

Christian-Albrechts-Universität zu Kiel

---

## **Masterprojekt**

# **Virtual-Explorer**

## **Ein virtueller Rundgang**

### **Dokumentation**

Betreuer: Prof. Dr. Thomas Slawig  
Institut für Informatik  
Technische Fakultät  
Christian-Albrechts-Universität zu Kiel

**Linn Schwartkop und Marius Rasch**

---

18. April 2016

# Inhaltsverzeichnis

<b>1 Einführung</b>	<b>4</b>
1.1 Szenario . . . . .	4
1.2 Aufbau . . . . .	5
<b>2 Server</b>	<b>6</b>
2.1 Express . . . . .	6
2.2 Ordnerstruktur . . . . .	6
2.3 Konfiguration von Karten und Sphären . . . . .	7
2.3.1 Konfiguration einer Karte . . . . .	7
2.3.2 Konfiguration einer Sphäre . . . . .	9
2.3.3 Konfiguration eines Pfeils . . . . .	11
2.3.4 Konfiguration eines Info-Markers . . . . .	12
2.3.5 Konfiguration des onClick-Events für Pfeile und Info-Marker	14
2.3.6 Erstellen von Informationen für die Info-Marker . . . . .	15
2.4 Grunt . . . . .	16
2.4.1 Aufgaben . . . . .	16
2.4.2 Gruppierung von Aufgaben . . . . .	17
2.4.3 Verwendete Plugins und npm-Pakete . . . . .	18
2.4.4 Zusätzliche Software . . . . .	19
2.4.5 Globale Konfiguration . . . . .	19
<b>3 Browseranwendung</b>	<b>20</b>
3.1 Browserkompatibilität . . . . .	21
3.2 Verwendete JavaScript-Bibliotheken . . . . .	21
3.3 Kartenansicht . . . . .	22
3.3.1 Konfiguration einer Karte . . . . .	23
3.4 Sphärenansicht . . . . .	25
3.4.1 Konfiguration einer Sphäre . . . . .	27

3.4.2	Koordinaten in Three.js . . . . .	29
3.4.3	Anzeigen der Koordinaten in einer Sphäre . . . . .	31
3.5	Menü . . . . .	32
3.6	Mehrsprachigkeit . . . . .	33
<b>4</b>	<b>Glossar</b>	<b>34</b>
<b>5</b>	<b>Referenzen</b>	<b>37</b>

## Abbildungsverzeichnis

1	Beispiel einer Sphäre . . . . .	4
2	Sphäre mit Pfeil in die nächste Sphäre . . . . .	11
3	Sphäre mit zusätzlichen Informationen . . . . .	13
4	Karte mit Vorschaubildern . . . . .	22
5	Ausschnitt einer Sphäre in der 360°-Ansicht . . . . .	25
6	Sphäre in Rektangularprojektion . . . . .	26
7	Sphäre als Cubemap . . . . .	26
8	Ausrichtung des Koordinatensystems und der Cubemap . . . . .	30
9	Beschreibung von Punkten mittels Kugelkoordinaten . . . . .	31
10	Koordinatensystem in einer Sphäre zum Ablesen von Winkeln . . . . .	32
11	Das Menü . . . . .	33

## Tabellenverzeichnis

1	Node.js Pakete . . . . .	6
2	Ordnerstruktur . . . . .	7
3	Spezifikation des JSON-Formats zur Konfiguration der Karten . . . . .	8
4	Spezifikation des JSON-Formats zur Konfiguration der Sphären . . . . .	10
5	Spezifikation des JSON-Formats für Pfeile . . . . .	12

6	Spezifikation des JSON-Formats für Info-Marker in einer Sphäre . . . . .	14
7	Spezifikation des onClick-Events für die Aktion newSphere . . . . .	15
8	Spezifikation des onClick-Events für die Aktion showPopup . . . . .	15
9	Auflistung der Grunt-Aufgaben . . . . .	17
10	Auflistung der zusammengesetzten Grunt-Aufgaben . . . . .	18
11	Benötigte Pakete zum initialen Erstellen von Ressourcen . . . . .	19
12	Liste kompatibler Browser . . . . .	21
13	Verwendete Bibliotheken der Browseranwendung . . . . .	22
14	Spezifikation des JSON-Formats für die Sphären auf einer Karte . . . . .	24
15	Spezifikation des JSON-Formats einer Sphäre . . . . .	28
16	Referenzen . . . . .	38

## Programmcodeverzeichnis

1	JSON-Beispiel zur Konfiguration einer Karte . . . . .	9
2	JSON-Beispiel zur Konfiguration einer Sphäre . . . . .	10
3	JSON-Beispiel zur Konfiguration eines Pfeils . . . . .	12
4	JSON-Beispiel zur Konfiguration eines Info-Markes . . . . .	14
5	Markdown-Beispiel für Informationen zu einem Info-Marker . . . . .	16
6	Beispiel der globalen Konfigurationsdatei für Grunt . . . . .	20
7	JSON-Beispiel der Sphären auf der Karte . . . . .	24
8	JSON-Beispiel einer Sphäre . . . . .	29
9	JSON-Beispiel einer i18next-Datei . . . . .	33

# 1 Einführung

Der Virtual-Explorer ist eine Webanwendung, die das Erkunden eines Geländes mit Hilfe von 360° Sphären erlaubt. Eine Sphäre besteht aus Bildern, die von einem Punkt in alle Richtungen aufgenommenen wurden und einen Rundumblick ermöglichen. Abbildung 1 zeigt hierzu ein Beispiel. Über die Webanwendung kann der Nutzer sich die Position aller verfügbaren Sphären auf einer Karte anschauen, sich in den Sphären umschauen und sich von einer in die nächste Sphäre bewegen. Zusätzlich können in einer Sphäre Informationen zu bestimmten Orten angeschaut werden.

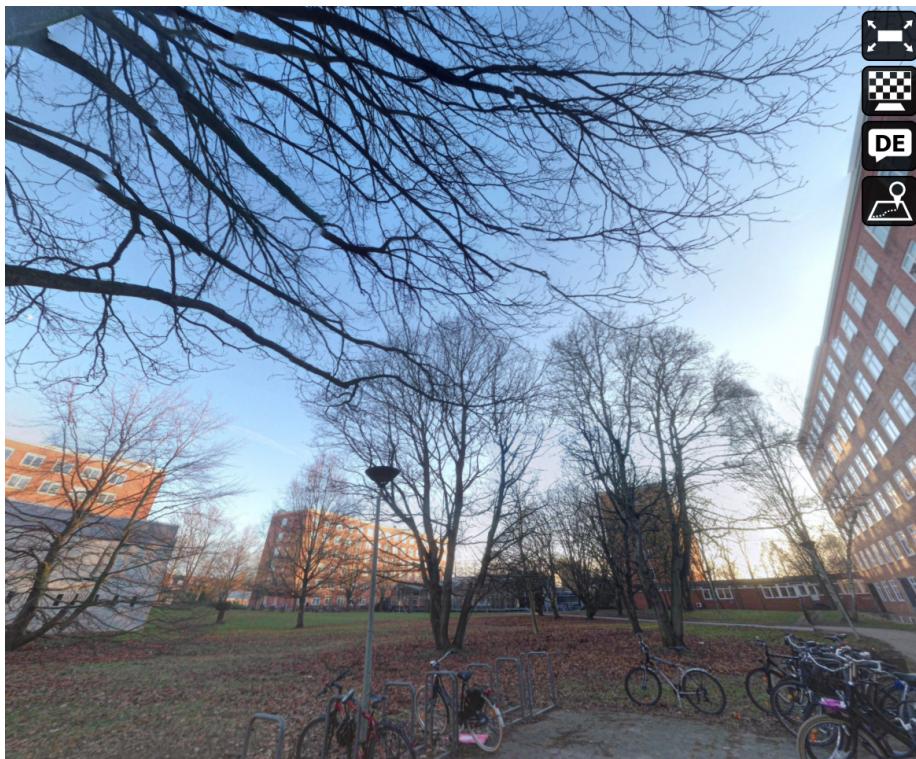


Abbildung 1: Beispiel einer Sphäre

## 1.1 Szenario

Ein Einsatzgebiet ist die virtuelle Darstellung des Universitätscampus. Hier können sowohl Außenansichten des Geländes sowie Innenansichten von Gebäuden und Räumen bereitgestellt werden, wodurch sich eine Vielzahl von Anwendungsszenarien des Virtual-Explorers ergeben.

So kann der *Virtual-Explorer* von Externen, zum Beispiel Studieninteressierten oder Vortragenden, zum Besichtigen der Universität genutzt werden. Diese haben die Möglichkeit die Uni schon vor ihrem ersten Besuch interaktiv online zu erkunden und als Vortragende im Voraus ihren Vortragsort zu besichtigen. Außerdem können zusätzliche nützliche Informationen an bestimmten Orten abgerufen werden.

Auch kann die Anwendung von Dozenten genutzt werden, um sich vor einer Vorlesung die Gegebenheiten in einem Hörsaal anzuschauen und bei Bedarf einen passenden Hörsaal zu suchen. ebenso können Studierende mit dem *Virtual-Explorer* leicht neue Ecken des Campus entdecken oder sich Wege zu Räumlichkeiten anschauen.

Es gibt somit allein für den Campus eine große Gruppe von unterschiedlichen Anwendern und Nutzungsszenarien.

## 1.2 Aufbau

Der *Virtual-Explorer* besteht aus zwei Komponenten. Zum einen aus dem Server, der die Inhalte generiert, bereithält und ausliefert und zum anderen aus der Browseranwendung, die an den Nutzer ausgeliefert wird und die Inhalte darstellt.

In dieser Dokumentation wird in Kapitel 2 zunächst auf den Server eingegangen. Kapitel 2.1 beschreibt dabei das verwendete Framework und die verwendeten Pakete. Die Ordnerstruktur des Projektes wird anschließend in Kapitel 2.2 aufgelistet.

In Kapitel 2.3 liegt das Augenmerk vor allem auf dem Hinzufügen von Karten und Sphären in den passenden Formaten. Dafür werden die Spezifikationen der Konfigurationsdateien beschrieben und die verschiedenen Möglichkeiten aufgelistet.

In Kapitel 2.4 werden die automatisierten Aufgaben vorgestellt, die das Anlegen und Überprüfen von Dateien in den richtigen Formaten vereinfachen und das damit verbundene Generieren von Dateien und Bildern übernehmen.

Im zweiten Teil der Dokumentation wird in Kapitel 3 die Browseranwendung ein- und in Kapitel 3.2 die verwendeten JavaScript Bibliotheken aufgeführt. Kapitel 3.3 befasst sich mit den Karten, auf denen das Gelände und die verfügbaren Sphären gezeigt werden. Dabei wird auch auf die Spezifikationen des Formats der in Kapitel 2.4 generierten Dateien eingegangen.

In Kapitel 3.4 wird die Sphärenansicht eingeführt und auch hier wird dabei auf die in Kapitel 2.4 generierten Dateien eingegangen. Zusätzlich gibt es eine kurze Einführung in das vom 3D-Framework verwendete Koordinatensystem und die Konvertierung von Kugelkoordinaten in dieses System.

Im Glossar (Kapitel 4) sind einige wichtige Begriffe aufgeführt und erklärt. Unter Referenzen (Kapitel 5) befindet sich abschließend eine Auflistung aller verwendeten Pakete und Software.

## 2 Server

Mit Hilfe von Node.js wird ein Webserver realisiert. Node.js stellt grundlegende Netzwerk- und Serverkomponenten zur Verfügung und basiert auf der JavaScript V8 Engine, die ursprünglich für Google Chrome entwickelt wurde. Node.js ist modular aufgebaut, wobei mit dem npm (Node Package Manager) gewünschte Module nachinstalliert werden können.

### 2.1 Express

Eines dieser Pakete ist das Express-Framework, welches einfache Werkzeuge für das Entwickeln von Webanwendungen bereitstellt. Der Einstiegspunkt ist hierfür die Datei app.js in welcher der Server initialisiert wird. Express verwendet zusätzlich noch weitere Pakete, die in Tabelle 1 aufgeführt sind.

Paket	Version	Beschreibung
body-parser	~1.13.2	Parsed den Body eines Strings.
cookie-parser	~1.3.5	Dient zum Parsen von Cookies in ein JSON-Objekt.
debug	~2.2.0	Stellt Funktionen zum Debuggen der Anwendung und zum Ausgeben von Warn- und Fehlermeldungen bereit.
express	~4.13.1	Das Express-Framework selber.
jade	~1.11.0	Wandelt Jade-Code in HTML um.
morgan	~1.6.1	Protokolliert HTTP-Anfragen.
serve-favicon	~2.3.0	Liefert das Favicon aus.

Tabelle 1: Node.js Pakete

### 2.2 Ordnerstruktur

Zur besseren Übersicht ist das Projekt in mehrere Hauptordner gegliedert. Hierbei gibt das Express-Framework eine Struktur vor, die um die Ordner extra, raw und tasks erweitert wurde. Tabelle 2 listet diese Ordner auf und beschreibt deren Inhalt.

Ordner	Beschreibung
bin	Enthält die Datei, die zum Starten des Servers ausgeführt wird.
extra	Dient zur Ablage von nicht einsortierbaren Dateien, zum Beispiel die Blender-Projektdateien zur Generierung der Vorschaubilder (siehe Kapitel 2.4.4).
node_modules	Enthält die über den Node-Package-Manager installierten Pakete.
public	Enthält die Dateien, die an den Nutzer ausgeliefert werden.
raw	Enthält die Rohdateien zur Auslieferung an den Nutzer. Diese müssen vor dem Serverstart mit Hilfe von Grunt (siehe 2.4) angepasst und in den Ordner public verschoben werden.
routes	Enthält die Routenbeschreibungen zur Spezifizierung der auszuliefernden Dateien.
tasks	Enthält die selbstdefinierten Grunt-Aufgaben.
views	Enthält die Jade-Dateien für die Webinhalte.

Tabelle 2: Ordnerstruktur

## 2.3 Konfiguration von Karten und Sphären

Ein wichtiges Feature des Virtual-Explorers ist die einfache Konfigurierbarkeit der darzustellenden Karten und Sphären. Um diese zu gewährleisten, wird das JSON-Format verwendet. Die genauen Spezifikationen der Konfigurationsdateien werden in den folgenden Kapiteln beschrieben.

Jede JSON-Datei, die im entsprechenden Unterordner vom raw-Ordner liegt, wird dabei als Beschreibung für eine Karte oder eine Sphäre interpretiert. Die vorhandenen Karten und Sphären müssen somit nicht noch an anderer Stelle aufgeführt werden. Hieraus ergibt sich auch, dass es genau eine Datei für jede Karte und eine für jede Sphäre gibt.

Zu beachten ist noch, dass diese Dateien mit denen in Kapitel 2.4 beschriebenen Aufgaben überprüft, zusammengefasst und erweitert werden. Die daraus resultierenden JSON-Dateien werden in Kapitel 3 vorgestellt. Dabei wird die Funktion einiger der hier aufgelisteten Attribute deutlich.

### 2.3.1 Konfiguration einer Karte

Zur Konfiguration einer Karte müssen verschiedene Attribute in der Konfigurationsdatei existieren und gesetzt sein. Diese Attribute legen das Verhalten der Karte fest

und definieren somit zum Beispiel welcher Kartenausschnitt gezeigt wird und welche Sphären auf der Karte angezeigt werden.

Tabelle 3 listet diese Attribute auf, wobei die Spalte „Gefordert“ angibt, ob das entsprechende Attribut existieren muss. Ist ein Attribut selber ein Objekt, werden die entsprechenden Objektattribute aufgelistet, indem sie mit einem Punkt beginnen und nach rechts eingerückt sind.

Attribut	Gefordert	Beschreibung
<code>id</code>	ja	Die ID der Karte.
<code>bound</code>	ja	Gibt den Kartenausschnitt an.
<code>.northWest</code>	ja	Gibt die obere linke Begrenzung der Karte an.
<code>.lat</code>	ja	Gibt den Breitengrad der oberen linken Begrenzung an.
<code>.long</code>	ja	Gibt den Längengrad der oberen linken Begrenzung an.
<code>.southEast</code>	ja	Wie <code>bound.northWest</code> nur für die untere rechte Begrenzung
<code>center</code>	ja	Mittelpunkt der Karte beim Aufrufen.
<code>.lat</code>	ja	Breitengrad des Mittelpunktes.
<code>.long</code>	ja	Längengrad des Mittelpunktes.
<code>minZoom</code>	ja	Minimales Zoom-Level der Karte.
<code>maxZoom</code>	ja	Maximales Zoom-Level der Karte.
<code>startZoom</code>	ja	Initiales Zoom-Level beim Aufrufen der Karte.

Tabelle 3: Spezifikation des JSON-Formats zur Konfiguration der Karten

Der Programmcode 1 zeigt ein Beispiel des in Tabelle 3 spezifizierten JSON-Formats.

```
{
  "id": "44c2e9bdcaf4c29b",
  "bound": {
    "northWest": {
      "lat": "54.349024",
      "long": "10.101001"
    },
    "southEast": {
      "lat": "54.335277",
      "long": "10.129754"
    }
  },
  "center": {
    "lat": "54.3389585",
    "long": "10.1190736"
  },
  "minZoom": "16",
  "maxZoom": "19",
  "startZoom": "16"
}
```

Programmcode 1: JSON-Beispiel zur Konfiguration einer Karte

### 2.3.2 Konfiguration einer Sphäre

Die Konfiguration der Sphären verhält sich ähnlich wie die in Kapitel 2.3.1 beschriebene Konfiguration der Karten. Hier werden jedoch andere Attribute verwendet, die in Tabelle 4 aufgelistet werden. Zusätzlich existiert eine Spalte „math“ in der angegeben wird, ob das entsprechende Attribut mit Hilfe der `math.js`-Bibliothek (siehe Glossar) ausgewertet wird.

Attribut	Gefordert	math	Beschreibung
<code>id</code>	ja	nein	ID der Sphäre.
<code>belongsToMap</code>	ja	nein	Array mit Karten-IDs zu denen die Sphäre gehört. Hier muss mindestens eine Karte angegeben sein.
<code>initialView</code>	nein	nein	Initiale Blickrichtung beim Betreten der Sphäre von der Karte aus.
<code>.lat</code>	nein	ja	Breitengrad der initialen Blickrichtung. Standard ist 0.
<code>.long</code>	nein	ja	Längengrad der initialen Blickrichtung. Standard ist 0.
<code>name</code>	nein	nein	Verlinkung auf einen Namen mittels i18next (siehe Glossar).

Attribut	Gefordert	math	Beschreibung
coords	ja	nein	Koordinaten der Sphäre.
.lat	ja	nein	Breitengrad der Sphäre.
.long	ja	nein	Längengrad der Sphäre.
arrows	nein	nein	In eine andere Sphäre weiterführende Pfeile. Siehe Kapitel 2.3.3.
infos	nein	nein	Zusätzliche Informationen zu einem Punkt in der Sphäre. Siehe Kapitel 2.3.4.
objects	nein	nein	Beliebige Three.js Objekte.

Tabelle 4: Spezifikation der JSON-Formats zur Konfiguration der Sphären

In Programmcode 2 ist ein Beispiel des in der vorherigen Tabelle spezifizierten JSON-Formates zu sehen.

```
{
  "id": "13bad6b1e478b951",
  "belongsToMany": ["44c2e9bdcaf4c29b"],
  "initialView": {
    "lat": "30/90*PI/2",
    "long": "320/360*2*PI"
  },
  "name": "sphere.13bad6b1e478b951.name",
  "coords": {
    "lat": "54.337858",
    "long": "10.120956"
  },
  "arrows": [
    {
      "long": "2*PI*138/360",
      "next_sphere": "d875d5d81bbbc835"
    }
  ],
  "infos": [
    {
      "long": "2*PI*270/360",
      "lat": "PI/2*-10/90",
      "content": "fs-infmath.html"
    }
  ],
  "objects": []
}
```

Programmcode 2: JSON-Beispiel zur Konfiguration einer Sphäre

### 2.3.3 Konfiguration eines Pfeils

Zu einer Sphäre kann eine beliebige Anzahl an Pfeilen definiert werden, die in andere Sphären führen. Diese werden in einem Array aufgeführt, das in der Sphären-Konfiguration unter dem Attribut `arrows` abgelegt ist (siehe Kapitel 2.3.2). Abbildung 2 zeigt einen solchen Pfeil in einer Sphäre.



Abbildung 2: Sphäre mit Pfeil in die nächste Sphäre

Die für einen Pfeil benötigten und optionalen Attribute sind in Tabelle 5 aufgelistet. Die Konfiguration erlaubt es auch beliebige andere Bilder in einer Sphäre anzuzeigen. Die Spalten der Tabelle sind dabei wie in Kapitel 2.3.2 zu lesen.

Attribut	Gefordert	math	Beschreibung
lat	nein	ja	Position des Pfeils in der Sphäre entlang des Breitengrades (Höhe des Pfeils). Standard ist $-\frac{\pi}{8}$ (Siehe Kapitel 3.4.2).
long	ja	ja	Position des Pfeils in der Sphäre entlang Längengrades (Siehe Kapitel 3.4.2).
rotationX	nein	ja	Rotation des Pfeils entlang der X-Achse ausgehend vom Standard-Wert. Standard ist $\frac{\pi}{2}$ .

Attribut	Gefordert	math	Beschreibung
rotationY	nein	ja	Rotation des Pfeils entlang der Y-Achse. Standard ist 0.
rotationZ	nein	ja	Rotation des Pfeils entlang der Z-Achse ausgehend vom Standard-Wert. Standard ist -long.
radius	nein	ja	Entfernung des Pfeiles vom Mittelpunkt. Standard ist 90. (Siehe Kapitel 3.4.2).
size	nein	ja	Größe des Pfeils. Standard ist 30.
texture	nein	nein	Verwendete Textur. Standard ist /images/objects/arrow.png.
onClick	nein	nein	Gibt an was beim Klicken auf den Pfeil passieren soll (Siehe Kapitel 2.3.5).

Tabelle 5: Spezifikation des JSON-Formats für Pfeile

Als Beispiel zeigt Programmcode 3 die Konfiguration eines Pfeils. In den meisten Fällen wurde der Standardwert verwendet, wodurch der Eintrag auch weggelassen werden könnte.

```
{
    "lat": "-PI/8",
    "long": "0",
    "rotationX": "0",
    "rotationY": "0",
    "rotationZ": "0",
    "radius": "90",
    "size": "30",
    "texture": "/images/objects/arrow.png",
    "onClick": {
        "type": "newSphere",
        "nextSphere": "db352ed5a0fee174",
        "nextCameraLat": "0",
        "nextCameraLong": "2*PI*120/360"
    }
}
```

Programmcode 3: JSON-Beispiel zur Konfiguration eines Pfeils

### 2.3.4 Konfiguration eines Info-Markers

Zusätzlich können in einer Sphäre Info-Marker positioniert und damit Informationen zu einem Sphäreninhalt hinterlegt werden. Durch den Klick auf einen Marker wer-

den die Informationen in der oberen linken Ecke angezeigt. Im Gegensatz zu einem Pfeil ist ein Info-Marker ein Sprite, was bedeutet, dass er in Richtung der Kamera ausgerichtet und somit immer frontal zu sehen ist. In Abbildung 3 ist ein solcher Info-Marker mit den zusätzlichen Informationen zu sehen.

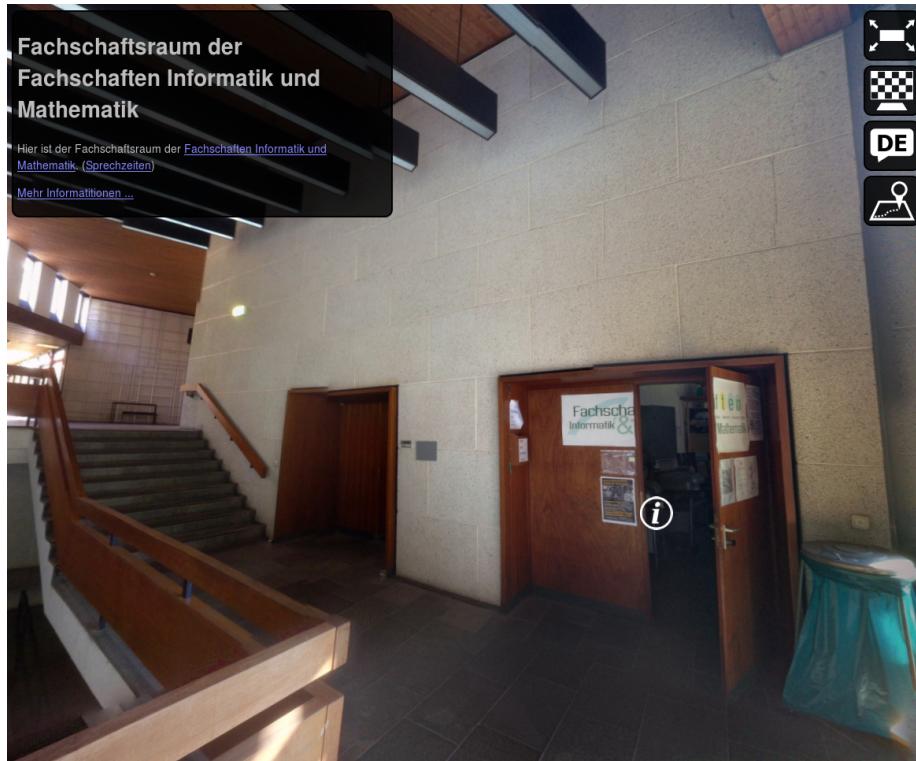


Abbildung 3: Sphäre mit zusätzlichen Informationen

In Tabelle 6 wird analog zu Kapitel 2.3.3 die Spezifikation für Info-Marker angegeben. Auch diese sind in einem Array aufzuführen, das in der Sphären-Konfiguration unter dem Attribut `infos` abgelegt ist (siehe Kapitel 2.3.2).

<b>Attribut</b>	<b>Gefordert</b>	<b>math</b>	<b>Beschreibung</b>
lat	ja	ja	Breitengrad des Info-Markers in der Sphäre (Siehe Kapitel 3.4.2).
long	ja	ja	Längengrad des Info-Markers in der Sphäre (Siehe Kapitel 3.4.2).
size	nein	ja	Größe des Info-Marker-Symbol. Standard ist 60.
texture	nein	nein	Verwendete Textur. Standard ist /images/objects/info.png.

Attribut	Gefordert	math	Beschreibung
onClick	nein	nein	Gibt an was beim Klicken auf den Info-Marker passieren soll (Siehe Kapitel 2.3.5).

Tabelle 6: Spezifikation des JSON-Formats für Info-Marker in einer Sphäre

Der Programmcode 4 zeigt eine beispielhafte Konfiguration für einen Info-Marker.

```
{
    "lat": "PI/2*-10/90",
    "long": "2*PI*270/360",
    "size": "60",
    "texture": "/images/objects/info.png",
    "onClick": {
        "type": "showPopup",
        "content": "fs-infmath"
    }
}
```

Programmcode 4: JSON-Beispiel zur Konfiguration eines Info-Markers

### 2.3.5 Konfiguration des onClick-Events für Pfeile und Info-Marker

Zu einem Pfeil und einem Info-Marker kann eine Aktion hinzugefügt werden, die beim Klicken auf das Objekt ausgeführt wird. Es gibt hier die Aktionen `newSphere`, die den Wechsel in eine neue Sphere auslöst, sowie die Aktion `showPopup`, die Informationen in einem Popup-Fenster anzeigt.

Die Aktion `newSphere` ist hierbei primär für Pfeile und `showPopup` für Info-Marker vorgesehen, aber nicht daran gebunden. Das bedeutet das auch ein Info-Marker die Aktion `newSphere` erhalten kann.

Tabelle 7 zeigt die möglichen Attribute für die `newSphere`-Aktion, Tabelle 8 zeigt dies für die `showPopup`-Aktion.

Attribut	Gefordert	math	Beschreibung
type	ja	nein	Muss auf <code>newSphere</code> gesetzt sein.
nextSphere	ja	nein	ID der Sphäre, in die durch das Klicken auf den Pfeil gewechselt werden soll.

Attribut	Gefordert	math	Beschreibung
nextCameraLat	nein	ja	Ausrichtung der Kamera beim Betreten der verlinkten Sphäre entlang des Breitengrades. Wird nichts angegeben, wird die Kamera nicht gedreht.
nextCameraLong	nein	ja	Ausrichtung der Kamera beim Betreten der verlinkten Sphäre entlang des Längengrades. Standardmäßig wird die Kamera um 180° entgegen des Pfeils ausgerichtet, der aus der verlinkten Sphäre in die verlassene Sphäre zurückführt. Existiert dieser nicht, wird die Kamera nicht gedreht.

Tabelle 7: Spezifikation des onClick-Events für die Aktion newSphere

Attribut	Gefordert	Beschreibung
type	ja	Muss auf showPopup gesetzt sein.
content	ja	Dateiname des anzuzeigenden Inhaltes. Der Inhalt muss als html-Datei (Siehe Kapitel 2.3.6) im Ordner /public/infos/{language} vorliegen. Die Bedeutung von {language} wird in Kapitel 3.6 beschrieben. Die Dateiendung wird nicht mit angegeben.

Tabelle 8: Spezifikation des onClick-Events für die Aktion showPopup

### 2.3.6 Erstellen von Informationen für die Info-Marker

Zum einfachen Erstellen von Informationen für die Info-Marker wird Markdown verwendet. Die im Ordner `raw/markdown` hinterlegten Markdown-Dateien werden dabei, wie in Kapitel 2.4 beschrieben, automatisch in HTML-Dateien umgewandelt. Dafür muss die Datei die Dateiendung `.md` benutzen. Der Name der Datei (ohne `.md`) wird dann für Verlinkung im `content`-Attribut der JSON-Datei (siehe Tabelle 6) und zum Verlinken auf weiterführende Informationen (siehe Programmcode 5) verwendet.

Programmcode 5 zeigt den zu Abbildung 3 passenden Markdown-Code. Neben der Standard Markdown-Syntax gibt es zusätzlich die Möglichkeit einen Link mit einer Raute zu beginnen, um so intern zu verlinken. Die aktuellen Informationen werden dann durch die verlinkten ersetzt (siehe letzte Zeile im Programmcode).

```
# Fachschaftsraum der Fachschaften Informatik und Mathematik  
Hier ist der Fachschaftsraum der [Fachschaften Informatik und Mathematik]  
(https://www.fs-infmath.uni-kiel.de/wiki/Hauptseite).  
([Sprechzeiten] (https://www.fs-infmath.uni-kiel.de/wiki/Sprechzeiten))  
  
[Mehr Informatitionen ...] (#fs-infmath_more)
```

Programmcode 5: Markdown-Beispiel für Informationen zu einem Info-Marker

## 2.4 Grunt

Vor dem Starten des Servers müssen verschiedene Skripte ausgeführt werden, die Dateien und Bilder generieren. Zum Beispiel werden JSON-Dateien geprüft und verändert, sowie Cubemaps (siehe Kapitel 3.4) und Vorschaubilder für die Sphären generiert. Dazu wird `Grunt` verwendet, mit dem vordefinierte Aufgaben ausgeführt werden können. Diese verwenden den Ordner `raw` als Quell- und den Ordner `public` als Zielverzeichnis.

### 2.4.1 Aufgaben

Es existieren die in Tabelle 9 aufgelisteten Aufgaben. Diese können mit dem Befehl `grunt` gefolgt vom Namen der Aufgabe im Projektverzeichnis ausgeführt werden. Dabei sollen vor allem die in Kapitel 2.4.2 beschriebenen Aufgaben ausgeführt werden, da diese sinnvoll gruppiert sind.

Aufgabe	Beschreibung
<code>concurrent:watch</code>	Überwacht die JavaScript- und Less-Dateien und führt <code>script</code> und <code>style</code> bei Veränderungen aus. Dies ist sinnvoll für die Entwicklung der Anwendung.
<code>concurrent:watchDev</code>	Wie <code>concurrent:watch</code> , außer dass <code>scriptDev</code> und <code>styleDev</code> ausgeführt werden.
<code>copy</code>	Kopiert Dateien.
<code>generatecubemap</code>	Generiert aus Sphären in Rektangularprojektion Cubemaps in verschiedenen Auflösungen. Hierfür wird zusätzliche Software benötigt (siehe Kapitel 2.4.4).

Aufgabe	Beschreibung
generatemapjson	Generiert aus den JSON-Dateien der Sphären eine JSON-Datei für jede Karte mit gesammelten Informationen (siehe 3.3.1).
generatepreview	Generiert aus Sphären in Rektangularprojektion Vorschaubilder für die Karte. Hierfür wird zusätzliche Software benötigt (siehe Kapitel 2.4.4).
markdown2html	Compiliert die Markdown-Dateien mit auf den Sphären anzuseigenden Informationen zu HTML. Hierzu wird die marked-Bibliothek verwendet, wobei das Verhalten für Links angepasst wurde. Externe Seiten sollen in neuem Tab geöffnet werden, wohingegen Links, die mit einer Route beginnen, zu JavaScript werden, welches die neue Seite im selben Infofenster anzeigt (siehe 2.3.6).
prepairmapjson	Prüft und bearbeitet die JSON-Dateien für die Karten.
preairspherejson	Prüft und bearbeitet die JSON-Dateien für die Sphären.
script	Hängt die einzelnen JavaScript-Dateien hintereinander und minimiert sie. Hierfür werden die Grunt-Plugins concat und uglify verwendet.
scriptDev	Wie script, außer dass die Datei nicht minimiert wird.
style	Compiliert die einzelnen Less-Dateien zu CSS, hängt diese hintereinander und minimiert sie. Hierzu werden die Grunt-Plugins concat, less und cssmin verwendet.
styleDev	Wie style, außer dass die Datei nicht minimiert wird.

Tabelle 9: Auflistung der Grunt-Aufgaben

#### 2.4.2 Gruppierung von Aufgaben

Die im vorherigen Kapitel aufgeführten Aufgaben sind in sinnvollen Gruppen zusammengefasst, um die Verwendung zu vereinfachen. Tabelle 10 listet diese Aufgaben auf und gibt an, welche der in Tabelle 9 aufgeführten Aufgaben nacheinander ausgeführt werden.

Aufgabe	Auflistung der Aufgaben
all	copy, script, style, generatecubemap, generatepreview, prepairspherejson, prepairmapjson, generatemapjson, markdown2html
allDev	copy, scriptDev, styleDev, generatecubemap, generatepreview, prepairspherejson, prepairmapjson, generatemapjson, markdown2html
default	copy, script, style, prepairspherejson, prepairmapjson, generatemapjson, markdown2html, concurrent:watch
dev	copy, scriptDev, styleDev, prepairspherejson, prepairmapjson, generatemapjson, markdown2html, concurrent:watchDev
init	copy, script, style, prepairspherejson, prepairmapjson, generatemapjson, markdown2html
initDev	copy, scriptDev, styleDev, prepairspherejson, prepairmapjson, generatemapjson, markdown2html

Tabelle 10: Auflistung der zusammengesetzten Grunt-Aufgaben

#### 2.4.3 Verwendete Plugins und npm-Pakete

Grunt verwendet ein modulares Plugin System, sodass für viele Aufgaben vorgefertigte Plugins verwendet werden können. Diese können, wie andere Node.js-Module, mit dem npm installiert werden. Die vom Virtual-Explorer verwendeten vorgefertigten Grunt-Plugins und andere zum initialen Erstellen von Ressourcen verwendete Pakete sind in Tabelle 11 aufgeführt.

Paket	Version	Beschreibung
grunt	^0.4.5	Das Grunt-Paket selber.
grunt-concurrent	^2.1.0	Erlaubt es mehrere watch-Aufgaben gleichzeitig auszuführen.
grunt-contrib-concat	^0.5.1	Erlaubt das Zusammenhängen von mehreren Dateien zu einer Datei.
grunt-contrib-copy	^0.8.2	Erlaubt das Kopieren von Dateien.
grunt-contrib-cssmin	^0.14.0	Erlaubt das Minimieren von CSS-Code.
grunt-contrib-jshint	^0.12.0	Erlaubt das Testen des JavaScript Codes auf häufige Fehler.

Paket	Version	Beschreibung
grunt-contrib-less	^1.2.0	Erlaubt das Umwandeln von Less- in CSS-Code zur Auslieferung an einen Browser.
grunt-contrib-uglify	^0.11.1	Erlaubt das Minimieren von JavaScript-Code.
grunt-contrib-watch	^0.6.1	Erlaubt das Überwachen von Dateien, um bei Änderungen andere Aufgaben anzustoßen.
marked	^0.3.5	Erlaubt das Umwandeln von Markdown-Code zu HTML.
mathjs	~ 2.6.0	Wandelt mathematische Ausdrücke in Gleitkommazahlen um.
mkdirp	~ 0.5.1	Erlaubt das Erstellen von Ordnern inklusive der Oberverzeichnisse.

Tabelle 11: Benötigte Pakete zum initialen Erstellen von Ressourcen

#### 2.4.4 Zusätzliche Software

Das Generieren der Cubemaps verwendet zusätzlich noch die Programme `Panotools` und `Hugin`. Für das Generieren der Vorschaubilder wird die Anwendung `Blender` benötigt. Diese Anwendungen müssen zusätzlich installiert sein. Liegen die Cubemaps und Vorschaubilder bereits in allen erforderlichen Auflösungen vor, werden diese Anwendungen nicht benötigt.

#### 2.4.5 Globale Konfiguration

Neben den einzelnen Konfigurationen für die Grunt-Aufgaben gibt es eine globale Konfigurationsdatei (`raw/config.json`), in der für alle Aufgaben hinterlegt ist, welche Bilder es geben soll und wo diese abgelegt werden sollen.

```
{
  "cubemapSizes": [
    {"size": 2048, "path": "/images/spheres/$id$/cubemap/2048"},  

    {"size": 1024, "path": "/images/spheres/$id$/cubemap/1024"},  

    {"size": 512, "path": "/images/spheres/$id$/cubemap/512"}  

  ],  

  "previewSizes" : [  

    {"size": 256, "path": "/images/spheres/$id$/preview/256", "count": 30},  

    {"size": 128, "path": "/images/spheres/$id$/preview/128", "count": 30},  

    {"size": 64, "path": "/images/spheres/$id$/preview/64", "count": 10}
  ]
}
```

Programmcode 6: Beispiel der globalen Konfigurationsdatei für Grunt

Die beispielhafte Konfiguration gibt an, dass es drei Auflösungen (2048x2048, 1024x1024 und 512x512) aller Cubemaps gibt und den Pfad wo diese abgelegt sind. Dabei wird \$id\$ durch die jeweilige ID der Sphäre ersetzt. Genauso gibt es die Vorschaubilder der Sphären auf der Karte in drei Auflösungen. Daneben wird hier zusätzlich noch mit count angegeben, in wie vielen Schritten die Drehung des Vorschaubildes animiert werden soll (siehe Kapitel 3.3). Die Aufgaben generatecubemap und generatepreview prüfen diese Konfiguration und generieren fehlende Bilder bei Bedarf.

### 3 Browseranwendung

Der Server liefert HTML, CSS und JavaScript an den Browser des Nutzers aus. Die Visualisierung der Daten erfolgt dann vollständig durch den Browser. Hierbei gibt es zwei wesentliche Ansichten.

In der Kartenansicht erhält der Nutzer einen Überblick über das Gelände der Universität. Er kann den Kartenausschnitt verändern und an beliebigen Stellen hineinzoomen. Dabei werden Vorschauen der Sphären angezeigt, die an den entsprechenden Orten verfügbar sind. Diese sind nach Bedarf animiert, um eine bessere Vorschau der 360°-Bilder zu gewährleisten. Klickt der Nutzer auf die Vorschau, so wechselt er in die Sphärenansicht.

In der Sphärenansicht kann sich der Nutzer virtuell in der Sphäre umschauen. Hier kann der Sichtausschnitt des 360°-Bildes verändert und hineingezoomt werden. Zusätzlich können sich noch Objekte, zum Beispiel Pfeile oder Info-Marker, in der Sphäre befinden, mit denen der Nutzer interagieren kann. Hiermit ist es zum Beispiel möglich – ohne Umweg über die Karte – zu einer benachbarten Sphäre zu wechseln oder sich zusätzliche Informationen zu einem Ort anzeigen zu lassen.

### 3.1 Browserkompatibilität

Die Browseranwendung verwendet insbesondere HTML5 und WebGL, wodurch beide Technologien für den Betrieb benötigt werden. Daraus – sowie aus Tests – ergeben sich die in Tabelle 12 aufgeführten Browserkompatibilitäten.

Browser	Version
Chrome	ab Version 18
Chrome (Android)	ab Version 49
Edge	ab Version 12
Firefox	ab Version 4
Internet Explorer	ab Version 11
Opera	ab Version 15
Opera Mobile	ab Version 36
Safari (iOS)	ab Version 8
Safari (MacOS)	ab Version 8

Tabelle 12: Liste kompatibler Browser

### 3.2 Verwendete JavaScript-Bibliotheken

Die Clientanwendung verwendet die in Tabelle 13 aufgelisteten JavaScript-Bibliotheken.

Bibliothek	Version	Beschreibung
i18next	1.11.0	Ermöglicht eine einfache Implementierung von Mehrsprachigkeit.
jQuery	2.1.4	Das jQuery-Framework ist eine Erweiterung von JavaScript, um häufig benötigte Funktionen kürzer implementieren zu können.
leaflet	0.7.7	Die Leaflet-Bibliothek stellt eine interaktive leicht konfigurierbare Karte bereit.
leaflet.markercluster	0.4.0	Erweitert Leaflet, um die Möglichkeit Markierungen auf der Karte in Gruppen zusammenzufassen.
leaflet-canvasicon	0.1.4	Erweitert Leaflet, um die Möglichkeit als Markierung auf der Karte ein Canvas zu verwenden. Die Inhalte eines Canvas können dynamisch verändert und animiert werden.

Bibliothek	Version	Beschreibung
math.js	2.4.2	Wandelt mathematische Ausdrücke in Gleitkommazahlen um.
three.js	r75	Eine JavaScript-3D-Bibliothek, die das Erstellen und Anzeigen von 3D-Szenen erlaubt. So können zum Beispiel dreidimensionale Sphären erstellt werden, um eine 360°-Sicht zu ermöglichen.

Tabelle 13: Verwendete Bibliotheken der Browseranwendung

### 3.3 Kartenansicht

Die Karte basiert auf Leaflet, einem freien Kartenframework, welches die Grundfunktionalitäten der Karte bereitstellt. In Abbildung 4 ist eine beispielhafte Karte der Universität zu Kiel zu sehen.

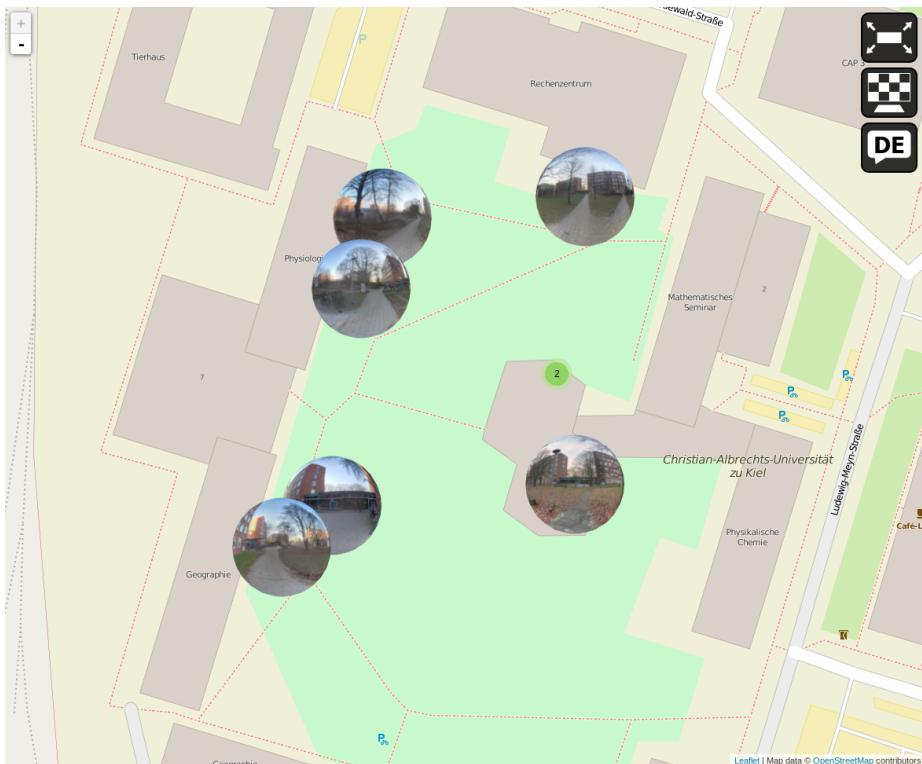


Abbildung 4: Karte mit Vorschaubildern

Die runden Vorschaubilder erlauben einen Eindruck von den dahinter liegenden Sphären. Diese erscheinen – aus Gründen der Übersichtlichkeit – erst bei einem ausreichenden Zoom-Level. Zusammengefasste Sphären erscheinen als farbiger Punkt, der mit der Anzahl der dort vorhandenen Sphären beschriftet ist. Wird auf einen solchen Punkt geklickt, werden die zugehörigen Sphären angezeigt.

Fährt man mit der Maus über eines der Vorschaubilder, so beginnt es sich zu drehen, um einen besseren Eindruck des 360°-Bildes zu gewährleisten. Zusätzlich wird die Beschreibung der Sphäre – falls hinterlegt – angezeigt.

### 3.3.1 Konfiguration einer Karte

Zu einer Karte gehören zwei JSON-Dateien. Die eine definiert grundsätzliche Eigenschaften der Karte, zum Beispiel deren ID oder den anzuzeigenden Kartenausschnitt.

In Kapitel 2.3.1 wird die erste JSON-Datei spezifiziert. Diese wird mit Grunt nur überprüft und nicht modifiziert, sodass die entsprechende Tabelle 3 hier nicht erneut aufgeführt ist.

Die auf der Karte angezeigten Informationen werden aus einer vom Server bereitgestellten zweiten JSON-Datei entnommen, die zuvor generiert werden muss (siehe Abschnitt 2.4). Dies hat den Vorteil, dass zum Anzeigen der Karte nicht die JSON-Datei zu jeder auf der Karte befindlichen Sphäre geöffnet werden muss, sondern alle benötigten Informationen in einer einzigen Datei bereitstehen.

Die generierte JSON-Datei hat den in Tabelle 14 gezeigten Aufbau. Hierbei ist zu beachten, dass jede Sphäre unter ihrer ID in dieser Datei abgelegt ist (vergleiche Programmcode 7). Die Tabelle zeigt nur die Attribute eines unter einer solchen ID abgelegten Objektes.

Die Spalte „Gefordert“ gibt hier an, ob das entsprechende Attribut existieren muss. Wenn ein Attribut selber ein Objekt ist, werden die entsprechenden Objektattribute aufgelistet, indem sie mit einem Punkt beginnen und nach rechts eingerückt sind.

Attribut	Gefordert	Beschreibung
id	ja	ID der Sphäre.
name	nein	Verlinkung auf einen Namen mittels i18next.
coords	ja	Koordinaten der Sphäre.
.lat	ja	Breitengrad der Sphäre.
.lat	ja	Längengrad der Sphäre.
images	ja	Zur Sphäre gehörende Bilder.
.preview	ja	Zur Sphäre gehörende Vorschaubilder.

Attribut	Gefordert	Beschreibung
.256	ja	Vorschaubilder der Sphäre in der Auflösung 256x256. Hier muss für jedes in der Konfiguration angegebene Auflösung (siehe Kapitel 2.4.5) ein Eintrag existieren.
.size	ja	Auflösung der Bilder. Muss mit dem Namen des Eintrages übereinstimmen.
.path	ja	Pfad zu den Vorschaubildern. Unter dem angegebenen Pfad müssen dazu entsprechende Bilder mit dem Namen iconxxxx.png liegen, wobei zunächst das Bild mit Namen icon0001.png angezeigt wird.
.count	ja	Fährt der Nutzer mit der Maus über die Vorschau, so werden die Bilder in aufsteigender Reihenfolge bis zur Nummer images.preview.256.count angezeigt. Anzahl der Bilder für die Animation der Vorschau.

Tabelle 14: Spezifikation des JSON-Formats für die Sphären auf einer Karte

Programmcode 7 zeigt einen Beispielcode für eine Sphäre. Die drei Punkte zeigen an an, dass hier weitere Sphäre auf die selbe Art hinzugefügt werden können.

```
{
  "13bad6b1e478b951": {
    "id": "13bad6b1e478b951",
    "name": "sphere.13bad6b1e478b951.name",
    "coords": {
      "lat": "54.337858",
      "long": "10.120956"
    },
    "images": {
      "preview": {
        "256": {
          "size": 256,
          "path": "images/spheres/13bad6b1e478b951/preview/256",
          "count": 30
        }
      }
    }
  }
}
```

Programmcode 7: JSON-Beispiel der Sphären auf der Karte

### 3.4 Sphärenansicht

Die Sphärenansicht verwendet die Three.js-Bibliothek und basiert auf PhotoSphereViewer. Die Sphären werden in einer 3D-Umgebung dargestellt, die den Eindruck vermittelt, vor Ort zu sein. Die Verwendung von Three.js vereinfacht es zum Einen ein 360°-Bild dreidimensional darzustellen und zum Anderen auch beliebige andere Objekte (wie zum Beispiel Pfeile) zur Szene hinzuzufügen.

Durch gedrückt halten und ziehen der Maus kann sich der Nutzer in der Sphäre umschauen und den Bildausschnitt verändern. Mit dem Mausrad kann er hinein und heraus zoomen. Durch einen Klick auf die Pfeile kann sich der Nutzer in eine benachbarte Sphäre bewegen.

Abbildung 5 zeigt eine solche Sphäre inklusive Pfeile. In Abbildung 3 (Kapitel 2.3.4) ist zusätzlich noch das Infosymbol zu sehen, mit dem der Nutzer zusätzliche Informationen abrufen kann. Diese erscheinen in der oberen linken Ecke.



Abbildung 5: Ausschnitt einer Sphäre in der 360°-Ansicht

Zur Darstellung wird eine sogenannte Cubemap der Sphären verwendet. Aus einem 360°-Bild in Rektangularprojektion (siehe Abbildung 6) werden dazu sechs entsprechend projizierte quadratische Bilder generiert (siehe Abschnitt 2.4).



Abbildung 6: Sphäre in Rektangularprojektion

Diese stellen die sechs Seiten eines Würfels da, welcher aufgeklappt in Abbildung 7 zu sehen ist.



Abbildung 7: Sphäre als Cubemap

Durch die Darstellung als Cubemap wird eine bessere Performance erzielt, da das 3D-Modell einer Kugel aus wesentlich mehr Flächen besteht. Die Kamera befindet sich in der Mitte des Würfels, auf dessen sechs Innenseiten die Bilder gelegt werden.

Durch die geschickte Projektion als Cubemap wirkt das Resultat für den Betrachter, als würde er sich in einer Kugel befinden (siehe Abbildung 5).

### 3.4.1 Konfiguration einer Sphäre

Zu jeder Sphäre existiert eine JSON-Datei, die die erforderlichen Informationen beinhaltet. Diese wird mit Hilfe von Grunt (Kapitel 2.4) aus der in Kapitel 2.3.2 gezeigten JSON-Datei generiert. So werden zum Beispiel die Pfade der Bilder ergänzt und die angegebenen Informationen geprüft. Der Vollständigkeit halber listet Tabelle 15 diese Spezifikation auf.

Um zum Beispiel die Koordinaten in den JSON-Dateien gut lesbar und nicht als Dezimalzahl angeben zu müssen, wird die `math.js`-Bibliothek verwendet. Diese wandelt Strings in Dezimalzahlen um und macht es so einfacher nachzuvollziehen, wo sich welche Objekte in der Sphäre befinden. In der Tabelle gibt die Spalte „math“ an, ob der hier eingetragene Wert mit `math.js` interpretiert wird. „Gefordert“ gibt an, ob dieser Eintrag existieren muss.

Attribut	Gefordert	math	Beschreibung
<code>id</code>	ja	nein	ID der Sphäre.
<code>belongsToMap</code>	ja	nein	Array mit Karten-IDs zu denen die Sphäre gehört. Hier muss mindestens eine Karte angegeben sein.
<code>initialView</code>	nein	nein	Initiale Blickrichtung beim Betreten der Sphäre von der Karte.
<code>.lat</code>	nein	ja	Breitengrad der initialen Blickrichtung. Standard ist 0.
<code>.long</code>	nein	ja	Längengrad der initialen Blickrichtung. Standard ist 0.
<code>name</code>	nein	nein	Verlinkung auf einen Namen mittels <code>i18next</code> .
<code>coords</code>	ja	nein	Koordinaten der Sphäre.
<code>.lat</code>	ja	nein	Breitengrad der Sphäre.
<code>.long</code>	ja	nein	Längengrad der Sphäre.
<code>images</code>	ja	nein	Zur Sphäre gehörende Bilder.
<code>.cubemap</code>	ja	nein	Zur Sphäre gehörende Cubemaps.
<code>.2048</code>	ja	nein	Cubemaps der Sphäre in der Auflösung 2048x2048. Hier muss für jedes in der Konfiguration angegebene Auflösung (siehe Kapitel 2.4.4) ein Eintrag existieren.

<b>Attribut</b>	<b>Gefordert</b>	<b>math</b>	<b>Beschreibung</b>
.size	ja	nein	Auflösung der Bilder. Muss mit dem Namen des Eintrages übereinstimmen.
.path	ja	nein	Pfad zu den Cubemaps. Unter dem angegebenen Pfad müssen dazu entsprechende Ordner ( $px$ , $nx$ , $py$ , $ny$ , $pz$ , $nz$ ) existieren, in denen jeweils ein Bild mit dem Namen $0.0.jpg$ liegt. Die Ordner stehen hierbei für die sechs Seiten der Cubemap, wobei $x$ , $y$ und $z$ die Koordinatenachsen beschreiben und $p$ (positiv) und $n$ (negativ) die Richtung angeben. Siehe auf Kapitel 3.4.2.
arrows	nein	nein	In eine andere Sphäre weiterführende Pfeile. Siehe Kapitel 2.3.3.
infos	nein	nein	Zusätzliche Informationen zu einem Punkt in der Sphäre. Siehe Kapitel 2.3.4.
objects	nein	nein	Beliebige Three.js Objekte.

Tabelle 15: Spezifikation des JSON-Formats einer Sphäre

In Programmcode 8 ist eine beispielhafte Konfiguration zu sehen.

```
{
  "id": "13bad6b1e478b951",
  "belongsToMany": [
    "44c2e9bdcaf4c29b"
  ],
  "initialView": {
    "lat": "30/90*PI/2",
    "long": "320/360*2*PI"
  },
  "name": "sphere.13bad6b1e478b951.name",
  "coords": {
    "lat": "54.337858",
    "long": "10.120956"
  },
  "images": {
    "cubemap": {
      "2048": {
        "size": 2048,
        "path": "images/spheres/13bad6b1e478b951/cubemap/2048"
      }
    }
  },
  "arrows": [
    {
      "long": "2*PI*138/360",
      "next_sphere": "d875d5d81bbbc835"
    }
  ],
  "infos": [
    {
      "long": "2*PI*270/360",
      "lat": "PI/2*-10/90",
      "content": "fs-infmath.html"
    }
  ],
  "objects": []
}
```

Programmcode 8: JSON-Beispiel einer Sphäre

### 3.4.2 Koordinaten in Three.js

Das Three.js-Framework verwendet ein rechtshändiges Koordinatensystem, welches in Abbildung 8 dargestellt ist. In dieser Abbildung ist zusätzlich die Position der Cubemap innerhalb dieses Koordinatensystems zu sehen, wobei der Würfel zur besseren Übersicht auseinandergezogen ist. Die Seiten des Würfels sind nach ih-

rer Position im Koordinatensystem benannt, so steht  $px$  für die Seite, die von der positiven x-Achse durchstoßen wird.

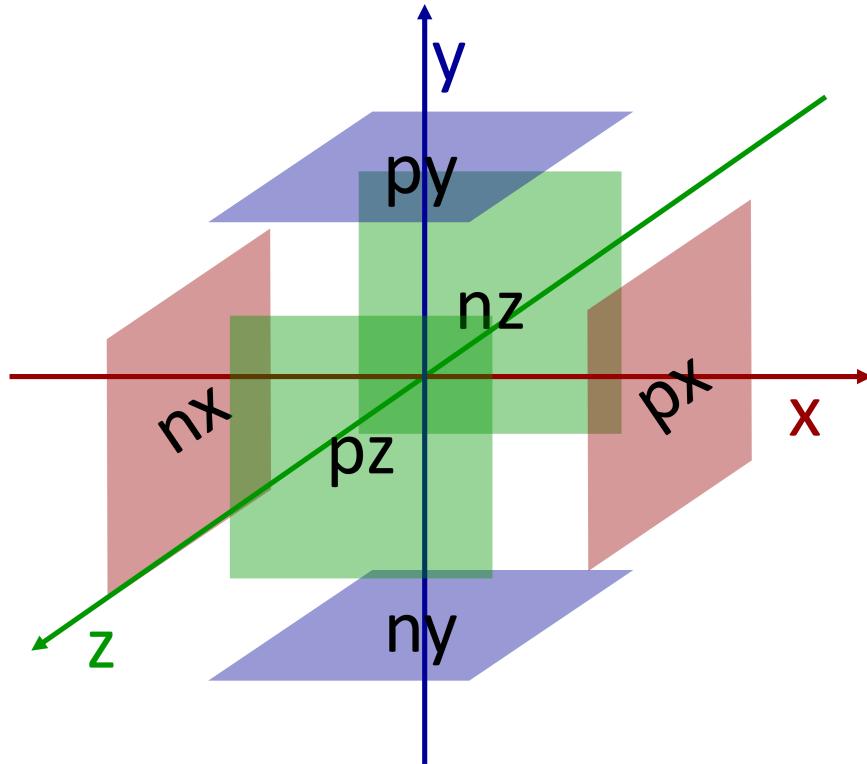


Abbildung 8: Ausrichtung des Koordinatensystems und der Cubemap

Die Kamera befindet sich in der Mitte des Koordinatensystems und ist somit von allen Seiten des Würfels gleich weit entfernt. Der Würfel hat eine Kantenlänge von 2000. Auf die Würfelseiten sind von innen die entsprechenden Texturen der Sphären gelegt, sodass durch die entsprechende Projektion der Eindruck entsteht, sich innerhalb einer Kugel zu befinden (Siehe Kapitel 3.4).

Da der Würfel als Kugel wahrgenommen wird, ist es einfacher für die Positionierung von Objekten (Pfeile, Infos, usw.) innerhalb des Würfels Kugelkoordinaten zu verwenden. Abbildung 9 zeigt die Angabe eines Punktes  $P$  im Koordinatensystem mit Hilfe von Kugelkoordinaten.

- Der `radius` gibt die Entfernung des Punktes vom Mittelpunkt an.
- Der Wert `lat` beschreibt den Winkel des Punktes zur x-z-Ebene, wobei der Winkel in Richtung y-Achse positiv ist.

- Der Wert long beschreibt den Winkel zur z-Achse, wobei nach links gemessen wird und keine negativen Werte zugelassen sind.

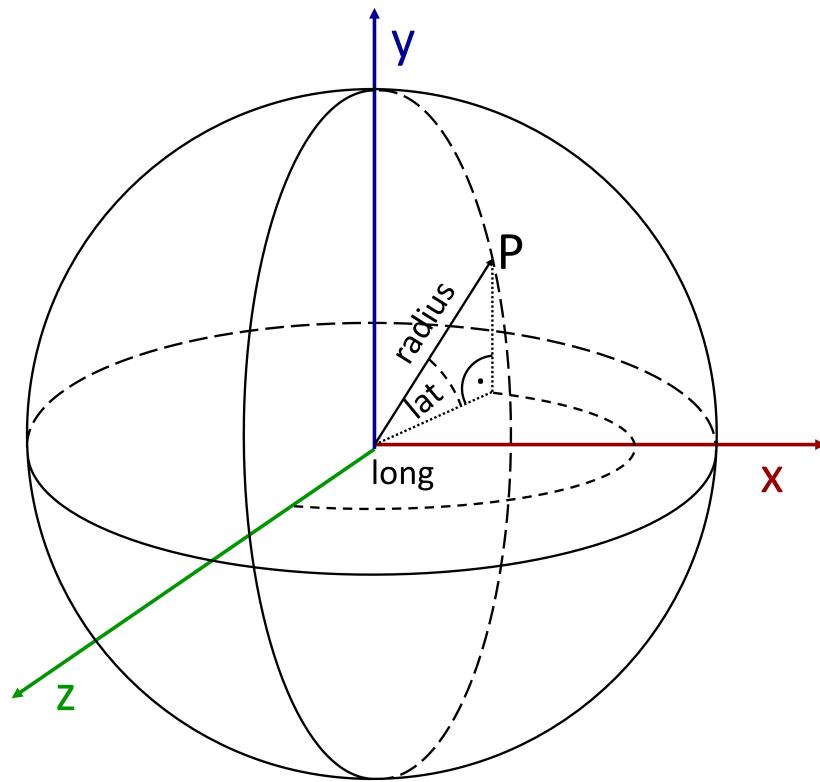


Abbildung 9: Beschreibung von Punkten mittels Kugelkoordinaten

### 3.4.3 Anzeigen der Koordinaten in einer Sphäre

Um das Einbinden von Objekten in einer Sphäre bei der Konfiguration zu erleichtern, existiert die Möglichkeit sich das Koordinatensystem anzeigen zu lassen. An diesem können die Winkel zur Positionierung einfach abgelesen werden. Abbildung 10 zeigt ein solches Koordinatensystem in einer Sphäre.

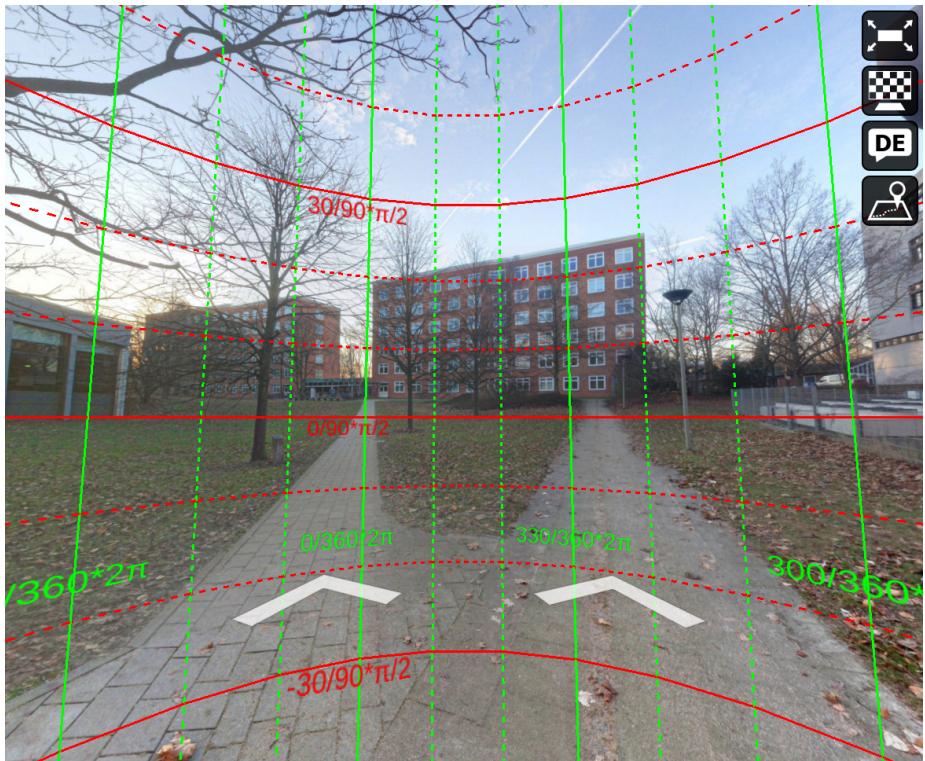


Abbildung 10: Koordinatensystem in einer Sphäre zum Ablesen von Winkeln

Das Koordinatensystem kann in der JavaScript-Konsole des Browsers mit dem Befehl `explore.sphere.sphere.showGrid();` angezeigt und mit `explore.sphere.sphere.removeGrid();` wieder entfernt werden.

### 3.5 Menü

In der rechten oberen Ecke befindet sich das Menü mit verschiedenen Auswahlmöglichkeiten. Hier kann der Nutzer in den Vollbildmodus und zurück wechseln, die Auflösung der Bilder ändern, die Sprache anpassen und – falls er sich in einer Sphäre befindet – zurück zur Kartenansicht wechseln. Abbildung 11 zeigt das Menü in einer Sphäre mit höchster Auflösung und deutscher Sprache.



Abbildung 11: Das Menü

### 3.6 Mehrsprachigkeit

Zur Umsetzung von Mehrsprachigkeit wird das i18next-Framework verwendet. Im Ordner raw/locales befinden sich Unterordner für verschiedene Sprachen. So zum Beispiel de-DE. In diesen Unterordnern befindet sich jeweils eine Datei translation.json, in der Begriffe in der jeweiligen Sprache angegeben sind. Programmcode 9 zeigt beispielhaft eine solche Datei.

```
{
  "menu": {
    "fullscreen": {
      "enter": "Vollbild",
      "exit": "Vollbild verlassen"
    },
    "sphere": {
      "13bad6b1e478b951": {
        "name": "Vor dem Fachschaftsraum"
      }
    }
  }
}
```

Programmcode 9: JSON-Beispiel einer i18next-Datei

Die Codezeile `$.i18n.t("menu.fullscreen.enter");` liefert nun – sofern auf deutsch eingestellt – den String „Vollbild“. Daneben können diese Werte in der Sphären-Konfiguration (siehe Kapitel 2.3.2) für das Attribut name eingetragen werden. So führt eine Eintragung von `"name": "sphere.13bad6b1e478b951.name"`

dazu, dass die Bezeichnung „Vor dem Fachschaftsraum“ als Beschreibungstext der Sphäre angezeigt wird.

Die für die Aktion `showPopup` benötigten `html`-Dateien (siehe Kapitel 2.3.5) müssen ebenfalls in allen Sprachen vorliegen. Dafür muss im Ordner `/public/infos` ein Unterordner für jede Sprache angelegt werden.

Mithilfe von Grunt (siehe Kapitel 2.4) werden die benötigten Dateien und Ordner automatisch angelegt, sofern im Ordner `/raw/markdown` entsprechende Unterordner für alle Sprachen und Markdown-Dateien vorliegen (siehe Kapitel 2.3.6).

## 4 Glossar

### Canvas

Ein Canvas ist ein HTML-Element, in das mit Hilfe von JavaScript gezeichnet werden kann.

### Express

Express ist ein Node.js Framework, welches einfache Werkzeuge für das Entwickeln von Webanwendungen bereitstellt. Siehe Kapitel 2.1.

### Grunt

Grunt ist ein Node.js-Framework zur Verwaltung von Aufgaben, die meist vor dem Serverstart ausgeführt werden. Beispiele sind das Kopieren oder Aneinanderhängen von Dateien oder auch das Generieren von benötigten Ressourcen. Die Aufgaben werden in der Datei `Gruntfile.js` verwaltet und mit dem Befehl `grunt` ausgeführt. Die Verwendung von Grunt im Virtual-Explorer wird im Kapitel 2.4 beschrieben.

### i18next

Mit Hilfe des i18next-Framework kann Mehrsprachigkeit realisiert werden. Im Ordner `public/locales` werden dazu Dateien im JSON-Format angelegt, die zu einem Attribut-Namen einen Text in verschiedenen Sprachen bereitstellen.

## ID

Ein Identifikator (ID) ist eine eindeutige Bezeichnung für ein Objekt. Im Beispiel des **Virtual-Explorers** hat jede Karte und Sphäre eine eindeutige ID, die eine 16-stellige Hexadezimalzahl ist.

## Jade

Jade ist die von Express bevorzugt verwendete Template-Engine. In Jade lassen sich HTML-Inhalte einfacher und kürzer formulieren. Da nicht eine XML ähnliche Syntax mit Klammerung verwendet wird, sondern eine auf Formatierung beruhende Darstellung, ist dies somit einfacher zu überblicken. Daneben erlaubt Jade das Einbinden anderer Jade-Dateien, womit zum Beispiel auf jeder einzelnen Seite leicht das selbe Grundlayout verwendet werden kann.

Um Jade vor dem Ausliefern an einen Browser in HTML zu übersetzen, wird das in Tabelle 1 aufgeführte `jade` Paket verwendet.

## JSON

JavaScript Object Notation (JSON) dient zum einfachen Austauschen und Abspeichern von Daten beziehungsweise JavaScript-Objekten.

Der **Virtual-Explorer** verwendet JSON für Konfigurationsdateien, vor allem zum Beschreiben der einzelnen Sphären (Koordinaten, zugehörige Bilder, Ausrichtungen, Verknüpfungen zu anderen Sphären, usw.) und Karten (Markierungen auf der Karte, Vorschaubilder, usw.).

## Less

Less ist eine Erweiterung von CSS und bietet damit mehr Funktionalitäten. Vor der Auslieferung an den Browser muss Less in CSS übersetzt werden. Diese Aufgabe übernimmt das in der Tabelle 11 aufgelistete Paket `grunt-contrib-less`.

## Math.js

Math.js ist eine JavaScript-Bibliothek, die es erlaubt, mathematische Ausdrücke die als String vorliegen, in Gleitkommazahlen zu übersetzen. So wird zum Beispiel  $\pi/2$  zu 1.570796 ausgewertet. Damit ist zum Beispiel die Konfiguration von Winkeln innerhalb einer Sphäre einfacher.

## **Node.js**

Node.js stellt grundlegende Netzwerk- und Serverkomponenten zur Verfügung und basiert auf der JavaScript V8 Engine, die ursprünglich für Google Chrome entwickelt wurde. Siehe Kapitel 2.

## **Rendern**

Erstellen des fertigen Bildes aus einer 3D-Szene. Eine Sphäre wird im Virtual-Explorer mit Hilfe von Three.js modelliert und muss zur Anzeige auf dem Bildschirm gerendert werden.

## 5 Referenzen

Der Quellcode der Anwendung befindet sich im Git des Institutes für Informatik unter folgender Adresse: <https://git.informatik.uni-kiel.de/mras/virtual-explorer>. Im Branch final liegt die hier beschriebene Version.

Verweise zu den verwendeten Paketen, Bibliotheken und anderer Software sind in Tabelle 16 aufgelistet.

Software	Link
Blender	<a href="https://www.blender.org">https://www.blender.org</a>
body-parser	<a href="https://www.npmjs.com/package/body-parser-json">https://www.npmjs.com/package/body-parser-json</a>
cookie-parser	<a href="https://www.npmjs.com/package/cookie-parser">https://www.npmjs.com/package/cookie-parser</a>
debug	<a href="https://www.npmjs.com/package/debug">https://www.npmjs.com/package/debug</a>
express	<a href="http://expressjs.com">http://expressjs.com</a> <a href="https://www.npmjs.com/package/express">https://www.npmjs.com/package/express</a>
grunt	<a href="http://gruntjs.com">http://gruntjs.com</a> <a href="https://www.npmjs.com/package/grunt">https://www.npmjs.com/package/grunt</a>
grunt-concurrent	<a href="https://www.npmjs.com/package/grunt-concurrent">https://www.npmjs.com/package/grunt-concurrent</a>
grunt-contrib-concat	<a href="https://www.npmjs.com/package/grunt-contrib-concat">https://www.npmjs.com/package/grunt-contrib-concat</a>
grunt-contrib-copy	<a href="https://www.npmjs.com/package/grunt-contrib-copy">https://www.npmjs.com/package/grunt-contrib-copy</a>
grunt-contrib-cssmin	<a href="https://www.npmjs.com/package/grunt-contrib-cssmin">https://www.npmjs.com/package/grunt-contrib-cssmin</a>
grunt-contrib-jshint	<a href="http://jshint.com">http://jshint.com</a> <a href="https://www.npmjs.com/package/grunt-contrib-jshint">https://www.npmjs.com/package/grunt-contrib-jshint</a>
grunt-contrib-less	<a href="http://lesscss.org">http://lesscss.org</a> <a href="https://www.npmjs.com/package/grunt-contrib-less">https://www.npmjs.com/package/grunt-contrib-less</a>
grunt-contrib-uglify	<a href="https://www.npmjs.com/package/grunt-contrib-uglify">https://www.npmjs.com/package/grunt-contrib-uglify</a>
grunt-contrib-watch	<a href="https://www.npmjs.com/package/grunt-contrib-watch">https://www.npmjs.com/package/grunt-contrib-watch</a>
Hugin	<a href="http://hugin.sourceforge.net">http://hugin.sourceforge.net</a>
i18next	<a href="http://i18next.com">http://i18next.com</a>
jade	<a href="http://jade-lang.com">http://jade-lang.com</a>

Software	Link
	<a href="https://www.npmjs.com/package/jade">https://www.npmjs.com/package/jade</a>
jQuery	<a href="https://jquery.com">https://jquery.com</a>
Leaflet	<a href="http://leafletjs.com">http://leafletjs.com</a>
Leaflet-canvasicon	<a href="https://github.com/sashakavun/leaflet-canvasicon">https://github.com/sashakavun/leaflet-canvasicon</a>
Leaflet.markercluster	<a href="https://github.com/Leaflet/Leaflet.markercluster">https://github.com/Leaflet/Leaflet.markercluster</a>
marked	<a href="https://www.npmjs.com/package/marked">https://www.npmjs.com/package/marked</a>
math.js	<a href="http://mathjs.org">http://mathjs.org</a> <a href="https://www.npmjs.com/package/math-js">https://www.npmjs.com/package/math-js</a>
mkdirp	<a href="https://www.npmjs.com/package/mkdirp">https://www.npmjs.com/package/mkdirp</a>
morgan	<a href="https://www.npmjs.com/package/morgan">https://www.npmjs.com/package/morgan</a>
node.js	<a href="https://nodejs.org">https://nodejs.org</a>
Panotools	<a href="http://search.cpan.org/~bpostle/Panotools-Script-0.25">http://search.cpan.org/~bpostle/Panotools-Script-0.25</a>
PhotoSphereViewer	<a href="https://github.com/JeremyHeleine/Photo-Sphere-Viewer">https://github.com/JeremyHeleine/Photo-Sphere-Viewer</a>
serve-favicon	<a href="https://www.npmjs.com/package/serve-favicon">https://www.npmjs.com/package/serve-favicon</a>
three.js	<a href="http://threejs.org">http://threejs.org</a>

Tabelle 16: Referenzen