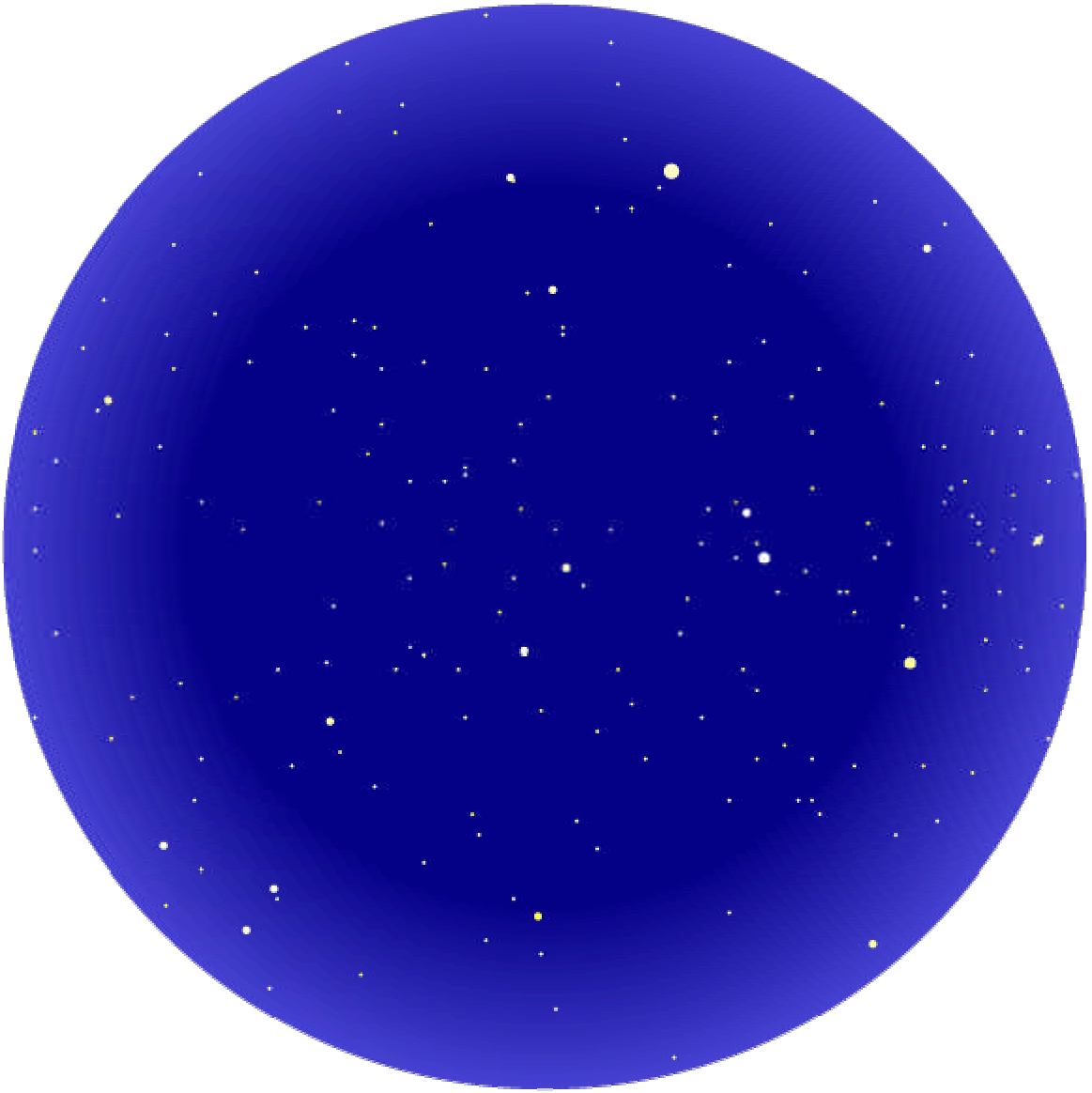
Interaktive Sternkarte fürs WWW

Diplomarbeit in Kartographie

Institut für Kartographie

ETH Zürich



Nico Hauri

Juli 2003

Autor:
Nico Hauri

e-Mail:
nico.hauri@gmx.ch
WWW:
http://www.karto.ethz.ch/nico/

Betreuung:
Prof. Dr. Lorenz Hurni, Zürich
Tobias Dahinden, Zürich

Zusammenfassung

Deutsch

Sterne begleiten uns auf Schritt und Tritt. Wir sehen sie zwar am Tag nicht, aber sie sind trotzdem hier. Aus diesem Grund möchten wir immer wieder gerne wissen, wie der Sternenhimmel aussieht, wenn wir ihn nicht sehen können. Ebenso möchten wir den Sternenhimmel für zukünftige oder vergangene Zeitpunkte oder für andere Orte kennen. Daher befassen wir uns mit Astronomie und versuchen, möglichst genaue Sternkataloge und Sternkarten zu erstellen. Um diese Arbeit für das WWW zu machen sind jedoch noch weitere Anforderungen nötig.

Zuerst müssen Sterndaten aus den Katalogen für Computer verständlich eingelesen werden. Danach folgen einige Berechnungen, um die Koordinaten der Sterne zu einem gewünschten Zeitpunkt und an einen gewünschten Ort zu transformieren. Diese neu errechneten Koordinaten werden zum Schluss für den Web-Browser verständlich ausgegeben. Dies soll so plattformunabhängig wie möglich geschehen.

Ebenfalls soll die Ausgabe für den Menschen lesbar sein. Dies geschieht normalerweise als bildliche Darstellung, die soweit als möglich dem „richtigen" Sternenhimmel ähnlich ist. Zusätzliche Informationen sollen zudem abrufbar sein. Als Versuch, dies auf eine andere Art als bisher zu tun, wurde diese Diplomarbeit geschrieben.

English

Stars attend us at every turn. Although we don't see them every day, they are always here. Therefore, we like to know at any time how the starry sky looks like, whether we see it or not. In addition, we want to know how the starry sky changes according to time and position. This is the main reason why we deal with astronomy and why we try to create precise card indexes and maps of stars. But in order be able to adapt this work on the Internet, other conditions have to be met.

Firstly, all the data material about the stars has to be imported from its sources by computers so that they can understand it. Next, some calculations follow in order to transform the coordinates of the stars at a certain point of time and a certain place. Finally, these newly calculated coordi-

nates should be emitted comprehensively by the Web-Browser. This process should happen as independent of the platforms as possible.

Furthermore, the result of the calculations has also to be legible for the human recipient, what is usually achieved by a figurative account that represents the real starry sky as best as possible. Moreover, additional information should be available. This degree dissertation attempted to find new ways of dealing with these transmission processes.

Inhaltsverzeichnis

1. Vorwort

Die Kombination von interaktiven WWW-Inhalten und Kartographie hat mich schon während der ganzen Studienzeit interessiert. Im Rahmen meiner Diplomarbeit an der ETH Zürich konnte ich mich nun mit dieser Materie vertieft auseinandersetzen. Zusätzlich kam noch die für mich ziemlich unbekannte Astronomie hinzu.

Für die Erstellung einer „Interaktiven Sternkarte" wurde volle Methodenfreiheit gegeben. Einzig einige Ziele waren schon vorgegeben. Die Arbeit, eine vollständige „Applikation" von Grund auf zu designen, programmieren und zu dokumentieren hat mir viel Spass und Freude bereitet. Neben der Vertiefung in für mich neue Themenkreise konnte ich zum ersten Mal eine eigene „Web-Applikation" erstellen und so sehr viel in diesem Bereich lernen.

Danken möchte ich Prof. Dr. Lorenz Hurni für den Vorschlag des Themas dieser Arbeit und für das Ermöglichen der intensiven Auseinandersetzung mit diesem Thema. Ebenso geht mein Dank an Tobias Dahinden für die Betreuung und die Unterstützung, die er mir jederzeit geboten hat.

Zürich, 18. Juli 2003
Nico Hauri

2. Problemstellung und Zielsetzung

2.1. Aufgabenstellung

„Es soll eine Webseite erstellt werden, auf der Sternkonstellationen von beliebigen Zeiten des 20./21. Jahrhunderts und Orten angesehen werden können. Die Karten können direkt aus einer Datenbank oder einem Textfile generiert werden" (Auszug aus der Aufgabenstellung). Um die Aufgabe besser zu verstehen, wird hier der Titel, der das wichtigste schon in sich aussagt, analysiert und erklärt.

Interaktive...

„Es soll für jede beliebige Zeit und jeden beliebigen Ort und für eine beliebige Zoomstufe eine Sternkarte erstellt werden können". Der User soll Ort und Zeit interaktiv eingeben können, um eine Sterndarstellung nach seinen Wünschen zu erstellen. Nicht nur voreingestellte Werte werden erwartet, sondern auch direkte Eingaben nach den Wünschen des Benutzers. Weitere Interaktionen wie Zoom und Pan (Verschieben des Ausschnitts) sind natürlich erwünscht.

...Sternkarte...

Das Hauptziel besteht darin, Sterne und zusätzliche Informationen darzustellen. Die Sichten sollen interaktiv an die eigenen Bedürfnisse angepasst werden. Um die Sterne zu allen gewünschten Zeitpunkten und Orten darstellen zu können, sind im Hintergrund einige Operationen und Umrechnungen durchzuführen.

„Die Karten müssen kartographischen Ansprüchen genügen." Dies beinhaltet sowohl die Benützung, wie auch das Gesamtbild. Es soll nicht einfach eine Karte generiert werden, sondern sie soll auch noch schön, einfach lesbar und ansprechend sein.

...fürs WWW

Da das WWW für praktisch jedermann zugänglich ist und die „Interaktive Sternkarte" auch möglichst vielen Leuten einen Nutzen bringen soll, setzt dies eine möglichst einfache und intuitive Benützung voraus.

Des weiteren heisst das (für den Autor auf jeden Fall), dass die Sternkarte plattformunabhängig nutzbar sein soll. Jeder, der Zugang zum WWW hat, sollte sie auch anschauen und nützen können. Das heisst, es sollten auch die Standards des Internets genützt werden. Dies stellt jedoch schon die ersten Probleme. Es würde viele verschiedene Möglichkeiten geben – nur eine kann aber implementiert werden – und das soll die beste sein (weitere Überlegungen zu diesem Thema im Abschnitt 4.2, Seite 17).

2.2. Stand der Dinge

Es gibt viele Arbeiten, Bücher und Programme, die sich dem Thema der Sterne widmen. Ebenfalls finden sich im Internet sehr viele Links zu diesem Thema. Was sie jedoch fast alle gemeinsam haben, ist, dass sie sich nur mit der Theorie befassen und sich selten jemand an die Darstellung der Resultate (für das WWW) gewagt haben. Anders sieht es natürlich bei den vielen Programmen aus, die offline auf dem lokalen Computer ausgeführt werden können.

Die Darstellung von solchen Informationen im WWW stellt sich jedoch als äusserst schwierig heraus. Die einfachste Variante ist, auf dem Server ein Bild (`*.jpg`, `*.png`) zu generieren und dieses dann zum Client zu schicken. Diese Methode ist jedoch nicht ganz zufriedenstellend, da jedes Mal ein ganzes Bild geladen werden muss. Ebenfalls sind mit dieser Methode Zoomen, Panen und andere Interaktionen sehr schwer vorstellbar – ausser mit einer erneuten Generierung eines Bildes. Diese Situation findet sich übrigens auch bei der Versorgung mit Landkarten und (Stadt-) Plänen im WWW.

Mit der Formulierung von SVG ist es dem W3C („World Wide Web Consortium“) [13] im Jahr 1999 gelungen, eine Darstellungssprache für das WWW zu erfinden, welche die Darstellung von 2D-Objekten sehr stark vereinfacht. Natürlich gab es schon zuvor Vergleichbares, das jedoch auf proprietären Standards beruhte und schwierig dynamisch herzustellen war. (vgl. Abschnitt 4.2 Seite 17)

Nach langer Suche ist ein einziges Projekt aufgefallen, welches dynamischen Inhalt auf einer Webseite darstellt. Das „SkyViewCafe“ [9] läuft als Java-Applet auf dem Client-Computer. Es bringt eine Hülle von Informationen und hat unzählige Einstellungsmöglichkeiten. Gleichzeitig bringt aber die Architektur als Java-Applet einige Nachteile mit sich. Zum Beispiel

muss für jede Java-Version das Applet neu kompiliert werden, was bei der Update-Kadenz von Java mit der Zeit ziemliche Probleme mit sich bringt. Kartographisch gibt es auch einige Nachteile. So sucht man in einer ziemlich funktionalen, aber lieblosen Darstellung vergebens nach Zoomwerkzeugen. Trotzdem steht mit dem „SkyViewCafe" eine gute und informative interaktive Sternkarte im WWW.

2.3. Zielsetzung

Aus den vorgängigen Überlegungen resultieren nun folgende Ziele:

- Standardisiert: Es sollen keine proprietären Standards und Formate genutzt werden. Es wird speziell darauf geachtet, möglichst breit abgestützte Formate zu nutzen (vgl. Kapitel 4, Seite 16).
- Einfach: Die Struktur der Skripte soll einfach, klar und durchsichtig sein. Es sollte somit möglich sein, die Struktur zu verstehen um sie selber auch anwenden und erweitern zu können.
- Benutzbar: Die Bedienung muss für Benutzer einfach und intuitiv erfolgen, damit möglichst viele Interessenten Nutzen an einer interaktiven Sternkarte haben können.
- Anpassbar: Darstellungen sind nie perfekt. Eine einfache Anpassung des GUI (Graphisches User Interface) sollte möglich sein (vgl. Abschnitt 4.3, Seite 18).
- Erweiterbar: Jedes Produkt, jedes Programm kann verbessert werden. Dies gilt auch für die vorliegende Arbeit. Also sollten mit möglichst geringem Aufwand weitere Funktionen implementiert werden können.
- Erneuerbar: Neue Sterndaten sollen auf dem einfachsten Weg „importiert" bzw. bereitgestellt werden können (vgl. Abschnitt 4.1, Seite 16).

Eine vollständige Beendigung der Arbeiten würden natürlich den Rahmen einer Diplomarbeit sprengen. Daraus folgt, dass es sich bei den Arbeiten an der „Interaktiven Sternkarte fürs WWW" um die Arbeit an einem Prototypen handelt, der nicht alle Funktionalitäten beherrscht. Einige Wunschvorstellungen müssen leider zurückgesteckt werden.

2.4. Aufbau der Arbeit

Die Struktur des Berichts folgt im Grossen und Ganzen den Arbeitsabläufen der Arbeit. Während der ganzen Arbeit wurde ein zielorientierter Ansatz gewählt. Das heisst, zu Beginn wurde die Aufgabe analysiert und eine

adäquate Zielsetzung gefunden. Einige Ziele wurden dabei schon als „Nice to have" deklariert. Der Programmcode wurde so strukturiert, dass diese fehlenden Funktionen ohne weiteres eingefügt werden können.

Das Studium der astrogeodätischen Grundlagen hat einige Zeit in Anspruch genommen. Für einen (anfänglichen) Laien sind diese Überlegungen ziemlich schwer zu verstehen. Zum Schluss mussten die benötigten Programmier- und Auszeichnungssprachen erlernt, und danach angewandt werden. Dies ging hingegen relativ leicht von der Hand.

3. Astronomische Grundlagen

3.1. Grundlagedaten

Es gibt einige Kataloge, in denen Daten zu vielen uns bekannten Sternen gesammelt werden. Für diese Arbeit wurde mit der sechsten Auflage des **Fundamentalkatalogs der Sterne** (FK6) [12] gearbeitet.

Der FK6 ist eine Kombination aus erdgemessenen und von Satelliten aus gemessenen Daten. Einige Koeffizienten werden ebenfalls aus dem Katalog „Hipparcos" (HIP) übernommen. Der FK6 besteht aus drei verschiedenen Teilen (I, II und III, wovon der Teil II noch nicht veröffentlicht wurde). Der Einfachheit halber wurde in dieser Arbeit nur mit dem ersten Teil gearbeitet. Dieser beinhaltet 878 „einfache Sterne". Die Teile II und III, welche die komplizierten Sterne (Doppelsterne, bewegliche Sterne u.ä.) enthalten, können jedoch ohne grossen Aufwand ebenfalls eingefügt werden. Der FK6 gibt die Parameter zur Zeit des Äquinoktium 2000.0 (siehe Abschnitt 3.2, Seite 11) an.

Da das Ziel des FK6 die wissenschaftliche, exakte Beobachtung der Sterne ist, werden für eine Aufgabe wie die Vorliegende nur sehr wenige der 86 angegebenen Parameter benötigt. Alle Koeffizienten für mittlere Fehler (Felder 9, 10, 12, 13), Parallaxe (14 bis 16), Beschaffenheit der Sterne (19 bis 22) und Unterschiede zwischen verschiedenen Messmethoden, Katalogen u.ä. (23 bis 85) müssen nicht berücksichtigt werden.

1	2	3	4	5	6	7	8	11	18
FK6 No.	HIP No.	Name	α h m s	δ ° ' "	μ_α mas/yr	μ_δ mas/yr	T_α	T_δ	m_V
904	122	θ Oct	0 1 35	-77 3 56	-56	-176	91.18	91.21	4.78

Tabelle 1: Auszug aus dem FK6

1: FK6 No.

Die Nummer des Sterns im FK6. Sie ist identisch mit der Nummer im Vorgänger FK5.

2: HIP No.

Die Nummer des Sterns im HIP-Katalog.

3: Name des Sterns

Der Name setzt sich zusammen aus einer Identifikation des Sterns (in der Regel der Helligkeit nach geordnet, von α bis ω, danach Zahlen und römische Buchstaben) und dem Sternbild, dem der Stern zugeordnet wird.

4: a 2000

Rektaszension (Abbildung 1) zur Epoche des Äquinoktium 2000.0. α wird vom Frühlingspunkt aus gemessen und in Stunden, Minuten und Sekunden angegeben.

5: d 2000

Deklination (Abbildung 1) zur Epoche des Äquinoktium 2000.0. In Grad, Minuten und Sekunden vom Äquator aus in Richtung der beiden Pole (Norden = Positiv, Süden = Negativ).

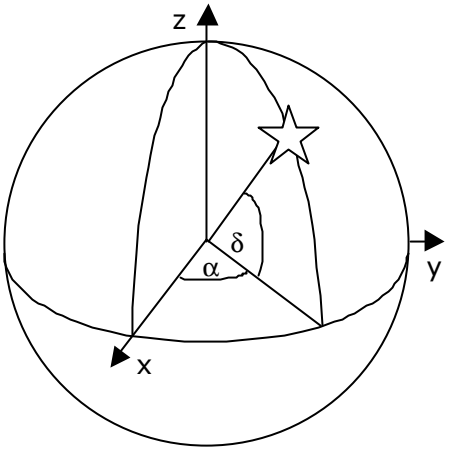


Abbildung 1: Rektaszension und Deklination

6 und 7: m_a 2000 und m_d 2000

Die Komponenten der Eigenbewegung in die Richtungen α und δ, in Millibogensekunden pro Jahr.

8 und 11: T_a und T_d

Die zentrale Epoche der Aufnahme. Mit 8, 11 und 6, 7 kann die Eigenbewegung des Sterns zwischen Bestimmung und dem „jetzigen" Zeitpunkt berechnet werden.

18: m_V

Die sichtbare Magnitude des Sterns (≈Helligkeit). Sie wird in dieser Arbeit für die Darstellung (≈Punktgrösse) benötigt. Ab einer Magnitude von ca. 6.5 verschwindet der Stern für das menschliche Auge.

3.2. Astrogeodäsie

Der Bereich der Astrogeodäsie wird in vielen Arbeiten dargelegt. Trotzdem werden hier die wichtigsten Berechnungsschritte von den gegebenen Parametern aus einem Sternkatalog bis zu den tatsächlich darzustellenden Werten wiederholt. Für genauere Angaben empfiehlt sich auch die Lektüre

des „Berichts 128 des Instituts für Geodäsie und Photogrammetrie der ETH Zürich" [10].

Äquinoktium, Ekliptik und Frühlingspunkt

Das **Äquinoktium** ist die Tag-undnachtgleiche [lateinisch]; Zeit des Durchgangs der Sonne durch die beiden Äquinoktialpunkte: Frühlingspunkt und Herbstpunkt, in denen Ekliptik und (Himmels-) Äquator sich schneiden (Abbildung 2). Von Bedeutung für die Astronomie ist das Äquinoktium im Frühling (daher auch „Frühlingspunkt").

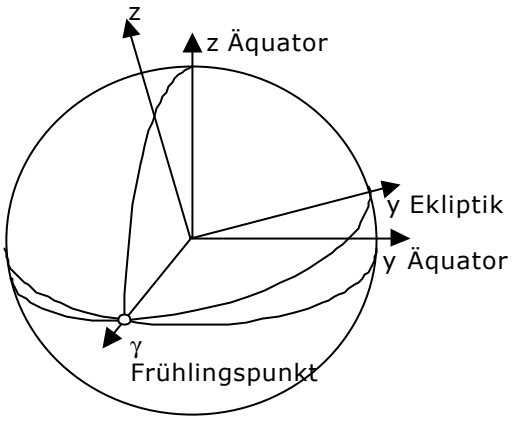


Abbildung 2: Frühlingspunkt

Als **Ekliptik** [griechisch] wird die scheinbare jährliche Bahn der Sonne an der Himmelskugel bezeichnet; ein grösster Kreis, der zum Himmelsäquator einen Winkel von ca. 23.5 Grad (Schiefe der Ekliptik) bildet und mit ihm die zwei Schnittpunkte (Äquinoktia) gemeinsam hat.

Der **Frühlingspunkt** wird definiert durch die Schiefe der Stehachse (⊥ Ekliptik) zur Rotationsachse (⊥ Äquator). Die Angaben im FK6 beziehen sich alle auf die Zeit T_0 = 2000.0, was der Zeit des Äquinoktium im Frühling 2000 entspricht.

Zeit

Die zeitlichen Komponenten werden in astronomischen Berechnungen immer in **Julianischen Daten** (JD) bzw. für Jahrhunderte in julianischen Jahrhunderten gerechnet. (bei den Angaben in eckigen Klammern werden nur die ganzen Zahlen benötigt.)

$$JD = 367J - \left[\frac{7 * (J + [(M + 9)/12])}{4}\right] + \left[\frac{375M}{9}\right] + T + 1721013.5 + \frac{(\text{Std in GM T}) + 12}{24} - \text{sign}(100J + M - 190002.5) + 0.5$$

Das **julianische Jahrhundert** berechnet sich aus dem JD der aktuellen Epoche ($T = JD_{jetzt}$) und dem JD der Standardepoche ($T_0 = JD_0$).

$$T_{Jhd} = \frac{JD - JD_0}{36525}$$

Eigenbewegung

Die Eigenbewegung der Sterne setzt sich zusammen aus der wirklichen eigenen Bewegung und der Bewegung der Sonne (mit den Planeten) zusammen. Sie besteht aus einem Teil in Rektaszension EB_α und einem Teil in Deklination EB_δ.

$$\alpha_{T_0}^{T} = \alpha_{T_0} + EB_\alpha * (T_{Jhd})$$

$$\delta_{T_0}^{T} = \delta_{T_0} + EB_\delta * (T_{Jhd})$$

Präzession und Nutation

Durch die Anziehungskräfte von Sonne, Mond und anderen Planeten auf die abgeplattete Erde wird diese leicht aus ihrer Bahn gelenkt. Dies verschiebt die Ekliptik aus der äquatorialen Lage und wird zu Teilen Präzession und zu Teilen Nutation benannt. Die Komponente der Präzession (Lunisolarpräzession) hat eine Periode von ca. 258000 Jahren mit einer Amplitude gleich der Schiefe der Ekliptik (˜23.5°). Dies wird überlagert von der kurzperiodischen Nutation des Himmelspols mit einer Periode von ca. 18.6 Jahren und einer maximalen Amplitude von 9''. Des weiteren wirken Planeten in der planetarischen Präzession auf die Erde. Diese Komponenten werden zusammengefasst in der allgemeinen Präzession.

Um diese Komponenten anwenden zu können, werden zwei Transformationen durchgeführt. Im ersten Schritt werden die Winkel von einem **mittleren Äquatorsystem zur Zeit T_0** (Aufnahmezeit des Sternkatalogs mit der Eigenbewegung korrigiert; MÄT0) zum **mittleren momentanen Äquatorsystem** (MÄT) durchgeführt. Danach wird die Transformation vom mittleren momentanen Äquatorsystem zum **wahren momentanen Äquatorsystem** (WÄT) durchgeführt.

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix}_{MÄT} = R_3(-z_A)R_2(\Theta_A)R_3(-\zeta_A)\begin{pmatrix} x \\ y \\ z \end{pmatrix}_{MÄT0}$$

$$\text{mit } (z_A, \Theta_A, \zeta_A) = F(T_0, T)$$

R_i sind dabei die Drehmatrizen um die Koordinatenachsen x_1, x_2, x_3.

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix}_{WÄT} = R_1(-\varepsilon_T - \Delta\varepsilon)R_3(-\Delta\psi)R_1(\varepsilon_T)\begin{pmatrix} x \\ y \\ z \end{pmatrix}_{MÄT}$$

$$\text{mit } (\varepsilon_T, \Delta\varepsilon, \Delta\psi) = F(T_0, T)$$

Nach diesen Transformationen haben wir nun 3D-Koordinaten der Sterne (x_1 = Richtung Frühlingspunkt, x_3 = Richtung Nordpol).

3.3. Projektion

Mit der Projektion der Sternkoordinaten sollen nun die 3D-Koordinaten auf eine 2D-Sicht „reduziert" werden. Dies muss in jedem Fall getan werden, ob die Karte auf Papier oder auf dem Bildschirm dargestellt werden soll. Abbildung 3 (nach Dutchs [3]) zeigt einige grundlegende Projektionsarten.

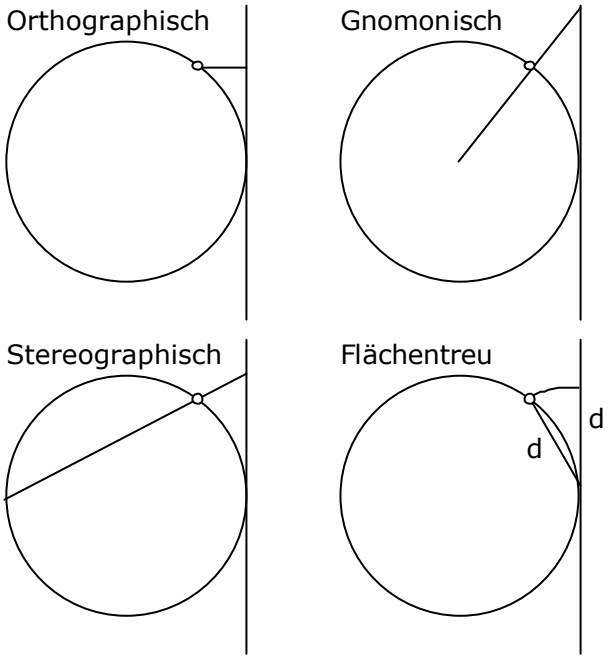


Abbildung 3: Beispiele Projektionen

Projektionen für dieses Anwendungsgebiet könnte man sich viele vorstellen. Jede Darstellung hat ihre Vor- und Nachteile. Fürs erste wurde nur eine Projektion, die stereographische, implementiert. Andere können problemlos später zugefügt werden.

Stereographische Projektion

Die stereographische Projektion stellt den gesamten Himmel als Kreis dar. In der Mitte kann man sich den Zenit vorstellen, am Rand ist der Horizont. Eine andere Vorstellung, die ebenfalls zur stereographischen Projektion führt, ist ein Blick durch ein Fernrohr.

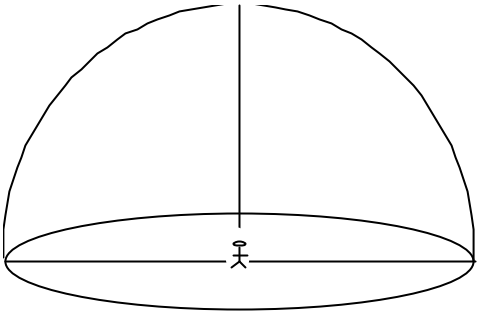


Abbildung 4: Betrachter in der stereographischen Projektion

Ein grosser Vorteil der stereographischen Projektion ist, dass Ellipsen in Ellipsen abgebildet werden und sie winkeltreu ist. Dies verfälscht das sichtbare Bild für den Betrachter am wenigsten.

In Abbildung 5 (nach Weisstein [11]) gelten S als Südpol, N als Nordpol, P' als wahrer Punkt, P als projizierter Punkt. Für Sternprojektionen gilt S = Zenit, N = Standpunkt. Daraus folgen sofort, dass Sterne mit $\overline{OS} < 0$ nicht projiziert werden dürfen (im „Schatten der Erde"). Bei $\overline{OS} \approx 0$ stehen die Sterne im Unendlichen (≈ auf dem Horizont). Es müssen nun noch alle gegebenen Formeln so angepasst werden, dass vorgängig eine Rotation stattfindet, damit der Zenitpunkt nicht gleich Südpol ist sondern in den „Standpunkt" gedreht wird.

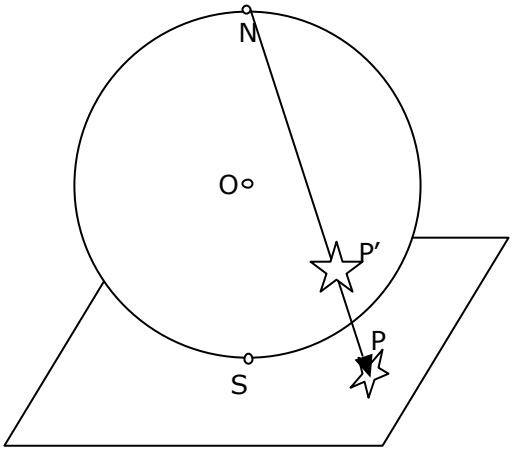


Abbildung 5: Stereographische Projektion

Als Berechnung gibt Burnett [4] an (ohne oben beschriebene Drehung):

$$X = \frac{x'}{1+z'} \text{ und } Y = \frac{y'}{1+z'}$$

für Nordpol (0,1,0), Südpol (0,-1,0) und Projektionspunkt (0,0,1). Auch diese Formeln müssen für die Berechnung in der vorliegenden Arbeit angepasst werden.

4. Technische Grundlagen

4.1. Skriptsprache PHP

PHP ist die Abkürzung für „PHP: Hypertext Preprocessor" und ist eine weit verbreitete Opensource Skriptsprache der „PHP Group" [8] für interaktive Webseiten. Die Struktur ist den bekannten (und älteren) Sprachen Perl, C, C++ und Java (-script) sehr ähnlich, und gehört wie diese auch zu den objektorientierten Programmiersprachen. Sie wird von der „Apache Software Foundation" [2] entwickelt und ist daher sehr eng mit dem Apache-Webserver verzahnt. Die Onlinedokumentation ist vortrefflich. Daten können mit PHP aus normalen Textfiles oder aus allen möglichen Datenbanken gelesen und geschrieben werden. Es sind sehr viele mathematische Funktionen implementiert und der Umgang mit mehrdimensionalen Arrays (Matrizen, Listen etc.) ist sehr einfach.

PHP wird als Skript geschrieben und dementsprechend vom Server interpretiert (nicht kompiliert). Dies hat den Vorteil, dass die Entwicklung leichter vonstatten gehen kann, da beim Testen auf das jeweilige Kompilieren verzichtet wird. Weil die Applikation auf dem **Server** abgearbeitet wird, ist es ein Leichtes, während dem laufenden Betrieb Änderungen vorzunehmen.

Natürlich ergeben sich aus dieser Struktur auch Nachteile. Zum Beispiel wird der Server bei starker Frequentierung mehr ausgelastet als bei einem Servlet oder gar einem Applet (kompilierte Programme, die entweder auf dem Server oder dem Client ablaufen), und es braucht für die Applikation einen Server (und somit für die meisten Benutzer einen Zugang zum WWW). Der Offline-Betrieb ist also stark erschwert.

Für alle grösseren Berechnungen wurde auch in der vorliegenden Arbeit PHP eingesetzt (auf die einzelnen Schritte wird in Kapitel 5, Seite 20 eingegangen).

Vorteile von PHP

Grundsätzlich gilt es zu sagen, dass sich PHP und z.B. Perl sehr ähnlich sind. Es gibt technisch kaum mehr Vorteile für das eine als für das andere. Einige Ideen dazu:

- PHP unterscheidet sich von anderen für CGIs verwendete Sprachen in einem Punkt sehr stark: Der Programmcode kann problemlos in eine Seite mit anderer Auszeichnungssprache eingebettet werden.
- PHP ist ziemlich neu und genau darauf ausgelegt, externe Dateien einzulesen, zu verarbeiten und wieder auszugeben. Es wird immer häufiger gebraucht um interaktive, dynamische Webinhalte zu produzieren.
- Die Programmierung nicht als CGI sondern als einfaches Skript vereinfacht die Struktur wesentlich.

Textfile oder Datenbank?

Nun sollen die Sterndaten aus einer Datenbank oder einem Textfile herausgelesen werden. An und für sich ist eine Datenbank zwar schnell mit Daten gefüllt und sie kann universeller gebraucht werden. Jedoch spricht für diese Aufgabe dagegen, dass dann auf dem Server zusätzlich eine Datenbank unterstützt werden muss. Da eines der Ziele aber war, die Struktur so einfach wie möglich zu behalten, wurde auf die Datenbank verzichtet. Ein Textfile kann einfach in die richtige Form gebracht werden und am richtigen Ort abgelegt werden – und schon arbeiten die Skripte mit den neuen Daten.

4.2. Auszeichnungssprache SVG

SVG steht für „Scalable Vector Graphics" und wurde 1999 erstmals vom „Worldwide Web Consortium" (W3C) vorgestellt [16]. Mit SVG können 2D-Graphikobjekte ziemlich einfach erstellt und dargestellt werden. Es werden Linien, Punkte und komplizierte Pfade (Bézier etc.), Flächen, Symboldefinitionen etc. unterstützt. SVG ist eine XML-Sprache und unterstützt dementsprechend Feinheiten von XML (fremde Namensräume wie CSS, Xlink u.ä.). Mit SVG ist eine standardisierte und starke Auszeichnungssprache für 2D-Vektor- und Rastergraphiken auf dem Markt. Einige Hersteller unterstützen diese Philosophie, und so gibt es schon einige SVG-Viewer (Stand Alone oder als Browser Plug-In). Mit ihnen können SVG-Dokumente sehr einfach angezeigt werden. Leider haben alle dieser Viewer ihre Mängel, so dass noch nicht die volle Syntax des W3C wirklich durchgehend gebraucht werden kann – einige Befehle werden ignoriert.

Vorteile von SVG

Warum wurde nun gerade SVG genutzt und nicht etwas wie Flash, womit ziemlich das gleiche machbar wäre? Es gibt verschiedene Gründe dafür, eine kleine Auswahl folgt hier:

- SVG ist ein offener Standard. Für das offene WWW sollten (meiner Meinung nach) auch offene Standards genutzt werden.
- Es ist sehr viel einfacher, per PHP o.ä. ein SVG zu schreiben. Dies liegt daran, dass SVG ganz normal im Texteditor geschrieben werden kann (ASCII), Flash und andere brauchen einen eigenen Editor und werden dann als Binär-Files gespeichert. Diese können nicht so einfach ausgegeben werden.
- Objekte lassen sich in „Gruppen" zusammenfassen, was vergleichbar ist mit Layern oder Frames in Graphikprogrammen. Diese Gruppen können als ganzes mit Styles versehen, verschoben, skaliert werden usw.
- In SVG lassen sich vorgängig verschiedene Definitionen wie Symbole, Verläufe oder Animationen machen, auf die später per `<xlink>` und/oder `<use>` verwiesen und zugegriffen werden kann. Dies kann in kartographischen Darstellungen eine grosse Vereinfachung bringen.
- Mit Hilfe von **Viewboxes** können sehr einfach für Teile des SVG eigene, interne Koordinatensysteme definiert werden. Dies ist natürlich für kartographische Anwendungen sehr interessant. Es müssen keine Umrechnungen der Masse vorgenommen werden, sondern es werden einfach die Masse angepasst. SVG bietet sich auch aus diesem Grund für die Kartographie sehr gut an.

4.3. Darstellungssprache CSS

„Cascading Style Sheets" (**CSS**) sind ebenfalls ein Vorschlag des W3C zur Ergänzung von HTML, XHTML und XML (inkl. SVG) [14]. Es dient dazu, relativ einfach Style für Web-Dokumente Global zu definieren. So können Farben, Schriften, Schriftgrössen und -aussehen, Strichstärken etc. definiert und sehr einfach den Objekten zugewiesen werden. So kann die Darstellung von (Web-) Dokumenten mit wenigen Änderungen schnell global für die ganze Web-Site geändert werden. Ein gutes und bekanntes Tutorial wurde von Stefan Münz [6] ins Internet gestellt.

4.4. Skriptsprache JavaScript

JavaScript wurde ursprünglich von Netscape als plattformübergreifende, objektorientierte Skriptsprache erfunden. Es wurde konzipiert, um in anderen Produkten eingebettet auf deren Objekte zuzugreifen. Es wurde in Anlehnung an die Struktur und Syntax von Java zusammengestellt, sollte jedoch einfacher, kleiner und flexibler sein. Zusammen mit dem DOM (siehe Abschnitt 4.5, Seite 19) wird JavaScript zu einem sehr starken Werkzeug für interaktive Seiten und Dokumente. JavaScript ist eine interpretierte, nicht kompilierte, Scriptsprache, die beim Aufruf jeweils nur die benötigten Funktionen neu interpretiert.

Die meisten SVG-Viewer nützen (wie schon die meisten WWW-Browser) JavaScript für Interaktionsmöglichkeiten. Am vollständigsten wurde JavaScript im SVG-Plug-In von Adobe implementiert.

4.5. Dokumentenmodell DOM

Das „Document Object Model" (**DOM**) ist eine plattform- und sprachneutrale Beschreibung, die Programmen und Skripten den Zugriff auf jeden Inhalt, jede Struktur und die Styles eines Dokuments erlaubt [15]. Es können sowohl Änderungen vorgenommen, wie auch Objekte erstellt oder gelöscht werden. Damit kann Interaktivität durch jede Programmier- oder Skriptsprache in ein DOM-fähiges Dokument gebracht werden. XHTML, XML und somit auch SVG unterstützen das DOM. DOM wurde ebenfalls vom W3C entwickelt und publiziert. Grundsätzlich könnte mit jeder objektorientierten Skript-/Programmiersprache auf das DOM zugegriffen werden, mit JavaScript geht dies jedoch am einfachsten.

5. Aufbau des Prototypen

5.1. Architektur

Für die Bewältigung der Aufgabe werden sowohl Fähigkeiten auf dem Server wie auch auf dem Client gefordert. Dies ist gegeben durch die Anforderung „Generierung der Sternkarte on demand" – auf Anfrage.

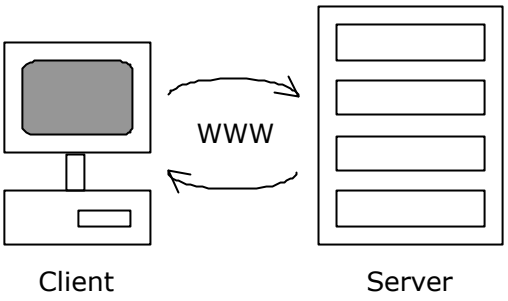


Abbildung 6: Client und Server

- SVG und JavaScript verlangen einen Client, der imstande ist, diese zu interpretieren.
- Das Auslesen der Sterndaten aus einem Textfile erfordert PHP oder eine andere serverseitige Skriptung/Programmierung.
- Leider setzt dies eine Leitung zum WWW voraus (oder einen laufenden Server mit PHP-Fähigkeiten auf dem eigenen Computer).

Diese Struktur vereinfacht ebenfalls die Handhabung mit den grossen Datenmengen der vielen Sterne, welche nämlich schon auf der Serverseite gefiltert werden können auf diejenigen Sterne, die auch wirklich für die angefragte Sicht vonnöten sind.

In den beiden nächsten Kapiteln werden die einzelnen Berechnungsschritte und Funktionen erläutert. Eine Übersicht über die Struktur der Quelltexte findet sich im Anhang 3.

5.2. Server

Auf dem Server werden die grösseren Berechnungen mit PHP gemacht. Gleichzeitig werden die Sterndaten gefiltert, so dass nur die für die angeforderte Sicht relevanten Daten fixfertig zum Client geschickt werden. Dies reduziert die Datenmenge auf rund die Hälfte der Sterne (jeweils nur die angeforderte „Hemisphäre"). Da die Sterne fast die gesamte Datenmenge ausmachen, ist dies sicher eine günstige Vorgehensweise. Im vorliegenden Fall wird dadurch die Datenmenge auf 200 kB reduziert.

Des Weiteren rechnet natürlich der Server meist schneller als ein Client. Weil die angeforderte Sicht fixfertig vom Server berechnet und als SVG ge-

schrieben wird, wird auch die Rechenzeit verkürzt. Die Geschwindigkeit der neuen Darstellung wird also nur noch durch die Verbindungsgeschwindigkeit zwischen Client und Server vorgegeben.

Konfiguration

stern.php

Als eine Art „Main" funktioniert die Datei `stern.php`. Sie wird aufgerufen, um die Sternkarte zu generieren. In ihr werden zuerst alle anderen Skriptdateien eingebunden, danach werden die einzelnen Berechnungsskripte aufgerufen. Da sie sozusagen als Steuerzentrale für die Erstellung fungiert, können in ihr auch verschiedene Variablen definiert werden: `$datei` für die einzulesende Sterndatei und `$svg` für verschiedene SVG-Variablen.

definitionen.php

Einige Funktionen werden immer wieder verwendet. Diese werden vorgängig definiert, um sie danach immer wieder aufrufen zu können. Mit dieser Methode wird der Quelltext der anderen Funktionen sehr viel übersichtlicher und Fehler in Berechnungsschritten müssen nur an einem Ort geändert werden. Man könnte zu diesem Zweck noch einige dieser Berechnungsschritte aus anderen Funktionen auslagern.

Übernahme der Parameter

parameter.php

Für die Berechnungen aller weitern Schritte, müssen als erstes die Übergabeparameter der Webseite übernommen werden. Grundsätzlich gibt es dafür zwei Methoden (`POST` und `GET`). Diese zwei Methoden bewirken zwar das gleiche, werden aber anders übertragen. Mit `GET` werden die Parameter über die Adresszeile des Browsers (`dokument?parameter1=wert1&parameter2=wert2` etc.) übertragen. Dies macht die Parameter für den Benutzer sichtbar und vereinfacht die Struktur. Zum Beispiel könnte man mit SVG keine `POST`-Variablen abschicken, was hingegen mit `GET` sehr einfach geht (der Link beinhaltet die Parameter). Ausserdem kann so einfacher von anderen Webseiten direkt ein gewünschter Ort zu einer bestimmten Zeit aufgerufen werden.

Diese Parameter werden nun vom PHP-Skript auf dem Server eingelesen und für die weitere Verarbeitung bereitgemacht. Leer- und andere störende Zeichen werden entfernt und danach die Daten in Arrays gespeichert.

`$zeit['sec','min','std','day','mon','yr','tz']` = Sekunde, Minute, Stunde, Tag, Jahr, timezone (Zeitverschiebung zu GMT)

`$ort['ld','lm','we','bd','bm','sn','name','land','kontinent']` = Länge (Grad und Minute) und West/Ost, Breite (Grad und Minute) und Nord/Süd; Name, Land und Kontinent des Ortes

`$other['projection','ausgabe']` = Projektion, Ausgabemedium (`screen`, `pda`)

Umrechnungen der Parameter

`umrechnungen.php` – `ort_neu()`, `zeit_neu()`, `other_neu()`

Die eingelesenen Parameter müssen nun auf ihre Richtigkeit überprüft werden. Einerseits kann man sich hier unmögliche Parameter vorstellen (z.B. der 35.12.2004), andererseits leere Eingaben. Bei fehlerhaften Übergaben werden die Eingaben vom Skript bereinigt. Das heisst zum Beispiel überzählige Stunden werden den Tagen angerechnet, überzählige Minuten den Studnen etc. In den Eingabemöglichkeiten der „Interaktiven Sternkarte" werden leere Eingaben zwar verunmöglicht, beim erstmaligen Aufruf der Seite sind jedoch noch alle Felder leer. In diesem Fall wird die aktuelle Serverzeit, als Ort Zürich (Standort der ETH Zürich) und als Projektion eine stereographische Projektion voreingestellt.

Zum Schluss werden noch einige Vereinfachungen für die Berechnungen vorgenommen. Datum und Uhrzeit werden zu einem Julianischen Datum und in einen Unix Timestamp umgerechnet. Aus den Grad-, Minuten- und Hemisphärenangaben werden absolute Winkel berechnet (W-O beginnt im 0-Meridian und geht 360° um die Erde, N-S beginnt im Nordpol und geht 180° bis zum Südpol).

Einlesen der Sterndaten

einlesen.php – sterne_einlesen($datei,$trennzeichen)

fk06	hip	name	alpha_h	alpha_m	alpha_s
904	122	Theta Oct	0	1	35.7029
1630	154	30 Psc	0	1	57.6198
905	301	2 Cet	0	3	44.3881
1001	377	45 G. Tuc	0	4	41.3082
1003	560	9 G. Cet	0	6	50.0860
4	841	22 And	0	10	19.2464
6	950	theta Scl	0	11	44.0095
1655	1007	o Oct	0	12	33.9700

Tabelle 2: FK6, umformatiert

Die in der Variablen `$datei` gespeicherte Textdatei wird nun eingelesen. Die Felder in der Datei müssen mit dem in `$trennzeichen` gespeicherten Zeichen getrennt sein, z.B. wie vorgegeben ein „|" (vgl. Tabelle 2). Es werden vorerst alle Felder eingelesen. Danach werden die Spaltennamen in der ersten Zeile als Namen für die jeweiligen Felder eingesetzt. So kann später am einfachsten auf die richtigen Parameter zugegriffen werden. Gleichzeitig werden nun auch alle Spalten gelöscht, die für diese Arbeit nicht relevant sind (siehe auch Abschnitt 3.1, Seite 10). Die Sterne stehen nun in einem zweidimensionalen Array (Anzahl Zeilen = Anzahl Sterneinträge) mit dem Namen `$sid`.

Astronomische Berechnungen

astro.php – winkel()

Alle folgenden Berechnungen verändern nur den Array `$sid`. Schritt für Schritt werden die astronomischen Umrechnungen vorgenommen (siehe Abschnitt 3.2, Seite 11). Nach jedem Berechnungsschritt werden die nicht mehr benötigten Spalten aus dem Array gelöscht, um so den benötigten Speicherplatz möglichst klein zu halten.

astro.php – zeitkorrektur()

Nach der Umrechnung der Winkel wird die Zeitkorrektur (Eigenbewegung der Sterne) angebracht. Hier gilt zu beachten, dass das Aufnahmedatum (t_delta und t_alpha), um Platz zu sparen, im FK6 (meist) ohne vorangehendes Jahrhundert (1900) geschrieben werden.

Nach dieser Rechnung haben wir nun die Winkel, unter denen die Sterne zum „momentanen Zeitpunkt" T_1 stehen. Diese Winkel (α, Rektaszension

und δ, Deklination) sind noch immer relativ zum Frühlingspunkt, bzw. zum Äquator. Deshalb werden hier die Deklinationen umgerechnet (vom Nordpol aus 180° zum Südpol). Dies vereinfacht weitere Berechnungen, da keine Winkel mehr negativ sind.

astro.php – xyz()

Da die weiteren Berechnungen nicht mehr mit Winkeln, sondern mit Kugelkoordinaten (x, y, z) durchgeführt werden, müssen die Deklinationen und Rektaszensionen im folgenden umgeformt werden unter der Annahme von r = 1 (Einheitskugel).

astro.php – praezession(), maeto_maet()

Nun folgt die erste Transformation vom **mittleren Äquatorsystem der Epoche T_0** (Aufnahmezeit) (MÄT0) in das **mittlere momentane Äquatorsystem** (Zeit „T"; MÄT). Dazu müssen zuerst die Präzessionsparameter berechnet werden (`praezession()`, `$praezession`), welche von der Zeitdifferenz T_0 bis „jetzt" abhängig sind.

astro.php – anomalie(), maet_waet()

Als weitere Korrektur wird die Transformation vom **mittleren momentanen Äquatorsystem** in das **wahre momentane Äquatorsystem** (WÄT) angebracht. Diese Transformation ist abhängig von den Anomalien, Distanzen und Entfernungen von Sonne, Mond und Erde, welche vorgängig berechnet werden (`anomalie()`, `$anomalie`). Auch hier sind diese Parameter nur von der Zeitdifferenz abhängig. Die Koordinaten sind nun in den Feldern `$sid[][x_korr]`, `$sid[][y_korr]`, `$sid[][z_korr]` gespeichert.

Projektion

projektionen.php – projektion($projection)

Die Funktion `projektion()` erwartet als Übergabe die Angabe über die Art der Projektion. So könnte man ohne weiteres noch mehr Projektionen implementieren, ohne die Struktur der Skripte gross ändern zu müssen. Aus Zeitgründen wurde bisher erst die stereographische Projektion gemacht.

Nach der eigentlichen Projektion wird hier ebenfalls eine grosse Umstrukturierung des Arrays `$sid` gemacht. Alle Sterne, die „hinter dem Betrachter" stehen, wo also für seine Sicht die Erde im Weg ist, werden hier gelöscht. Dies hat den Vorteil, dass im später geschriebenen SVG nur die

wirklich relevanten Sterne drin sind, das heisst also, dass so die Datenmenge, welche zum Client geschickt wird, um fast die Hälfte reduziert wird. Dieser Schritt sollte bei der Implementation von weiteren Projektionen nicht vergessen werden.

SVG schreiben

svg.php – svg($ausgabe,$projektion), animate($stern)

Nun sind alle Berechnungsschritte getan. Es bleibt nur noch, die Darstellung, bzw. in diesem Fall das SVG zu schreiben. Die Funktion `svg()` erwartet die Übergabe des Parameters `$ausgabe`, worin gesteuert wird, auf welche Art das SVG geschrieben werden soll. Momentan werden als Ausgabegrössen `screen` und `pda` unterstützt. Ebenfalls Einfluss auf die Darstellung hat die Art der Projektion, welche als Parameter `$projektion` übergeben wird.

body.php

Es ist relativ schwierig, in PHP den SVG-Code zu schreiben, noch schwieriger ist es, den Quellcode sauber zu schreiben. Dazu kommt auch, dass im Code immer wieder PHP-Variablen gebraucht werden. Dies führt schnell zu unübersichtlichem Code. Sowohl PHP wie auch SVG benützen einfache oder doppelte Anführungszeichen, die jedoch bei PHP anders interpretiert werden (Code in doppelten Anführungszeichen wird interpretiert, Code in einfachen wird eins zu eins übernommen). Hier muss exakt darauf geachtet werden, dass die richtigen Anführungszeichen gebraucht werden – und mit den gleichen die Angabe wieder geschlossen wird. Es gibt sonst schnell ein Durcheinander zwischen SVG- und PHP-Code, der zu nicht darstellbaren Ergebnissen führt.

Etwas verwirrlich ist es, dass keine Datei ein eigentliches SVG ist. Zu finden sind nämlich vornehmlich `*.php`'s. Um verschiedene Versionen zu vereinfachen und dynamische von nichtdynamischen Inhalten zu trennen wurden ganze SVG-Codeteile in PHP-Variablen gespeichert (im File `body.php`), die dann nacheinander aufgerufen und geschrieben werden. Hier liessen kleine PHP-Routinen noch eine grosse Vereinfachung des Quellcodes zu.

`defs.php`, `svgcss.css`

SVG lässt die Vordefinition von Symbolen (zu finden in `defs.php`) und Styles (`svgcss.css`) zu. Natürlich wurde auch in dieser Arbeit auf diese Feinheiten zurückgegriffen. Übrigens hat dies noch einen weiteren Vorteil: die HTML-Webseite kann auf dasselbe `*.css` zurückgreifen...

Um die Verknüpfung der Hauptkarte und der Übersichtskarte zu gewährleisten (siehe auch Abbildung 12, Seite 30), werden im SVG sogenannte **Viewboxes** definiert. Mit Hilfe dieser kann im SVG für einen bestimmten Teil ein eigenes, neues Koordinatensystem definiert werden. Da diese Systeme bei der Hauptkarte und bei der Übersichtskarte (bzw. die Koordinaten des Ausschnittszeigers in der Übersichtskarte) dieselben sind, wird Zoomen und Panen sehr einfach zu implementieren (siehe Abschnitt „Zoom und Pan", Seite 28).

5.3. Client

Als Client wird eine Art einfacher „Thick Client" vorausgesetzt (Thick Client = Client mit, Thin Client = Client ohne eigene Möglichkeiten). Ein reiner WWW-Browser reicht für die Darstellung der „Interaktiven Sternkarte" meist nicht, da bisher noch kein Browser von Haus aus SVG darstellen kann. Es ist also ein SVG-Plug-In (oder mit einigen Umwegen ein externer SVG-Viewer) nötig, was jedoch nur einmal heruntergeladen und installiert werden muss. Hier gibt es verschiedene Varianten, wobei der SVG-Viewer aus dem Hause Adobe [1], mittlerweile in der Version 3.0, die vollständigste Implementation des SVG-Standards besitzt.

SVG unterstützt von sich aus keine Texteingaben und Formulare. Es würde zwar eine Möglichkeit geben, Texteingaben zu übernehmen (gesteuert mit JavaScript-Events), diese Möglichkeit ist jedoch sehr aufwändig und braucht einige Zeilen an Code. Um dies zu umgehen, wurde die Eingabe von Koordinaten und Zeitangaben über den kleinen Umweg von Schaltflächen gelöst. Dies verringert den Programmier- und Datenaufwand erheblich.

Datum und Zeit

main.js

Die Eingabe von Zeit und Datum über kleine Schaltflächen ist sehr einfach (Abbildung 7). Als Vorgabe beim ersten Starten wird die aktuelle Serverzeit übernommen. Weiter muss die Zeitverschiebung zur GMT (Greenwich Meantime) angegeben werden. Ein kleines Problem dabei ist, dass dabei die Sommerzeit nicht berücksichtigt wird. Dies verfälscht die Eingabe um bis zu 1666 km (am Äquator). Es muss jedoch auch gesagt werden, dass diese Methode sowieso nicht genau ist. Die genauste Methode wäre, die Zeitverschiebung direkt über die Koordinateneingabe (W-O-Verschiebung) zu berechnen (vgl. Abschnitt „Zeiteingabe", Seite 31).

Abbildung 7: Einstellungen für Zeit und Datum

Bei der Eingabe wird über einige JavaScript-Funktionen überwacht, dass keine unmöglichen Zeiten und Daten eingegeben werden. Über ein Button kann die gerade aktuelle Zeit des Client-Computers eingestellt werden.

Ort

main.js

Um einen Ort eingeben zu können, gibt es zwei verschiedene Möglichkeiten (Abbildung 8). Entweder werden die gewünschten Koordinaten direkt über die Schaltflächen eingegeben.

Abbildung 8: Einstellungen für den Ort

koord.js, selectionList.js

Die zweite Möglichkeit ist, über die Auswahllisten (`selectionList.js` [7]) einen vordefinierten Ort auszuwählen. Über die Auswahl des Kontinenten wird die jeweilige Auswahlliste per Skript eingeschaltet (`visibility="visible"`) und für die weitere Auswahl angezeigt. Koordinaten und Zeitverschiebung zu GMT werden aus `koord.js` gelesen und automatisch angepasst.

Zoom und Pan

main.js

Das Zoomwerkzeug (Abbildung 9) ist sehr einfach gemacht. Mit einem Klick kann man hier den Sichtwinkel (View) verändern. Dank JavaScript und DOM ist dies sehr einfach zu implementieren. Die **Viewbox** (siehe Abschnitt „SVG schreiben", Seite 25) des Hauptfensters wird einfach mit den gleichen Zahlen bestückt wie die Koordinaten des Ausschnittzeigers. Auf dieselbe Art funktioniert grundsätzlich das Panen. Zu den Originalkoordinaten wird die relative Verschiebung des (gedrückten) Mauszeigers gezählt. Fehlend hier ist das adaptive Zooming (siehe auch Abschnitt „Zoom", Seite 32).

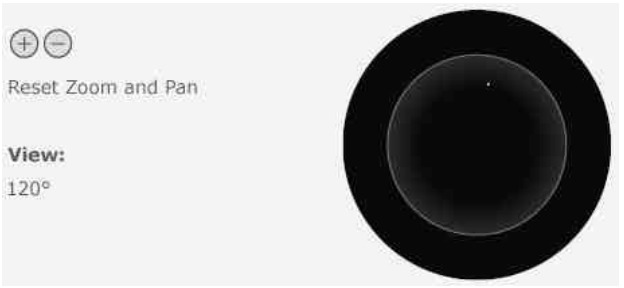


Abbildung 9: Zoom und Pan

Informationen

main.js

Im Informationsteil werden allgemeine Informationen über die aktuelle Projektion, Zeit und Datum der Sternkonstellation und den Ort angezeigt (Abbildung 10). Beim Überfahren von Sternen mit dem Mauszeiger wird zusätzliche Information über den Stern angezeigt. Momentan beinhaltet dies erst den Namen und das Sternbild. Denkbar wären hier noch weitere Informationen wie Winkel (Deklination und Rektaszension), unter denen der Stern gesehen wird, Helligkeit, Distanz etc. Dies würde einige weitere Codezeilen und einen vollständigen Sternkatalog voraussetzen.

Projection:	Stereographische Projektion
Time:	7.7.2003, 9:20 (GMT1)
Locality:	Zuerich, CH
Star:	gamma UMa

Abbildung 10: Informationen

Eingabe von Orten

`locality.php, dynkoord.js`

Die Auswahlliste der Orte ist natürlich erweiterbar. Über ein HTML-Eingabeformular (Abbildung 11) kann der User Orte und ihre Zusatzinformationen eingeben. Dies erscheint in einer eigenen Auswahlliste.

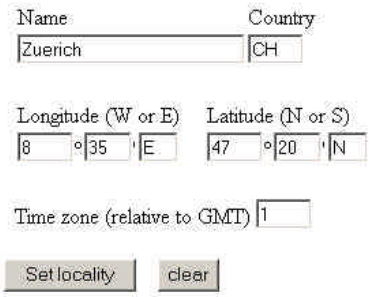


Abbildung 11: Orteingabe

Karte generieren

`main.js`

Sobald eine Einstellung bezüglich Ort oder Zeit gemacht wird, erscheint ein neuer Link („Click here to update your settings...!"). Die Einstellungen werden nicht sofort sichtbar, sondern müssen vom Server bearbeitet werden. Um die Parameter korrekt zum Server zu schicken wird der Link dynamisch mit den getätigten Einstellungen erstellt. Dies ersetzt sozusagen die Formularfunktion und die Methode `GET`.

6. Ergebnisse und Schlussfolgerungen

6.1. Ergebnis

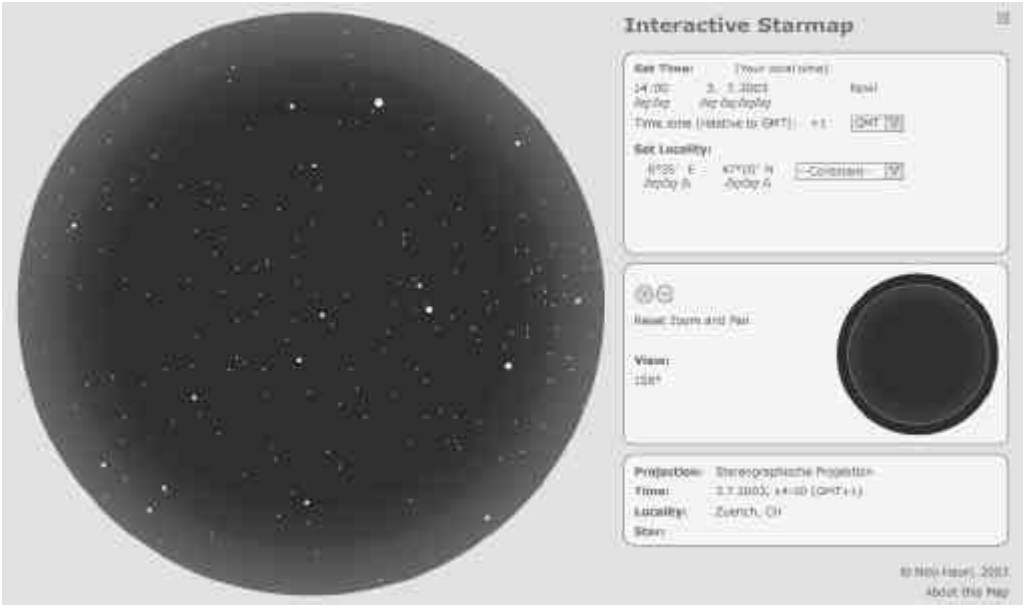


Abbildung 12: „Interaktive Sternkarte"

Nach einer Browserabfrage [1], bzw. im Microsoft Explorer einem eventuellen Download des SVG-Plug-Ins, gelangt man auf die Seite mit der „Interaktiven Sternkarte" (siehe Abbildung 12, bzw. Anhang 1). Sie wird auf Anfrage vom Server jedes Mal für jede Sicht (Ort, Zeit) neu generiert.

Die „Interaktive Sternkarte" besteht aus einer Sammlung von Skripten (Funtionen und Variablen), welche – in der richtigen Reihenfolge aufgerufen – zu einem guten Ergebnis führen. Die Skriptsruktur ist ziemlich einfach zu durchschauen und somit einfach erweiterbar. Die Ziele (siehe Abschnitt 2.3, Seite 8) **standardisiert** (Web-Standards), **benutzbar** (GUI), **anpassbar** (GUI), **erweiterbar** (Funktionalität) und **erneuerbar** (Sterndaten) werden soweit eingehalten. Der Prototyp ist somit fertig und für weitere Entwicklungen (siehe unten) zu gebrauchen.

6.2. Fehlende Funktionalitäten

Aus zeitlichen Gründen konnten nicht alle gewünschten Funktionen implementiert werden, was schon zu Beginn klar war. Leider hat aber die Zeit

bis zum Abgabetermin auch nicht gereicht, alle Fehler in den Berechnungen und Skripten zu korrigieren und alle vorhandenen Funktionen zu vollenden.

Zeiteingabe

Wie schon angetönt (Abschnitt „Datum und Zeit", Seite 27), gibt es mit der Zeiteingabe ein Problem. Grundsätzlich soll die lokale Zeit eingegeben werden können. Nun sind aber die Zeitzonen nicht immer gleich von der Greenwich Meantime (GMT) abhängig. Z. B. haben wir in der Schweiz normalerweise eine Stunde Zeitverschiebung – in der Sommerzeit jedoch zwei. Dies verschiebt unsere Zeitrechnung, bzw. unseren relativen Standpunkt am Äquator um bis zu 1666 km. Diese Ungenauigkeit ist an sich zu verschmerzen, denn auf dem Bildschirm macht es nur wenige Pixel aus – für den ganzen Sternenhimmel gleich, was relativ das gleiche Bild liefert. Die feinste Variante wäre es, die Zeitverschiebung direkt aus der Position (Länge in Grad geteilt durch 15) zu ermitteln. Leider ist dann jedoch die Angabe der Zeitverschiebung als Zusatzinformation etwas schwieriger.

Projektion

In den Formeln der Projektion hat sich ein Fehler eingeschlichen. Diesen zu finden war leider nicht mehr möglich. Als Folge davon wird der Sternenhimmel nicht korrekt angezeigt. Sowohl in der Hauptkarte wie in der Übersicht wird ein falscher Ausschnitt dargestellt, was die Wiedererkennung des Sternenhimmels beinahe verunmöglicht.

Orientation

Die Orientation im jeweiligen Kartenausschnitt ist sehr schwierig. Dies hängt sowohl mit dem oben beschriebenen Fehler in der Projektion zusammen wie auch mit fehlenden Orientierungshilfen. Ein Muss wären hier die Angaben der vier Himmelsrichtungen und eines Zenitpunktes. Zusätzlich wäre eine symbolische Darstellung des Horizontes (bei der bisher implementierten stereographischen Projektion am Rande des Kreises in der Übersicht) und ein (ein- und ausschaltbares) Netz eine grosse Orientierungshilfe. Die Darstellung solcher Hilfen ist jedoch nur sinnvoll, wenn die Projektion stimmt.

Zoom

Das eigentlich grösste Manko ist ein fehlendes **adaptives Zooming**. Diese Funktion bringt jedoch einige (natürlich lösbare) Probleme mit sich. Bei jeder Zoomstufe sollte eine andere Sterndichte und andere Grössen für die Sterndarstellungen gebraucht werden. Dafür gibt es zwei Lösungsansätze:

- Das Zoomen erledigt der Server. Durch einen Klick auf das „+" oder das „-" wird auf dem Server eine neue Darstellung generiert. Der grösste Vorteil dieser Methode besteht darin, dass so jeweils nur die Sterne den Weg vom Server zum Client nehmen, die in dieser Stufe auch dargestellt werden. Gleichzeitig wird die Darstellung (Grösse der Sterne) vom Server errechnet. Dies ist jedoch auch der Nachteil, weil dann bei jeder neuen Zoomstufe eine ca. 200 kB grosse Datei auf den Client-Computer geladen werden muss.
- Das Zoomen wird auf dem Client gemacht. Alle Sterne sind in einem JavaScript-Array gespeichert. Beim Zoomen muss nun eine JavaScript-Funktion die SVG-Gruppe „Sterne" durch eine neue Gruppe ersetzen. Dies wird auf einigen Computern bzw. in einigen SVG-Viewern, die schon mit wesentlich einfacheren JavaScripten ihre Mühe bekunden, sehr langsam ablaufen.

Ortseingabe

Bisher sind in den Auswahllisten nur sehr wenig Ortschaften eingegeben. Hier sollten mindestens pro Land die Hauptstadt, bei grösseren Ländern noch sonstige wichtige Städte erscheinen. Es ist jedoch nicht die Idee, mehrere Hundert Städte einzugeben. Die Liste kann auf dem Server editiert werden (`koord.js`).

Über Auswahllisten kommt man mit Continent -> UserDefined -> Customize zu einem (momentan noch sehr versteckten, da nicht funktionierenden) Eingabeformular, in dem Orte (Name, Land, Breite, Länge und Zeitverschiebung) eingegeben werden können. Diese neuen Orte sollten dann in der Auswahlliste Continent -> UserDefined erscheinen. Die Idee wäre es, diese Orte für einige Wochen zu speichern, um sie dann aus der Liste zu nehmen. Es muss dazu (hier über ein PHP-Skript) bei jedem Aufruf der Seite die gesamte Liste (`dynkoord.js`) eingelesen, neu sortiert und von alten Einträgen befreit werden, bevor sie wieder neu geschrieben wird.

Leider hat die Zeit nicht mehr gereicht, die Funktionen für das Sortieren und Schreiben zu coden.

PDA

Es ist eine nette Vorstellung: mit einem PDA (muss leider über Internetzugang verfügen) in der Gegend herumzuspazieren und sich die Sterne anzeigen zu lassen. Die Orientierung wäre so sicher gewährleistet!

Der kleine Bildschirm und ein langsamer Prozessor verhindern jedoch, die gleiche Darstellung wie auf einem Computerbildschirm zu gebrauchen. Ein weiteres Problem ist die fehlende Unterstützung durch vollständige SVG-Viewer. Um dem zu begegnen ist die „Interaktive Sternkarte" fähig, eine sehr abgespeckte Version auszugeben. Es wird dabei auf jegliche JavaScripte und platzverzehrende Informationen verzichtet. Für die Eingabe von Ort und Zeit ist ein kleines HTML-Formular zuständig. Da diese Funktionalität nie getestet werden konnte, gibt es auch noch keine PDA-Einstiegsseite (Aufruf mittels `cgi/stern.php?vp=pda` oder `cgi/stern.php?other=pda`).

6.3. Ausblick

Einige Ideen, die zu Beginn gesammelt wurden, konnten aus zeitlichen Gründen gar nicht mehr realisiert werden. Hier soll eine Liste folgen, was eine Weiterarbeit an der „Interaktiven Sternkarte" zu einer „interaktiven Astrokarte" oder einem „Astrolabor" beinhalten müsste.

Fehlerbereinigung und Fertigstellung

- Korrektur der stereographischen Projektion
- Fertigstellung der Ortseingabe
- PDA-Funktion testen
- Ortslisten erweitern
- Zeitverschiebung und Sommerzeit bereinigen
- Adaptives Zooming implementieren
- Orientation in der Karte verbessern

(genauere Erklärungen zu den oben genannten Punkten in Abschnitt 6.2, Seite 30)

Verfeinerung

Um nicht bei jeder Eingabe die gesamte Seite neu generieren zu müssen und trotzdem immer die aktuellen Sterndaten zu haben bzw. die grösseren Berechnungen auf dem Server zu machen (wie bisher), soll hier auf eine Möglichkeit des DOM (siehe Abschnitt 4.5, Seite 19) aufmerksam gemacht werden. Ein grosser Teil der Interaktivität der „Interaktiven Sternkarte" wird mit JavaScript gemacht. Es können so Objekte geändert, erstellt und gelöscht werden (auf dem Client). Nun gibt es theoretisch auch die Möglichkeit, direkt mit irgendeiner anderen objektorientierten Programmiersprache auf das DOM zuzugreifen, und zwar vom Server direkt auf ein schon angezeigtes SVG auf dem Client. Diese Variante wurde in der Arbeit noch nicht so gemacht, weil dies wesentlich bessere Programmierkenntnisse voraussetzen würde. Vorteile dieser Methode sind aber ersichtlich: Es wird nicht für jede kleine Änderung eine ganze Seite generiert, sondern nur die veränderten Daten (hier: die Sterne und die Viewboxes von Hauptkarte und Übersichtskarte). Die Datenmenge, welche über das WWW geschickt werden muss, würde so noch weiter schrumpfen. Erkenntnisse aus einer solchen Arbeit könnten in viele weitere ähnliche Projekte einfliessen und dynamische Kartenabfragen über das WWW erleichtern.

Erweiterung

Erweiterungen für die „Interaktive Sternkarte" sind wie schon angetönt ziemlich leicht zu machen. Die Ideen sind nicht neu und werden von anderen ähnlichen Projekten schon zur Verfügung gestellt.

Mehrsprachigkeit

SVG unterstützt von sich aus mehrsprachige Texte. Verschiedene Texte werden mit einer `switch`-Abfrage direkt im Text als eine Art Auswahl durch eine Browserabfrage der Sprache automatisch gesetzt. Leider wird dadurch auch das SVG-File grösser. Daher wäre es noch etwas besser, da auf jeden Fall Serverabfragen geschehen, dies direkt auf dem Server von einem PHP-Skript zu erledigen.

Sterninformation

Bis jetzt wird für die Information über die Sterne lediglich diese Information gebraucht, die im FK6 als Sternname enthalten ist (z.B. α UMa – was bedeutet, dass dies der hellste Stern des Sternbildes Ursus Major ist –

die gesamte Liste aller Sternbilder liegt im Anhang 2 bei). Diese Information ist jedoch etwas spärlich. Mit anderen Listen wäre eine genauere Information möglich, bzw. wenigstens ein Ersetzen des Namens „UMa". Weiter wären die Winkel, unter denen die Sterne am gewählten Ort zur gewählten Zeit gesehen werden sehr hilfreich.

Sternbilder

Eine weitere hilfreiche Funktion, die bisher nicht in der „Interaktiven Sternkarte" enthalten ist, wäre ein Ein- und Ausschalten von Sternbildern. Um die Verbindungslinien darstellen zu könne, müssen diese jedoch zusätzlich digitalisiert und in einer neuen Ebene angezeigt werden. Dazu braucht es ebenfalls zusätzliche Informationen, die aus dem FK6 nicht ersichtlich sind.

Drehen

Neben Zoom und Pan gibt es (z.B. in einer stereographischen Projektion) auch noch die Drehung. Diese Funktion (vergleichbar zu einer Drehung um die eigene Achse) müsste als Erweiterung in der Übersichtskarte machbar sein.

Projektionen

Bisher wird als Projektion nur die Stereographische angeboten. Um weitere Projektionen anbieten zu können, wird der Funktion `projektion()` die Art der Projektion mitgegeben. Die Abfrage vor der Berechnung kann durch weitere Arten erweitert werden. Ebenfalls wird in der Funktion `svg()` der Parameter `$projektion` übergeben, da dies ebenfalls Einfluss auf die Darstellung der gesamten Seite (Hauptkarte und Übersichtskarte) hat. Hier müssten neue PHP-Variablen in `body.php` für das Schreiben des SVG definiert werden und die Aufteilung der Darstellung berücksichtigt werden. Um auch mit anderen Projektionen Zoom- und Panfunktionalitäten zu haben müssten ebenfalls die JavaScript-Funktionen `zoomIt()` und `doPan()` in `main.js` angepasst werden.

Horizont

Der Rand der Übersichtskarte, bzw. je nach Zoomfaktor oder Sichtbereich (Pan) in der Hauptkarte wird momentan einfach als Ende des Kartenbereichs abgeschnitten. Um die Übersicht zu vereinfachen wäre hier eine stilisierte Horizontdarstellung (Berge, Häuser, Bäume o.ä.) eine grosse Hilfe. Dies gibt ein wenig Programmieraufwand für eine zufällig gewählte Li-

nie. Diese muss als eigene Liniendefinition in den <def>-Bereich von SVG gespeichert werden. Der Aufwand ist relativ gross – aber auch der Nutzen wäre gross.

Himmelskörper

Wie der Titel der Arbeit sagt, werden in der Karte nur Sterne angezeigt. Möchte man jedoch die Karte zu einer „Interaktiven Astrokarte" ausbauen, wären weitere Objekte ein Muss. Planetenbahnen können ziemlich einfach berechnet und in die Daten integriert werden. Ebenfalls sind die Bahnen von Sonne und Mond relativ einfach zu berechnen. Hier stellt sich nur die Frage, wie diese beiden Himmelskörper dargestellt werden müssen, denn sie verdecken einiges. Ein weiteres Problem dabei ist das Licht der Sonne und des Mondes, welche noch mehr des Sternenhimmels verdecken. Weitere kleine Feinheiten wie ISS und andere sichtbare Satelliten könnten das Tüpfelchen auf das ι bringen...

6.4. Fazit

Die Kombination PHP mit SVG ist ein sehr starkes Gespann. Die Arbeit an der „Interaktiven Sternkarte" hat gezeigt, dass eigentlich alles machbar ist – falls genug Zeit vorhanden ist. Es gibt zwar einige Probleme, die jedoch alle lösbar sind. Durch den sehr hohen Grad der Skriptbarkeit von SVG durch das DOM, sind sehr viele Dinge möglich, welche mit anderen Darstellungssprachen o.ä. schwierig wären.

- Die Datenmenge für eine gesamte Seite ist mit 200 kB für die gesamte Seite sehr klein. Dies liegt daran, dass nicht ein Bild generiert wird, sondern nur die Anweisungen, wie der Client ein Bild machen soll.
- Die hier gewählte Struktur mit server- und clientseitigen Aufgaben lässt zwar eine sehr einfache Programm-Struktur zu, hat jedoch andere Nachteile (z.B. Datenmenge, die bei jeder neuen Generierung ansteht).
- Mit einem direkten Zugriff eines serverseitigen Skriptes auf das DOM der angezeigten Seite könnte diese Datenmenge noch sehr viel mehr verkleinert werden, indem nicht die ganze Seite neu erstellt wird, sondern lediglich der geänderte Teil.
- Durch das Zusammenspiel von JavaScript und SVG werden praktisch alle erwünschten Interaktionen ermöglicht.

- Die Struktur von Skripten kann viele verschiedene Formen haben. Hier kommen die Präferenzen der Programmierer ins Spiel, wobei jedoch ein Augenmerk auf die Ausbaubarkeit gelegt werden muss.
- Einzelne Teile der Skripte sind problemlos übertragbar auf andere Projekte (wie z.B. die Übernahme von „`selectionLists.js`" zeigt). Solche Module müssten jedoch noch allgemeiner Programmiert werden.
- Erweiterungen sind dank der einfachen Struktur sehr einfach implementierbar.
- SVG bietet sich dank all seinen Spezifikationen sehr gut für kartographische Anwendungen und Darstellungen an. Dies ist jedoch ziemlich abhängig von der Qualität der vorhandenen SVG-Viewern und –Plug-Ins (z.B. ob JavaScript unterstützt wird usw.).

7. Verzeichnisse

7.1. Abbildungsverzeichnis

7.2. Tabellenverzeichnis

7.3. Anhangsverzeichnis

7.4. Literaturverzeichnis

[1] Adobe Systems Inc.
http://www.adobe.com/svg/

[2] Apache Software Foundation
http://www.apache.org

[3] Dutchs (1999),
http://www.uwgb.edu/dutchs/structge/sphproj.htm

[4] Burnett Keith (2002),
http://www.xylem.f2s.com/kepler/

[5] Morrison James (2002),
http://www.astrolabes.org/astrolab.htm

[6] Münz Stefan (2001),
http://selfhtml.teamone.de

[7] Neumann Andreas, Winter André (2003),
http://carto.net

[8] PHP Consortium
http://www.php.net

[9] Shetline Kerry (2003),
http://www.skyviewcafe.com

[10] Walser Felix (1987). Automatisation im Bereich der Astrogeodätischen Mess- und Auswertetechnik. Bericht Nr. 128 zum Geodätischen Seminar I WS 86/87, Institut für Geodäsie und Photogrammetrie, ETH Zürich

[11] Weisstein Eric, Wolfram Research (2003),
http://mathworld.wolfram.com/StereographicProjection.html

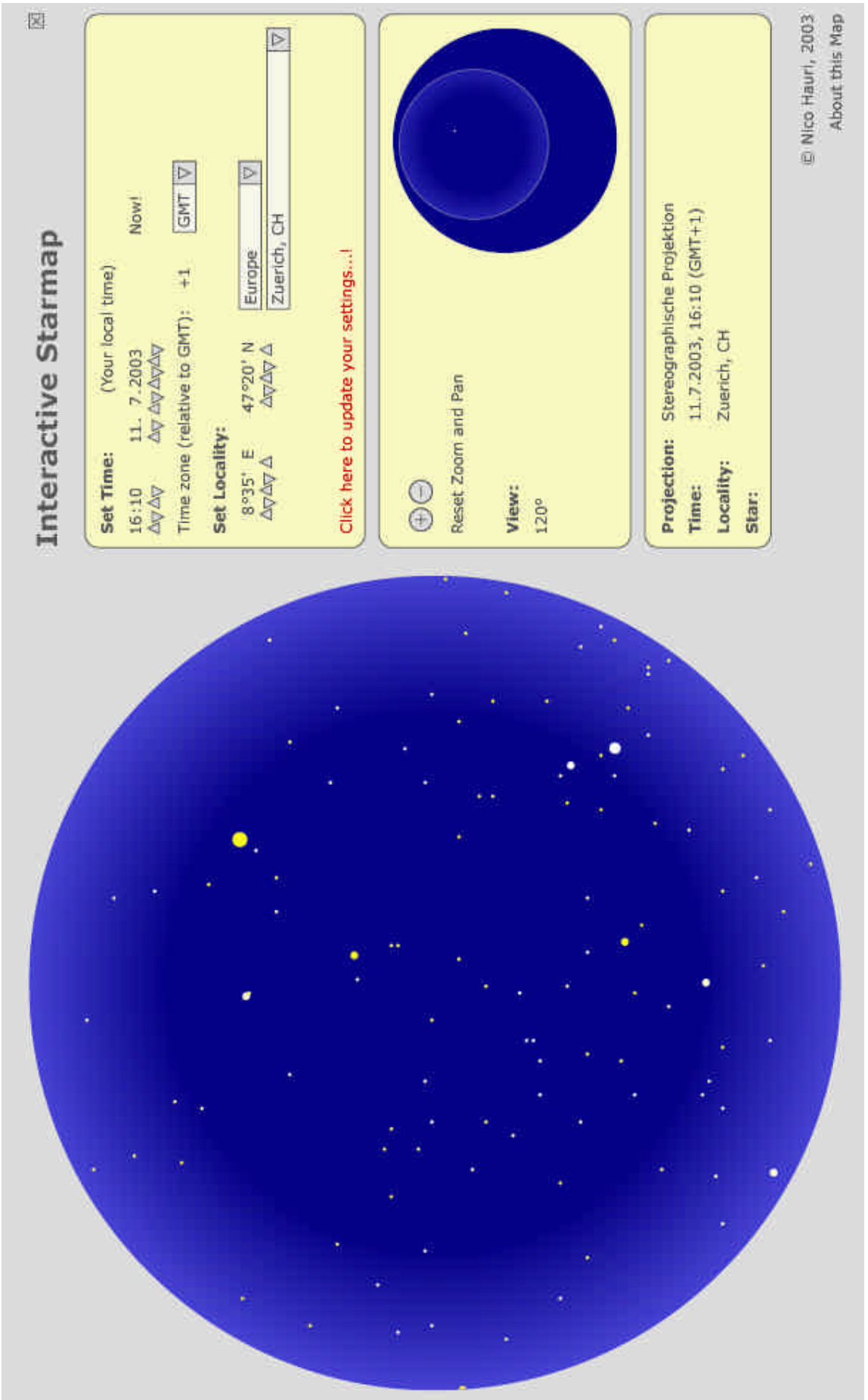
[12] Wielen R., et al. (1999). Sixth Catalogue of Fundamental Stars (FK6), Astronomisches Rechen-Institut Heidelberg. Kommissions-Verlag G. Braun, Karlsruhe

[13] World Wide Web Consortium
http://www.w3c.org

[14] World Wide Web Consotium, CSS
http://www.w3.org/Style/CSS/

[15] World Wide Web Consotium, DOM
http://www.w3.org/DOM/

[16] World Wide Web Consotium, SVG
http://www.w3.org/Graphics/SVG/Overview.htm8

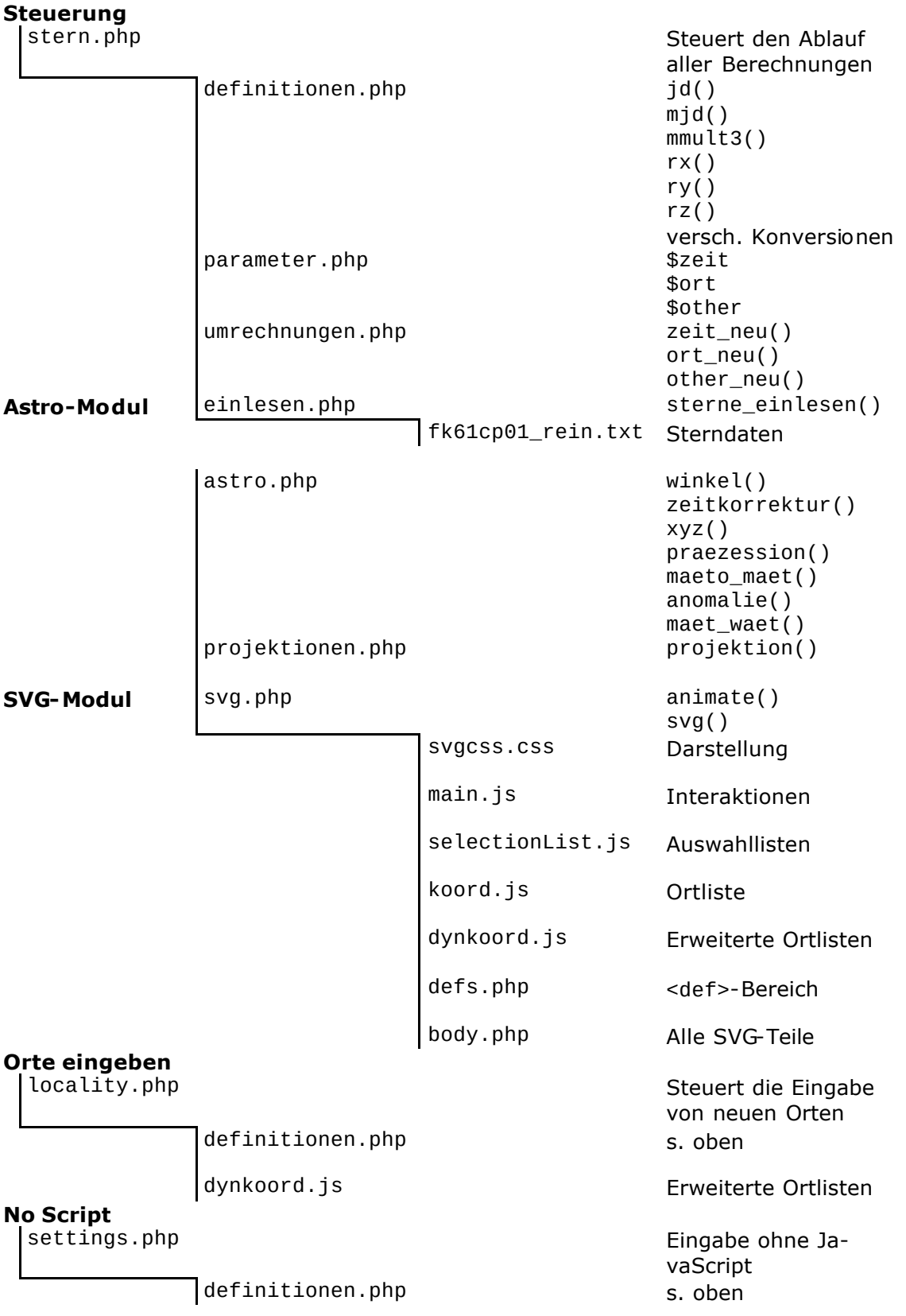

Interactive Starmap
Set Time: (Your local time)
16:10 11. 7.2003 Now!
Time zone (relative to GMT): +1 GMT
Set Locality:
8°35' E 47°20' N
Europe
Zuerich, CH
Click here to update your settings...!
Reset Zoom and Pan
View:
120°
Projection: Stereographische Projektion
Time: 11.7.2003, 16:10 (GMT+1)
Locality: Zuerich, CH
Star:
© Nico Hauri, 2003
About this Map

Constellation Names
Compiled by R. Block

Name (Short form, Genitive form): meaning

1. ANDROMEDA (And, ANDROMEDAE): the daughter of Cassiopeia
2. ANTLIA (Ant, ANTLIAE): the pump
3. APUS (Aps, APODIS): the bird of paradise
4. AQUARIUS (Aqr, AQUARII): the water carrier
5. AQUILA (Aql, AQUILAE): the eagle
6. ARA (Ara, ARAE): the altar
7. ARIES (Ari, ARIETIS): the ram
8. AURIGA (Aur, AURIGAE): the charioteer
9. BOOTES (Boo, BOOTIS): the oxherd
10. CAELUM (Cae, CAELI): the graving tool
11. CAMELOPARDALIS (Cam, CAMELOPARDALIS): the giraffe
12. CANCER (Cnc, CANCRI): the
13. CANES VENATICI (CVn, CANUM VENATICORUM): the hunting dogs
14. CANIS MAJOR (CMa, CANIS MAJORIS): the greater dog
15. CANIS MINOR (CMi, CANIS MINORIS): the lesser dog
16. CAPRICORNUS (Cap, CAPRICORNI): the horned goat
17. CARINA (Car, CARINAE): the keel of the Argo
18. CASSIOPEIA (Cas, CASSIOPEIAE): the wife of Cepheus
19. CENTAURUS (Cen, CENTAURI): the centaur
20. CEPHEUS (Cep, CEPHEI): the king of Ethiopia or Joppa
21. CETUS (Cet, CETI): the whale (or whatever) menacing Andromeda
22. CHAMAELEON (Cha, CHAMAELEONTIS): the chamaeleon
23. CIRCINUS (Cir, CIRCINI): a pair of compasses
24. COLUMBA (Col, COLUMBAE): the dove
25. COMA BERENICES (Com, COMAE BERENICES): Bernice's lock of hair
26. CORONA AUSTRALIS (CrA, CORONAE AUSTRALIS): the southern crown
27. CORONA BOREALIS (CrB, CORONAE BOREALIS): the northern crown
28. CORVUS (Crv, CORVI): the crow
29. CRATER (Crt, CRATERIS): the cup
30. CRUX (Cru, CRUCIS): the cross
31. CYGNUS (Cyg, CYGNI): the swan
32. DELPHINUS (Del, DELPHINI): the dolphin
33. DORADO (Dor, DORADUS): the goldfish
34. DRACO (Dra, DRACONIS): the dragon
35. EQUULEUS (Equ, EQUULEI): the little horse
36. ERIDANUS (Eri, ERIDANI): the river
37. FORNAX (For, FORNACIS): the furnace
38. GEMINI (Gem, GEMINORUM, or -IUM): the twins, Castor and Pollux
39. GRUS (Gru, GRUIS): the crane
40. HERCULES (Her, HERCULIS): Heracles, the hero
41. HOROLOGIUM (Hor, HOROLOGII): the clock
42. HYDRA (Hya, HYDRAE): water serpent (female)
43. HYDRUS (Hyi, HYDRI): the water serpent (male)
44. INDUS (Ind, INDI): the (American) Indian
45. LACERTA (Lac, LACERTAE): the lizard
46. LEO (Leo, LEONIS): the lion
47. LEO MINOR (LMi, LEONIS MINORIS): the lesser lion
48. LEPUS (Lep, LEPORIS): the hare
49. LIBRA (Lib, LIBRAE): the scales (weighing)
50. LUPUS (Lup, LUPI): the wolf
51. LYNX (Lyn, LYNCIS): the lynx
52. LYRA (Lyr, LYRAE): the lyre
53. MENSA (Men, MENSAE): the Table (mountain)

54. MICROSCOPIUM (Mic, MICROSCOPII): the microscope
55. MONOCEROS (Mon, MONOCEROTIS): the unicorn
56. MUSCA (Mus, MUSCAE): the fly
57. NORMA (Nor, NORMAE): the carpenter's level
58. OCTANS (Oct, OCTANTIS): the oct
59. OPHIUCHUS (Oph, OPHIUCHI): the snake holder
60. ORION (Ori, ORIONIS): the hunter
61. PAVO (Pav, PAVONIS): the peacock
62. PEGASUS (Peg, PEGASI): the winged horse
63. PERSEUS (Per, PERSEI): the rescuer of Andromeda
64. PHOENIX (Phe, PHOENICIS): the phoenix
65. PICTOR (Pic, PICTORIS): the painter (originally his easel)
66. PISCES (Psc, PISCIUM): the fish
67. PISCIS AUSTRINUS (PsA, PISCIS AUSTRINI): the southern fish
68. PUPPIS (Pup, PUPPIS): the poop deck of the Argo
69. PYXIS (Pyx, PYXIDIS): the box, or compass, of the Argo
70. RETICULUM (Ret, RETICULI): the net (optical instrument)
71. SAGITTA (Sge, SAGITTAE): the arrow
72. SAGITTARIUS (Sgr, SAGITTARII): the archer
73. SCORPIUS (Sco, SCORPII): the scorpion
74. SCULPTOR (Scl, SCULPTORIS): the sculptor (originally his studio)
75. SCUTUM (Sct, SCUTI): the shield
76. SERPENS (Ser, SERPENTIS): the snake
77. SEXTANS (Sex, SEXTANTIS): the sextant
78. TAURUS (Tau, TAURI): the bull
79. TELESCOPIUM (Tel, TELESCOPII): the telescope
80. TRIANGULUM (Tri, TRIANGULI): the triangle
81. TRIANGULUM AUSTRALE (TrA, TRIANGULI AUSTRALIS): southern triangle
82. TUCANA (Tuc, TUCANAE): the toucan
83. URSA MAJOR (UMa, URSAE MAJORIS): the greater bear
84. URSA MINOR (UMi, URSAE MINORIS): the lesser bear
85. VELA (Vel, VELORUM): the sails of the Argo
86. VIRGO (Vir, VIRGINIS): the virgin
87. VOLANS (Vol, VOLANTIS): the flying fish
88. VULPECULA (Vul, VULPECULAE): the little fox


Steuerung
stern.php
Steuert den Ablauf aller Berechnungen
definitionen.php
jd()
mjd()
mmult3()
rx()
ry()
rz()
versch. Konversionen
parameter.php
$zeit
$ort
$other
umrechnungen.php
zeit_neu()
ort_neu()
other_neu()
Astro-Modul
einlesen.php
sterne_einlesen()
fk61cp01_rein.txt
Sterndaten
astro.php
winkel()
zeitkorrektur()
xyz()
praezession()
maeto_maet()
anomalie()
maet_waet()
projektionen.php
projektion()
SVG-Modul
svg.php
animate()
svg()
svgcss.css
Darstellung
main.js
Interaktionen
selectionList.js
Auswahllisten
koord.js
Ortliste
dynkoord.js
Erweiterte Ortlisten
defs.php
<def>-Bereich
body.php
Alle SVG-Teile
Orte eingeben
locality.php
Steuert die Eingabe von neuen Orten
definitionen.php
s. oben
dynkoord.js
Erweiterte Ortlisten
No Script
settings.php
Eingabe ohne JavaScript
definitionen.php
s. oben