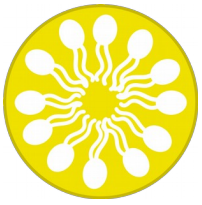
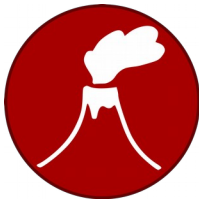
JuniorAkademie Adelsheim

15. Science Academy Baden-Württemberg 2017

Astronomie

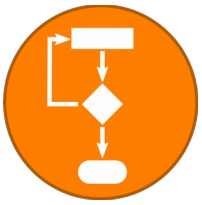


Chemie



Geophysik

Geschichte/ Amerikanistik



Informatik

TheoPrax

HOPP FOUNDATION for computer literacy & informatics

Regierungspräsidium Karlsruhe
Abteilung 7 – Schule und Bildung

Stiftung Bildung und Jugend

Dokumentation der JuniorAkademie Adelsheim 2017

15. Science Academy Baden-Württemberg

Veranstalter der JuniorAkademie Adelsheim 2017:
Regierungspräsidium Karlsruhe
Abteilung 7 –Schule und Bildung–
Hebelstr. 2
76133 Karlsruhe
Tel.: (0721) 926 4245
Fax.: (0721) 933 40270
www.scienceacademy.de
E-Mail: joerg.richter@scienceacademy.de
monika.jakob@scienceacademy.de
rico.lippold@scienceacademy.de

Die in dieser Dokumentation enthaltenen Texte wurden von den Kurs- und Akademieleitern sowie den Teilnehmern der 15. JuniorAkademie Adelsheim 2017 erstellt. Anschließend wurde das Dokument mit Hilfe von LaTeX gesetzt.

Gesamtredaktion und Layout: Jörg Richter

Vorwort

Seit nunmehr 15 Jahre findet die Junior Akademie Adelsheim im Landesschulzentrum für Umwelterziehung (LSZU) in Adelsheim statt. Der offizielle Startschuss für die diesjährige Akademie fiel schon mit dem Initiationstreffen im Januar, doch das erste Zusammentreffen der Kursleiter, Schülermentoren, Leiter der kursübergreifenden Angebote (KüAs) und 72 Teilnehmerinnen und Teilnehmern fand im Juni statt: Gemeinsam legten sie am Eröffnungswochenende den Grundstein für die erfolgreiche Kursarbeit der zweiwöchigen Sommerakademie. Im Oktober wurden Ergebnisse und Erlebnisse am Dokumentationswochenende festgehalten, das zugleich ein schönes Wiedersehen und ein Abschluss der Junior Akademie Adelsheim war.

In den Kursen, die im Rahmen der JuniorAkademie angeboten werden, bekommen die Jugendlichen die Möglichkeit, wissenschaftlichen Fragestellungen auf den Grund zu gehen, eigenständig zu arbeiten und die aufgeworfenen Fragen zu beantworten. Die Teilnehmer profitieren nicht nur von einem fachlichen Wissenszuwachs und neu erlerntem Methodenwissen, sondern entwickeln sich auch auf persönlicher Ebene weiter. Neue Freundschaften gehören ebenso zu den Dingen, die die Teilnehmerinnen und Teilnehmer von der Akademie mitnehmen, wie auch ein gestärktes Selbstbewusstsein und das Erlangen einer neuen Entwicklungsperspektive. Dies reflektieren die Jugendlichen auch für sich in der abschließenden Evaluation.

Das Erlangen eines neuen Standpunktes spiegelte sich auch in dem diesjährigen Motto der Akademie „Horizonte“ wider: Seit dem Eröffnungswochenende begleitete uns dieses Motto in Form von wunderschönen von vielen Teilnehmern eingesendeten Bildern zu diesem Thema, passenden philosophischen Gedanken und Sprüchen sowie gemeinsamen Aktionen durch die Akademiezeit.

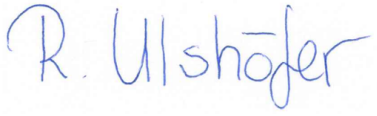
In jedem Kurs fand sich das Thema „Horizonte“ wieder: Der Astronomiekurs blickte nachts regelmäßig an den Himmelshorizont, die Geophysiker klärten darüber auf, dass es Gesteinshorizonte auch unter der Erdoberfläche gibt, und die Chemiker beschäftigten sich mit einem Horizont, der viel kleinere Dimensionen hat, den Nanopartikeln. Im Kurs Geschichte-/Amerikanistik setzten die Kursteilnehmer sich damit auseinander, wie eingeschränkt der geistige Horizont beispielsweise während der Hexenverfolgung war, in nur wenigen Disziplinen verändert sich der Horizont so schnell wie in der Informatik und im TheoPrax-Kurs vermischten sich zwei Horizonte zu einem Gesamtkonzept.

Das Motto sollte Anlass zum Reflektieren und Nachdenken sein, aber auch besonders die schönen Momente der Akademiezeit und das ganz besondere „Akademiegefühl“ begleiten und einrahmen. Wie sich der Horizont aller Teilnehmer während der Akademie verändert hat, wird jeder für sich selbst herausfinden, aber obwohl mit der Dokumentation die Akademie zu Ende gegangen ist, geht es hinterm Horizont ja immer weiter, und wir hoffen, dass für alle an der Akademie Beteiligten die Zeit in Adelsheim noch lange in guter Erinnerung bleiben wird.

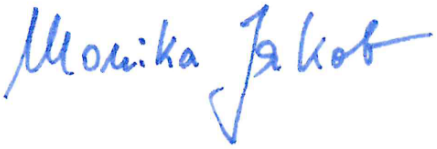
Aber jetzt wünschen wir euch viel Spaß beim Lesen, Schmökern und Erinnern!

Eure/Ihre Akademieleitung

Johanna Kroll (Assistenz)



Rebecca Ulshöfer (Assistenz)



Dr. Monika Jakob

Jörg Richter

Inhaltsverzeichnis

Kurs 1 – Die Kartenmacher: Die Kunst der Erstellung von Sternkarten

1. Einführung

Olaf, Tatjana, Ranran

In ähnlicher Weise, wie uns Landkarten auf der Erde zur Orientierung und als Medium der Datenerfassung und -visualisierung dienen, helfen uns Sternkarten am Himmel.

Der Astronomiekurs hatte das Ziel, eine Sternkarte des über Deutschland sichtbaren Sternenhimmels (eine polzentrierte Karte, wie sie bei drehbaren Sternkarten zum Einsatz kommt) selbst zu erzeugen.

Zur Bewältigung dieser astronomisch motivierten Aufgabe benötigten wir Kenntnisse und Fähigkeiten aus Kartografie, Mathematik, Kunst und vor allem aus der Informatik. Diese wurden zunächst durch die Kursteilnehmer geliefert.

Danach bestand die zentrale Aufgabe des Kurses darin, mittels der Computersprache Python in der Entwicklungsumgebung Spyder ein Programm zur Ausgabe einer Karte zu erzeugen, welche die hellsten 1000 Sterne zeigt, die ein Beobachter bei einem Breitengrad von 50° N sehen kann. Scheinbare Helligkeit und Farbe der Sterne sollen ersichtlich werden. Die Sternbilder sollen einmal durch Verbindungsstriche erkennbar, aber auch durch die namengebenden mythischen Figuren illustriert werden.

Im Folgenden wird beschrieben, wie sich der Kurs für die Aufgabe fit machte, wie die Aufgabe umgesetzt wurde und was sonst noch passierte.

2. Sternkarten in Kunst und Geschichte

Ole, Hannah D.

2.1. Was ist eine Sternkarte?

So, wie eine irdische Landkarte den gewünschten Teil der Oberfläche der Erde darstellt, zeigt eine Sternkarte den gewünschten Teil des Sternenhimmels, den man sich als scheinbare Himmelskugel vorstellt.

Eine Sternkarte gibt dann die Position und einige beobachtbare Merkmale der Sterne, wie scheinbare Helligkeit, Farbe oder Sterntyp wieder.

Die traditionelle Sternkarte liegt meist in zweidimensionaler, also gezeichneter oder ausgedruckter Form vor. Die moderne Sternkarte existiert digital und wird auf dem Bildschirm angezeigt, z. B. durch Planetariumssoftware.

2.2. Werdegang der Sternkarten

Schon in den frühen Hochkulturen wurden die hellsten Sterne beobachtet, zu Sternbildfiguren gruppiert und aufgezeichnet. Diese ersten Sternkarten hatten jedoch weniger einen naturwissenschaftlichen als einen lebenspraktischen und religiösen Nutzen. Die Sternbilder wurden mit mythischen Figuren identifiziert. Claudius Ptolemäus erweiterte dann um 150 n. Chr. den bereits von Hipparchos aufgestellten Sternkatalog (siehe Abschnitt 4.2). In diesem verzeichnete er die Positionen von über 1000 Sternen.

Im Mittelalter wurde die Gestaltung von Sternkarten stark durch das Christentum geprägt. Aufgrund des strengen Glaubens wurden Stern- und Sternbildbezeichnungen aus der Antike teilweise aberkannt.

In der Renaissance erhielten auch in der Astronomie die Erkenntnisse der Antike wieder Beachtung. 1515 erschien die erste gedruckte Sternkarte von Albrecht Dürer (siehe Abb. 2.1). Seine Daten bezog der Maler aus den Aufzeichnungen bedeutender Astronomen der Antike, wie z. B. Claudius Ptolemäus.

Heutige Sternkarten verzeichnen auch Sterne, die mit dem bloßen Auge nicht erkennbar sind.

Abbildung 2.1: Holzdruck einer Sternkarte der nördlichen Hemisphäre um 1515 von Albrecht Dürer. In den Ecken sind vier berühmte Astronomen zu sehen.

2.3. Sternbilder und Sternnamen

Die Namen der insgesamt 88 Sternbilder und ihrer Sterne sind historisch gewachsen und haben verschiedene kulturelle Wurzeln. Eine weltweite Vereinheitlichung wurde durch die Anfang des 20. Jahrhunderts gegründete internationale astronomische Union vorgenommen. Die meisten Sternbilder behielten ihren Namen seit der Antike. Die heute noch gebräuchlichen Eigennamen der hellen Sterne haben oft einen arabischen Ursprung, wie z. B. der Stern Deneb im Sternbild Schwan. Die erste systematische Sternbezeichnung führte Johann Bayer im Jahre 1603 ein. Danach werden die Sterne eines Sternbildes nach dessen Namen und der scheinbaren Helligkeit wie folgt benannt: Die Helligkeiten der Sterne innerhalb eines Sternbildes werden dem griechischen Alphabet zugeordnet. Dabei entspricht die Bezeichnung „Alpha“ dem hellsten Stern, „Beta“ dem zweithellsten usw. Diese Sternbezeichnungen werden heute noch genutzt.

2.4. Sternkarten in der gestaltenden Kunst

Sternkarten werden seit der Antike erstellt. Dabei schritt ihre Gestaltung durch viele ver-

schiedene Epochen der Kunstgeschichte. Die Ausgestaltung erfolgte durch Federzeichnungen, Wandmalereien, Holzdrucke und viele weitere Techniken. Ein Beispiel für eine Wandmalerei ist in der Kuppel der alten Sakristei in Florenz (1434–1440) von Giovanni Pesello zu sehen.

Typisch für alte Sternkarten war die Perspektive des Betrachters außerhalb der scheinbaren Himmelskugel - also nicht von der Erde aus. So sind diese Sternkarten im Abgleich mit dem Nachthimmel spiegelverkehrt.

2.5. Sternkarte Albrecht Dürers

Der Holzdruck Albrecht Dürers ist als erste gedruckte Sternkarte (siehe Abb. 2.1) wohl eine der bekanntesten.

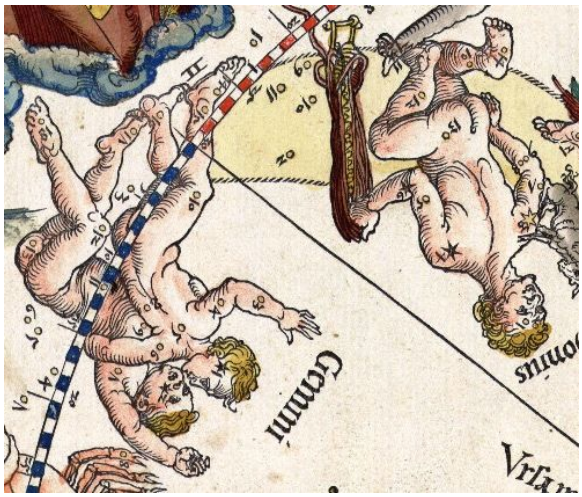


Abbildung 2.2: Ausschnitt aus der in Abb. 2.1 gezeigten Sternkarte. Zu sehen sind die verdrehten Köpfe der Sternbildfiguren.

Dürer wurde am 21. Mai 1471 in Nürnberg geboren. 1515 gestaltete er mit wissenschaftlicher Unterstützung von Johannes Stabius und Conrad Heinfogel seine berühmten Karten der südlichen und nördlichen Hemisphäre (siehe auch Abb. 2.1). In den Ecken der Karte sind die vier berühmten Astronomen Aratus Cilix, Ptolemeus Aegyptus, Marcus Manilius und Addorhaman Al-Suphi abgebildet.

Auffällig an der Karte Albrecht Dürers sind die Figuren mit merkwürdig verdrehten Köpfen. Diese Darstellung zeigt den eigenen Stil des Künstlers, welcher trotz der Perspektive außerhalb der gedachten Himmelskugel versucht, die Sternbilder frontal oder von der Seite zu zeigen (siehe Abb. 2.2). [2.1], [2.2]

2.6. Sternbildmythen

Wie die anderen Kursteilnehmer bei unserer Nachtwanderung erfuhren, stecken hinter den Sternbildern viele Mythen. Verschiedene Kulturen hatten unterschiedliche Vorstellungen von den Sternbildern. Bekannt sind heute vor allem griechische Mythen wie die der Sternbilder Perseus, Andromeda, dem Walfisch Cetus und Cassiopeia. Die äthiopische Königin Cassiopeia behauptete, schöner als die Nereiden zu sein. Dies zog den Zorn des Poseidon auf sich, welcher das Ungeheuer Cetus schickte. Um das Unheil abzuwenden, wollte ihm Cassiopeia ihre Tochter Andromeda opfern. Diese wurde an einen Felsen gekettet und stand kurz vor ihrem Tod, als Perseus mit dem Kopf der Medusa kam, das Ungeheuer versteinerte und Andromeda rettete.

Bis heute ist diese Geschichte am Nachthimmel verewigt [2.1]. In unserer eigenen Sternkarte haben wir versucht, diese Mythen in Form von selbstgestalteten Bildern einfließen zu lassen.

3. Sternkarten heute

Simon, Paula

3.1 Die scheinbare Helligkeit

Schon früh entwickelte sich der Wunsch, die unterschiedlich hell erscheinenden Sterne zu klassifizieren. Im alten Babylonien entwickelte man deshalb eine sechsteilige Helligkeitsskala, die die Sterne wie folgt einteilte: Gerade noch sichtbaren Sternen wurde die Größenklasse Sechs zugewiesen. Den hellsten Sternen, wie beispielsweise Wega im Sternbild Leier, wurde die Größenklasse Eins vergeben (siehe auch Abb. 3.1 – nach Korrektur hat Wega heute die Größenklasse 0).

Auch Ptolemäus und Hipparchos teilten in ihren Sternkatalogen die mit bloßem Auge sichtbaren Sterne in sechs Größenklassen ein und gaben diese in „Magnituden“ (*mag*) an. Der Begriff „Magnitude“ leitet sich vom lateinischen

Wort *Magnitudo* ab, was so viel wie Größe bedeutet. Später wurde die Skala nach beiden Seiten hin erweitert. Die logarithmische Einteilung der Sternhelligkeiten blieb weiterhin erhalten. Da für das menschliche Auge nur Sterne bis zu sechs mag (kurz: m) erkennbar sind, kamen für unsere Sternkarte nur Sterne infrage, die heller sind. Diese scheinbare Helligkeit visualisierten wir auf unserer Sternkarte mit der Größe der Scheibchen. Eine große Scheibe bedeutet einen Stern mit einer hohen scheinbaren Helligkeit bzw. einem kleinen Magnitudenwert.

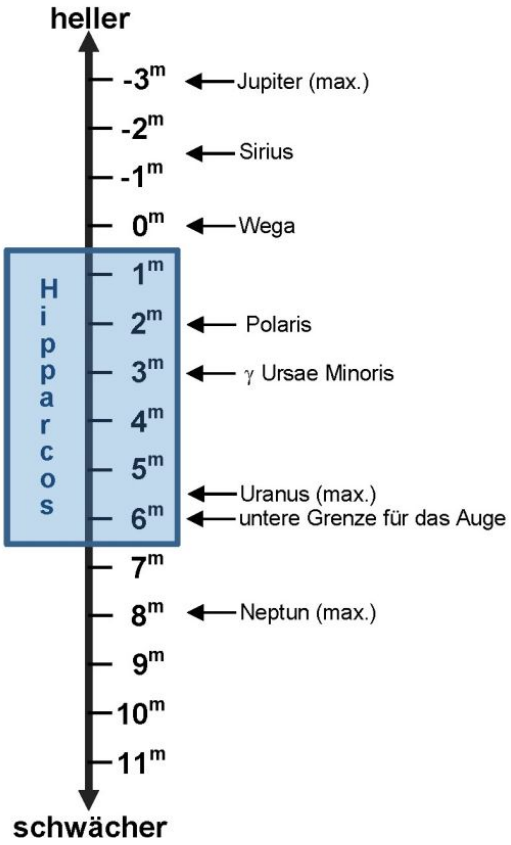


Abbildung 3.1: Magnitudenskala der scheinbaren Helligkeit.

3.2 Der B-V-Farbindex

Zusätzlich zur scheinbaren Helligkeit wollten wir auf unserer Sternkarte den Farbindex der Sterne visualisieren. Mithilfe des Farbindex B-V lässt sich die Farbe eines Sterns bestimmen. Genauer gesagt gibt dieser den Helligkeitsunterschied eines Sterns zwischen verschiedenen Wellenlängenbereichen an.

Im Hipparcos-Katalog (siehe Abschnitt 4.2) ist der B-V Farbindex angegeben, also die Differenz zwischen der Helligkeit im blauen Spektralbereich des Lichts (B = Blau) und der Helligkeit im gelben Spektralbereich (V = Visuell). So hat beispielsweise der Stern Spica im Sternbild Jungfrau, der einen Farbindex von $-0,23$ mag besitzt, also im blauen Spektralbereich mehr Licht emittiert, die Farbe Blau. Wohingegen der Stern 119 Tauri im Sternbild Stier einen Farbindex von 2,06 mag hat. Er emittiert also viel mehr Licht im roten Spektralbereich, erscheint daher auch tiefrot.

3.3 Projektionsarten

Weitere wichtige Sterndaten, die in den modernen Sternkarten verwendet werden, sind die Positionen der einzelnen Sterne am Nachthimmel. Um verschiedene Objekte, zum Beispiel den Stern Atair, zu finden, genügt es, zwei Winkel in Erfahrung zu bringen und diese auf den Sternenhimmel zu übertragen. Dabei wird mit einem ähnlichen Koordinatensystem wie dem auf der Erde gearbeitet, nur dass an der Himmelskugel die Breitengrade Deklinationen genannt und in Grad gemessen werden und man die Längengrade als Rektaszensionen bezeichnet und in Stunden misst. Dieses Koordinatensystem umspannt die scheinbare Himmelskugel wie ein Netz.

Koordinaten an einer Kugeloberfläche lassen sich nicht so einfach auf einer zweidimensionalen Karte darstellen wie die Helligkeit oder Farbe eines Sterns. Um die Positionen der Himmelsobjekte auf die Sternkarte zu übertragen, wird mit verschiedenen Projektionen gearbeitet. Zum einen benötigt man in die Ebene abrollbare Projektionsflachen. Das heißt, die Fläche, auf der die Projektion später abgebildet werden soll, wird bildlich gesehen um die Kugel gewickelt und später abgerollt. Je nach Art der Abbildungsfläche kann man zwischen drei Kartenprojektionen unterscheiden (siehe Abb. 3.2): Ebenenprojektion, Zylinderprojektion und Kegelprojektion.

Um die Sterne von der Himmelskugel auf eine Ebene möglichst realitätsgetreu abzubilden, müsste die Projektion folgende zwei Kriterien erfüllen: Winkeltreue und Flächentreue. Ist

eine Karte winkeltreu, so entsprechen alle Winkel auch den Winkeln in der Wirklichkeit. Auf einer Landkarte behalten dann alle Kontinente die gleiche Form. Das ist sehr hilfreich zur Orientierung und Navigation.

In flächentreuen Karten bleibt der Flächeninhalt im Verhältnis zur Realität erhalten. Das kann man z. B. bei Landvermessungen gut gebrauchen. Aber nur eine Kugeloberfläche erfüllt beide Kriterien gleichzeitig, denn bei jeder Abbildung der gekrümmten Himmelskugel auf eine ebene Karte muss ein Kriterium vernachlässigt werden, je nachdem, welchen Zweck die Karte erfüllen soll.

Besonders an den Kartenrändern kommt es, je nachdem welche Verzerrung man in Kauf nimmt, zu Abweichungen des naturgetreuen Abbilds. Beispielsweise wirkt Grönland bei der Winkeltreue viel größer, als es eigentlich ist.

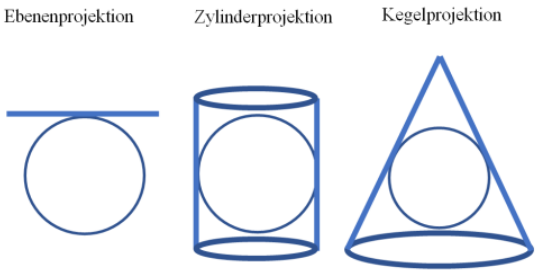


Abbildung 3.2: Darstellung der drei am häufigsten verbreiteten Projektionsarten.

Da eine Sternkarte zur Orientierung dient, haben wir für unsere Sternkarte eine winkeltreue Projektion gewählt, die man hauptsächlich für das Herstellen von Sternkarten verwendet: die stereographische Zentralprojektion. Bildlich gesehen liegt bei dieser Ebenenprojektion das Zentrum der Projektionsstrahlen im Süd- oder Nordpol der Himmelskugel. So besteht die Möglichkeit, die komplette Kugel auf eine gegenüber anliegende Ebene zu projizieren. Jetzt kann mithilfe der Projektionslinien die Abbildung der Kugel auf die Fläche übertragen werden.

Durch die Verzerrung wird der Abstand zwischen den einzelnen Deklinationskreisen nach außen hin immer größer, daher haben wir am Rand unserer Sternkarte die Skalierung gestaucht (äquidistante Ebenenprojektion). Dadurch verliert die Karte am Rand ihre Winkeltreue.

4. Sternkataloge als Quellen für Sternkarten

Mattea, Hannah B.

Als nächstes beschäftigten wir uns mit der Datenbeschaffung für unsere Sternkarte. Da wir dafür den Hipparcos-Katalog benutzen wollten, hörten wir Vorträge zur Satellitenmission Hipparcos und zum dazugehörigen Sternkatalog.

4.1 Die Satellitenmission Hipparcos

Schon in den 70er Jahren begann die Europäische Raumfahrtorganisation ESA ein bis dahin einzigartiges Projekt zu planen – die exakte astrometrische Vermessung der Sterne vom Weltraum aus. Sie entwickelte den Astrometrie-Satelliten Hipparcos, dessen Aufgabe es sein sollte, hochgenaue Entfernungsdaten von über 100.000 Sternen zu sammeln.

Dieser Satellit wurde kurz Hipparcos genannt, was sich ableitet aus den Anfangssilben seiner technischen Bezeichnung „High Precision Parallax Collecting Satellite“ und gleichzeitig an den bedeutenden griechischen Astronomen Hipparchos von Nikäa (ca. 200 v. Chr.) erinnert.

Mithilfe eines hochgenauen Verfahrens zur Winkelmessung konnte der Satellit die scheinbare Verschiebung der Sterne vor einem sehr fernen Hintergrund (Bezugssystem) feststellen – die Parallaxenwinkel (siehe Abb. 4.1). Dabei konnten Winkel von unter 0,0001” (Bogensekunden) ermittelt werden, was dem 1/36.000.000 eines Grades entspricht.

Gleichzeitig wurden bei der Mission auch die Helligkeiten und die Farben der Sterne erfasst.

Der Satellit wurde am 8. August 1989 mit einer Ariane 44 LP Trägerrakete in einen geostationären Transferorbit gebracht. Mithilfe eines Boost-Motors (Hilfstriebwerk) sollte er in eine geostationäre Umlaufbahn wechseln. Doch dieses Manöver versagte und die Mission drohte zu scheitern. Nur durch eine Korrektur der Umlaufbahn, eine Umprogrammierung der Sensoren und die Einrichtung zusätzlicher Bodenstationen gelang es der ESA, die Mission zu einer der erfolgreichsten der modernen Astrometrie

zu machen. Der Satellit lieferte trotz ungünstiger Umlaufbahn weitaus mehr und präzisere Ergebnisse als im Vorfeld erhofft. Von 1989 bis 1993 wurden mehr als 2,5 Millionen Sterne vermessen bis am 15. 8. 1993 die Kommunikation endgültig ausfiel.

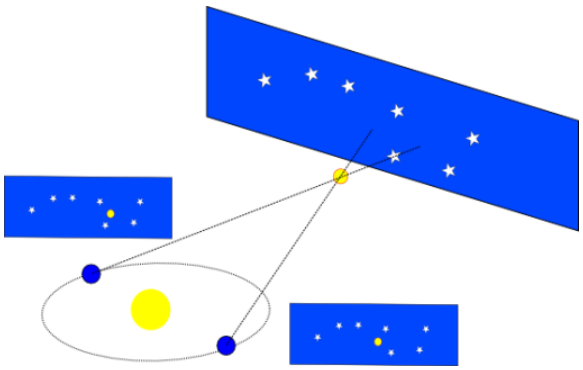


Abbildung 4.1: Prinzip der Sternparallaxe: Durch die jährliche Bewegung der Erde um die Sonne verschiebt sich ein naher Stern vor dem entfernten Hintergrund im Halbjahresrhythmus (hier stark übertrieben)[1]

Ein Team aus über 200 Wissenschaftlern wertete in den folgenden vier Jahren die Daten aus und erstellte drei Sternkataloge: den Hipparcos-Hauptkatalog und zwei Tycho-Kataloge.

4.2. Sternkataloge und der Hipparcos-Katalog

Sternkataloge im Allgemeinen dienen dazu, eine große Anzahl an Sternen nach verschiedenen Eigenschaften aufzulisten und als Buch oder Datenbank darzubieten.

Der erste Sternkatalog überhaupt war der von Timocharis von Alexandria ca. 300 vor Christus. Dieser Katalog wurde auch 100 Jahre später von Hipparchos von Nikäa für seinen eigenen Sternkatalog benutzt. 200 Jahre nach Christus entstand der Sternkatalog von Ptolemäus. Er enthielt 1022 über Alexandria mit dem bloßen Auge sichtbare Sterne und war über 1000 Jahre der Standardkatalog in der westlichen und arabischen Welt. Spätere Versionen kennt man heute unter dem Namen „Almagest“ (siehe Abb. 4.2).

[1]Bild: Wikimedia, Wikimedia-User WikiStefan, CC-BY-SA

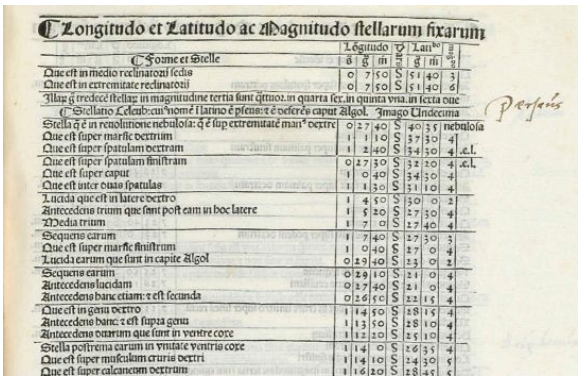


Abbildung 4.2: Ausschnitt aus dem Ptolemäus-Katalog: oben noch 2 Sterne aus dem Sternbild Kassiopeia, dann folgen Sterne aus dem Sternbild Perseus.

Allgemein kann man sagen, dass die früheren Sternkataloge immer nur wenige, an einem Ort und ohne technische Hilfsmittel sichtbare Sterne enthielten, wobei die Positionen von der Erde aus bestimmt wurden. Da es auch heute unmöglich ist, alle, teils nicht mal mehr sichtbaren Sterne zu katalogisieren, muss man eine Auswahl treffen. Dazu arbeiten die meisten Sternkataloge mit einer Grenze der scheinbaren Helligkeit, es werden also nur Sterne vermerkt, die heller als diese Grenze sind.

Heutzutage werden Daten aus Sternkatalogen benutzt, um u.a. die Position eines neuen Objekts oder eines Ereignisses vor dem Hintergrund zu beschreiben. So kann man z. B. die Bewegungen der Spiralarme unserer Galaxie beobachten, wenn sich nähere Sterne im Vergleich zu ihrem Hintergrund bewegen.

Häufig sind Sternkataloge, wie der Hipparcos-Katalog, im Internet über Datenbanken abrufbar. Parameter können zum Beispiel die Sternörter sein. Diese werden in den Himmelskoordinaten Rektaszension (in Stunden, Minuten und Sekunden) und Deklination (in Grad, Bogenminuten und Bogensekunden) angegeben (siehe Abb. 4.3, Spalten 2 und 3). Weitere wichtige Parameter sind die scheinbare Helligkeit des Sterns, die in Magnituden angegeben wird (Spalte 4), und der Farbindex des Sterns, welcher Auskunft gibt über die Farbe des Sternlichts, die von der Temperatur des Sterns abhängt (Spalte 5).

Der Hipparcos-Katalog enthält die Daten der gleichnamigen ESA-Mission und wurde 1997

HIP	RAhms	DEdms	Vmag mag	B-V mag
32349	06 45 09.25	-16 42 47.3	-1.44	0.009
30438	06 23 57.09	-52 41 44.6	-0.62	0.164
69673	14 15 40.35	+19 11 14.2	-0.05	1.239
71683	14 39 40.90	-60 50 06.5	-0.01	0.710
91262	18 36 56.19	+38 46 58.8	0.03	-0.001
24608	05 16 41.30	+45 59 56.5	0.08	0.795
24436	05 14 32.27	-08 12 05.9	0.18	-0.030
37279	07 39 18.54	+05 13 39.0	0.40	0.432
7588	01 37 42.75	-57 14 12.0	0.45	-0.158
27989	05 55 10.29	+07 24 25.3	0.45	1.500

Abbildung 4.3: Ausschnitt aus dem Hipparcos-Katalog, geordnet nach scheinbarer Helligkeit (Spalte 4).[2]

veröffentlicht. Aus dieser Mission entsprangen die folgenden drei Kataloge:

- Hipparcos-Hauptkatalog: 118.218 Sterne. Sternörter, Parallaxen und Eigenbewegungen mit sehr hoher Präzision (Messfehler von etwa 0,001”, also einer Millibogensekunde).
- Tycho-Katalog: mehr als 1 Million Sterne (statt der geplanten 400.000) Astronomische Vermessung und Bestimmung von Helligkeit und Farbe.
- Tycho-2-Katalog: 2,5 Millionen Sterne. Astronomischer und photometrischer Referenzkatalog, enthält Positionen, Bewegungen, Helligkeit und Farben.

5. Sternkarten im Computer

Adrian, Lilly

Nachdem wir durch verschiedenste Beiträge gelernt hatten, was man alles beachten muss, wenn man Sterne beobachtet, und wie man Daten über Sterne findet, konnte es endlich mit dem Programmieren losgehen. Um unsere Sternkarte zu programmieren, benutzten wir die Programmiersprache „Python“, welche Guido van Rossum Anfang der 1990er entwickelt hatte. Wir verwendeten diese Sprache, weil unsere Vorlage, die Sternkarte von Thomas Müller (siehe Abschnitt 8), in dieser Sprache verfasst war. Das traf sich ganz gut, da manche aus unserem Kurs noch nie programmiert hatten und sich Python gut für Einsteiger eignet. Außerdem mussten wir dank der großen Standardbibliothek, also den möglichen Zugriff auf eine Vielzahl von vordefinierten Befehlen und Funktionen, kein langes Programm schreiben und konnten einfach verschiedenste Module in unser Endprogramm importieren. Ein versteckter Nachteil ist die vereinfachte Syntax und die wenigen Schlüsselwörter, die Python verwendet. Die Syntax ist vergleichbar mit unserer Rechtschreibung. Schlüsselwörter dagegen sind vordefinierte Befehle und Funktionen, die man als Programmierer sehr einfach aufrufen kann. Was oft passieren kann, sind kleine Fehler, wenn man die falschen Klammern benutzt oder die Anführungszeichen vergisst. Dieses Problem kann man aber ganz leicht beheben, indem eine große Gruppe, das ganze Programm noch überprüft, denn 24 Augen sehen mehr als nur zwei.

[2]Bild: Suchergebnisse mit VizieR, http://vizier.u-strasbg.fr/viz-bin/VizieR-3?-source=I/239/hip_main

5.1 Grundlagen der Programmierung

Als ersten Befehl erlernten wir ***print()*** (die Ausgabe-Funktion). Danach beschäftigten wir uns noch mit ***type()***, was den Typ der Variablen ausgibt. Hierbei muss man wissen, dass es bei Python mehrere Datentypen gibt, beispielsweise den Typ ***String***, welcher ein beliebiges Zeichen oder eine Zeichenfolge sein kann. Er wird durch Anführungszeichen gekennzeichnet. Strings lassen sich durch „Addition“ verketten: Wenn man z. B. „x“ + „y“ „rechnet“, kommt „xy“ raus. Ein anderer Typ ist ***Boolean***, welcher entweder den Wert ***True*** oder ***False*** haben kann. Weitere Typen sind Zahlentypen wie ***integer***, was ganze Zahlen sind, und ***float***, was Dezimalzahlen sind.

5.2 Öffnen von Dateien

Nach dem Kennenlernen von Python und dessen Grundbefehlen war die nächste Aufgabe das Öffnen und Schließen von Dateien, sowie das Einlesen von Daten. Diese Fähigkeit

brauchten wir, um die in einer Datei gespeicherten Daten vom Hipparcos-Katalog einzulesen und weiterzuverarbeiten.

Um dieses Wissen zu erlangen, erlernten wir die Grundkenntnisse mithilfe von einfachen Textdokumenten. Zum Öffnen und Schließen von Dateien gibt es zwei verschiedene Möglichkeiten:

1. Man öffnet und schließt die Dateien manuell mithilfe von ***open()*** und ***close()***. Dabei wird die geöffnete Datei in einer Variable gespeichert (hier: x). Am Ende wird mit ***x.closed*** überprüft, ob die Datei geschlossen wurde (Abb. 5.1).

```
x = open("Beispiel 1.txt", "r")
x.close()
print x.closed
```

Abbildung 5.1: Öffnen und Schließen einer Textdatei mithilfe der Funktionen ***open*** und ***close***.

2. Man öffnet die Datei wieder mit ***open()*** und speichert sie in einer Variable. Dieses Mal öffnet man aber die Datei mithilfe eines ***with***-Satzes. Dadurch wird die Datei automatisch geschlossen, sobald man den ***with***-Satz verlässt (Abb. 5.2).

```
with open("Beispiel 1.txt", "r") as x:
    print x.closed
print x.closed
```

Abbildung 5.2: Öffnen und Schließen einer Textdatei mithilfe eines ***with***-Satzes.

5.3. Lesen von Dateien

Die Funktion ***open()*** besitzt zwei sogenannte Parameter, also spezifische Informationen zur Ausführung der Funktion. Bei ***open()*** braucht man einerseits den Dateinamen, sowie den Modus, in dem die Datei geöffnet werden soll. Dabei kann man zwischen read, write und append wählen.

Bei dem Modus ***r*** für read kann man nur die Datei lesen und nichts hinzufügen oder ändern. Zum Lesen von Dateien gibt es verschiedene Möglichkeiten. Der einfachste Befehl ist ***read()***. Dabei wird der gesamte Inhalt der Datei ausgegeben. Jedoch kann man ***read()*** auch einen Parameter übergeben, z. B. ***read(10)***. Dabei werden die ersten zehn Zeichen der Datei, die Buchstaben, Ziffern oder auch Leerzeichen sein können, ausgegeben.

5.4 Schreiben von Dateien

Wenn man die Datei aber im Modus ***w*** für write öffnet, kann man den Inhalt verändern. Bei ***write()*** gibt man zusätzlich als Parameter an, was man schreiben will.

Ein weiterer Modus ist ***a*** für append. Auch hier kann man mit dem Befehl ***write()*** in die Datei schreiben. Dabei wird das Geschriebene immer hinten angehängt, wodurch man nicht entscheiden kann, wo man in die Datei schreibt wie beim Modus write.

5.5. Modul *astropy*

Nachdem wir uns mit den Grundkenntnissen vertraut gemacht haben, griffen wir auf die Dateien des Hipparcos-Kataloges zu, welche in einer Datei mit dem Dateiformat VOTable oder FITS gespeichert wurden. Um auf diese Dateiformate zugreifen zu können, brauchten wir in Python sogenannte Module, das sind Sammlungen von Definitionen und Anweisungen, welche die in Python bereits integrierten Befehle und Funktionen erweitern. Für VOTable und FITS brauchen wir konkret das Modul ***astropy.io***. Um auf Module zugreifen zu können, muss man diese erst innerhalb des Programms importieren (siehe Abb. 5.3). Dabei gibt es drei Möglichkeiten:

1. Generic Imports: Dabei wird lediglich das Modul importiert und über das Modul kann man auf dessen Funktionen zugreifen.
2. Function Imports: Nur die benötigten Funktionen werden importiert.
3. General Imports: Sämtliche Funktionen des Moduls werden importiert.

```
import math
print math.sqrt(25)
from math import sqrt, sin, cos
print sqrt(25)
from math import *
```

Abbildung 5.3: Verschiedene Möglichkeiten zum Importieren des benötigten Moduls mithilfe des Befehls ***import***.

Wenn die Daten des Hipparcos-Kataloges im Dateiformat FITS gespeichert wurden, muss man zuerst die Funktion ***fits*** vom Modul ***astropy.io*** importieren. Mithilfe dieser Funktion kann man dann die Datei öffnen und speziell auf die Tabelle mit den Daten des Kataloges zugreifen. Man kann entweder die gesamte Tabelle ausgeben, nur eine bestimmte Spalte, eine bestimmte Zeile oder einen bestimmten Wert mithilfe der Zeilenangabe und des Spaltennamens. Diese Fähigkeit, auf Daten einer bestimmten Spalte zugreifen zu können, brauchten wir in unserem nächstem Schritt: Der Verarbeitung der Daten.

6. Programmablaufsteuerung und Einstieg in die Grafikprogrammierung

Timo, Corvin

Anschließend war es wichtig zu erfahren, wie man die Daten weiterverarbeiten kann. Dazu ist vor allem die gezielte Programmablaufsteuerung essenziell. In einem weiteren Vortrag wurden uns deshalb nach einer kurzen Erklärung dazu, wie man in Python rechnet, drei verschiedene Werkzeuge näher erläutert, die helfen, den Ablauf eines Programms durch Gliederung in bestimmte Abschnitte, sogenannte Codeblöcke, zu steuern.

6.1 Schleifen

Zuerst wurde gezeigt, wie man durch Nutzung einer Schleife einen Programmteil schreibt, der immer wieder wiederholt wird. Damit wird der Code durch Vermeidung unnötiger Wiederholungen verkürzt und vereinfacht.

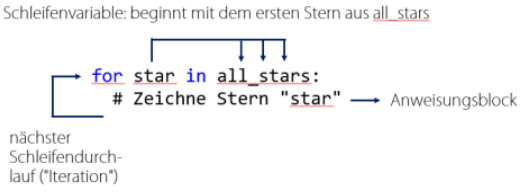


Abbildung 6.1: Eine Schleife hilft, nicht jeden Stern einzeln zeichnen zu müssen.

In unserer Sternkarte half uns die Schleife zum Beispiel dabei, nicht jeden Stern einzeln zeichnen zu müssen, sondern einmal zu definieren, was sich für jeden Stern ändern soll. Das wurde dann vom Programm für jeden Stern wiederholt. (s. Abb. 6.1)

6.2 Entscheidungen

Nachdem alle die erste kleine Übungsaufgabe zu den Schleifen gelöst hatten, erfuhren wir, wie ein Programm Entscheidungen treffen kann. Dazu führt man einen Programmteil nur dann aus, wenn eine bestimmte Bedingung erfüllt ist (s. Abb. 6.2).

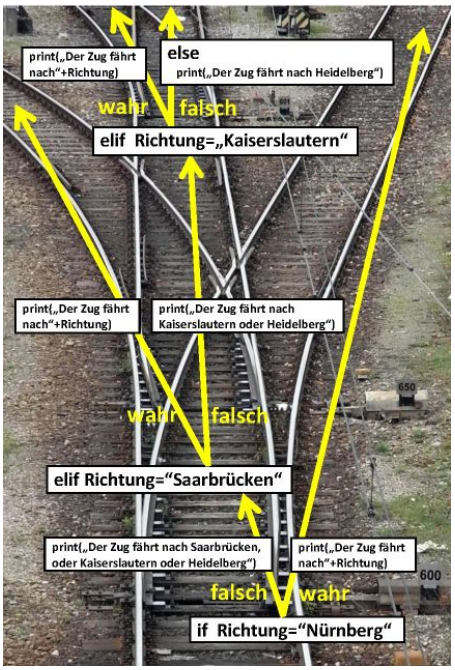


Abbildung 6.2: So wie eine Weiche einen Zug auf ein bestimmtes Gleis lenkt, wenn eine Bedingung zutrifft, so dirigieren die Schlüsselwörter ***if***, ***elif*** und ***else*** die Programmverlauf.[3]

Dazu verwendet man die Schlüsselwörter ***if***, ***elif*** und ***else***. Nach ***if*** oder ***elif*** gibt man die Bedingung und den dann ausgeführten Codeblock an, nach ***else***, was passieren soll falls die obigen Bedingungen nicht zutreffen. Ein Beispiel ist in Abb. 6.3 zu sehen.

```
if antwort >= 10:
    print "Mindestens 10!"
```

[3]Bild basierend auf: Wikimedia, Wikimedia-User Asiano, CC-BY-SA

```
elif antwort >= 5:
    print "Mindestens 5!"
else:
    print "Kleiner als 5 ..."
```

Abbildung 6.3: Die erste Bedingung trifft zu, wenn die Zahl mindestens zehn ist, die zweite, wenn sie mindestens fünf ist, und falls beide falsch sind, die letzte.

6.3 Funktionen

Zum Schluss des Vortrags wurden wir in ein weiteres Hilfsmittel zur Strukturierung von Programmen eingeführt: Funktionen. Diese ermöglichen es Programmierern, bestimmte Codeabschnitte, die für eine übergeordnete Aufgabe zuständig sind, zusammenzufassen. Ist eine Funktion einmal definiert, kann sie im Programm mehrmals aufgerufen werden. Dabei können der Funktion immer unterschiedliche Werte als Argumente mitgegeben werden, die die Funktion dann entsprechend der Definition verarbeitet und zurückgibt. Für unser Projekt der Sternkartenerstellung war dieses Prinzip essenziell, da wir für fast alle Teilaufgaben Funktionen geschrieben haben.

Die neu gelernten Werkzeuge wurden dann für ein Übungsprogramm benutzt, das verschiedene Sterndaten aus dem Hipparcos-Katalog liest, verarbeitet und ausgibt.

6.4 Grafik mit *Matplotlib*

Anschließend haben wir in der nächsten Präsentation eine Einführung in das Grafikmodul ***Matplotlib*** bekommen, das für unseren Kurs Grundlage für die Grafikausgabe war. Ein Modul stellt Definitionen und Anweisungen zur Verfügung, ähnlich wie bei Heimwerkern der Werkzeugkoffer, die in diesem Fall für die Grafikprogrammierung dienen. Angefangen hat das Ganze mit einer Übersicht der mit ***Matplotlib*** möglichen Daten-Darstellungsarten. So bekamen die Teilnehmer verschiedene Diagrammtypen gezeigt.

Ein Diagrammtyp war das Streudiagramm, das später auch direkt seinen Zweck in der Sternkarte gefunden hatte. Damit ist es möglich, an definierten Positionen Scheibchen mit bestimmten Farben und einer bestimmten Größe zu drucken. Dies wurde in der Sternkarte für die Darstellung der unterschiedlich hellen und farbigen Sterne benutzt (siehe Abb. 6.4).

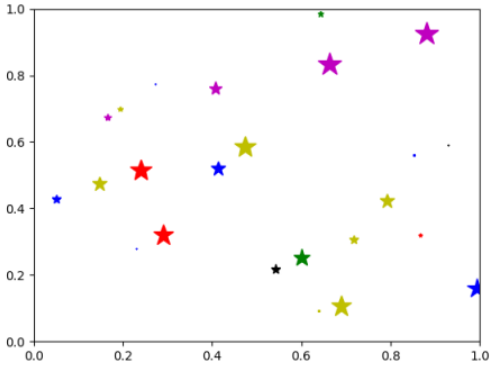


Abbildung 6.4: gefärbte Sterne an zufälligen Positionen mit zufälliger Größe.

Eines der einfachsten Diagramme ist der ganz normale Graph. Man kann benutzerdefinierte Linien erzeugen, die ihren Zweck in der Darstellung der Sternkonstellation fanden. So war es uns möglich, mithilfe der Sternkoordinaten und einer zusätzlichen Datei die sternbildorientierten Verbindungslinien zu zeichnen. Diese Datei stammt aus Stellarium, einer Software, die eine virtuelle Himmelsbeobachtung ermöglicht.

Weitere Diagramme waren das Balkendiagramm, das benutzt wird, um große Mengen an Daten zu visualisieren. Das Stapeldiagramm wird verwendet, um Zusammenhänge zwischen zwei Größen darzustellen, beispielsweise die Stundenzahl eines Tages und die verschiedenen Tätigkeiten (Schlafen, Essen etc.). Das letzte Diagramm war das Kuchendiagramm.

Für unsere Sternkarte nicht unbedingt notwendig aber trotzdem nützlich sind sogenannte Subplots (Plot = Zeichenfeld, der Bereich auf dem ein Graph dargestellt wird). Diese können verwendet werden, um mehrere Diagramme in einer Grafik darzustellen. Das heißt, es ist nicht notwendig, die verschiedenen Plots extern miteinander zu einer großen Grafik zusammenzufügen.

Ein weiteres Tool ist das Raster, das ***Matplotlib*** zur Verfügung stellt. Sofern man es aktiviert, dient es dem besseren Zurechtfinden in der Grafik. Mit ihm kann man die Werte für bestimmte Punkte besser ablesen.

Am Ende wurden auch noch 3D-Grafiken vorgestellt, die für unsere Sternkarte aber keine direkte Verwendung fanden (siehe Abb. 6.5).

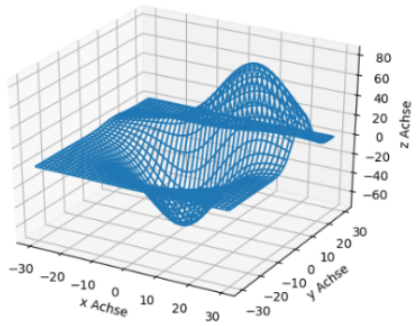


Abbildung 6.5: 3D-Graph mit Testwerten.

7. Grafikprogrammierung für die Sternkarte

Annika, Filip

Um unser Kursziel zu erreichen, mussten wir uns zunächst darüber informieren, wie sich die visuelle Darstellung der Sternkarte in Python gestaltet. Hierzu war eine Menge Vorarbeit nötig, um herauszufinden, welche Funktionen für das Zeichnen einer Sternkarte geeignet sind und welche nicht.

7-1 Sterne in einer Karte – wie und wo

Zum Zeichnen der Sterne in einer Sternkarte in Python bietet sich die Funktion ***scatter*** an. Wird sie aufgerufen, gibt die Funktion beliebig viele Scheibchen, die die Sterne darstellen, aus. Die für eine Sternkarte relevanten Parameter sollen im Folgenden kurz benannt und erklärt werden.

Die wichtigsten Parameter sind die x- und die y-Koordinate und somit die Position der Sternscheibchen auf der Karte. Die Daten, die man für die Karte verwendet, müssen erst umgerechnet werden (siehe Kapitel 9) und werden dann in Listen gespeichert (D in Abb. 7.1). ***scatter*** verwendet dann x- und y-Werte aus den Listen für die Sternscheibchen.

```
import matplotlib.pyplot as plt
''' ... '''
''' A '''
def convertPosition(ra, dec):
        ''' ... '''
        return x, y
''' B '''
def getPointSize(m)
       ''' ... '''
       return mag_size[imag]
''' ... '''
''' C '''
ra = hip_field['RArad'][idx_list]
dec = hip_field['DErad'][idx_list]
mag = hip_field['Hpmag'][idx_list]
''' D '''
x, y = convertPosition(ra, dec)
m = [getPointSize(mag[n]) for n in
    range(len(x))]
''' E '''
plt.scatter(x, y, s=m)
```

Abbildung 7.1: Beispielhafte Verwendung der Scatter-Funktion in Python.

7.2 Darstellung der Sterneigenschaften Helligkeit und Farbe

Am Nachthimmel erscheinen die Sterne unterschiedlich hell (siehe Abschnitt 3.1). Um eine naturgetreue Darstellung zu erhalten, stellt man das grafisch dar, indem man zum Beispiel unterschiedlich große Scheibchen verwendet. Auch hierfür kann man eine Funktion (B in Abb. 7.1) schreiben, in der man die scheinbare Helligkeit der Sterne einer sinnvollen Scheibchengröße zuordnet.

Eine weitere Eigenschaft des Sternlichts ist seine Farbe (siehe Abschnitt 3.2). Die Sternscheibchen in der Karte sollen entsprechend eingefärbt werden. Da es keinen vorgefertigten Farbverlauf mit den Sternfarben gibt, muss man einen eigenen erstellen, den man ***Colormap*** nennt. Die ***Colormap*** bildet einen normierten Wertebereich von 0 bis 1 auf Farben ab. Die Farbe bestimmt sich durch die Kombination von Intensitätswerten für die Farben Rot, Grün und Blau sowie den Transparenzwert Alpha (A in Abb. 7.2). Der Intensitätswert einer Farbe wird über Stützstellen definiert. Zwischen die Stützstellen wird dann eine gedachte Gerade gelegt, um die Zwischenwerte zu berechnen. Beim Aufrufen der ***scatter***-Funktion muss man nun die Liste mit den Eingangswerten für die Farbe

aus dem Sternkatalog (B-V-Werte) und den Namen der selbst im Programm erstellten ***Colormap*** angeben (B in Abb.7.2). Die Funktion bildet die Werte dann automatisch auf die ***Colormap*** ab.

```
''' ... '''
''' A '''
cdict1 = {
    'red': ((0.0, 0.0, 0.0),
            (0.17, 0.3, 0.3),
            (0.4, 1.0, 1.0),
            (1.0, 1.0, 1.0)),
    'green': ((0.0, 0.0, 0.0),
            (0.17, 0.3, 0.3),
            (0.4, 1.0, 1.0),
            (1.0, 0.0, 0.0)),
    'blue':  ((0.0, 1.0, 1.0),
            (0.17, 0.0, 0.0),
            (0.4, 0.0, 0.0),
            (1.0, 0.0, 0.0)),
    'alpha': ((0.0, 1.0, 1.0),
            (0.17, 0.0, 0.0),
            (0.4, 0.0, 0.0),
            (1.0, 0.0, 0.0))
 }

 blue_red1 = LinearSegmentedColormap(
    ´BlueRed1´, cdict1)
''' ... '''
''' B '''
plt.scatter(x, y, s=m, c=color, cmap=
    blue_red1)
```

Abbildung 7.2: Erstellung und Verwendung einer beipielhaften Colormap in Python.

7.3 Bilder und Beschriftungen für die Sternkarte

Ein weiteres Ziel unseres Kurses war es, die Sternkarte mit selbst gezeichneten Sternbildfiguren zu bebildern und gegebenenfalls auch zu beschriften.

Bevor man damit anfangen konnte, musste man sich überlegen, welche Module hierfür notwendig sind und welchen Grafikdateityp man für die einzufügenden Bilder verwendet. Die Entscheidung fiel dabei auf das Grafikformat „png“, da es viele unterschiedliche Farbtiefen sowie transparente Bilder unterstützt.

Die folgenden Abb. 7.3 und 7.4 zeigen die programmtechnische Umsetzung, wie in Python mit Bildern und Beschriftungen umgegangen wird.

```
# .*. coding: utf-8 -*-
''' A '''
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import numpy as np
from scipy import ndimage
plt.axis([0, 800, 0, 600])
''' B '''
bild0 = mpimg.imread("bild.png")
bild1 = mpimg.imread("bild.png")
''' C '''
bild0 = ndimage.rotate(bild0, 45)
''' D '''
plt.imshow(bild0, alpha=0.5, zorder=1,
     extent=[500, 800, 300, 500])
plt.imshow(bild1, alpha=0.7, zorder=0,
     extent=[400, 700, 300, 500])
```

Abbildung 7.: Beispielhafte Verwendung von Bildern in Python 2.7 (siehe [7.1] und [7.2]).

In A werden die notwendigen Module in das Programm inkludiert. Mit Hilfe der Funktionen in B werden die Bilder in das Programm importiert. Die dort verwendete Funktion ***mpimg.imread*** benötigt lediglich den Dateinamen als Argument. Danach wird in C eine Funktion zur geometrischen Bildtransformation verwendet. Diese rotiert das als Parameter übergebene Bild um einen bestimmten Wert, welcher als zweites Argument übergeben wird. In diesem Fall wird „bild0“ um 45° gedreht.

Die Funktion ***plt.imshow*** in D gibt die Bilder schließlich aus. Hierbei wurde der Parameter ***alpha*** verwendet, um die Transparenz der Bilder anzupassen, wobei 0 durchlässig und 1 undurchlässig entspricht. Mit Hilfe des zweiten Parameters ***zorder*** kann man die Abfolge der Bilder bestimmen. Das Bild mit dem kleinsten ***zorder***-Wert wird als erstes ausgegeben.

Zuletzt werden die Bilder im Koordinatensystem mittels des Parameters ***extent*** verschoben. Dieser Parameter gibt an, an welchen Koordinaten sich die Seiten des Bildes befinden sollen.

```
# .*. coding: utf-8 -*-
''' A '''
import matplotlib.pyplot as plt
''' B '''
plt.axis([0,10,0,10])
''' C '''
plt.text(1,1,"Text0")
''' D '''
plt.text(3,8,"Text1", fontsize=20)
''' E '''
plt.text(4,6,"Text2", rotation=30)
```

```
''' F '''
plt.show()
```

Abbildung 7.4: Die Programmzeilen zeigen, wie man in Python 2.7 Beschriftungen in ein Koordinatensystem einfügt.

Wie in Abbildung 7.1 wird auch hier in A das notwendige Modul eingefügt und in B ein Koordinatensystem erstellt. In C, D und E wird die Funktion ***plt.text*** für das Einfügen von Text verwendet. Diese Funktion benötigt mindestens drei Parameter: Die x- und y-Koordinate sowie den einzufügenden Text. ***plt.text*** besitzt zudem weitere Parameter, welche in D und E verwendet werden. Mittels der Parameter ***fontsize*** und ***rotation*** kann die Schriftgröße und die Zeilenneigung des Textes geändert werden. Zum Schluss wird das Endergebnis in F mit der Funktion ***plt.show*** ausgegeben.

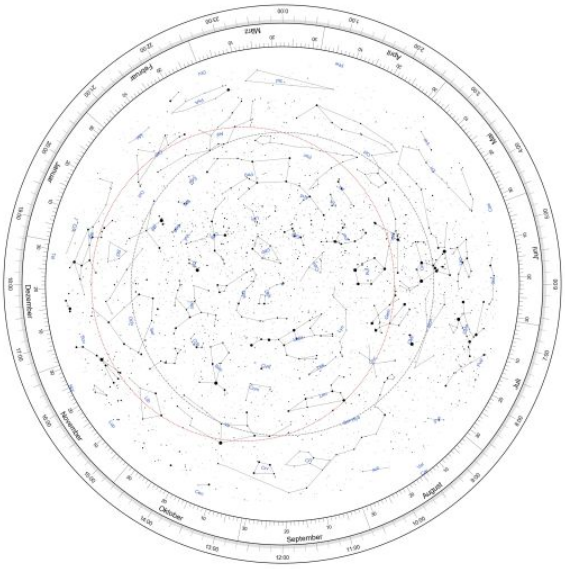


Abbildung 8.1: Grafikausgabe des Python-Programms von Dr. Thomas Müller – die HdA-Sternkarte.

8. Die HdA-Sternkarte

Annika

Als Vorlage für die Programmierung unserer eigenen Sternkarte verwendeten wir einen Python-Code, der uns von Herrn Dr. Thomas Müller vom Haus der Astronomie in Heidelberg freundlicherweise zur Verfügung gestellt wurde. Dieser gibt eine Sternkarte (Abb. 8.1) aus, die Sterne, Konstellationen und Konstellationsnamen anzeigt und auf den Daten des Hipparcos-Katalogs basiert. Zusätzlich hat die Sternkarte einen Datums- und Uhrzeitring und kann damit als Grundscheibe einer drehbaren Sternkarte genutzt werden.

Im Folgenden werden Aufbau und Funktion des Python-Programms von Dr. Thomas Müller in groben Zügen beschrieben. Zu Beginn werden alle wichtigen Module, wie z. B. das Grafikmodul ***Matplotlib***, importiert. Nun werden direkt nach den Importen allen Größen, die veränderbar sein sollen, Variablen zugewiesen. Thomas Müller hat die meisten Größen variabel gehalten, damit man die Karte nach den persönlichen Wünschen anpassen kann, wie zum Beispiel der geographischen Breite des Nutzungsortes.

Des Weiteren ist es möglich, die Verbindungslinien für die Sternbilder zu deaktivieren, indem man eine Variable auf „false“ setzt. Auch graphische Details wie Farben und Linien- bzw. Schrifttypen kann man gesammelt direkt unterhalb der Importe ändern.

Außerdem werden die Namen der nötigen Tabellen und Textdokumente in Variablen gespeichert. Im Anschluss werden mit Hilfe dieser Namen alle gewünschten Daten aus dem Hipparcos-Katalog ins Programm eingelesen und in Listen gespeichert. Auch Monatsnamen und Konstellationen mit zugehörigen Namen werden aus separaten Dateien eingelesen und gespeichert.

Danach werden alle notwendigen Funktionen definiert. Mithilfe einer Funktion wird die Größe der Scheibchen, die die Sterne darstellen, errechnet. Dazu werden die Magnituden in ganze Zahlen umgewandelt und auf eine Skala mit verschiedenen Scheibchengrößen projiziert.

Eine andere Funktion (in Abb. 8.2) errechnet die Position der Scheibchen auf der Karte, indem sie die Kugelkoordinaten Rektaszension und Deklination in kartesische Koordinaten umrechnet. Außerdem gibt es Funktionen, mit denen die Kreise von Himmelsäquator und Ekliptik (die jährliche scheinbare Bahn der Sonne) dargestellt werden. Im Programm ist des Weiteren eine Funktion zu finden, die ausgibt, welchen Bereich der Karte man am Nachthim-

mel sehen kann. Auch dieser Bereich muss für die Karte berechnet werden, wofür die Eingabe der geographischen Breite im Variablen-Block unter anderem nötig ist.

Nun wird die Scheibe, auf der die Karte abgebildet werden soll, mit allen Ringen für Datum und Uhrzeit gezeichnet und die Markierungen und die Beschriftung werden erstellt.

```
def convertPosition(ra, dec):
    """
     Convert position from
     (right ascension, declination)
     to Cartesian coordinates (x,y)
     depending on the global
     variable 'useStereoscopicProj'.

     Args:
     ra Right ascension
     dec Declination

     Return:
     Cartesian coordinates x, y in
         the domain [-1,1] x [-1,1]
    """

theta = 90. - dec
rar = np.radians(90 - ra)

if useStereoscopicProj:
    theta = np.radians(theta)
    x = np.sin(theta) * np.cos(rar) /
        (1 + np.cos(theta)) /
        scale_max
    y = np.sin(theta) * np.sin(rar) /
        (1 + np.cos(theta)) /
        scale_max
else:
   theta = theta / (90.0 -
       min_declination)
   x = theta * np.cos(rar)
   y = theta * np.sin(rar)

return x, y
```

Abbildung 8.2: In Python programmierte Funktion zur Errechnung der Positionen der Sterne auf der Sternkarte aus dem Code von Thomas Müller.

Dann werden je nach dem, wie es in den Variablen festgelegt wurde, die Funktionen für das Zeichnen von Sternen, Konstellationen mit Namen und dem Himmelsäquator und der Ekliptik aufgerufen. Die fertige Karte kann man sich direkt in der Konsole als Grafik ausgeben lassen oder man speichert sie in einem Format seiner Wahl ab.

9. Projekt Kurssternkarte

Adrian, Filip, Timo

Nachdem wir uns die Grundlagen zum Erstellen einer Sternkarte mithilfe von Python vorgestellt haben, waren wir in der Lage, in Form einer Projektarbeit eine solche Sternkarte selber zu programmieren.

Zuerst legten wir fest, wie unsere Sternenkarte am Ende aussehen sollte. Wir wollten eine Sternenkarte für 50° N und mit den 1000 hellsten Sternen erstellen. Dabei sollten die Sterne der jeweiligen Sternbilder verbunden und mit eigens gezeichneten Sternbildern hinterlegt sein. Zudem sollten die Sterne basierend auf ihrer scheinbaren Helligkeit unterschiedlich groß sein und mithilfe des Farbindexes ihre entsprechende Farbe annehmen.

Um dieses Ziel möglichst effektiv und innerhalb der uns zur Verfügung stehenden zwei Tage zu erreichen, teilten wir uns in drei Teilgruppen auf: Datenbringer, Datenmanipulierer und Datenzeichner.

Abbildung 9.1: Arbeit in Zweiergruppen.

Die Datenbringer sollten die Daten der Sterne beschaffen und innerhalb unseres Programms speichern. Die Datenmanipulierer wandelten die Daten daraufhin in eine für die Datenzeichner nützliche Form um. Die Datenzeichner visualisierten schlussendlich die Sternkarte.

Diese drei Teilgruppen griffen immer wieder auf die Informationen der zuvor gehaltenen Vorträge zurück, um das Ziel zu erreichen. Am Ende sollte jede Gruppe ihren Teil programmiert haben, welche dann zu einem großen Programm zusammengefügt wurden. Zudem zeichneten

einige parallel dazu Sternbilder, welche wir in unsere Sternkarte einbauen wollten.

9.1 Datenbeschaffung

Die erste Aufgabe war also, die Daten der 1000 hellsten Sterne in unser Programm zu laden. Diese Daten brauchten wir, um die Position der Sterne, ihre Größe und ihre Farbe zu ermitteln.

Dazu speicherten wir zuerst den Hipparcos-Index, die Helligkeit, die Rektaszension, die Deklination, den Farbindex und die scheinbare Helligkeit dieser 1000 hellsten Sterne in einer VOTable-Datei.

Um auf die dort gespeicherten Daten zugreifen zu können, mussten wir erst das entsprechende Modul ***astropy.io.votable*** importieren (siehe Abb. 9.2).

```
from astropy.io.votable import
parse_single_table
```

Abbildung 9.2: Programmzeile unseres Pythonprogramms, welche das Importieren des benötigten Moduls mit der Funktion ***import*** bewirkt.

Dadurch konnten wir nun mithilfe des Befehls ***parse_single_table*** die Datei in Python öffnen und mithilfe der Funktion ***array*** auf die in einer Tabelle gespeicherten Daten der Sterne zugreifen (Abb. 9.3).

```
HIPtable2 =
parse_single_table(data_name2)
HIPdata2 = HIPtable2.array
```

Abbildung 9.3: Öffnen der Datei und Zugriff auf die Tabelle mit den Sterndaten.

Als Nächstes erstellten wir vier Listen, in welche die Daten für die Verarbeitung gespeichert wurden: Eine Liste für die Helligkeit (*vmag*), eine für den Farbindex (*bv*) und zwei für die Koordinaten, also Rektaszension und Deklination (*rade* und *radec*) (Abb. 9.4).

```
vmag = []
rade = []
bv = []
radec = []
```

Abbildung 9.4: Erstellen der Liste für die scheinbare Helligkeit, die Rektaszension und der Deklination, sowie dem Farbindex der Sterne.

Nun griffen wir mithilfe einer Schleife auf jeden einzelnen Stern zu, holten uns die scheinbare Helligkeit, den Farbindex und die Koordinaten und speicherten diese in der entsprechenden Liste. Bei den Koordinaten speicherten wir die Rektaszension und Deklination jedes Sterns erst in einer Liste (*rade*) und fügten diese gemeinsam als Paar der finalen Liste für die Koordinaten hinzu (*radec*). Als letzten Schritt löschten wir die Daten immer aus *rade* (Abb. 9.5).

```
for star in range(lenght2):
  vmag.append(HIPdata2[star]["Vmag"])
  rade.append(HIPdata2[star]["RAhms"]
  rade.append(HIPdata2[star]["DEdms"])
  bv.append(HIPdata2[star]["B-V"])
  radec.append(rade)
  rade = []
```

Abbildung 9.5: Zugriff auf die Daten der Sternkarte mithilfe einer Schleife und speichern in den entsprechenden Listen durch ***append***.

Dadurch wurden am Ende sämtliche benötigte Daten der Sterne in diesen drei Listen gespeichert, auf welche man später leichter zugreifen kann als direkt auf die Datei.

9.2 Datenmanipulation

Während sich die erste Gruppe um den Programmabschnitt zur Beschaffung der Daten kümmerte, schrieb die Gruppe zur Datenmanipulation bereits einige Werkzeuge zur Veränderung und Umrechnung der Rohdaten zur Position der Sterne. Dafür war viel Denkarbeit zur Theorie notwendig, die in zwei Kernfunktionen mündete.

Umrechnung von sexagesimalen in dezimale Winkelangaben

Die Sternkoordinaten Rektaszension und Deklination sind im Hipparcos-Katalog in einem sogenannten sexagesimalen System angegeben, das heißt in Stunden, Minuten und Sekunden, die wie in Abbildung 9.6 jeweils mit dem Faktor sechzig umgerechnet werden (lat. *sexagesimus* = *der Sechzigste*).

Zur weiteren Berechnung und Projektion der Koordinaten benötigten wir diese allerdings in dem von Python genutzten dezimalen System, das heißt auf Zehnerbasis. Dafür haben

Abbildung 9.6: Umrechnung von sexagesimalen Werten jeweils mit dem Faktor 60.

wir eine eigene Funktion geschrieben (zu sehen in Abb. 9.7), die die Stunden-, Minuten- und Sekundenangaben der Daten des Hipparcos-Katalogs als Parameter annimmt und die Koordinaten als Dezimalzahl zurückgibt.

```
def hexToDecimal(hours, minutes,
    seconds):
        # Werte in Kommazahlen
            umwandeln:
        hours = float(hours)
        minutes = float(minutes)
        seconds = float(seconds)
        # Umrechnung in einen
            dezimalen Gradwert:
        h = hours + minutes/60 +
            seconds/3600
        return h
```

Abbildung 9.7: Python-Quelltext zur ersten Version der Funktion hexToDecimal().

Zusätzlich hatten wir das Problem, dass der erste Wert der Deklination zwar durchaus in Grad angegeben war, die beiden kleineren Angaben aber trotzdem in Minuten und Sekunden. Deshalb haben wir in unsere Funktion noch eine vierte Option eingebaut, ***unit***, bei der sich angeben lässt, ob der erste Wert in Stunden oder in Grad angegeben ist. Über eine ***if/else***-Entscheidung haben wir dann abhängig vom Wert des Parameters ***unit*** einen anderen Berechnungsweg gewählt (s. Abbildung 9.8). Am Ende hatten wir eine Funktion, die wir später bequem nutzen konnten, um einen sexagesimal angegebenen Winkel in einen dezimalen umzurechnen.

```
def hexToDecimal(hoursdegrees,minutes,
    seconds,unit):
    # Werte in Kommazahlen umwandeln:
    hoursdegrees = float(hoursdegrees)
    minutes = float(minutes)
    seconds = float(seconds)

    # Falls der erste Wert in Grad
        angegeben ist:
    if unit == "deg":
        h = hoursdegrees + minutes/60
            + seconds/3600

    # Falls der erste Wert in Stunden
        angegeben ist:
    elif unit == "hours":
        h = 15*hoursdegrees + 15*
            minutes/60 + 15*seconds
            /3600

    # Abfangen von Fehlern:
    else:
        print "unit needs to be either
            \"hours\" or \"deg\""
        h = 0
    return h
```

Abbildung 9.8: Endfassung der Funktion hexToDecimal().

Projektion der Koordinaten

Die zweite vom Datenverarbeitungsteam entwickelte Funktion diente der Projektion der beiden polaren Koordinaten Rektaszension und Deklination in die Ebene, also in kartesische x- und y-Koordinaten. Dementsprechend nannten wir unsere Funktion ***polarToCartesian()***. Als Argumente lassen sich Rektaszension und Deklination angeben (in Grad, deshalb unsere erste Funktion ***hexToDecimal***). Zurückgegeben werden die x- und y-Werte.

Aufruf der Funktionen

Nachdem unsere Funktionen einmal definiert waren, war es nicht mehr schwer, diese anzuwenden, um alle Sternkoordinaten, welche uns die Datenbeschaffungsgruppe in Form einer Liste gab, umzurechnen. Genutzt haben wir dazu eine Schleife, die jeden Wert der Liste durchläuft und diesen jeweils als *star_coords* speichert. *star_coords* wiederum enthält die beiden Sternkoordinaten, die wir den Variablen *ra* und *de* zuwiesen. Diese teilten wir mit der Funktion ***split()*** in die einzelnen Stunden- (bzw. Grad für die Deklination), Minuten- und Sekundenwerte auf. Anschließend konnten wir unsere zuvor programmierten Funktionen aufrufen und die Werte in dezimale Grad und anschließend x- und y-Koordinaten umrechnen. Diese Werte wurden in der Liste xs und ys gespeichert, die die Grafikgruppe für die Positionsangaben der Sterne nutzen konnte (alles zu sehen in Abb. 9.9).

```
xs = []
ys = []

for star_coords in radec:
    ra = star_coords[0]
    de = star_coords[1]
    ra_h, ra_m, ra_s = ra.split(" ")
    ra_degrees = hexToDecimal(ra_h,
        ra_m,ra_s,"hours")
    de_d, de_m, de_s = de.split(" ")
    de_degrees = hexToDecimal(de_d,
        de_m,de_s,"deg")
    x,y = polarToCartesian(ra_degrees,
        de_degrees)
    xs.append(x)
    ys.append(y)
```

Abbildung 9.9: Aufruf der Funktionen in einer Schleife.

9.3 Datenzeichner

Als „Datenzeichner“ waren im Großen und Ganzen die folgenden Aufgaben zu erledigen:

1. Wie erstellt man die Grundfläche, auf die die Sterne, Sternbilder und Verbindungslinien gezeichnet werden?
2. Wie zeichnet man die Sterne mit unterschiedlicher Farbe und Größe?
3. Wie erstellt man die Verbindungslinien für die einzelnen Sternbildkonstellationen?
4. Wie kommen die selbstgezeichneten Sternbilder in das Programm?

Wie erstellt man die Grundfläche?

In der Abbildung 9.10 wird gezeigt, wie man in Python mittels des Moduls ***Matplotlib*** eine Grundfläche für die Sternkarte erstellt. Hierbei wird zunächst ein Koordinatensystem in A erstellt, wobei sowohl die x-Achse, als auch die y-Achse von -1.6 bis 1.6 durchnummeriert wird. Anschließend wird mit Hilfe des ***print***-Befehls eine Meldung ausgegeben, dass die Grundfläche gezeichnet wird.

In B wird die schwarze Grundfläche in Form einer großen Scheibe erstellt. Innerhalb dieser schwarzen Grundfläche werden in C Kreise für den Himmelsäquator und die Ekliptik konstruiert, dabei wird der Himmeläquator in der Farbe Rot (***r***) und die Ekliptik in der Farbe Gelb (***y***) erstellt. In D werden diese Kreise in das Koordinatensystem eingefügt.

```
(A): ax=plt.axes(xlim=(-1.6, 1.6),
    ylim=(-1.6, 1.6))
ax.set_xticks(np.arrange(-1.0, 1.5,
    0.5))
ax.set_yticks(np.arrange(-1.0, 1.5,
    0.5))
ax.set_facecolor('w')

(B): print "Draw map ..."
ax.scatter(0,0, s=20000, color='k')

(C): equator=plt.Circle((0,0),0.69,
    color='r',fill=False,linewidth
    =0.3, alpha=1)
ecliptic=plt.Circle((-0.18,0),0.69,
    color='y',fill=False,linewidth
    =0.3, alpha=1)
circle=plt.Circle((0,0),1.4,color='w',
    linewidth=48,fill=False)

(D): ax.add_artist(circle)
ax.add_artist(equator)
ax.add_artist(ecliptic)
```

Abbildung 9.10: Ausschnitt des fertigen Programms, das zeigt, wie die Grundfläche für die Sternkarte entsteht.

Wie zeichnet man die Sterne mit unterschiedlicher Farbe und Größe?

Für die Farbe der Sterne wurde eine ***Colormap*** verwendet. Diese wurde so eingestellt, dass ein Stern mit dem Farbwert 0 blau ist und ein Stern mit dem Farbwert 1 rot. Abbildung 9.11 zeigt, wie eine solche ***Colormap*** in Python aussieht.

```
cdict = {
         'red':   ((0.0, 0.2, 0.2),
                   (0.2, 1.0, 1.0),
                   (0.4, 1.0, 1.0),
                   (0.8, 1.0, 1.0),
                   (1.0, 0.6, 0.6)),
         'green': ((0.0, 0.2, 0.2),
                   (0.2, 1.0, 1.0),
                   (0.4, 1.0, 1.0),
                   (0.8, 1.0, 1.0),
                   (1.0, 0, 0)),
         'blue':  ((0.0, 0.4, 0.4),
                   (0.2, 1.0, 1.0),
                   (0.4, 1, 1),
                   (0.6, 1, 1),
                   (0.8, 1, 1),
                   (1.0, 0.1, 0.1)),
```

```
        'alpha': ((0.0, 1.0, 1.0),
                  (0.2, 1.0, 1.0),
                  (0.4, 1.0, 1.0),
                  (1.0, 1.0, 1.0))
}
```

Abbildung 9.11: Colormap aus dem fertigen Programm.

Um diese Colormap konkret verwenden zu können, wurden die B-V-Werte der Sterne mit Hilfe der Funktion in Abbildung 9.12 in eine Dezimalzahl zwischen 0 und 1 umgerechnet.

```
def CalcColor(bv):
        color = (bv + 0.3) / 2.4
        return color
```

Abbildung 9.12: Umrechnung der B-V-Werte in für die Colormap brauchbare Dezimalzahlen.

Die Größe der Sterne wurde ebenfalls mit einer Funktion berechnet. Hierbei wurde den Sternen mit einer Magnitude kleiner als 0 oder größer als 6, als auch den Sternen mit einer gerundeten Magnitude von 1, 2, 3, 4 und 5 eine bestimmte Größe zugeordnet. Diese Größenzuordnung erfolgt in der Funktion ***CalcSize*** und ist in der Abbildung 9.13 zu sehen.

```
def CalcSize(vmag):
    sizes = []
    for i in range(countStars):
        if (vmag[i]<=0):
            sizes.append(mag_size[0])
        elif (vmag[i]>=6):
            sizes.append(mag_size[5])
        elif (vmag[i]<6 and vmag[i]>0):
            if (vmag[i]>0 and vmag[i]<2):
                sizes.append(mag_size[1])
            elif (vmag[i]>1 and vmag[i
                ]<3):
                sizes.append(mag_size[2])
            elif (vmag[i]>2 and vmag[i
                ]<4):
                sizes.append(mag_size[3])
            elif (vmag[i]>3 and vmag[i
                ]<5):
                sizes.append(mag_size[4])
    return sizes
```

Abbildung 9.13: Größe der Sternscheibchen nach ihrer scheinbaren Helligkeit bestimmen.

Mit Hilfe dieser beiden Funktionen wurde die Farbe und Größe jedes Sterns berechnet und mittels der ***scatter***-Funktion in Form einer kleinen Scheibe ausgegeben.

Wie erstellt man die Verbindungslinien für die einzelnen Sternbildkonstellationen?

Um nicht von Hand jedes einzelne Sternbild herauszusuchen und jeden dem Sternbild zugehörigen Stern in der richtigen Reihenfolge zu verbinden, haben wir auch hierfür eine Funktion geschrieben namens ***drawConstellations()***. Diese Funktion verwendet die Datei „constellationship.fab“ aus der Planetariumssoftware „stellarium“, um die Sterne richtig zu verbinden. In dieser Datei sind blockweise die Hipparcos-Nummern für jedes Sternbild gegeben und zwar in der zu verbindenden Reihenfolge. Die Funktion liest die Hipparcos-Nummern der einzelnen Sterne jedes Sternbildes heraus und sucht für die entsprechenden Hipparcos-Nummern die Rektaszension und Deklination jedes Sterns. Diese Polarkoordinaten werden dann in kartesische Koordinaten umgerechnet.

Zeichnen der Sternbilder

Während die Programmierteams sich noch um die technische Umsetzung kümmerten, haben einige der künstlerisch begabten Kursteilnehmer für einen Teil der Sternbilder eigene Illustrationen gezeichnet (Beispiel siehe Abb. 9.15). Dazu wurde erst nach Referenzbildern aus dem Internet recherchiert, die anschließend möglichst detailgetreu nachgezeichnet wurden. Die Zeichnung erfolgte auf zuvor ausgedruckten Sternkartenabschnitten, damit Bild und Sterne zueinander passen.

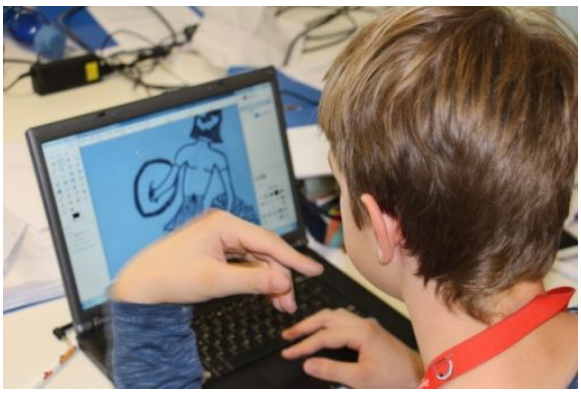


Abbildung 9.14: Ausschneiden des Perseus-Sternbilds in einem Bildbearbeitungsprogramm.

Um diese zu digitalisieren, fotografierten wir die Bilder ab. Damit diese dann auch ohne den

Tisch im Hintergrund in die Sternkarte eingefügt werden konnten, mussten wir sie dann noch im Bildbearbeitungsprogramm GIMP freistellen (siehe Abb. 9.14).

Abbildung 9.15: Zeichnung des Sternbilds Perseus.

Wie kommen die selbstgezeichneten Sternbilder in das Programm?

Bevor man die Bilder an die Sternbildkonstellationen anpassen konnte, mussten diese erst einmal in das Programm geladen werden. Abbildung 9.16 zeigt einen Ausschnitt, wie die Bilder mit Hilfe der Funktion ***mpimg.imread()*** in das Programm importiert wurden.

```
drache = mpimg.imread("image\Drache.
    png")
pegasus = mpimg.imread("image\Pegasus.
    png")
fische = mpimg.imread("image\Fische.
    png")
```

Abbildung 9.16: Ausschnitt des fertigen Programms, das zeigt, auf welche Art und Weise die Bilder in das Programm geladen wurden.

Die nun im Programm implementierten Bilder konnten jetzt an die Sternbildkonstellationen angepasst werden (Ort, Größe, Drehung).

Hierzu wurde im Allgemeinen die Funktion ***ndimage.rotate()*** verwendet, um die Bilder richtig zu drehen, sowie die Parameter der Funktion ***plt.imshow()***. Mit Hilfe des alpha-Parameters konnte man die Transparenz der Bilder justieren und somit ihre visuelle Dominanz an die der Sterne anpassen. Des Weiteren kam es zur aktiven Verwendung des Parameters ***extent***, welcher es uns ermöglichte, die Eckpunkte der Bilder mit denen der Sternbildkonstellationen gleichzusetzen. In Abbildung 9.17 sieht man, wie diese Mittel zum Anpassen der Bilder in der Praxis zur Verwendung kamen.

```
drache = ndimage.rotate(drache, 95,
    reshape = False)
plt.imshow(drache, extent = [-0.85,
    0.5, -0.6, 0.5], alpha =
    constAlpha, zorder = 6)
print "\t2 /", allImages
pegasus = ndimage.rotate(pegasus, 345,
     reshape = False)
plt.imshow(pegasus, extent = [0.7, -1,
     0.1, 0.95], alpha = constAlpha,
    zorder = 8)
print "\t4 /", allImages
fische = ndimage.rotate(fische, 20,
    reshape = False)
plt.imshow(fische, extent = [-0.1,
    0.6, 0.425, 0.74], alpha =
    constAlpha, zorder = 9)
print "\t5 /", allImages
```

Abbildung 9.17: Ausschnitt des fertigen Programms, in dem die Bilder an die Sternbildkonstellationen angepasst wurden.

Nachdem alle Teilgruppen ihre zugeteilten Programmabschnitte abgeschlossen hatten, wurden diese zusammengefügt. Nach einem letzten Feinschliff sowie ergänzten Kommentaren war das Programm bereit, die Sternkarte zu generieren. Das Ergebnis ist am Ende der Dokumentation unseres Kurses zu sehen

Abbildung 9.18: Begeisterung bei erfolgreicher erster Ausgabe der Sternkarte.

10. Exkursion zum MPIA

Simon, Hannah B., Corvin

Am Montag, den 04.09.2017 ging es für den Astronomiekurs im Rahmen einer Exkursion zum Max-Planck-Institut für Astronomie (MPIA) auf den Königstuhl in Heidelberg. Nach circa zwei Stunden Fahrt mit Bus und Bahn waren wir auf dem Gelände angelangt und konnten uns erstmal die Beine ein bisschen vertreten.

Unser Kursleiter Olaf führte uns zu Beginn auf dem Gelände herum, angefangen in der Landessternwarte Heidelberg. So erfuhren wir, dass die am 20. Juni 1898 durch Großherzog Friedrich I. von Baden eingeweihte Sternwarte durch den Astronomen Max Wolf betrieben wurde. Aufgrund der während der Industrialisierung ausgebauten Straßenbeleuchtung und der starken Luftverschmutzung konnte er in der Stadt, in der er davor den Sternenhimmel beobachtete, die Sterne nicht mehr klar erkennen.

Schon früh hatte Wolf begonnen, den Sternenhimmel zu fotografieren. Da er jedoch leistungsfähigere Teleskope brauchte, warb er um Sponsoren. In der Tochter von George Bruce, Catherine Wolfe Bruce, fand er eine Sponsorin. Catherine Wolfe Bruce, die auch anderen Sternwarten Geld gesponsert hat, unterstützte den Erwerb eines größeren, leistungsfähigeren Teleskops, das der Stifterin zu Ehren „Bruce-Teleskop“ genannt wurde. Dabei handelt es sich um einen Doppelastrographen. Im Laufe der Zeit gelang es so Wolf und seinen Mitarbeitern, den Sternenhimmel präzise abzulichten und Kleinplaneten zu entdecken. Dieses Teleskop besichtigten wir im Laufe des Nachmittags in einer Kuppel des Observatoriums. Die Tuben des Doppelastrografen, mit denen Bilder vom nächtlichen Sternhimmel geschossen werden, waren riesig. So lang wie eineinhalb Kursleiter.

Nachdem wir uns dieses imposante Teleskop angeschaut haben, gingen wir zu einem weiteren außergewöhnlichen Exemplar. Dieses war nämlich in einem Haus so installiert, dass es nur entlang des Himmelsmeridians bewegt werden konnte. Wollte man nun mit diesem den Nachthimmel betrachten, musste man die gesamte Hauswand beiseiteschieben. Anschließend betrachteten wir auch die aufgenommenen Fotoplatten, nahmen einen echten Meteoriten in die Hand, lernten den Unterschied zwischen Kometen, Meteoriten und Meteoren kennen und Zeit zum Herumalbern blieb natürlich auch.

Abbildung 10.1: Der Astronomiekurs auf dem Gelände der Landessternwarte Heidelberg.

Gegen Nachmittag sind wir ins Planetarium im Haus der Astronomie gegangen. Dort trafen wir, wie verabredet, den Astronomen und Informatiker Thomas Müller, der die Vorlage für unsere eigene Sternkarte programmiert hatte und uns sein Programm zur Verfügung stellte. Zuerst hielt er einen Vortrag über sich und seine Arbeit am Haus der Astronomie. Nach seinem Physikstudium in Tübingen und einer Doktorarbeit zur Visualisierung in der Relativitätstheorie beteiligte er sich an der Entwicklung zweier Einsteinausstellungen in Ulm und Bern. Nachdem er drei Jahre lang Software für Planetarien entwickelte, forschte er am Visualisierungsinstitut der Universität Stuttgart. Seit April 2016 ist er am Haus der Astronomie in Heidelberg tätig. Dort befasst er sich mit dem Visualisieren von astronomischen und astrophysikalischen Phänomenen, wie z. B. Schwarzen Löchern oder Wurmlöchern. Außerdem entwickelt er Visualisierungen der Relativitätstheorie und der Oberflächenstruktur von z. B. Erde und Mond, Software und Apps für Planetarien sowie Programme für Schulen.

Nach einer anschließenden Fragerunde, bei der wir Thomas Müller Fragen zum Programm, aber auch zu seiner Arbeit am MPIA stellen konnten, musste er leider auch schon wieder gehen. Programme programmieren sich leider nicht von allein. Da unser Kursleiter Olaf auch im Haus der Astronomie arbeitet, konnte er uns

noch das Planetarium zeigen. Zuerst wiederholten wir die unterschiedlichen Sternbilder des nördlichen Himmels, die wir von unserer Nachtwanderung kannten, und stellten verschiedene Sternbilder dar. Zum Beispiel ließen wir die Tierkreissternbilder auf der Ekliptik erscheinen. Anschließend schwenkte Olaf zum südlichen Himmel, der für uns neu war, und erzählte uns mehr über die Sternbilder und Geschichten dahinter. Nachdem wir noch ein bisschen Zeit mit der Reflexion des Besuchs verbrachten, posierten wir mit unserer fertig programmierten und ausgedruckten Sternkarte, die wir auch Thomas Müller überreichten, vor dem Haus der Astronomie. Nach einem kurzen kulinarischen Ausflug in der Stadt, in der wir uns die unterschiedlichsten Arten von Burgern schmecken ließen, nahmen wir die Bahn zurück nach Adelsheim. Ein aufregender Tag voller neuer Erkenntnisse über den Nachthimmel endete.

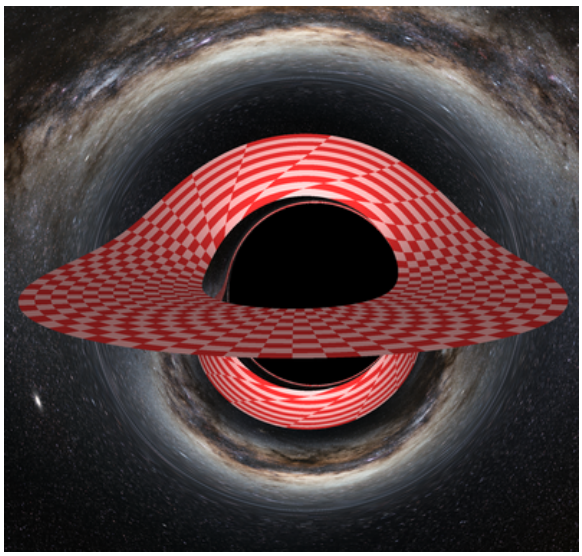


Abbildung 10.2: Visualisierung eines Schwarzen Lochs (Thomas Müller, Universität Stuttgart).

Kursteilnehmer Astronomie

Mattea, Ole, Hannah D.

Hannah Mit ihrem Humor wusste sie an vielen langen Arbeitstagen die Stimmung im Kurs aufzuheitern. Noch dazu arbeitet sie hervorragend im Team. Sie brachte sich vor allem auch organisatorisch bei der Bewältigung von kursinternen Aufgaben ein. Ihr musikalisches Talent durften wir hautnah am Hausmusikabend erleben. Dort spielte sie vier Instrumente. Darüber hinaus konnte sie sich im Spiel „Sternenstaub“, bei welchem man mit Zahnbürsten Astros in „Sternenstaub“ verwandeln sollte, gemeinsam mit Corvin bis ins Finale durchkämpfen.

Adrian Adrian ist sehr politikinteressiert, weswegen er scherzhaft „der Politiker“ genannt wurde. Da diese Science Academy kurz vor der Bundestagswahl stattfand, verging kein Tag, an dem er nicht mindestens einmal den „Wahl-O-Mat“ benutzt hatte. Darüber hinaus programmiert er gerne und hat uns mit seinem Votrag über die Computersprache Python einen tollen Einstieg in diese ermöglicht.

Simon Der „Kaffeemensch“, der mit seiner Kaffeemaschine jederzeit bereit war, unsere Adenosinrezeptoren zu binden. Er ist außerdem ein begeisteter Physiker und interessiert sich sehr für schwarze Löcher und die Quantenmechanik. Mit seinem Wissen brachte er sich gerne ein und sorgte außerdem für eine humorvolle Arbeitsatmosphäre. Das half dem Kurs sehr.

Paula Sie zeichnete sich im Kurs vor allem durch ihre Kreativität aus. So gestaltete sie das Sternbild der großen Bärin und hängte ihr einen riesigen Fuchsschwanz an. Das sorgte im ganzen Kurs für Erheiterung. Außerdem war sie immer gut drauf und förderte den Kurs vorallem durch ihre Kenntnisse in den verschiedensten Kartenprojektionen.

Ole Der „sprücheklopfende Biochemiker“. Mit seiner Leidenschaft für biochemische Prozesse im Körper ergänzte er die Truppe. Er hatte trotz der vielen Arbeit für seine Präsentation immer gute Laune und viele Sprüche auf Lager, von denen er gerne Gebrauch machte. Auch war er bei der Sport KüA immer vorne mit dabei und beim Zeichnen von Sternbildern stach wohl seine Cassiopeia, deren Extremitäten fehlten und die aufgehängt war, besonders heraus. Und in Wirklichkeit isch sein Bein auch ganz gerade.

Annika Annika hat uns alle mit ihren Programmkenntnissen und ihrer schon angefangenen Sternkarte ganz schön überrascht. Zusammen mit Filip legte sie so manche

Nachtschichten ein, um unser Projekt rechtzeitig zur Exkursion zu vollenden. Neben ihrer Arbeit im Kurs hat sie uns auch mit ihrer Geige am Hausmusikabend begeistert. Mit ihrer freundlichen und hilfsbereiten Art ist sie definitiv eine Bereicherung für unseren Kurs gewesen.

Corvin Ein weiterer Programmierexperte, vor allem für Grafiken, war Corvin. Neben vielen Fun Facts konnte er uns allen beibringen wie man Diagramme und Schaubilder möglichst optisch ansprechend in der Programmiersprache Python erstellt. Trotzdem schaffte er es, auf seinen Computerbildschirm des Öfteren die Aussage „kernel died“ zu zaubern. Bei all diesen Fähigkeiten war er außerdem auch ein großartiger Wildhüter.

Mattea Mattea kann man wohl als ein wahres Energiebündel bezeichnen. Immer neue Ideen und Einfälle sind aus ihr herausgesprudelt und haben die Zusammenarbeit positiv geprägt. Beim Wildhüten war sie ein echter Profi und motivierte auch die anderen dazu die Umwelt mit offenen Augen zu betrachten. Sie teilte mit uns allen eine Vorliebe für Schoko- Cookies.

Hannah Hannah ist die Kreativität in Person. Egal ob sie sich in ihrem Vortrag mit der Kunst in Sternkarten beschäftigte oder für die kurseigene Sternkarte Bilder zeichnete. In ihrem Vortrag ließ sie uns an ihrem Wissen über die Mythen und Sagen der Antike teilhaben, die den Sternbildern ihre Namen geben. Außerdem haben wir es nur Hannahs hervorragendem Orientierungssinn (nicht nur am Nachthimmel) zu verdanken, dass wir nicht immer noch durch die Wälder rund um Adelsheim irren. Denn sie führte uns auch ohne Taschenlampe sicher zum Ziel.

Timo Timo ist ein echter Gewinner-Typ. Mit seiner weltrekordverdächtigen Leistung beim Teebeutelweitwurf sicherte er unserem Kurs den Sportfest-Sieg. Trotz dieser herausragenden Leistung konnte man ihn jedoch nicht bei den Sport-KüAs antreffen, denn sein Herz schlug für das Theater.

Außerdem gab es kein Thema, über das man nicht mit ihm diskutieren konnte. Dabei hat er seinen Standpunkt nicht nur verbal sondern auch anhand von umfangreichen Tafelanschrieben (Skizzen) verdeutlicht.

Lilly Lilly ist der wertvollste französische Re-Import der letzten Jahre. Nach 12 Jahre „Baguette und Camembert“ kehrte sie zu „Spätzle und Vollkornbrot“ zurück – welch ein Glück. Denn sie leistete in jedem Bereich des Kurses wertvolle Beiträge und fand für alles eine Erklärung. Sowohl mit ihrem Vortrag zur Einführung in die Programmiersprache Python als auch mit ihrer netten und freundlichen Umgangsart oder auch durch viele selbstgezeichnete Sternbilder trieb sie die Kursarbeit voran. Dass sie selbst nach der Abend-KüA noch die Energie aufbrachte, Klavier zu üben, haben wir alle sehr bewundert.

Filip Wenn jemand Vatergefühle für eine Sternenkarte entwickeln kann, dann Filip. Als der „Papa“ unserer Sternkarte, konnte er nach deren Geburt (es handelte sich dabei lediglich um einen ersten Ausdruck) unser „Baby“ nicht mehr aus der Hand legen. Beim Programmieren leistete er gerne Überstunden und war manchmal noch bis nach Mitternacht an seinem PC zu finden. Er wusste selbst dann noch weiter, wenn manche nur noch „Bahnhof“ verstanden. Auch durch seine ruhige und entspannte Art hat er die Produktivität unseres Kurses deutlich gesteigert.

Tatjana Tatjana hatte als Leiterin für uns alle ein offenes Ohr. Egal ob es dabei um Probleme im Programm ging oder um die Nervosität vor der Abschlusspräsentation. Klasse war auch ihr kursinterner Vortrag: „Wie geht Studieren?“. Die Leiter konnten sie bis spät in die Nacht knobeln sehen. Allerdings hatte sie so ihre Probleme beim Wildhüten. Beim Quidditch macht sie Ginny Weasley Konkurrenz.

Olaf Olaf ist der „Herr der Sterne“. Als unser Kursleiter war er immer bemüht, sein unglaubliches Wissen über das Weltall mit uns zu teilen und uns alle an seiner Begeisterung für die Astronomie teilhaben zu lassen. Dabei hat er jede noch so „doofe“ Frage geduldig beantwortet und immer ein

Lächeln für uns gehabt.

Wenn wir mal wieder vom Thema abkamen, sorgte Olaf dafür, dass wir konzentriert blieben und uns an den Zeitplan hielten – zum Glück, denn sonst wären wir niemals so weit gekommen.

Ranran Sie ist die Motivation in Person, denn als unsere Schülermentorin wusste sie uns immer zu motivieren. Besonders kam uns das beispielsweise beim Sportfest und bei der Kursarbeit allgemein zugute. Ein Markenzeichen sind ihre lustigen Sternbildsocken, welche im Astronomiekurs natürlich nicht fehlen dürfen. Doch auch nach der Kursarbeit brachte sie uns mit lustigen Spielen wie dem Evolutionsspiel zum Lachen.

Nachtwanderung

Lilly

Kurze Zeit nach unserer Ankunft in der Akademie wurde der Astrokurs mit der Planung und Vorbereitung der Nachtwanderung, beauftragt.

Dabei begleiteten je zwei Astros eine Gruppe von 10 Schülern und ein paar Leitern auf eine Wanderung und brachten ihnen das im Kurs erlernte, astronomische Wissen über Sterne, Sternbilder und ihren Geschichten näher. Am Montag, den 28. August sollten die Teilnehmer der Akademie um 22 Uhr zu einem 30-minütigen Marsch durch den dunklen Wald aufbrechen. Dieser endete auf einem freien Feld, das sich sehr gut für das Beobachten der sichtbaren Objekte, wie zum Beispiel Sterne, Planeten oder Galaxien, am Himmel über Adelsheim eignete.

In den drei zur Verfügung stehenden Vorbereitungstagen erarbeitete sich die Astros jegliches benötigtes Wissen für die anstehende Nachtwanderung. Das schloss sowohl die Orientierung am Sternenhimmel, als auch die Route und den Aufbau eines Teleskops mit ein. Dabei hatten alle viel Spaß, nach und nach immer mehr Sternbilder am Nachthimmel zu erkennen und den teilweise sehr phantasievollen Mythen über ihre Herkunft zu lauschen. Um die verschiedenen Sternbilder zu zeigen, nahmen die Kursleiter stark lichtbündelnde Taschenlampen zur Hilfe, deren Strahlen weit in den Himmel hinein ragten. Dieselben nutzen auch die Astros als Hilfsmittel während ihrer Vorträge. Tagsüber, wenn es unmöglich war Sterne zu beobachten, beschäftigten sie sich mit dem Auf- und Abbau von Teleskopen, weil diese schließlich auch bei der Nachtwanderung zum Einsatz kamen. Außerdem unternahm der gesamte Kurs eine Probewanderung zum gewünschten Beobachtungsplatz, damit sich jeder den Weg einprägt und sich keiner in der Dunkelheit verirrt. Das wiederholten die Astros am selben Tag nochmal bei Nacht als „Generalprobe“ und überprüften, ob alles sitzt für die Nachtwanderung. In dieser Nacht kam der Astrokurs erst um 23:30 Uhr auf dem Campus zurück, während alle anderen schon lange schlafend in ihren Betten lagen. Was tut man nicht alles für den besten Kurs?

Am nächsten Tag als die Nachtwanderung bevor stand und der gesamte Kurs bereit war, das heißt alles nochmal im Detail durchgegangen wurde, konnte es los gehen. Nach der Abend-KüA-Schiene gingen die einzelnen Gruppen im zehnminütigen Takt los. Um die nachtaktiven Waldbewohner nicht zu stören und unsere Augen an die Dunkelheit zu gewöhnen, gingen wir mit nur einer Taschenlampe los, die nur zum Zeigen der Sterne benutzt wurde. Während der Wanderung führten die zwei Astronomen die Gruppe an verschiedene Zwischen-schauplätze, an denen man einen Teil des Sternhimmels erhaschen und die Grundlagen lernen konnte (z. B. der große Wagen, der Nordpolarstern). Auf dem Weg zur letzten Station kamen dann die erlernten Sternbilder zum Einsatz. Auf dem Feld angekommen, wurden die Zusammenhänge zwischen den Sternbildern erklärt und man konnte die verschiedene Besonderheiten des Nachthimmels beobachten. Bevor schließlich alle zurück zum Campus liefen, kam die Hauptattraktion dieses Abends, die Teleskope, mit denen man Saturn, Algol (auch der Teufelsstern genannt) und andere spannende Himmelskörper beobachten konnte. Anschließend gingen die einzelnen Gruppen wieder zurück zum Campus und legten uns sofort schlafen, da alle sehr müde waren von diesem anstrengenden, aber erlebnisreichen Tag.

Quellen

[2.1] Martin Rees: Universum die große Bild-Enzyklopädie; Dorling Kindersley, München, 2014, Seiten 368, 389, 346

[2.2] https://www.astronomie-nuernberg.de/index.php; 11.10.2017, 20.42 Uhr

[3.1] https://de.wikipedia.org/wiki/Scheinbare_Helligkeit, besucht am 17.10.17

[3.2] https://www.leifiphysik.de/astronomie/fixsterne/scheinbare-sternhelligkeit, besucht am 17.10.17

[3.3] http://physik.cosmos-indirekt.de/Physik-Schule/Farbindex, besucht am 17.10.17

[4.1] http://www.cosmos.esa.int/web/hipparcos (21.08.2017)

[4.2] Hipparcos: Die wissenschaftliche Ernte beginnt. Von Ulrich Bastian in: Sterne und Weltraum, Bd. 36, Nr. 11, S. 938–941 (1997)

[4.3] http://www.ianridpath.com/startales/almagest.htm (18.09.2017)

[4.4] https://www.bernd-leitenberger.de/astonomische-satelliten-speziell.shtml (18.09.2017)

[4.5] http://vizier.u-strasbg.fr/viz-bin/VizieR-3?-source=I/239/hip_main

[5.1] http://www.itwissen.info/Prozedurale-Programmierung-procedural-programming.html

[5.2] https://docs.python.org/2/library/functions.html#func-list

[5.3] https://www.python-kurs.eu/python3_klassen.php

[7.1] https://matplotlib.org/devdocs/api/_as_gen/matplotlib.pyplot.imshow.html

[7.2] https://matplotlib.org/users/image_tutorial.html

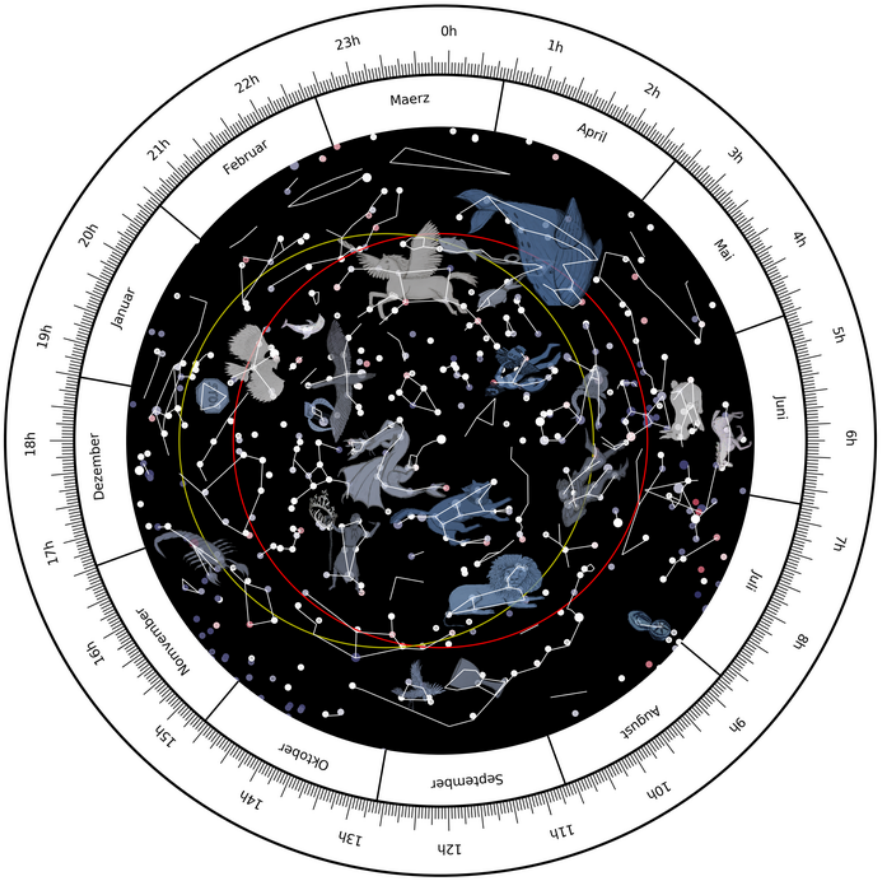
[7.3] https://www.pyimagesearch.com/2014/11/03/display-matplotlib-rgb-image

https://matplotlib.org/users/text_intro.html

http://www.labri.fr/perso/nrougier/teaching/matplotlib

[7.4] https://matplotlib.org/api/pyplot_api.html?highlight=scatter#matplotlib.pyplot.scatter

[10.1] https://www.lsw.uni-heidelberg.de/foerderkreis/geschichte, besucht am 14.10.17



Fertige Sternkarte – selbst programmiert vom Astronomiekurs.

Danksagung

Wir möchten uns an dieser Stelle bei denjenigen herzlich bedanken, die die 15. JuniorAkademie Adelsheim / Science Academy Baden-Württemberg überhaupt möglich gemacht haben.

Finanziell wurde die Akademie in erster Linie durch die Stiftung Bildung und Jugend, die Hopp-Foundation, den Förderverein der Science Academy sowie durch den Fonds der Chemischen Industrie unterstützt. Dafür möchten wir an dieser Stelle allen Unterstützern ganz herzlich danken.

Die Science Academy Baden-Württemberg ist ein Projekt des Regierungspräsidiums Karlsruhe, das im Auftrag des Ministeriums für Kultus, Jugend und Sport Baden-Württemberg und mit Unterstützung der Bildung & Begabung gGmbH Bonn für Jugendliche aus dem ganzen Bundesland realisiert wird. Wir danken daher dem ehemaligen Leiter der Abteilung 7 des Regierungspräsidiums Karlsruhe, Herrn Abteilungspräsidenten Vittorio Lazaridis, der Leiterin des Referats 75 – allgemein bildende Gymnasien, Frau Leitende Regierungsschuldirektorin Dagmar Ruder-Aichelin, Herrn Dr. Hölz vom Ministerium für Kultus, Jugend und Sport Baden-Württemberg sowie dem Koordinator der Deutschen Schüler- und JuniorAkademien in Bonn, Herrn Volker Brandt, mit seinem Team.

Wie in jedem Jahr fanden die etwas über einhundert Gäste sowohl während des Eröffnungswochenendes und des Dokumentationswochenendes als auch während der zwei Wochen im Sommer eine liebevolle Rundumversorung am Eckenberg-Gymnasium mit dem Landesschulzentrum für Umwelterziehung (LSZU) in Adelsheim. Stellvertretend für alle Mitarbeiter möchten wir uns für die Mühen, den freundlichen Empfang und den offenen Umgang mit allen bei Herrn Oberstudiendirektor Meinolf Stendebach, dem Schulleiter des Eckenberg-Gymnasiums, besonders bedanken.

Zuletzt sind aber auch die Kurs- und KüA-Leiter gemeinsam mit den Schülermentoren und der Assistenz des Leitungsteams diejenigen, die mit ihrer hingebungsvollen Arbeit das Fundament der Akademie bilden.

Diejenigen aber, die die Akademie in jedem Jahr einzigartig werden lassen und die sie zum Leben erwecken, sind die Teilnehmerinnen und Teilnehmer. Deshalb möchten wir uns bei ihnen und ihren Eltern für ihr Engagement und Vertrauen ganz herzlich bedanken.

Bildnachweis

S. 12: Sternparallaxe
https://commons.wikimedia.org/wiki/File:ParallaxeV2.png
Wikimedia-User WikiStefan
CC-BY-SA (https://creativecommons.org/licenses/by-sa/3.0/legalcode)

S. 15: Doppelweiche
Bild basierend auf: https://commons.wikimedia.org/wiki/File:Plektita_trakforko_14.jpeg
Wikimedia-User Asiano
CC-BY-SA (https://creativecommons.org/licenses/by-sa/3.0/legalcode)

S. 19: Dr. Thomas Müller – mit freundlicher Genehmigung

S. 27: Dr. Thomas Müller – mit freundlicher Genehmigung

S. 56: Schnittbild durch einen Vulkan
https://commons.wikimedia.org/wiki/File:Submarine_Eruption-blank.svg
Wikimedia-Nutzer Sémhur
CC-BY-SA (https://creativecommons.org/licenses/by-sa/4.0/legalcode)

S. 56: Eyjafjallajökull first crater 20100329
https://commons.wikimedia.org/wiki/File:Eyjafjallajökull_first_crater_20100329.jpg
Wikimedia-Nutzer David Karnå
CC-BY (https://creativecommons.org/licenses/by/3.0/legalcode)

S. 56: Ruinen von Pompeji
https://commons.wikimedia.org/wiki/File:Vesuvius_from_Pompeii_(hires_version_2_scaled).png
Wikimedia-Nutzer Morn the Gorn
CC-BY-SA (https://creativecommons.org/licenses/by-sa/3.0/legalcode)

S. 65: Funktionsweise Geysir
https://commons.wikimedia.org/wiki/File:Funktionsweise_Geysir_de.svg
Wikimedia-Nutzer Huebi
CC-BY-SA (https://creativecommons.org/licenses/by-sa/2.5/legalcode)

S. 78: Arthur Miller – Hexenjagd: Mit freundlicher Genehmigung des S. Fischer Verlags

S. 116: Electromagnetic spectrum c
https://commons.wikimedia.org/wiki/File:Electromagnetic_spectrum_c.svg
Wikimedia-Nutzer Horst Frank / Phrood / Anony
CC-BY-SA (https://creativecommons.org/licenses/by-sa/3.0/legalcode)

Alle anderen Abbildungen sind entweder gemeinfrei oder eigene Werke.