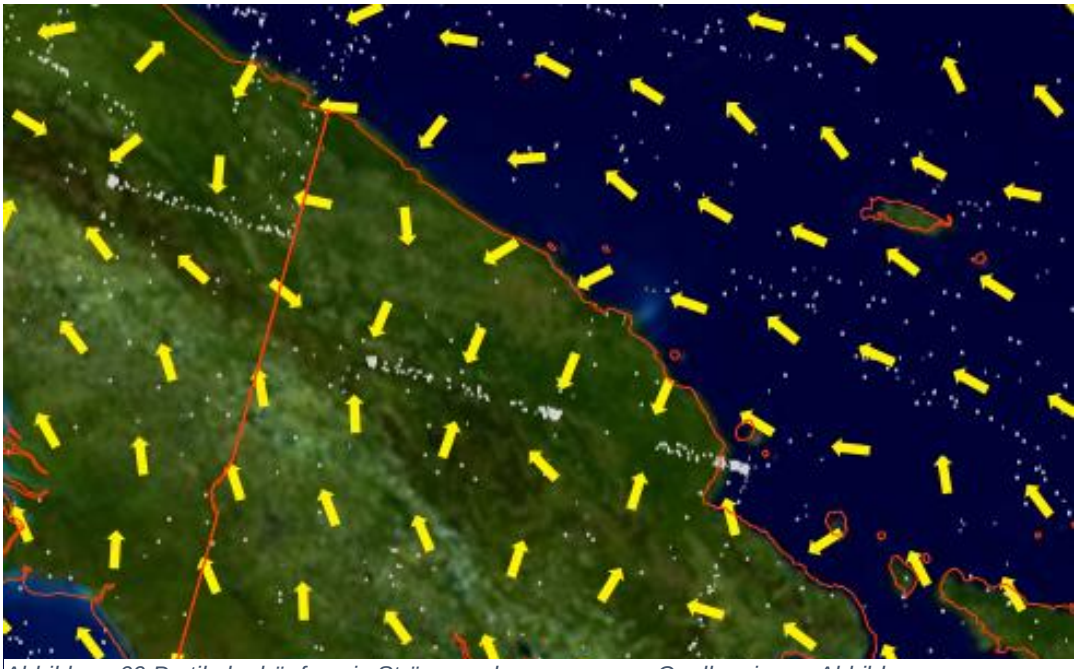

Abbildung 60 Partikelanhäufung in Strömungskonvergenzen. Quelle: eigene Abbildung

Ansätze zum Lösen dieses Problems wären eine weitere Überprüfung der Positionen der Partikel am Ende des Durchlaufes der Programmschleifen. Sollten Partikel ihre Positionen

nicht verändern, werden sie aus dem System gelöscht. Auch plötzliche Richtungsänderungen könnten überwacht und entsprechende Partikel gelöscht werden. Dies würden unnatürlich scharfe Flugbahnänderungen verhindern.

Grundsätzlich zeigt sich jedoch, dass sich das Partikelsystem den Erwartungen entsprechend verhält und ein Vektorfeld mithilfe von Partikeln dynamisch visualisiert. Weltweite Zusammenhänge innerhalb der atmosphärischen Strömung sind demnach mit dem Partikelsystem des virtuellen Globus optisch erfassbar.

7 Zusammenfassung und Diskussion der Ergebnisse

Die vorliegende Arbeit beschäftigte sich mit der Frage wie Partikelsystem zur Visualisierung von globalen Windströmungen auf Globen realisiert werden können. Im Vorfeld legte die Arbeit dazu ausführlich alle theoretischen Grundlagen. So wurden zunächst Globen im Allgemeinen und anschließend Hypergloben im speziellen besprochen. Auch verschiedene Möglichkeiten der Informationsvisualisierungen und deren Bedeutung wurde ausführlich dargelegt. Wesentliche Unterschiede in der Darstellung von statischen und dynamischen sowie zwei- und dreidimensionalen Visualisierungen wurden besprochen und als Ausgangslage für eine praktische Umsetzung dargelegt. Zuletzt wurden Vektorfelder und Partikelsysteme besprochen, welche zusammen zur Visualisierung von Strömungen innerhalb der Atmosphäre genutzt wurden.

Nachdem die Ergebnisse bereits ausführlich beschrieben wurden, werden sie nun diskutiert. Gegebenenfalls werden zudem Verbesserungsvorschläge ausgesprochen. Im Rahmen der Diskussion werden zusätzlich die Forschungsfragen, welche zu Beginn der Arbeit gestellt wurden, beantwortet. Die deskriptive Ausführung zum Ergebnis des taktilen Globus zeigte, dass auch nach Beurteilung der Herausforderungen an den Globus, welche durch Recherche vorangegangener Abschlussarbeiten zusammengetragen wurden, ein Punkt nicht berücksichtigt wurde. Denn nach der Betrachtung der Ergebnisse wurde festgestellt, dass neben der Verzerrungsproblematik sowie der Positionsneuberechnung am Kartenrand des Basismediums, auch eine Abbildung der Partikel, die durch

Übertragung eines Rasters auf den Globuskörper erfolgt, eine weitere Herausforderung darstellt. Denn während sich das Partikel in der Nähe des 180. Längengrades befindet und sein horizontaler Radius kleiner ist als der absolute Abstand zu dem erwähnten Meridian, wird das Partikel nur unvollständig angezeigt. Da der Rest des Partikels außerhalb der Karte gezeichnet wird, ist er auf dem Globus nicht zu sehen und das Partikel erscheint an einer Seite abgeschnitten. Sobald der Mittelpunkt des Partikels den 180. Längengrad überschritten hat, wird nunmehr die andere Seite des Kreises abgeschnitten dargestellt.

Ein möglicher Ansatz um dieses Problem zu lösen, ist die Fortsetzung der Zeichnung Partikel, obwohl sie den Kartenrand bereits überschritten haben. Erst wenn sie sich in einer Entfernung zum Kartenrand befinden, die größer als ihr Radius ist, können sie gelöscht werden. Gleichzeitig muss in diesem Fall ein korrespondierendes Partikel außerhalb des gegenüberliegenden Kartenrandes erstellt werden, das ebenfalls außerhalb des dortigen Kartenrandes gezeichnet werden muss und die Bewegungen des gegenüberliegenden Partikels spiegelt.

Ein weiteres Problem, dass die Visualisierung auf dem taktilen als auch auf dem virtuellen Globus betrifft, ist die Ansammlung von Partikeln in Konvergenzzonen (siehe Kapitel 6.4.4). Dort wo Luftmassen auf unterschiedlichen Richtungen aufeinandertreffen, entstehen demzufolge unnatürlich wirkenden Bewegungen der Partikel. Dabei sei darauf hingewiesen, dass die Partikel, dem Vektorfeld entsprechend, den richtigen Flugbahnen folgen. Dennoch kann in diesem Fall überlegt werden, ob ein softwareseitiger Eingriff an diesen Stellen zu einem natürlicheren Bild der Partikel führen kann. Im ersten Teil der Arbeit in Kapitel 4.2 (*Darstellung von Vektorfeldern*) wurde bereits erwähnt, dass *Streamlines* nach einer Methode von *Turk&Banks* gleichmäßig innerhalb eines Vektorfeldes verteilt werden können. Angelehnt daran, wäre es denkbar auch Partikel so zu verteilen, dass eine extreme Anhäufung wie in der vorliegenden Visualisierung ausgeschlossen ist.

Dennoch zeigt die Arbeit, wie Partikelsysteme auf Globen visualisiert werden können. Die Ergebnisse schließen damit eine Forschungslücke in der Visualisierung von globalen atmosphärischen Darstellungen. Bisher sind derartige Visualisierungen auf Karten gängig, wie die Betrachtung aktueller globaler Darstellung von Winden ergeben hat. Das Ergebnis der Arbeit zeigt, dass es auch möglich ist, globale Windströmungen auf Globen

darzustellen. Um dies zu erreichen mussten während des Arbeitsprozesses Fragen beantwortet werden. Die erste Forschungsfrage lautete:

- Wie müssen Partikelsysteme gestaltet sein, um den Anforderungen zur Visualisierung am Hyperglobus gerecht zu werden?

Diese Frage wurde in den Kapiteln zu den Anforderungen an ein Partikelsystem am jeweiligen Globus beantwortet (Kapitel 6.3.3 & 6.4.1.). Es stellte sich heraus, dass die Verzerrungsproblematik am Globus ein wesentlicher Punkt bei der Erstellung eines solchen Systems war. Da im Falle des virtuellen Globus mit Vektordaten für die Darstellung der Partikel gearbeitet wurde, entfiel der Punkt für diesen Globus. Auf dem taktilen Hyperglobus hingegen, mussten die Kreisformen der Partikel angepasst werden um der Problematik zu entgegnen. Je nach Entfernung des Partikels zum Äquator, wurde dazu der vertikale Durchmesser der Signatur berechnet und entsprechend aktualisiert. Dabei gilt: je größer die Distanz zum Äquator, desto größer die Verzerrung. Die genaue Formel dazu findet sich in Kapitel 6.4.1. Da Partikelbahnen innerhalb der Grenzen der sphärischen Koordinaten berechnet wurden, musste darüber hinaus darauf geachtet werden, dass bei Übertritt der minimalsten oder maximalsten Koordinaten, eine entsprechende Umrechnung stattfand.

Eine weitere Frage, die durch diese Arbeit beantwortet werden sollte war:

- Welchen Anforderungen müssen die Daten entsprechen, um mit Partikelsystemen visualisiert zu werden?

Ein wesentlicher Punkt, um Daten als Partikelsystem darstellen zu können, ist, dass sie Bewegungen beschreiben. Im konkreten Fall dienten Vektoren als Vorlage der Partikelbahnen. Auch Angaben über Winkel und Länge der Bewegung sind verwertbare Datengrundlagen, um eben jene als Partikelsystem zu visualisieren. Da Wetterdaten standartmäßig in einem, außerhalb der Meteorologie, unpopulären Datenformat ausgegeben werden, das sich nur schwer mit der verwendeten Softwareumgebung bearbeiten ließ, wurde zudem festgelegt, dass die Dateien vorher bearbeitet werden müssen. Standardmäßig muss nun ein CSV Format mit festgelegter Struktur als Grundlage zur Visualisierung eingegeben werden. Dieses muss die Vektoren für einen bestimmten Zeitpunkt enthalten. Wichtig ist ebenso, dass die Vektoren im 1° abstand aufgelistet sind. Die genauen Anforderungen finden sich im Kapitel 6.1.3 *Beschreibung der Daten*.

Aufgrund der globalen Darstellung, müssen die Daten zudem weltweit zur Verfügung stehen. Anderenfalls würden sich deutlich sichtbare Lücken in der Visualisierung zeigen. Des Weiteren sei zur Beantwortung der Frage der Blick auf das Kapitel 6.1 (*Ausgangsdaten*) gelenkt. Dort wurde beschrieben, dass Daten, die auf einem Globus abgebildet werden sollen, der Globenwürdigkeit entsprechen müssen. Zusammengefasst müssen die Daten also verzerrungsfrei dargestellt werden, auch im kleinen Maßstab dargestellt Sinn ergeben, global verfügbar und mit anderen globalen Themen kombinierbar sein.

8 Ausblick

Nachdem in der Diskussion bereits einige technische Mängel besprochen wurden, sind im weiteren Verlauf auch visuelle Verbesserungen möglich. Beispielsweise wurde bisher nicht in Betracht gezogen, der Visualisierung *Streamlines* beizufügen. Diese würden die Bahnen der Partikel verdeutlichen. Bisher wurde dies nicht implementiert, da sich die Visualisierung zunächst auf die Partikel konzentrierte. Eine möglicher Ansatz *Streamlines* realisieren zu können ist, indem Partikel nicht nach jedem Frame gelöscht und dann versetzt werden, sondern sie zu versetzten und die alten Positionen beizubehalten. Dies würde den Effekt eines Schweifs erzeugen, der sich hinter den Partikeln bildet.

Nachdem aber die technische Funktionalität sichergestellt ist, können derlei kosmetische Maßnahmen hinzugefügt werden. Eine weitere Verbesserung der Visualisierung wäre die Einführung von Partikelvisualisierungen in verschiedenen atmosphärischen Höhen. So könnten Luftströmungen innerhalb der Atmosphäre verglichen werden, welches einen weiteren didaktischen Wert hätte.

Neben technischen Verbesserungen kann auch darüber diskutiert werden, wie derartige Visualisierungen auf Betrachter wirken. Die Darstellung der atmosphärischen Strömungen auf dem virtuellen sowie auf dem taktilen Globus zeigen eine dynamische Visualisierung. Wie in Kapitel 3.2.4 (*Statische und Dynamische Geovisualisierung*) zu lesen ist, beinhalten dynamische Abbildungen zumeist die zeitliche Komponente. In dem vorliegenden Fall wird jedoch ein statischer Datensatz mit einer dynamischen Visualisierung dargestellt. Der Datensatz beschreibt Vektoren an einem bestimmten Punkt in der Atmosphäre zu einem bestimmten Zeitpunkt. Die Visualisierung suggeriert jedoch Bewegung und damit

temporalen Fortschritt. Der Betrachter könnte demnach nicht ohne Weiteres wahrnehmen, dass die dargestellten Szenen nur Momentaufnahmen des gesamten Strömungsgeschehen der Atmosphäre darstellen. Vielmehr könnte der Eindruck entstehen, dass die Visualisierung ein Geschehen der Atmosphäre innerhalb eines Zeitraums, anstatt eines Zeitpunktes zeigt. Die Visualisierung kann daher zu Fehlinterpretationen führen. Um diesen zuvor zu kommen, muss daher kommuniziert werden, dass es sich um einen bestimmten Zeitpunkt handelt. Zudem könnte der Eindruck entstehen, dass Partikel tatsächlich für Wind stehen, was wiederum zu der Fehleinschätzung führen könnte, dass dort wo sich keine Partikel befinden, kein Windgeschehen vorherrscht. Um einschätzen zu können, wie derartige Visualisierungen wahrgenommen werden, bietet sich im Anschluss an die Umsetzung dieser Darstellung eine Analyse an, wie Betrachter die Animation empfinden. Bisher sind keine solchen Untersuchungen bekannt.

Literatur

Card, S. (2008): Information Visualization. In: The Human Computer Interaction Handbook: Fundamentals, Evolving Technologies and Emerging Applications.Edition: 2ndChapter: 26Publisher: Mahwah, NJ: Lawrence Erlbaum AssociatesEditors: Jacko, Julie and Sears, Andrew

Carvalho, L. (2011). Evaluation of 2D and 3D Map Presentation for Geo-Visualization (Dissertation). Abgerufen von http://urn.kb.se/resolve?urn=urn:nbn:se:hig:diva-9448

Cauvin C., Francsisco E., Aziz S.: Thematic Cartography: 3: New Approaches in Thematic Cartography. 1. Publ. ed. London [u.a.]: Wiley, 2010.

Crespel T., Reuter P., Granier X.: A low-cost multitouch spherical display: hardware and software design. Display Week 2017, May 2017, Los Angeles, California, United States. pp.619-622

Crawfis R., Max N. (1992): Direct volume visualization of three-dimensional vector fields. In Proceedings of the 1992 workshop on Volume visualization (VVS '92). Association for Computing Machinery, New York, NY, USA, 55–60.

Cuntz, Nicolas, Echtzeit-Partikelsysteme und deren Anwendung auf Stromungsvisualisierung. Dissertation, Universität Siegen, 2009

Dodge, M., McDerby, M., and Turner, M. "The Power of Geographical Visualizations." In Geographic Visualization, 1-10. Chichester, UK: John Wiley & Sons, 2008.

Dorffner, L.: Der digitale Behaim-Globus - Visualisierung und Vermessung des historisch wertvollen Originals. In: Cartographica Helvetica. Vol. 14, 1996, S. 20 – 24

Dorsey J., Pederseny H, Hanrahan P. (1996): Flow and changes in appearance. In ACM SIGGRAPH 2006 Courses (SIGGRAPH '06). Association for Computing Machinery, New York, NY, USA

Dübel S., Röhlig M., Schumann H., Trapp M.:2D and 3D presentation of spatial data: A systematic review, In: 2014 IEEE VIS International Workshop on 3DVis (3DVis), 2014, pp. 11-18

Fabrikant, Sara I (2005). Towards an understanding of geovisualization with dynamic displays: issues and prospects. In:. Reasoning with Mental and External Diagrams: Computational Modeling and Spatial Assistance. Papers from the AAAI Spring Symposium.

Ferschin, P.: Visualisierung von Verkehrsströmen mit Hilfe von Partikelsystemen. In: CORP´99: COMPUTERGESTÜTZTE RAUMPLANUNG, 1999, S. 301 – 306

Friendly, M.(2006): A Brief History of Data Visualization. In: Handbook of Computational Statistics: Data Visualization. Springer Handbooks of Computational Statistics. Berlin [u.a.]: Springer

Friendly, M. (2009). Milestones in the History of Thematic Cartography, Statistical Graphics, and Data Visualization.

Gelder A., Wilhelms J. (1992): Interactive visualization of flow fields. In: Proceedings of the 1992 workshop on Volume visualization (VVS '92). Association for Computing Machinery, New York, NY, USA, 47–54.

Hake, G., Grunreich, D., Meng, L.: Kartographie: Visualisierung Raum-zeitlicher Informationen. De Gruyter Lehrbuch. De Gruyter, 1994.

Hastings E. J., Guha R. K., Stanley K. O. (2009): Interactive evolution of particle systems for computer graphics and animation. Trans. Evol. Comp 13, 2 (April 2009), 418–432.

Helman J. L, Hesselink, L.: Representation and display of vector field topology in fluid flow data sets. In Computer, vol. 22, no. 8, pp. 27-36, Aug. 1989, doi: 10.1109/2.35197.

Helman J. L, Hesselink, L.: Visualizing vector field topology in fluid flows,In: IEEE Computer Graphics and Applications, vol. 11, no. 3, pp. 36-46, May 1991

Herman, L., Vojtěch J., Zdeněk S., Daniel V., Jan R., Tomáš Ř: Evaluation of User Performance in Interactive and Static 3D Maps. In: ISPRS International Journal of Geo-Information 7, 2018, no. 11

Hirayama, R., Martinez P., D., Masuda, N., and Subramanian, S.: "A Volumetric Display for Visual, Tactile and Audio Presentation Using Acoustic Trapping." Nature (London) 575, no. 7782 (2019): 320-23.

Hodgeson, M, Gaile, G: Characteric mean and dispersion in surface orientations for a zone. In: International Journal of Geographical Information Systems, 1996, Vol. 10, Issue 7, S. 817-830

Hruby F., Miranda R., Riedl A.: (2009), Bad Globes & Better Globes – multilingual categorization of cartographic concepts exemplified by “map” and “globe” in English, German and Spanish. Proceedings, 24. ICA Cartographic Conference, 2009. Santiago, Chile, 2009

Hruby F. (2013): Globen vs. Karten: Grundlagen Zum Potenzial Sphärischer Displays Für Die Kartographie. Geographica; 8. Hamburg: Kovač

HUBER, T: Spezialeffekte in der Geovisualisierung. 2012, Dipl.-Arbeit Universität Wien

Klink, K. and Willmott, C.J. (1989), Principal components of the surface wind field in the United States: A comparison of analyses based upon wind velocity, direction, and speed. Int. J. Climatol., 9: 293-308.

Kolb A., Latta L., Rezk-Salama C. (2004): Hardware-based simulation and collision detection for large particle systems. In Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware (HWWS '04). Association for Computing Machinery, New York, NY, USA, 123–131.

Kraak, M.: Geovisualization Illustrated. In: ISPRS Journal of Photogrammetry and Remote Sensing, Volume 57, Issues 5–6, 2003, S. 390-399

Kuester F., Bruckschen R., Hamann B., Joy, K. (2001): Visualization of particle traces in virtual environments. In Proceedings of the ACM symposium on Virtual reality software and technology (VRST '01). Association for Computing Machinery, New York, NY, USA, 151–157.

Leggewie, C.: Zur Einleitung: Von der Visualisierung zur Virtualisierung des Erinnerns. In: MEYER, E.: Erinnerungskultur 2.0: kommemorative Kommunikation in digitalen Medien, Frankfurt am Main, 2009

Laidlaw, D.H, Kirby, R.M, Jackson, C.D, Davidson, J.S, Miller, T.S, Da Silva, M, Warren, W.H, and Tarr, M.J. "Comparing 2D Vector Field Visualization Methods: A User Study." IEEE Transactions on Visualization and Computer Graphics 11, no. 1 (2005): 59-70.

Latham J. -P., Munjiza A.: The Modelling of Particle Systems with Real Shapes.In: Philosophical Transactions: Mathematical, Physical and Engineering Sciences 362, no. 1822 (2004): 1953-972. Accessed July 23, 2021.

de Leeuw W., van Liere R.: Comparing LIC and spot noise," Proceedings Visualization '98 (Cat. No.98CB36276), 1998, pp. 359-365

Li X., Hodgson M (2000): Data model and operations for vector fields. Ph.D. Dissertation. University of South Carolina, USA.

Longley, Paul A. Geographical Information Systems and Science. 2.nd ed. Chichester [u.a.]: Wiley, 2005.

MacEachren A.: Visualization in modern cartography: setting the agenda, in MacEachren. A. M., and Taylor, D. R. F., eds., Visualization in Modem Cartography: Pergamon, Oxford, p. I-12, 1994

MacEachren A.: How Maps Work: Representation, Visualization, and Design. New York, NY [u.a.]: Guilford Press, 1995.

MacEachren, A. and Kraak, M.- J., 2001, Research challenges in geovisualization, Cartography and Geographic Information Science, Vol. 28, No. 1, pp.

Max N., Crawfis R, Williams D. (1992): Visualizing wind velocities by advecting cloud textures. In Proceedings of the 3rd conference on Visualization '92 (VIS '92). IEEE Computer Society Press, Washington, DC, USA, 179–184.

Muris, O., Saarmann, G. Der Globus Im Wandel Der Zeiten: Eine Geschichte Der Globen. Berlin [u.a.]: Columbus-Verl., 1961.

Nöllenburg, M. (2006): Geographic Visualization.In: Human-Centered Visualization Environments, 257-94. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg.

Ploetzner R., Lippitsch S., Galmbacher M., Heuer D.,Scherrer S.: Students' difficulties in learning from dynamic visualisations and how they may be overcome. In: Computers in Human Behavior, Volume 25, Issue 1, 2009, S. 56-65

Prandini M., Blom H. A. P., Bakker G. J.: Air traffic complexity and the interacting particle system method: An integrated approach for collision risk estimation, Proceedings of the 2011 American Control Conference, 2011, pp. 2154-2159.

Reeves, W. T..: Particle Systems – A Technique for Modeling a Class of Fuzzy Objects. In: Computer Graphics, Vol: 17, Nr. 3, 1983, S. 359 – 375

Reeves, W. T., Blau R. (1995): Approximate and probabilistic algorithms for shading and rendering structured particle systems. SIGGRAPH Comput. Graph. 19, 3 (Jul. 1985), 313–322.

Riedl A. (2000): Virtuelle Globen in der Geovisualisierung – Untersuchungen zum Einsatz von Multimediatechniken in der Geopräsentation. – Dissertation, Grund- und Integrativwissenschaftliche Fakultät der Universität Wien, 196 S

Riedl A. (2004): Entwicklung und aktueller Stand digitaler Globen. In: Kainz W., Kriz K., Riedl A. (Hrsg.): Aspekte der Kartographie im Wandel der Zeit. Wien, Institut für Geographie der Universität Wien, 2004 (=Wiener Schriften zur Geographie und Kartographie, Band 16) S. 256 - 263.

Riedl A. (2008): Entwicklung und Perspektiven von Taktilen Hypergloben. In: Mitteilungen der Österreichischen Geographischen Gesellschaft, Bd. 150. Wien, 2008, S. 339-356.

Riedl, A. (2009): Taktile Hypergloben – die nächste Stufe in der Globenevolution. In: Kriz K., Kainz W., Riedl A. (Hrsg.): Geokommunikation im Umfeld der Geographie. Wien, Institut für Geographie der Universität Wien, 2009

Riedl, A.: Entwicklungsgeschichte digitaler Globen. In: Der Globusfreund 57/58, Wien Internat. Coronelli-Ges. für Globen u. Instrumentenkunde 2010

Riedl, A. (2011): Der Globus ist tot, es lebe der Globus!.In: Wiender Schriften zur Grographie und Kartographie, Band 20, S. 1 – 9

Scheepens, R. J., Hurter, C., van de Wetering, H. M. M., van Wijk, J. J. (2016). Visualization, selection, and analysis of traffic flows. IEEE Transactions on Visualization and Computer Graphics, 22(1), 379-388.

Schöning, J, Hecht, B., Raubal, M., Krüger, A., Marsh, M, Rohs, M.: Improving Interaction with Virtual Globes through Spatial Thinking: Helping Users Ask "Why?". In: Proceedings of the 13th international conference on intelligent user interfaces, 2008-01-13, p.129-138

Schnotz, W., & Lowe, R. (2008). A unified view of learning from animated and static graphics. In R. Lowe & W. Schnotz (Eds.), Learning with animation: Research implications for design (pp. 304–356). Cambridge University Press

Schumann, H., Müller, W.: Visualisierung: Grundlagen Und Allgemeine Methoden. Berlin [u.a.]: Springer, 2000

Schweikart, J., Pieper, J. & Schulte, B. Virtuelle Globen: Entwicklungsgeschichte und Perspektiven. j. Cartogr. Geogr. inf. 59, 129–136 (2009)

Smith M.J., Hillier J.K., J.-C. Otto, M. Geilhausen: Geovisualization,In: Treatise on Geomorphology, Editor(s): John F. Shroder, S. 299-325, 201

Smutny, T. (2017): Optimierungsvorschläge zur Real Time Visualisierung von Punktsignaturen gezeigt
am Beispiel von Luftfahrzeugdaten. Masterarbeit. Universität Wien.

Stalling, D. (1998): Fast Texture-Based Algorithms for Vector Field Visualization, Dissertation, Fachbereich Mathematik und Informatik der Freien Universitat Berlin

Sumira, S, and Schneider, N.G.. Der Globus: 400 Jahre Geschichte, Macht, Entdeckungen. Darmstadt: Philipp Von Zabern, 2016.

Szeliski R., Tonnesen D. (1992): Surface modeling with oriented particle systems. In Proceedings of the 19th annual conference on Computer graphics and interactive techniques (SIGGRAPH '92). Association for Computing Machinery, New York, NY, USA, 185–194.

Tory M., Kirkpatrick A. E., Atkins M. S., Moller T.: "Visualization task performance with 2D, 3D, and combination displays," in IEEE Transactions on Visualization and Computer Graphics, vol. 12, no. 1, pp. 2-13, Jan.-Feb. 2006, doi: 10.1109/TVCG.2006.17.

Turk G., Banks,D (1996): Image-guided streamline placement. In Proceedings of the 23rd annual conference on Computer graphics and interactive techniques (SIGGRAPH '96). Associat

Unwin, A., Härdle, W. K., Chen, C.: Computational Statistics and Data Visualization (April 19, 2007). SFB 649 Discussion Paper 2007-020

Wahyudi, H. B., Ramdani, F., & Bachtiar, F. A. (2020): 2D and 3D Geovisualization: Learning user Preferences in Landslide Vulnerability. Journal of Information Technology and Computer Science, 5(1), 75–85.

Wang X., Pullar D.: Describing dynamic modeling for landscapes with vector map algebra in GIS. In: Computers & Geosciences, Volume 31, Issue 8, 2005, Pages 956-967

Weinhandl, R.: Die Verebnung der Welt –Kartographie im Mathematikunterricht. 2011, Diplomarbeit, Universität Wien

Yaeger L., Upson C., Myers R. (1986): Combining physical and visual simulation—creation of the planet Jupiter for the film “2010” In Proceedings of the 13th annual conference on Computer graphics and interactive techniques (SIGGRAPH '86). Association for Computing Machinery, New York, NY, USA, 85–93.

Onlinequellen

ArcGis: VectorField – Online unter: https://pro.arcgis.com/de/pro-app/latest/arcpy/spatial-analyst/vector-field.htm (01.08.2021)

ArcGis: Partikelverfolgung – Online unter: https://desktop.arcgis.com/de/arcmap/10.6/tools/spatial-analyst-toolbox/how-particle-track-works.htm (01.08.2021)

Globoccess: Showroom- Online unter: http://globoccess.at/showroom (01.08.2021)

Lookingglassfactory: Produktübersicht – Online unter: https://lookingglassfactory.com/product/overview (01.08.2021)

Pufferfish: Produktübersicht – Online unter https://pufferfishdisplays.com/solutions/displays/ (01.08.2021)

Anhang

1. Partikelsystem für den taktilen Globus

1.1 Index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Global Particle System</title>

    <link rel="stylesheet" href="./css/style.css">

    <script src="lib/jquery.js"></script>
    <script src="lib/papaparse.min.js"></script>
    <script src="lib/chroma.js"></script>
</head>
<body>


    <canvas id="canvas"  class="canvas" ></canvas>

    <div id = "parameter" class = "parameter">
        <div class = "para-side">
            <p>Partikelgröße:</p>
            <div style="display:flex; justify-content: space-around;">
                <input id="particle-
size" type="range" min="1" max="25" value="3">
                <label id="label-size"></label>
            </div>
            <p>Anzahl Partikel</p>
            <div style="display:flex; justify-content: space-around;">
                <input id="particle-
count" type="range" min="10" max="20000" value="5000">
                <label id="label-count"></label>
            </div>
        </div>
        <div class ="para-side" style ="margin-left:15px;">
            <p>Partikelgeschwindigkeit:</p>
            <div style="display:flex; justify-content: space-around;">
                <input id="particle-
speed" type="range" min="1" max="200" value="100">
                <label id="label-speed"></label>
            </div>
            <label>Datei wählen</label>
            <div style="display:flex; justify-content: space-around;">

                <input type="file" name="myfile" id="selectFile"  />
            </div>
        </div>

    </div>
    <script src="js/index.js"></script>
    <script src="js/windmap.js"></script>
    <script src ="js/emitter.js"></script>
    <script src ="js/particle.js"></script>
```

```
    <script src="js/colorGradient.js"></script>
</body>
</html>
```

1.2 Index.js

```
let url = "./earth_surface_2048.jpg"
const canvas = document.querySelector("#canvas");

const ctx = canvas.getContext('2d');

let windTexture;

let windmap;
let minmax;
let width, height, background;
let width2, height2; // width/2

let emitter;
let particles = [];
let ParticleCount;
let ParticleSpeed;
let ParticleSize;


window.onload = async function () {

    width = document.body.clientWidth;
    width2 = width/2;
    height = window.innerHeight;
    height2 = height/2;

    canvas.width  = width;
    canvas.height = height;

    background = new Image();
    background.src = url;

    background.onload = function(){
        ctx.drawImage(background ,0,0,width,height);
    }

    if (ctx === null) {
        alert("Unable to initialize Webctx. Your browser or machine may not support it.")
    }else{
        init();
    }
}


function init() {

    emitter = new Emitter;
    //Partikel initialisieren
    ParticleCount = document.getElementById('particle-count').value;

```

```
50.     //label im Optionsfenster
51.     document.getElementById("label-count").textContent = ParticleCount;
52.     document.getElementById("label-
   size").textContent = slider.value + "px";
53.     ParticleSize = slider.value;
54.     document.getElementById("label-
   speed").textContent = slider3.value + "%";
55.     ParticleSpeed = document.getElementById("label-
   speed").textContent = slider3.value;
56.
57.     for (var i = 0; i < ParticleCount; i++){
58.
59.         emitter.emit()
60.
61.     }
62.
63.  window.requestAnimationFrame(animate)
64.}
65.
66.function animate(){
67.     ctx.canvas.width  = window.innerWidth;
68.     ctx.canvas.height = window.innerHeight;
69.     //Inhalte im Canvas vor jedem Draw löschen
70.     ctx.clearRect(0, 0, canvas.width, canvas.height);
71.     ctx.drawImage(background,0,0,width,height);
72.     if(windmap){
73.
74.
75.
76.         //für jedes Partikel neue Position zeichnen
77.
78.         particles.forEach(function(particle, index, object){
79.             //lifespan verkürzen
80.             particle.lifecircle();
81.
82.             // Lebenszeit des Partikels aufgebraucht?
83.             if (particle.life <= 0){
84.                 emitter.deleteParticle(index)
85.                 emitter.emit();
86.             }
87.             particle.getNewPosition();
88.             particle.draw();
89.         })
90.
91.         if (particles.length < ParticleCount){
92.             emitter.increaseParticles();
93.         }
94.         if (particles.length > ParticleCount){
95.             emitter.decreaseParticles();
96.         }
97.     }
98.
99.     window.requestAnimationFrame(animate);
100.  }
101.
102.  var slider = document.getElementById('particle-size');
103.  slider.addEventListener('mouseup', function () {
```

```
104.        particles.forEach(function(p){
105.            p.r = slider.value;
106.        });
107.        ParticleSize = slider.value;
108.        document.getElementById("label-
    size").textContent = slider.value + "px";
109.    });
110.
111.    var slider2 = document.getElementById('particle-count');
112.    slider2.addEventListener('mouseup', function () {
113.        ParticleCount = slider2.value
114.        document.getElementById("label-count").textContent = ParticleCount;
115.    });
116.
117.    var slider3 = document.getElementById('particle-speed');
118.    slider3.addEventListener('mouseup', function () {
119.        ParticleSpeed = slider3.value
120.        document.getElementById("label-
    speed").textContent = slider3.value + "%";
121.    });
122.
123.    // einstellungen anzeigen
124.    window.onkeypress = function(event) {
125.        let parameterClass = document.getElementById("parameter").className;
126.
127.        if (event.charCode == 32) {
128.            if (parameterClass == "parameter"){
129.                document.getElementById("parameter").className = "none";
130.            } else {
131.                document.getElementById("parameter").className = "parameter"
    ;
132.            }
133.        }
134.     }
135.
136.     //Animation starten nach File-Select
137.    $(document).ready(function(){
138.        $("#selectFile").change(handleFileSelect);
139.    });
140.
141.    async function handleFileSelect(evt) {
142.
143.        var file = evt.target.files[0];
144.        windmap = makeWindmap(file);
145.    }
```

1.3 Particle.js

```
let Vector = function(u,v) {
    this.x = u;
    this.y = v;
}

class Particle  {
    constructor(x,y){

        this.position = new Vector(x,y); // Position des Partikels auf dem Cavas
        this.r = document.getElementById("particle-size").value;
        this.life = Math.floor(Math.random() * (100 - 10 + 10) + 100);
        this.speed = 0; //magnitude
        this.posX = 0; //absolute Position: lon - 180° = posX 0 / lon 180° = posX 360
        this.posY = 0;
    }

    lifecircle(){
        this.life --;
    }

    getNewPosition(){

        // aktuelle Position ermitteln
        this.posX = Math.round(convertRange( this.position.x, [ 0, width ], [ 0, 359 ] ));
        this.posY = Math.round(convertRange( this.position.y, [ 0, height ], [ 0, 179 ] ));

        // Index in eindimensionaler Matrix ermitteln
        var index = this.posY * 360 + this.posX;

        //console.log(index);
        //console.log(windmap);
        this.speed = windmap[index][2];
        let normalized = {"u":windmap[index][3],"v":windmap[index][4]}

        this.position.x += normalized.u * (this.speed * 0.25) * (ParticleSpeed/100);
        this.position.y -= normalized.v * (this.speed * 0.25) * (ParticleSpeed/100);


        //Kanten prüfen
        let xy = this.edges(this.position.x, this.position.y)
        this.position.x = xy[0];
        this.position.y = xy[1];


    }

```

```
48.     draw(){
49.
50.         //Breitengrad ermitteln
51.         let latitude;
52.         if ( this.position.y > height2  ){
53.             latitude = convertRange( this.position.y, [ 0, height2 ], [ 90,
    0 ] )}
54.         else {
55.             latitude = convertRange( this.position.y, [  height2, height ],
    [0,-90 ] )
56.         }
57.
58.         // Verzerrung berechnen
59.         let verzerrung = 1 / Math.cos(latitude * (Math.PI / 180));
60.
61.         if (verzerrung < 0){console.log(verzerrung)}
62.
63.         // Partikel zeichnen
64.         ctx.beginPath();
65.         if (ParticleSize > 1){
66.             ctx.ellipse(this.position.x, this.position.y, this.r*verzerrung
   ,this.r,  0, 0, 2 * Math.PI);
67.         }
68.         else{
69.             ctx.rect(this.position.x, this.position.y,1,1)
70.         }
71.
72.         //Farbwert auf Gradient ermitteln. langsam = gelb; schnell=rot
73.         ctx.fillStyle = getColorRGBA(this.speed);
74.         ctx.fill();
75.
76.     }
77.
78.     edges(x,y){
79.
80.         if (x > width) {    x = 0 + (x - width)}
81.         if (x < 0)     {    x = x + width}
82.
83.
84.         if (y < 0 || y > height) {
85.
86.             y < 0 ? y = 1 : y = height;
87.
88.             if ( x > width2){
89.                 let diff = Math.abs(width2 - x)
90.                 x = width2 - diff
91.                 //return [x,y]
92.             }
93.             else if (x < width2){
94.                 x = width - x
95.                 //return [x,y]
96.             }
97.         }
98.
99.         return [x,y]
100.       }
101.   }
```

1.4 Emitter.js

```
1. class Emitter{
2.     // Erstellt und löscht Partikel
3.     emit(){
4.         // Partikel mit zufälliger Startposition erstellen
5.
6.         let n = Math.floor(Math.random() * width) + 1 ;
7.         let m = Math.floor(Math.random() * height) + 1 ;
8.         let particle = new Particle(n,m,2); //posX, posY, radius
9.         particles.push(particle);
10.    }
11.
12.    deleteParticle(index){
13.         //Partikel löschen
14.         particles.splice(index,1);
15.    }
16.
17.    increaseParticles(){
18.        //Partikel erstellen bis ParticelCount erreicht ist
19.        for (var i = 0; i <= ParticleCount - particles.length; i++){
20.            let n = Math.floor(Math.random() * width) + 1 ;
21.            let m = Math.floor(Math.random() * height) + 1 ;
22.
23.            let particle = new Particle(n,m,2); //posX, posY, radius
24.            particles.push(particle);
25.        }
26.    }
27.
28.    decreaseParticles(){
29.        //Partikel löschen bis PartikelCount erreicht ist
30.        while( particles.length > ParticleCount) {
31.            var index = Math.floor( Math.random()*particles.length );
32.            particles.splice( index, 1 ); // Remove the item from the array
33.        }
34.    }
35.}
```

1.5 windMap.js

```
1. function makeWindmap(csvFile){
2.     Papa.parse(csvFile, {
3.         dynamicTyping: true,
4.         dataType:"text",
5.         complete: function(results) {
6.             let data = results.data
7.             data.shift(); //Kopfzeile löschen
8.             windmap = data;
9.
10.            windmap = addMagnitude(windmap);
11.            windmap = normVector(windmap);
12.            minmax = getMinMax(windmap);
13.        }
14.    })
15.}
16.
17.function addMagnitude(list){
18.    let map = []
19.    list.forEach(a =>{
20.        a.push(Math.sqrt(a[0]*a[0] + a[1]*a[1]))
21.        map.push(a);
22.    });
23.    return map
24.}
25.
26.function normVector(list){
27.    list.forEach(field =>{
28.        field[0] === 0 ? field.push(0) : field.push(field[0] / field[2]); /
   / u Komponente normalisieren
29.        field[1] === 0 ? field.push(0) : field.push(field[1] / field[2]); /
   / v Komponente normalisieren
30.    })
31.    return list
32.}
33.
34.function getMinMax(field){
35.    let minU = 1000;
36.    let maxU = -1000;
37.
38.    let minV = 1000;
39.    let maxV = -1000;
40.
41.    let minS = 1000;
42.    let maxS = -1000;
43.
44.    field.forEach((f) => {
45.        f[0] < minU ? minU = f[0] : null;
46.        f[0] > maxU ? maxU = f[0] : null;
47.
48.        f[1] < minV ? minV = f[1] : null;
49.        f[1] > maxV ? maxV = f[1] : null;
50.
51.        f[2] < minS ? minS = f[2] : null;
52.        f[2] > maxS ? maxS = f[2] : null;
53.    });
```

```
54.
55.     return {minU,maxU,minV,maxV,minS,maxS}
56.
57.}
```

1.5 colorGradient.js

```
1. const gradient = chroma.scale(['yellow','red']);
2.
3. function getColorRGBA(speed){
4.     //ermittelt Farbe für Hintergundbild
5.     let scale = convertRange(speed, [minmax.minS, minmax.maxS], [0,1])
6.     return gradient(scale);
7. }
8.
9. function convertRange( value, r1, r2 ) {
10.     // Quelle: https://stackoverflow.com/questions/14224535/scaling-
   between-two-number-ranges
11.     return ( value - r1[ 0 ] ) * ( r2[ 1 ] - r2[ 0 ] ) / ( r1[ 1 ] - r1[ 0
   ] ) + r2[ 0 ];
12.}
```

2. Partikelsystem für den virtuellen Hyperglobus

2.1 Index.html

```
1   <!DOCTYPE html>
2   <html lang="en">
3   <head>
4       <meta charset="UTF-8">
5       <meta http-equiv="X-UA-Compatible" content="IE=edge">
6       <meta name="viewport" content="width=device-width, initial-scale=1.0">
7
8       <title>Virtual Globe Particle System</title>
9
10      <link rel="stylesheet" href = "style/main.css" />
11
12      <script
13      src="https://code.jquery.com/jquery-3.6.0.js"
14      integrity="sha256-H+K7U5CnXl1h5ywQfKtSj8PCmoN9aaq30gDh27Xc0jk="
15      crossorigin="anonymous"></script>
16      <script src="lib/PapaParse-5.0.2/papaparse.min.js"></script>
17      <script src="https://cdnjs.cloudflare.com/ajax/libs/chroma-
    js/2.1.0/chroma.min.js" integrity="sha512-
    yocoLferfPbcwpCMr8v/B0AB4SWpJlouBwgE0D3ZHaiP1nuu5djZclFEIj9znuqghaZ3tdCMRr
    reLoM8km+jIQ==" crossorigin="anonymous"></script>
```

```
18
19
20      <script src="lib/WebGL.js"></script>
21      <script src="lib/Threejs/three.js-
    master/build/three.min.js"></script>
22      <script src="lib/stats.min.js"></script>
23      <script src="lib/TrackballControls.js"></script>
24      <script type='text/javascript' src="lib/gui/dat.gui.min.js"></script>
25
26      <script src = "./js/utils.js"></script>
27      <script src="./js/particle.js"></script>
28      <script src="./js/emitter.js"></script>
29      <script src="./js/index.js"></script>
30
31
32
33  </head>
34  <body>
35          <input type="file"  style ="display:none" name="myfile" id="select
    File"/>
36
37
38  </body>
39  </html>
```

2.2 Index.js

```
1   const radius = 1;
2
3   let scene, camera, renderer, controls;
4   let windmap, minmax;
5   let points,particleGeometry;
6   let particleCount
7   let particleSpeed = 1;
8
9   let bildURL = "https://upload.wikimedia.org/wikipedia/commons/d/d6/Nasa_land_o
    cean_ice_8192.jpg";
10  window.onload = async function (){
11      if( WEBGL.isWebGLAvailable() == false ){
12          document.body.appendChild( WEBGL.getWebGLErrorMessage() );
13      }else{
14          init();
15      }
16  }
17
18  function init(){
19      let emitter;
20      let initParticleCount = 10000;
21      let initParticleSize = 1.5;
22      let initParticleSpeed = 100;
23
24      //Szene, Kamera, Licht und Renderer
25      scene = new THREE.Scene();
26
27      camera = new THREE.PerspectiveCamera( 45, window.innerWidth/window.innerHe
    ight, 0.1, 1000 );
```

```
28      camera.position.z = 3;
29
30      var mDirectionallight = new THREE.AmbientLight( 0xFFFFFF );
31      mDirectionallight.position.y = 5;
32      mDirectionallight.position.z = 13;
33      scene.add( mDirectionallight);
34
35      renderer = new THREE.WebGLRenderer();
36      renderer.setSize( window.innerWidth, window.innerHeight );
37      renderer.setPixelRatio( window.devicePixelRatio );
38      document.body.appendChild( renderer.domElement );
39
40
41
42      // Globus
43      var texture = new THREE.TextureLoader().load( bildURL );
44      var globeMaterial = new THREE.MeshBasicMaterial({
45          //color:0x12f51
46          //map: THREE.ImageUtils.loadTexture('./data/earth_surface_2048.jpg')
47          map:texture
48      });
49      var globeGeometry = new THREE.SphereGeometry( radius, 32, 32 );
50      globe = new THREE.Mesh( globeGeometry, globeMaterial );
51      scene.add( globe );
52
53
54      //emitter initialisieren
55      emitter = new Emitter();
56
57      //Partikelanzahl von GUI Element entnehmen
58      particleCount = initParticleCount;
59
60      //Partikel
61      particleGeometry = new THREE.BufferGeometry();
62      const pointsMaterial = new THREE.PointsMaterial( { sizeAttenuation:false,
    size: initParticleSize, color:"lightgrey"} );
63      points = new THREE.Points( particleGeometry, pointsMaterial );
64      scene.add( points );
65
66      // Partikel initialisieren
67      emitter.initialiseParticles(particleCount);
68
69      //GUI
70      var gui = new dat.GUI({
71          height : 100
72      });
73
74      var params = {
75          Partikel: initParticleCount,
76          loadFile : function() {
77              document.getElementById('selectFile').click();
78              },
79          size:initParticleSize,
80          speed:initParticleSpeed
81      };
82
83      const anzahlFolder = gui.addFolder("Partikelanzahl");
84      anzahlFolder.add(params, 'Partikel').min(100).max(100000).step(1).onFinish
    Change(function(){
```

```
85          let newCount = gui.__folders.Partikelanzahl.__controllers[0].object.Partikel;
86
87          if(newCount < particleCount){
88              emitter.decreaseParticles(newCount);
89          }else{
90              emitter.increaseParticles(newCount);
91          }
92      })
93      anzahlFolder.open()
94
95      const particleSize = gui.addFolder("Partikelgröße");
96      particleSize.add(params,'size').min(0.5).max(3).step(0.1).onFinishChange(function(){
97          let newSize = gui.__folders.Partikelgröße.__controllers[0].object.size;
98
99          points.material.size = newSize;
100     });
101     particleSize.open();
102
103     const speedFolder = gui.addFolder("Geschwindigkeit");
104     speedFolder.add(params,'speed').name('%').min(10).max(200).step(10).onFinishChange(function(){
105         let newSpeed = gui.__folders.Geschwindigkeit.__controllers[0].object.speed;
106         particleSpeed = newSpeed / 100;
107         console.log(particleSpeed);
108     });
109     speedFolder.open();
110
111     const uploadFolder = gui.addFolder("Upload File");
112     uploadFolder.add(params, 'loadFile').name('Vektorfeld laden');
113     uploadFolder.open();
114
115
116
117     //controls = new THREE.TrackballControls( camera );
118     controls = new THREE.TrackballControls(camera, renderer.domElement);
119     controls.rotateSpeed = 1.0;
120     controls.zoomSpeed = 1.2;
121     controls.panSpeed = 0.8;
122     controls.noZoom = false;
123     controls.noPan = false;
124     controls.dynamicDampingFactor = 0.3;
125     controls.keys = [ 65 /*A*/, 83 /*S*/, 68 /*D*/ ];
126
127     window.addEventListener( 'resize', onWindowResized, false );
128
129     animate();
130 }
131
132 function animate(){
133     points.geometry.attributes.position.needsUpdate = true;
134     points.geometry.attributes.options.needsUpdate = true;
135     if (windmap){
136         moveParticle();
137     }
138     controls.update();
139     renderer.render(scene, camera);
```

```
140     window.requestAnimationFrame(animate);
141 }
142
143 function onWindowResized() {
144     renderer.setSize( window.innerWidth, window.innerHeight );
145     camera.aspect = window.innerWidth / window.innerHeight;
146     camera.updateProjectionMatrix();
147
148     controls.handleResize();
149 }
150

151 $(document).ready(function(){
152     $("#selectFile").change(handleFileSelect);
153 });
154
155 async function handleFileSelect(evt) {
156
157     var file = evt.target.files[0];
158     Papa.parse(file, {
159         dynamicTyping: true,
160         dataType:"text",
161         complete: function(results) {
162             let data = results.data
163             data.shift(); //Kopfzeile löschen
164             windmap = data;
165
166             windmap = addMagnitude(windmap);
167             windmap = normVector(windmap);
168             minmax = getMinMax(windmap);
169
170             console.log(windmap)
171         }
172     });
173
174 }
175
176 function addMagnitude(list){
177     let map = []
178     list.forEach(a =>{
179         a.push(Math.sqrt(a[0]*a[0] + a[1]*a[1]))
180         map.push(a);
181     });
182     return map
183 }
184
185 function normVector(list){
186     list.forEach(field =>{
187         field[0] === 0 ? field.push(0) : field.push(field[0] / field[2]); // u
     Komponente normalisieren
188         field[1] === 0 ? field.push(0) : field.push(field[1] / field[2]); // v
     Komponente normalisieren
189     })
190     return list
191 }
192
193 function getMinMax(field){
194     let minU = 1000;
195     let maxU = -1000;
196
```

```
197     let minV = 1000;
198     let maxV = -1000;
199 
200     let minS = 1000;
201     let maxS = -1000;
202 
203     field.forEach((f) => {
204         f[0] < minU ? minU = f[0] : null;
205         f[0] > maxU ? maxU = f[0] : null;
206 
207         f[1] < minV ? minV = f[1] : null;
208         f[1] > maxV ? maxV = f[1] : null;
209 
210         f[2] < minS ? minS = f[2] : null;
211         f[2] > maxS ? maxS = f[2] : null;
212     });
213 
214     return {minU,maxU,minV,maxV,minS,maxS}
215 
216 }
217 
```

2.3 Particle.js

```
1. function moveParticle(){
2. 
3.     // neue Array für Positionen und Lifetime
4.     let newPositions = [];
5.     let newOptions = [];
6. 
7.     for(var index = 0; index <= points.geometry.attributes.position.array.l
   ength-1; index += 3){
8. 
9.         try{
10. 
11.             // wenn aktuelle Lebenszeit auf 1 ist dann Partikel mit neuer Z
   eit auf
12.             // zufälligen Punkt setzen
13.             if(points.geometry.attributes.options.array[index] - 1 < 0){
14. 
15.                 //Partikel an neue Position setzen + options zurücksetzen
16.                 //zufällige Position auf Globus
17.                 let pos = randomSpherePoint(0,0,0,radius)
18.                 newPositions.push(pos[0]);
19.                 newPositions.push(pos[1]);
20.                 newPositions.push(pos[2]);
21. 
22.                 //zufällige Lebenszeit zwischen 100 und 200 Frames
23.                 newOptions.push(THREE.Math.randInt(100,200));
```

```
24.
25.                 //sphärische Koordinaten ermitteln und zuweisen
26.                 let initLatLon = XYZ_to_LL(pos[0],pos[1],pos[2])
27.                 newOptions.push(initLatLon.lat);
28.                 newOptions.push(initLatLon.lon);
29.
30.
31.             }else{
32.
33.                 //lebenszeit senken
34.                 let life = points.geometry.attributes.options.array[index]
    - 1;
35.                 newOptions.push(life);
36.
37.                 //sphärische Koordinaten
38.                 let particleLat = points.geometry.attributes.options.array[
    index+1];
39.                 let particleLon = points.geometry.attributes.options.array[
    index+2];
40.
41.                 let absll = ll_to_absll(Math.floor(particleLat),Math.floor(
    particleLon));  // von -90/90 -180/180 zu 0-179 0-359 für Index
42.                 if (absll.lat > 180 || absll.lon > 360){
43.                     console.log(absll)
44.                     console.log(particleLat, particleLon)
45.                 }
46.                 let fieldIndex = absll.lat * 360 + absll.lon;
47.                 let u = windmap[fieldIndex][3];
48.                 let v = windmap[fieldIndex][4];
49.
50.
51.                 // Vektor mit aktueller Position addieren / mit Faktor der
    Geschwindigkeit
52.                 particleLat += (u*0.05) * particleSpeed;
53.                 particleLon += (v*0.05) * particleSpeed;
54.
55.                 // Kanten beachten
56.
57.                 if (particleLat > 90){
58.                     particleLat = 90
59.                     if (particleLon >= 0){ particleLon = -
    180 + particleLon}
60.                     else {particleLon = 180 - particleLon}
61.                 }
62.                 if (particleLat < -90){
63.                     particleLat = -90
64.                     if (particleLon >= 0){ particleLon = -
    180 + particleLon}
65.                     else {particleLon = 180 - particleLon}
66.                 }
67.
68.                 if (particleLon >= 180){particleLon = -180}
69.                 if (particleLon < -180){particleLon = 180}
70.
71.                 //sphärische Koordinaten in Array speichern
72.                 newOptions.push(particleLat);
73.                 newOptions.push(particleLon);
```

```
74.
75.                 //kartesische Koordinaten ermitteln und abspeichern
76.                 let newXYZ = LL_to_XYZ(particleLat, particleLon)

77.                 newPositions.push(newXYZ.x);
78.                 newPositions.push(newXYZ.y);
79.                 newPositions.push(newXYZ.z);
80.             }
81.
82.         }catch(err){
83.             console.log(err);
84.         }
85.     }
86.
87.     // Arrays der Eigenschaften auf entsprechende Eigenschaft zuweisen
88.     points.geometry.setAttribute( 'position', new THREE.Float32BufferAttrib
    ute( newPositions, 3 ) );
89.     points.geometry.setAttribute( 'options',  new THREE.Float32BufferAttrib
    ute(newOptions, 3 ) );
90.}
91.
92.
```

2.4 Emitter.js

```
1. class Emitter{
2.
3.     initialiseParticles(particleCount){
4.         // Array des Optionen
5.         let positions = [];
6.         let options = [];
7.
8.         for (var i = 0; i <= particleCount * 3; i++){
9.             let XYZpos = randomSpherePoint(0,0,0,radius)
10.            positions.push(XYZpos[0])
11.            positions.push(XYZpos[1])
12.            positions.push(XYZpos[2])
13.
14.            options.push(THREE.Math.randInt(100,200))
15.            let initLatLon = XYZ_to_LL(XYZpos[0],XYZpos[1],XYZpos[2])
16.            options.push(initLatLon.lat);
17.            options.push(initLatLon.lon);
18.
19.        }
20.
21.        particleGeometry.setAttribute( 'position', new THREE.Float32Buff
   erAttribute( positions, 3 ) );
22.        particleGeometry.setAttribute( 'options', new THREE.Float32Buffe
   rAttribute( options, 3 ) );
23.    }
24.
25.    decreaseParticles(max){
26.
27.        let newPositions = [];
28.        let newLifetime = [];
29.
30.        for(var i = 0; i <= points.geometry.attributes.position.array.le
   ngth-1; i++){
31.            newPositions.push(points.geometry.attributes.position.array[
   i]);
32.            //newColors.push(points.geometry.attributes.color.array[i])
33.            newLifetime.push(points.geometry.attributes.options.array[i]
   );
34.        }
35.
36.        newPositions.length = max*3;
37.        newLifetime.length = max*3;
38.
39.
40.        points.geometry.setAttribute( 'position', new THREE.Float32Buffe
   rAttribute( newPositions, 3 ) );
41.        points.geometry.setAttribute( 'options',  new THREE.Float32Buffe
   rAttribute(newLifetime, 3 ) );
```

```
42.
43.         particleCount = max;
44.     }
45.
46.     increaseParticles(max){
47.
48.         let newPositions = [];
49.         let newLifetime = [];
50.
51.         for(var i = 0; i <= points.geometry.attributes.position.array.le
    ngth-1; i++){
52.             newPositions.push(points.geometry.attributes.position.array[
    i]);
53.             newLifetime.push(points.geometry.attributes.options.array[i]
    );
54.         }
55.
56.         for (var i = 0; i < (max-particleCount)*3; i++){
57.
58.             let pos = randomSpherePoint(0,0,0,radius)
59.             let x = pos[0];
60.             let y = pos[1];
61.             let z = pos[2];
62.
63.             newPositions.push(x);
64.             newPositions.push(y);
65.             newPositions.push(z);
66.
67.             newLifetime.push(THREE.Math.randInt(100,200));
68.             let initLatLon = XYZ_to_LL(pos[0],pos[1],pos[2])
69.             newLifetime.push(initLatLon.lat);
70.             newLifetime.push(initLatLon.lon);
71.
72.         }
73.
74.         points.geometry.setAttribute( 'position', new THREE.Float32Buffe
    rAttribute( newPositions, 3 ) );
75.         points.geometry.setAttribute( 'options',  new THREE.Float32Buffe
    rAttribute(newLifetime, 3 ) );
76.         //points.geometry.setAttribute( 'color',  new THREE.Float32Buffe
    rAttribute(newColors, 3 ) );
77.
78.         particleCount = max;
79.     }
80. }
```

2.5 Utils.js

```
1. function randomSpherePoint(x0,y0,z0,radius){
2.     var u = Math.random();
3.     var v = Math.random();
4.     var theta = 2 * Math.PI * u;
5.     var phi = Math.acos(2 * v - 1);
6.     var x = x0 + (radius * Math.sin(phi) * Math.cos(theta));
7.     var y = y0 + (radius * Math.sin(phi) * Math.sin(theta));
8.     var z = z0 + (radius * Math.cos(phi));
9.     return [x,y,z];
10. }
11.
12.function ll_to_absll(lat,lon){
13.     // transformiert normale lat lon (90/-90 180/-
   180 zu 0 bis 180 & 0 bis 360)
14.     if (lat >= 0){
15.         lat = 90 - lat;
16.     }
17.     else{
18.         lat = 90 + Math.abs(lat);
19.     }
20.     lon = lon + 180;
21.
22.     return {lat,lon}
23.}
24.
25.function LL_to_XYZ(lat,lon){
26.
27.     let theta = THREE.Math.degToRad(lat -90);
28.     let phi = THREE.Math.degToRad(lon - 90);
29.
30.     let x = Math.sin(theta) * Math.sin(phi);     // X in Threejs |  Y auf Ab
   bildung
31.     let y = Math.cos(theta);                     // Y in Threejs |  Z auf Ab
   bildung
32.     let z = Math.sin(theta) * Math.cos(phi);     // Z in ThreeJS |  X auf Ab
   bildung
33.
34.     return {x,y,z}
35.}
36.
37.function XYZ_to_LL(x,y,z){
38.     //lat
39.     let thetaRad = Math.acos(y / Math.sqrt(x*x+y*y+z*z))
40.     let thetaDeg = THREE.Math.radToDeg(thetaRad);
41.     let lat = 90 - thetaDeg;
42.     //lon
43.     let phiRad = Math.atan2(x,z);
44.     let lon = THREE.Math.radToDeg(phiRad);
45.     lon = lon - 90; //verschiebung 0 Medidian um 90° bei ThreeJS
46.     lon < -
   180 ? lon += 360: null; //  Werte über 90° müssen angepasst werden
47.
48.     return {lat,"lon":lon}
49.}
```

EIGENSTÄNDIGKEITSERKLÄRUNG:

Ich versichere:

- dass ich die Masterarbeit selbstständig verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt und mich auch sonst keiner unerlaubten Hilfe bedient habe.
- dass alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten und nicht veröffentlichten Publikationen entnommen sind, als solche kenntlich gemacht sind.
- dass ich dieses Masterarbeitsthema bisher weder im In- noch im Ausland (einer Beurteilerin/ einem Beurteiler zur Begutachtung) in irgendeiner Form als Prüfungsarbeit vorgelegt habe.
- dass diese Arbeit mit der vom Begutachter beurteilten Arbeit übereinstimmt.

Datum Unterschrift