Visualisierung von 3D Strahlungstransportrechnungen

(3D visualisation of radiative transfer calculations)

Bachelorarbeit im Studiengang Physik

an der Universität Hamburg

Hamburger Sternwarte

2015

von Fiona Dorothea Elisabeth Prodöhl

geboren am 26. August 1990

1. Gutachter: Prof. Dr. Peter Hauschildt
2. Gutachter: Prof. Dr. Jürgen H.M.M. Schmitt

Zusammenfassung

Diese Bachelorarbeit beschäftigt sich mit dem dreidimensionalen Visualisieren von stellaren Atmosphären. Das Visualisierungsprogramm wurde in Python geschrieben, wobei das OpenGL API für das Rendern der Kugel eingesetzt wurde. Die Sphäre lässt sich per Tastatureingabe und Mausbewegung drehen.
Als Inputdaten für den Code dienten Strahlungsdaten eines mittels des Atmosphärencode PHOENIX generierten Modells eines Gasriesens, der von einem Stern bestrahlt wird. Die Intensitätsdaten beziehen sich auf einen unendlich weit entfernten Beobachter. Es lassen sich die Daten von drei verschiedenen Wellenlängen gleichzeitig betrachten.
Das Programm lässt sich für zukünftige Anwendungen auf einen endlich nahen Beobachter erweitern.

Abstract

This bachelor thesis deals with three-dimensional visualisation of stellar atmospheres. The visualisation program is written in Python and the OpenGL API is used for rendering the sphere. The sphere can be rotated by keyboard input and mouse movement.
As input data for the code radiation data of a model of an irradiated gas giant produced by the atmospheric code PHOENIX is used. The intensity data refers to an observer, who is infinitely far away from the planet. Up to three different wavelengths can be displayed simultaneously.
The Program can be extended for future applications to an observer who is finitely far away from the visualized object.

Inhaltsverzeichnis

1 Einleitung

Heutzutage sind dreidimensionale Rechnungen möglich und werden immer häufiger durchgeführt. Allerdings ergibt sich daraus das Problem, wie man aus den gewonnenen Daten Erkenntnisse gewinnen kann. Hierfür ist die Visualisierung, bei der die Daten in korrekter Geometrie dargestellt werden, eine sehr wichtige Methode. Simulationsergebnisse können so wesentlich einfacher analysiert werden.
In dieser Bachelorarbeit wird ein solches Visualisierungsprogramm für bereits berechnete Modelle erstellt. Details über die Erzeugung dieser Daten würden den Rahmen dieser Arbeit sprengen, weshalb an dieser Stelle auf Hauschildt und Baron [HB10], Hauschildt und Baron [HB99] und Hauschildt u.a. [HBA97] verwiesen sei. Die benutzten Daten wurden mit dem Atmosphärencode PHOENIX [HB13] berechnet, basierend auf hydrodynamischen 3D Strukturen von Showman [SFL+09].

Das Programm, das im Verlauf dieser Bachelorarbeit geschrieben wurde, kann sowohl zur 3D-Visualisierung von Sternen- als auch von Planetenatmosphären dienen. Die Visualisierung wurde beispielhaft für einen bestrahlten Gasplaneten durchgeführt. Hierfür standen fertige Intensitätsdaten der Planetenoberfläche im Wellenlängenbereich von 5000Å bis 149000Å zur Verfügung.

Die Arbeit teilt sich in drei Abschnitte auf. Zunächst werden im theoretischen Teil die Programmiersprachen und physikalische Grundlagen zum Strahlungstransport vermittelt. Im zweiten Teil wird die Durchführung dieser Bachelorarbeit erläutert, wobei auch auf aufgetretene Probleme eingegangen und die Bedienung des Programms erklärt wird. Im dritten Teil werden dann die Ergebnisse präsentiert. Nach der Zusammenfassung folgt ein Kapitel über mögliche Erweiterungen für das Projekt dieser Bachelorarbeit.

2 Theoretische Grundlagen

Die Darstellung der Strahlungsintensitäten auf einer 3D-Kugeloberfläche, die man manuell steuern kann, erfordert Kenntnisse der verwendeten Programmiersprache und Pakete sowie Kenntnisse über die zugrunde liegenden physikalischen Eigenschaften der zu beschreibenden Objekte.
Im ersten Teil wird ein Überblick über verwendete Software gegeben. Der zweite Teil umreißt die physikalischen Grundlagen dieser Bachelorarbeit.

2.1 Programmiergrundlagen

Das Visualisierungsprogramm wurde in Python geschrieben, wobei das OpenGL API für das Rendern der Kugeloberfläche eingesetzt wurde. Sowohl Programmiersprache als auch Pakete sind Open Source verfügbar und der Umgang mit ihnen einfach zu erlernen, weshalb sie für diese Bachelorarbeit ausgewählt wurden. Im Folgenden werden die wichtigsten Eigenschaften des Codes erwähnt.

2.1.1 Python

Python ist eine höhere Programmiersprache, die aufgrund ihrer vielseitigen Einsetzbarkeit weit verbreitet ist. Die Programmstruktur ist im Gegensatz zu vielen anderen Programmiersprachen durch Einrückungen gegeben, wodurch der Code zum Teil erheblich kürzer wird. Allerdings können schon kleine Einrückfehler zu Problemen führen. Python hat eine klare und übersichtliche Syntax und kommt mit wenigen Schlüsselwörtern aus, sodass es leicht zu erlernen ist [Kle09].
Anfang der 1990er Jahre wurde Python von Guido van Rossum am *Centrum Wiskunde & Informatica* in Amsterdam als Nachfolger für die Programmier-Lehrsprache ABC entwickelt und war ursprünglich für das verteilte Betriebssystem Amoeba gedacht [Kle09].
Der Name geht nicht etwa (wie das Logo auf den meisten Python-Büchern vermuten ließe) auf die gleichnamige Schlangengattung (Pythonschlange) zurück, sondern bezog sich ursprünglich auf die englische Komikergruppe Monty Python. In der Dokumentation finden sich daher auch einige Anspielungen auf Sketche aus dem Flying Circus.
Da Python ein Open Source Produkt ist und keine Registrierung verlangt wird, ist es schwer die

tatsächliche Anzahl der Nutzer zu bestimmen. Seit gut 25 Jahren gibt es Python, das eine große Nutzerbasis und eine sehr aktive Entwickler-Gemeinde besitzt. Es gibt ein breites Einsetzungfeld und arbeitet sehr stabil und robust.
Python wird nicht nur von Privatnutzern gebraucht, sondern findet auch Anwendung im Internet bei Firmen wie etwa *Google* oder *Yahoo!* oder bei der Produktion von Animationsfilmen bei Firmen wie zum Beispiel *Industrial Lights* oder *Magic* [Het09]. Durch die vielen verschiedenen Bibliotheken und Module ist Python universell und vielseitig einsetzbar.
Ein für diese Bachelorarbeit wichtiges OpenSource Erweiterungsmodul von Python ist Numpy. Numpy (Numeric Python) bietet schnelle vorkompilierte Funktionen für mathematische und numerische Rechnungen und erleichtert das Arbeiten mit großen Arrays und Matrizen.
Zurzeit sind zwei Versionen von Python parallel in Gebrauch. Die Version Python 2.7 aus der 2. Generation wird weiterhin gerne verwendet, obwohl es schon eine Version 3.4 aus der 3. Generation gibt. Python 3.0 wurde 2008 veröffentlicht. Die finale Version von Python 2 wurde Mitte 2010 als 2.7 veröffentlicht. Die neueste Python 3 Version wurde 2014 als Version 3.4 veröffentlicht. Allerdings gibt es jeweils vor- und Nachteile für Python 2.7 und Python 3. Dies wird auf der Webseite *Python 2 or Python 3* [Clo14] ausführlich diskutiert. Für diese Arbeit stand die Python-Version 2.7 zur Verfügung.

2.1.2 OpenGL

OpenGL (Abkürzung für **Open G**raphics **L**ibrary) wird zur Entwicklung von 2D- und 3D-Computergrafikanwendungen verwendet. OpenGL ist plattform- und programmiersprachenunabhängig und ermöglicht das Rendern komplexer 3D-Szenen mit Frameraten von weniger als 1/s.
Ein sehr wichtiger Vorteil von OpenGL ist, dass es als Schnittstelle für den Zugriff auf die GPU dient, ohne dass bestimmte Forderungen an die Grafikkarte geknüpft sind. Beim Fehlen einer Grafikkarte kann somit die CPU das Rendern übernehmen.
OpenGL ist eine reine Grafikbibliothek und benötigt, wie andere Software auch, betriebssystemabhängige Bibliotheken, die OpenGL mit dem darunter liegenden Betriebssystem verbinden. Diese Bibliotheken kümmern sich um die Verwaltung von Fenstern, das Buffern und das Rendern. OpenGL-Kommandos besitzen ein Präfix, das angibt aus welcher Bibliothek der Befehl stammt. Basis-Befehle beginnen mit *gl*. Sie befassen sich nur mit der Synthese. Die Zusatzbibliothek GLU (OpenGL Utility Library) beginnt mit *glu* und umfasst viele Zeichenprimitive wie Kurven, Flächen und andere Funktionen zur Gestaltung der 3D-Szenerie. Befehle von GLUT (OpenGL Utility Toolkit) beginnen mit *glut* und umfassen zusätzlich die Verwaltung von

Fenstern, den Umgang mit Maus- und Tastaturereignissen, das Aufbauen grafischer Benutzeroberflächen und einige komplexere geometrische Objekte.
Eine wichtige Eigenschaft von OpenGL ist dessen Erweiterbarkeit. OpenGL kann von Herstellern selbst erweitert werden und es gibt viele sogenannte *Extensions* für neue, noch nicht vom Standard unterstützte Funktionen. Die Bibliothek OpenGL ist in der Programmiersprache C geschrieben, da diese besonders gut für objektorientierte Codes geeignet ist. Außerdem ermöglicht die Programmiersprache C, OpenGL in andere Programmiersprachen zu integrieren.
Die erste Veröffentlichung von OpenGL war im Juli 1994, entwickelt von der Firma *Silicon Graphics Computer Systems* [S+13]. Seit 2006 ist die *Khronos Group* für die Weiterentwicklungen zuständig. Heute hat OpenGL vielfältige Anwendungen in Industrie, Forschung, Lehre und Spieleprogrammierung.

2.1.3 Schnittstelle Python und OpenGL

Python kann zur Benutzung von OpenGL eine Schnittstelle zu C (ctypes) verwenden und somit auf die Funktionen von OpenGL zugreifen.
Am häufigsten wird für diese Schnittstelle die plattformübergreifende Verbindung PyOpenGL verwendet. Ist diese installiert, so können am Anfang einer Python-Datei die entsprechenden Module importiert werden:

*from OpenGL.GL import **
*from OpenGL.GLUT import **
*from OpenGL.GLU import **,

wobei der Stern (*) der Wildcard-Platzhalter ist.
Da die OpenGL-Befehle mit ihren spezifischen Vorsilben unverwechselbar sind, ermöglicht dies einen direkten Zugriff auf die OpenGL-Bibliothek im gesamten Programm.

2.2 Physikalische Grundlagen

In diesem Teil wird das physikalische Verständnis für die Visualisierung von Strahlungsintensitäten aufgebaut. Dazu wird zunächst darauf eingegangen, was Intensität und Strahlungsfluss sind und danach wird der Strahlungstransport in Stern- oder Planetenatmosphären beschrieben. Die Literatur für diesen Abschnitt entstammt hauptsächlich aus dem Buch *Radiative Transfer in Stellar Atmospheres* von R. J. Rutten [Rut03].

2.2.1 Intensität

Die spezifische Intensität I_ν ist der Proportionalitätsfaktor folgender Energietransportgleichung:

$$\begin{aligned} dE_\nu &\equiv I_\nu \left(\vec{r}, \vec{l}, t\right) \left(\vec{l} \cdot \vec{n}\right) \, dA \, dt \, d\nu \, d\Omega \\ &= I_\nu(x, y, z, \theta, \phi, t) \, cos\theta \, dA \, dt \, d\nu \, d\Omega \, , \end{aligned} \tag{2.1}$$

wobei dE_ν die Energiemenge angibt, die durch die Fläche dA mit dem Normalenvektor $\vec{n}$ am Ort $\vec{r}$ in einem Zeitintervall dt und im Frequenzbereich zwischen ν und $\nu + d\nu$ über einen Raumwinkel $d\Omega$ um die Richtung $\vec{l}$ mit den Polarkoordinaten θ und ϕ transportiert wird. Die Einheiten hierfür sind:

$$\begin{aligned} dE_\nu &= \left[erg \; s^{-1} \; cm^{-2} \; Hz^{-1} \; ster^{-1}\right] \\ &= \left[W \; m^{-2} \; Hz^{-1} \; ster^{-1}\right] \end{aligned} \tag{2.2}$$

Dies definiert die monochromatische Intensität. Die totale Intensität erhält man, indem man die monochromatischen Intensitäten über alle Frequenzen integriert: $I \equiv \int_0^\infty I_\nu \, d\nu$.
Die Intensität I_ν repräsentiert den makroskopischen Part zum Beschreiben der Energie, die durch Photonen entlang eines Lichtstrahls transportiert wird. Da die Photonen Übermittler der elektromagnetischen Wechselwirkungen sind, ist die Intensität die grundlegende makroskopische Größe, die den Strahlentransport beschreibt.

Den monochromatische Fluss Φ_ν erhält man, indem die monochromatischen Intensitäten über den Raumwinkel integriert werden:

$$\begin{aligned} \Phi_\nu \left(\vec{r}, \vec{n}, t\right) &\equiv \int I_\nu \, cos\theta \, d\Omega \\ &= \int_0^{2\pi} \int_0^{\pi} I_\nu \, cos\theta \, sin\theta \, d\phi \, d\theta \, . \end{aligned} \tag{2.3}$$

Der monochromatische Fluss wird in nachfolgenden Einheiten angegeben:

$$\Phi_\nu = \left[erg \; s^{-1} \; cm^{-2} \; Hz^{-1}\right] = \left[W \; m^{-2} \; Hz^{-1}\right] \tag{2.4}$$

Die Intensität ist oftmals nicht messbar, da mit den beobachteten Teleskopen Sterne nicht aufgelöst werden können. Es wird also dann der Strahlungsfluss gemessen.

2.2.2 Strahlungstransport in Stern- und Planetenatmosphären

Der Strahlungstransport in Stern- und Planetenatmosphären beschreibt den Weg der Photonen bis sie bei einem Beobachter ankommen. Bei einem Stern werden die Photonen im Inneren erzeugt und müssen dann durch das Material bis an die Oberfläche gelangen. Bei einem bestrahlten Planeten müssen die Photonen des Muttersterns einmal komplett durch den Planeten hindurchwandern. Dies funktioniert gut bei Gasplaneten, während bei terrestrischen Planeten die Absorption und Reflexion von Photonen des Muttersterns so hoch ist, dass diese Photonen fast nie auf der anderen Seite heraustreten. Diese Photonen gehen dort den Weg des geringsten Widerstandes und werden entweder sofort zurück reflektiert oder ein geringer Teil kann durch Streuung in der äußeren Atmosphärenschicht einmal um den Kern herum wandern.
Auf dem Weg der Photonen zum Beobachter hindern die Prozesse von Absorption, Emission und Streuung den Photonenfluss. Durch diese Prozesse wird die Bewegungsrichtung der Photonen immer wieder geändert und die Energie der Photonen umverteilt. Dieser zufällige Pfad, den die Photonen beschreiben, wird ”random walk” genannt und ist in Abbildung 2.1 schematisch für ein Photon aus dem Inneren eines Sterns bis zu der Sternoberfläche dargestellt. Deshalb gibt es in einem Stern nicht den direkten Photonenstrahl, der die Energie mit Lichtgeschwindigkeit nach Außen an die Oberfläche transportiert. Stattdessen reisen die einzelnen Photonen nur temporär mit dem gedachten Lichtstrahl.

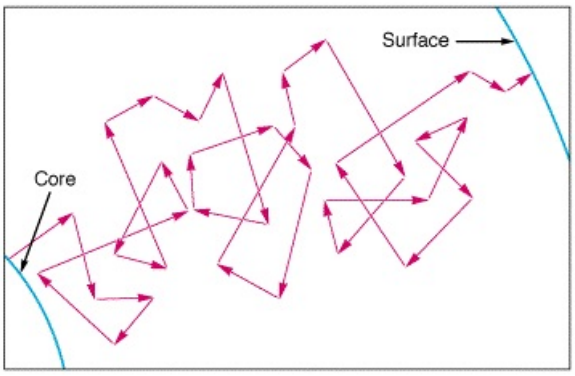


Abbildung 2.1: Schematischer ”random walk” eines Photons vom Inneren eines Sternes zur Oberfläche [Pog]

Da unterschiedliche Wellenlängen unterschiedlich absorbiert, emittiert und gestreut werden, diffundieren die Photonen unterschiedlicher Wellenlänge auch unterschiedlich gut durch das Material. Wie gut welche Wellenlänge passiert, kann durch Aufnahme eines Spektrums gemessen werden.

3 Durchführung

Das Problem, welches in dieser Bachelorarbeit behandelt wird, ist zunächst sechsdimensional, wobei davon drei Dimensionen die Position des Beobachters beschreiben und zwei Dimensionen den Ort auf der Kugeloberfläche angeben, von wo die Strahlungsintensität gemessen wird. Die sechste Dimension entspricht der Wellenlänge. Zunächst wird nur eine Wellenlänge benutzt, damit das Problem eine Dimension einfacher ist. An dieser Stelle sei schon einmal auf den sprachlichen Unterschied zwischen Beobachter und der Kamera hingewiesen. Auf den Beobachter, beziehen sich die Intensitätsdaten, wohingegen die Kamera wie ein externer Beobachter um die Kugel herumlaufen kann, um sich die Intensitätsdaten für den Beobachter von allen Seiten her anzusehen.
Die in dieser Bachelorarbeit benutzte Intensitätsmatrix ist vereinfacht als vierdimensionale Matrix angelegt. Diese besteht aus den Koordinaten für die Kugeloberfläche (y, z) und den Koordinaten für den Beobachter (θ, ϕ). Diese Koordinaten sind zur Veranschaulichung in Abbildung 3.1 als Skizze dargestellt. Dabei wird angenommen, dass sich der Beobachter in einer bestimmten Richtung unendlich weit von der Kugel entfernt befindet. In einer weiteren Vereinfachung wird die Breitenkoordinate des Beobachters auf $\theta = \pi/2$ gesetzt, sodass der Beobachter immer senkrecht auf den Äquator der Kugel blickt. Durch diese letzte Vereinfachung ist die vierdimensionale Matrix in Wirklichkeit schon dreidimensional.

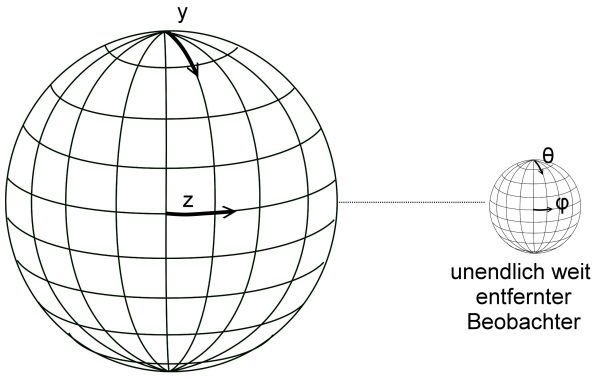


Abbildung 3.1: Skizze der Koordinaten der Intensitätsmatrix für einen unendlich weit entfernten Beobachter

Die Intensitätsdaten geben an, wie Licht aus der Sphäre in Richtung Beobachter austritt. Aus

diesem Grund ist eine rechnerische Sphäre, die für einen bestimmten Ort des Beobachters berechnet wird, auf der dem Beobachter zugewandten Seite hell und auf der anderen Seite dunkel. Die Auflösung der Daten, bzw. der Textur liegt bei 125 x 257.

In diesem Kapitel wird zunächst kurz auf die Anfangs- und Einarbeitungsphase eingegangen, danach wird erklärt, wie genau das Projekt realisiert wurde, mit was für Problemen umzugehen war und wie sich das Programm bedienen lässt.

3.1 Programmkonzepte

Die Grundlagen von Python konnten zum Beispiel durch das Buch *Beginning Python - From Novice to Professional* von Magnus Lie HetLand [Het09] erlernt werden.
Python, OpenGL und PyOpenGL können zudem sehr gut durch Online-Tutorien [z.B. Hoc13] erlernt und alle Befehle in der Dokumentation [z.B. Sou] nachgeschlagen werden.
Als grundlegender Code, auf den diese Bachelorarbeit aufbaut, wurde ein frei verfügbares Python-Skript zum Rendern von texturierten Objekten aus dem Internet gewählt [Bur07] und anschließend auf die gegebene Problemstellung angepasst. Die enthaltenen OpenGL Funktionen wurden verwendet, um eine einfache Sphäre zu erzeugen. Zusätzlich beinhaltete der Code bereits Funktionalitäten zur System-Event-Verwaltung, welches erlaubte, Tastatureingaben zu verwenden, um die Kameraposition zu beeinflussen. Es half auch zum Herumexperimentieren mit den Programmiersprachen und somit zum Erlernen der Befehle, die darin benutzt wurden.
Die bereits bestehende Datei "DDD.py" diente zum Speichern der von Phoenix berechneten Intensitätsdaten und wurde so umgeschrieben, dass in ihr die Daten geladen und bearbeitet werden konnten. Zunächst wurde so der Datensatz bei einer Wellenlänge von $\lambda = 10000$Å eingelesen. Die Intensitätsmatrix für diese erste Wellenlänge wurde soweit bearbeitet, dass sie als Textur auf der Oberfläche der Sphäre dargestellt werden kann. Sobald dies für eine Wellenlänge erfolgreich durchgeführt wurde, war es nicht mehr aufwendig, es für mehrere verschiedene Wellenlängen zu implementieren.
Die Intensitätsmatrix wurde auf diese drei Dimensionen reduziert und zunächst auf den Maximalwert dieser Matrix normiert.
Die Normierung der Daten war notwendig, da die Farben auf der Sphäre durch drei Farbkanäle mit Helligkeitswerten zwischen Null (dunkel) und 255 (hell) dargestellt wurden. Innerhalb dieses Intervalls mussten die Werte für die Intensitäten liegen. Die Intensitätsminima brauchten nicht verändert werden, da die Intensitäten auf der vom Beobachter aus rückseitigen Hemisphäre bereits den Wert Null annahmen.

Die Intensitätsdaten wurden erneut als numpy-Datei gespeichert, damit sie als solche innerhalb der Sphären-Datei zur Weiterverarbeitung aufgerufen werden konnten. Später wurde noch ein effizienterer Weg gefunden, bei dem die Daten mit dem *return*-Befehl direkt aus der "DDD.py" -Datei eingelesen werden, sodass viel Speicherplatz gespart werden kann.
In der Sphären-Datei wurde die nicht normierte Intensitätsmatrix aufgerufen und konnte dort skaliert und dann in eine Textur eingelesen werden. Dazu wurden den Intensitätswerten Farben zugeordnet. Diese Textur konnte dann auf der Sphäre abgebildet werden. Dieser Prozess, bei dem ein zweidimensionaler Zahlenraum mit einem dreidimensionalen Objekt verschränkt wurde, nennt man *Mapping*.
Sobald alles miteinander verknüpft war, konnte die fertige Sphäre mit Textur gerendert werden.

3.2 Bedienung

Eine bestimmte Wellenlänge kann durch den Befehl *DDD.__main__(a)* aufgerufen werden, wobei das a für die gewünschte Wellenlänge in kÅ steht. Mit Hilfe des Codeausschnittes aus Abbildung 3.2 können zum Beispiel die Daten für die Wellenlänge von 10000Å eingelesen, normiert und in diesem Fall eine Textur in Rottönen erstellt werden.

```
C = DDD.__main__(10)
Cnorm = C/C.max()

for j in range(257):
        for i in range(125):
                image[i,j,0] = Cnorm[i,j,k]
```

Abbildung 3.2: Teil des Source Codes zum Laden der Intensitätsdaten und späteren Darstellung in Rottönen

Wenn die Datei gestartet wird, dann erscheint nach einiger Rechenzeit die Sphäre. Um diese kann sich der Beobachter durch die Links- und Rechts-Pfeiltasten in und gegen den Uhrzeigersinn drehen. Hierbei wird für jede Blickrichtung des Beobachters eine neue Textur *k* erstellt, die sich gleichzeitig mit dem Wandern des Beobachters um die Sphäre ändert. Der dazugehörige Source Code kann in Abbildung 3.3 angesehen werden.

Mit den Tasten W und S kann sich die Kamera näher an die Sphäre heran und weiter weg von

```
for k in range(360):
    glBindTexture(GL_TEXTURE_2D, int(textures[k])
```

Abbildung 3.3: Teil des Source Codes für die verschiedenen Texturen

ihr bewegen.
Mit gedrückter linker Maustaste und gleichzeitigem Bewegen der Maus kann man die Kugel frei drehen, ohne dabei die Position des Beobachters zu ändern. Dies ermöglicht das Betrachten der theoretischen berechneten Strahlungsintensitäten, die für einen Beobachter immer auf der nicht sichtbaren Hemisphäre liegen. Dies kann zum Beispiel in den Abbildungen 3.9(a) und 3.9(b) gesehen werden.

3.3 Problem des Mappings

Die Intensitätsdaten aus der Intensitätsmatrix sollen als Textur dargestellt und dann auf der Sphäre abgebildet werden. Dabei gilt es, die richtigen Koordinaten im Code miteinander zu verknüpfen. In der Intensitätsmatrix werden die sphärischen Koordinaten $z = \phi_{Daten}$ und $y = \theta_{Daten}$ genannt, die der Textur ϕ_{Tex} und θ_{Tex} und die der Sphäre $\phi_{Sphäre}$ und $\theta_{Sphäre}$. Es erscheint logisch, dass die verschiedenen ϕ-Werte den gleichen Wert haben müssen, genauso wie die verschiedenen θ-Werte. Allerdings gilt es erst mal diese Variablen im Code miteinander zu verknüpfen. Dadurch sind zwischenzeitlich Sphären mit sehr interessanten Mustern entstanden. In Abbildung 3.4 ist in gelb-rot die Intensität bei einer Wellenlänge von $\lambda = 10000$Å dargestellt. Dabei ist rot keine Intensität und gelb maximale Intensität. Anhand der Spiralarme kann man erkennen, dass die Intensität falsch auf die Sphäre projiziert wurde. Dies ist dadurch entstanden, dass die Texturpunkte seriell gespeichert werden, sodass der Pitch-Wert (Breite der Textur) manuell übergeben werden muss. Wird der Pitch-Wert falsch angegeben, verursacht dies eine Sphäre wie in Abbildung 3.4, wohingegen eine Kombination mit um 90° verdrehter Textur die Spirale in Abbildung 3.5 erzeugt. Wie die Sphäre richtig aussieht, kann man in Abbildung 3.6 sehen.

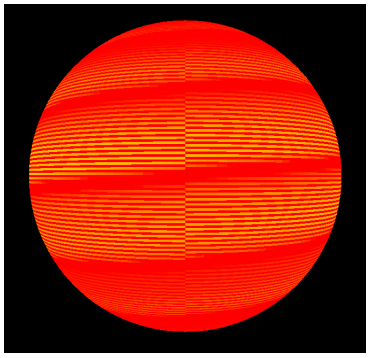


Abbildung 3.4: Fehlerhafte Darstellung der Intensitätsdaten bei $\lambda = 10000$Å auf der Sphäre, wobei Rot niedrige oder keine Intensität angibt und gelb hohe Intensität

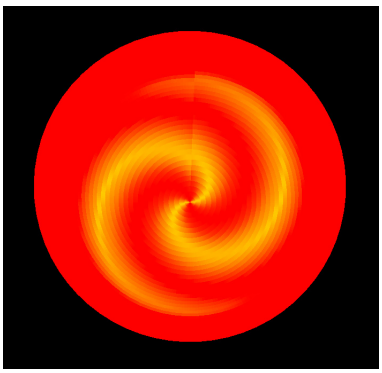


Abbildung 3.5: Fehlerhafte Darstellung der Intensitätsdaten bei $\lambda = 10000$Å auf der Sphäre bei Vertauschen der Bild- und Kugelkoordninaten in Kombination mit um 90° verdrehter Textur

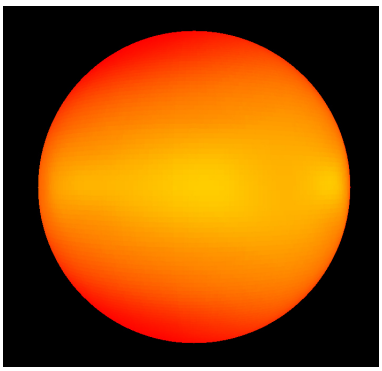


Abbildung 3.6: Korrekte Darstellung der Intensitätsdaten bei $\lambda = 10000$Å auf der Sphäre

3.4 Skalierung

Da die Intensitäten von Vorder- und Rückseite, sowie bei unterschiedlichen Wellenlängen sehr variieren, ist es nötig die Intensitäten gut zu skalieren. Die in dieser Bachelorarbeit verwendeten Intensitätsmaxima in Abhängigkeit der Wellenlänge sind in Abbildung 3.7 dargestellt. Man kann sehen, dass die Daten wie erwartet grob einem schwarzen Körper entsprechen.

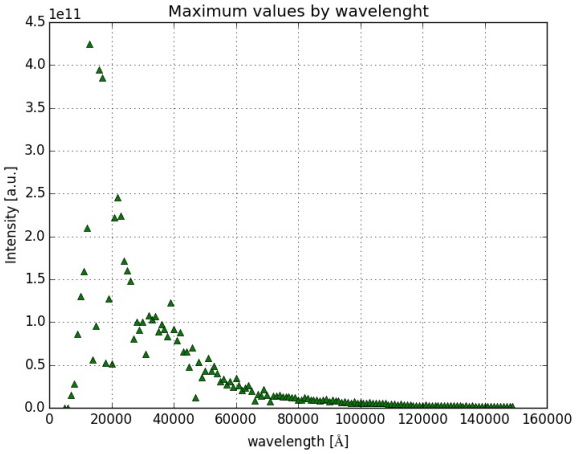


Abbildung 3.7: Intensitätsverteilung der Maximalwerte in Abhängigkeit der Wellenlängen

Die Intensitätsdaten wurden zunächst auf den Maximalwert der jeweiligen Wellenlänge normiert. Weiterhin wurde auch ausprobiert, die Intensitäten auf den Maximalwert aller Wellenlängen zu normieren. Wie in Abbildung 3.8 ersichtlich, wurden diese Sphären jedoch sehr dunkel.

Außerdem wurden die Intensitäten teilweise logarithmiert, wie in Abbildungen 3.9 (a) und (b) für die Wellenlänge von 10000Å und in der Abbildung 3.10 für die drei Wellenlängen von 5000Å (rot), 7000Å (grün) und 10000Å (blau) dargestellt ist. Die logarithmische Darstellung ist interessant, weil die Helligkeitsempfindlichkeit des menschlichen Auges auch logarithmisch ist.
Insgesamt waren die Ergebnisse am besten, als die Werte nicht logarithmiert und nur auf ihr eigenes Maximum normiert wurden. Deshalb wurde diese Einstellung verwendet.

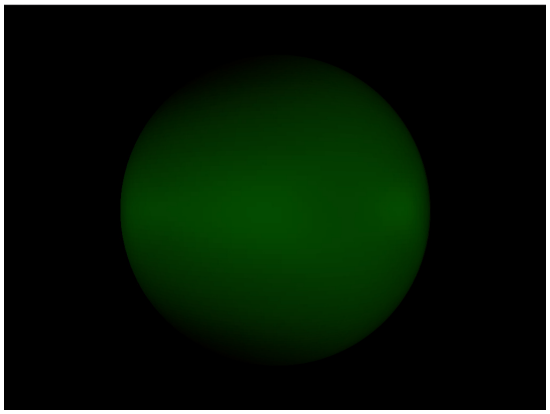
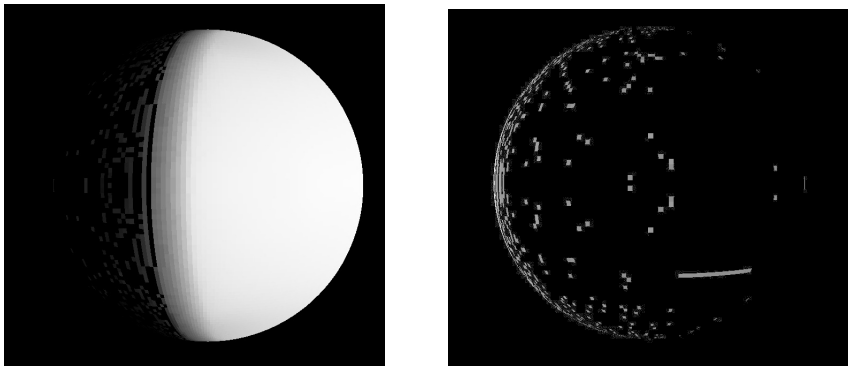


Abbildung 3.8: Sphäre mit Normierung auf das Maximum aller Wellenlängen für die Wellenlängen von λ=100000Å (rot), λ=10000Å (grün) und λ=5000Å (blau).



(a) (b)

Abbildung 3.9: Logarithmische Darstellung der Intensitätsdaten bei λ = 10000Å aus zwei verschiedenen Perspektiven, wobei Schwarz niedrige oder keine Intensität angibt und weiß hohe Intensität

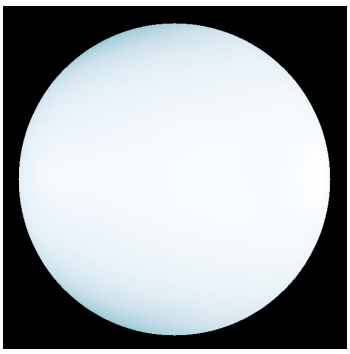


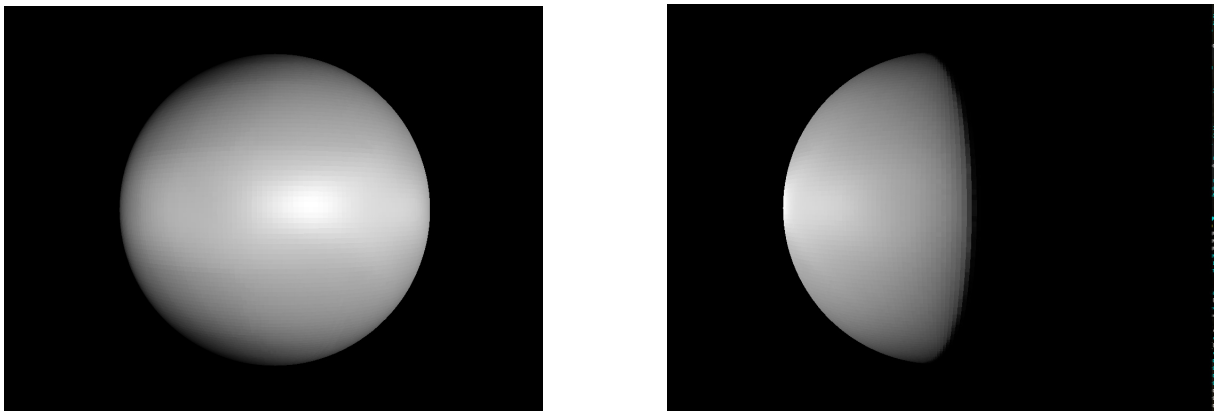
Abbildung 3.10: Sphären-Modell logarithmisch dargestellt bei Wellenlängen von λ = 5000Å (rot), λ = 7000Å(grün) λ = 10000Å (blau)

4 Ergebnisse

Das Ergebnis ist eine dreidimensional modellierte Sphäre, die man mit Hilfe von Maus und Tastatur von allen Seiten betrachten kann. Diese Sphäre kann Intensitätsdaten ein bis drei verschiedener Wellenlängen in verschiedenen Farbtönen darstellen.
In diesem Kapitel werden die Ergebnisse für eine und für drei verschiedene Wellenlängen präsentiert.

4.1 Modell für eine Wellenlänge

Abbildung 4.1(a) zeigt eine Momentaufnahme der Sphäre bei einer Wellenlänge von 10000Å. Auf diesem Bild kann man die Strahlungsquelle im Hintergrund des Gasriesens sehr gut als hellen weißen Fleck, auch *subsolar point* genannt, erkennen. Da dieser nicht ganz mittig ist, kann man erkennen, dass sich der Beobachter leicht versetzt zur Verbindungslinie zwischen Gasriese und Strahlungsquelle befindet. Außerdem kann man erkennen, dass die Intensität in Äquatornähe am größten ist, wohingegen sie zu den Polen hin abnimmt und es an den Polen beinah schwarz wird. Dies lässt sich dadurch erklären, dass die Strahlung am *subsolar point* senkrecht einfällt, jedoch zu den Polen hin der Einfallswinkel größer wird, der gleiche Raumwinkel eine größere Fläche abdecken muss und somit die Intensität dort abnimmt.



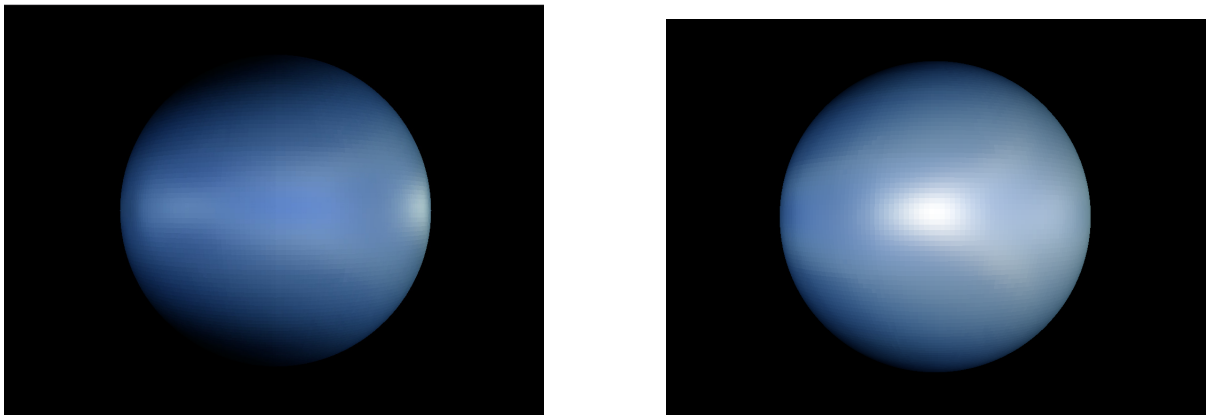
(a) (b)

Abbildung 4.1: Sphären-Modell der Strahlungsintensitäten bei einer Wellenlänge von λ = 10000Å in Beobachterposition (a) und als theoretisch berechnete Seitenansicht (b)

So wie es in Abbildung 4.1(b) dargestellt ist, kann ein Beobachter es niemals sehen, da die Textur für den entfernten Beobachter wahrnehmbare Intensität darstellt und nicht diejenige, die in Richtung der Kamera beobachtbar ist.

4.2 Modell mit drei verschiedenen Wellenlängen

Beim Modell mit drei verschiedenen Wellenlänge wird jeder Wellenlänge eine Farbe (rot, grün oder blau) zugeteilt. Sowohl die Abbildung 4.2(a) als auch die logarithmische Darstellung in Abbildung 3.10 zeigen die Sphäre unter gleichem Winkel zur Lichtquelle. Auch hier und in Abbildung 4.2(b) kann man, wie im Modell mit einer Wellenlänge, erkennen, dass die Polkappen etwas dunkler sind als die äquatoriale Ebene. Dies hat die selben Hintergründe.
Bei der Abbildung 4.2(a) und 4.2(b) kann man zusätzlich zu dem *subsolar point* an der rechten Seite noch eine Dreieckstruktur erkennen. Dies deutet darauf hin, dass der Gasriese im Uhrzeigersinn rotiert. Der etwas dunklere Bereich in dem Dreieck könnte von einer Wellenlänge mit Molekülband stammen. Dies müsste jedoch erst mit den Daten überprüft werden.



(a) (b)

Abbildung 4.2: Sphären-Modell in bei Wellenlängen von λ = 5000Å(rot), λ = 7000Å(grün) λ = 10000Å(blau) einmal mit seitlich einstrahlender Strahlenquelle (a) und einmal aus der gleichen Richtung wie der Beobachter kommenden Strahlenquelle (b)

Abbildung 4.3 zeigt eine Sphäre mit Intensitäten der Wellenlängen λ = 12000 Å (rot), λ = 22000 Å (grün), λ = 23000 Å (blau) aus vier verschiedenen Beobachterperspektiven. Hierbei ist sehr gut zu sehen, dass diese verschiedenen Wellenlängen bei unterschiedlichen Beobachterstandorten ihre Intensitätsmaxima aufweisen.

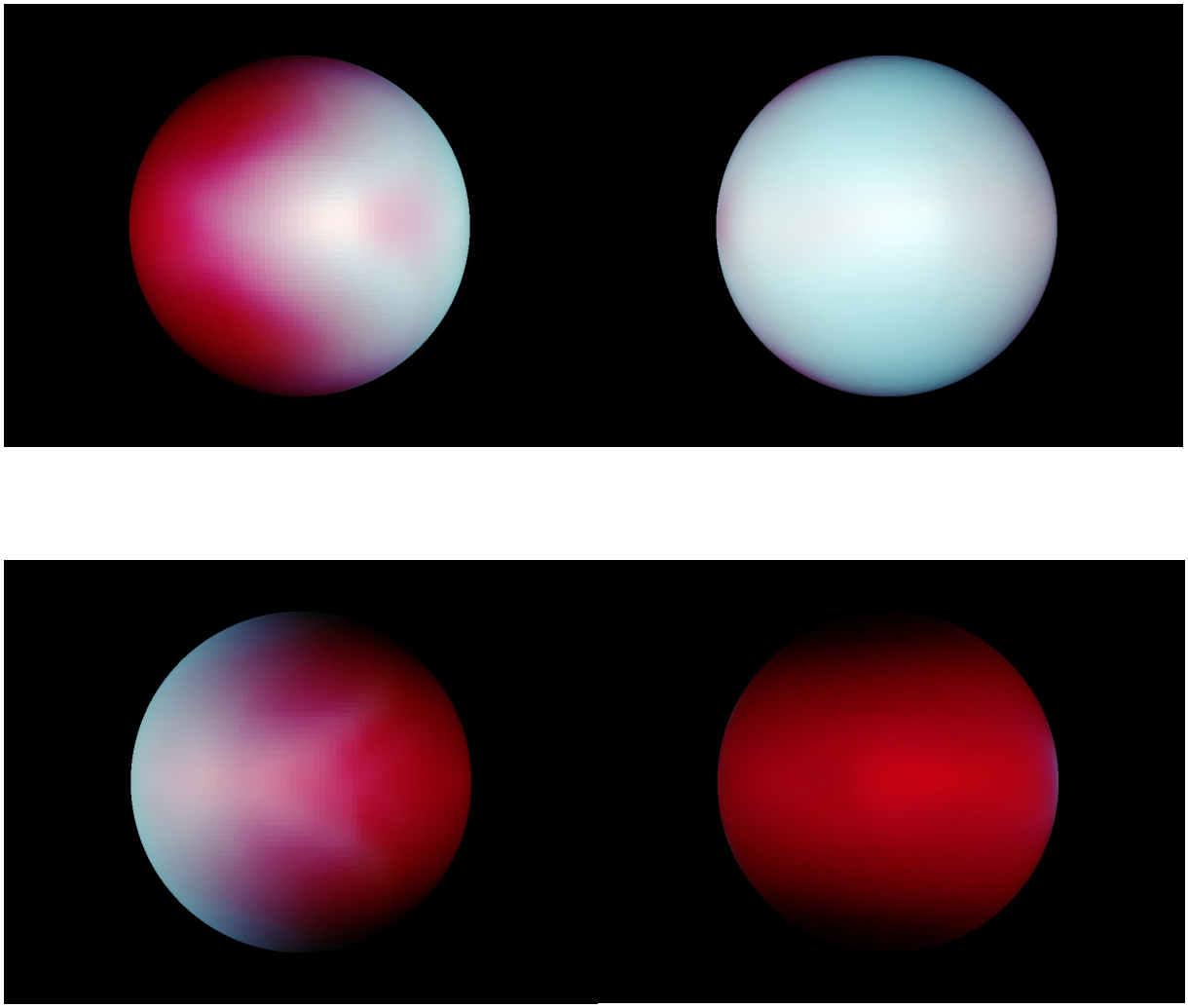


Abbildung 4.3: Darstellung einer Sphäre mit Intensitäten der Wellenlängen λ = 12000 Å (rot), λ = 22000 Å (grün), λ = 23000 Å (blau) aus vier verschiedenen Beobachterperspektiven bei 0°, 90°, 180° und 270°

5 Zusammenfassung

Es wurde ein Programm erstellt, das Intensitätsdaten bis zu dreier Wellenlängen auf einer Sphäre darstellt, die sich durch Tastatureingaben und Maus von allen Seiten betrachten lässt.
Zur Veranschaulichung des in dieser Bachelorarbeit entstandenen Codes dienen Intensitätsdaten eines Gasriesens. Die Intensitätsunterschiede zwischen *subsolar point*, Äquatorebene und den Polregionen können mit Hilfe des entstandenen Programms ausgemacht werden. Außerdem lässt sich erkennen, dass der Planet rotiert.

6 Erweiterungsmöglichkeiten

Man könnte noch die Strahlungsquelle im Hintergrund darstellen, sodass man sehen kann in welchem Winkel zur Hintergrundquelle man sich befindet. Es wurde angenommen, dass sich der Beobachter sehr weit weg vom Stern befindet, sodass der ϕ-Winkel nicht für jeden Punkt neu angepasst werden muss. In Abbildung 6.1 kann man sehen, wie sich der Winkel ändert, wenn der Beobachter unendlich weit entfernt ist (rot) oder wenn er sich endlich nah an der Sphäre befindet (blau).

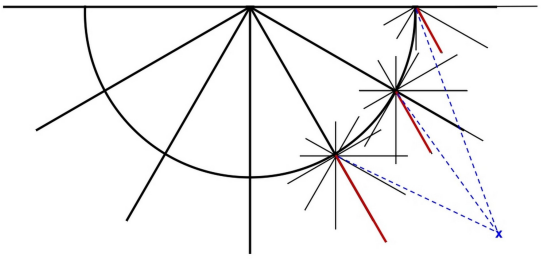


Abbildung 6.1: Skizze der Intensitätsstrahlen. In rot sieht man die Abnahme der Intensitäten für unendlich weit entfernten Beobachter, in blau wie sich der Winkel ändert, wenn sich der Beobachter endlich nah an der Sphäre befindet.

Es wäre interessant zu simulieren, wie es sich verhält wenn der Beobachter näher an die Sphäre heran geht. Hierfür ist es selbstverständlich notwendig, die präzise Geometrie zu berechnen und die Intensitäten für den Standort des Beobachters zu interpolieren. Mithilfe dieser Erweiterung wäre es auch möglich, zu simulieren, wie sich die Strahlung verhält wenn man theoretisch in den Stern hinein fliegt. Phoenix berechnet diese Daten bereits, allerdings werden sie nicht abspeichert, um zunächst noch Speicherplatz zu sparen.
Man könnte auch Darstellungen für das sichtbare Auge, bzw. für Teleskope erstellen. Hierfür wäre es notwendig, über mehrere Wellenlängen zu integrieren sowie Filterfunktionen zu beachten.
Ein anderes Projekt wäre, darzustellen, wie es aussieht wenn man sich auf der Oberfläche eines Planeten befindet und in den Himmel blickt.

Literaturverzeichnis

[Bur07] BURTON, Tarn W.: *Sphere example code*. http://pydoc.net/Python/PyOpenGL-Demo/3.0.0/PyOpenGL-Demo.NeHe.lesson18/. Version: Februar 2007

[Clo14] CLOUDREAM: *Python 2 or Python 3*. http://wiki.python.org/moin/Python2orPython3. Version: 2014

[HB99] HAUSCHILDT, P. H. ; BARON, E.: Numerical solution of the expanding stellar atmosphere problem. In: *Journal of Computational and Applied Mathematics* 109 (1999), September, S. 41–63

[HB10] HAUSCHILDT, P. H. ; BARON, E.: A 3D radiative transfer framework. VI. PHOENIX/3D example applications. In: *The Annals of Applied Probability* 509 (2010), Januar, S. A36. http://dx.doi.org/10.1051/0004-6361/200913064. – DOI 10.1051/0004–6361/200913064

[HB13] HAUSCHILDT, Peter ; BARON, Edward: *PHOENIX Manual*. www.hs.uni-hamburg.de/en/for/ThA/phoenix/manual.html. Version: 2013

[HBA97] HAUSCHILDT, P. H. ; BARON, E. ; ALLARD, F.: Parallel Implementation of the PHOENIX Generalized Stellar Atmosphere Program. In: *The Astrophysical Journal* 483 (1997), Juli, S. 390–398

[Het09] HETLAND, Magnus L.: *Beginning Python - From Novice to Professional*. München : Verlag, 2009

[Hoc13] HOCEVAR, Sam: *OpenGL Online Kurs*. www.opengl-tutorial.org/. Version: 2013

[Kle09] KLEIN, Bernd: *Python Online Kurs*. www.python-kurs.eu. Version: 2009

[Pog] POGOSIAN, Dmitri: *Astronomy of Stars and Galaxies, Vorlesung 8*. www.ualberta.ca/~Epogosyan/teaching/ASTRO_122/lect8/F13.03.gif

[Rut03] RUTTEN, R. J.: *Radiative Transfer in Stellar Atmospheres*. 2003

[S+13] SHREINER, Dave u. a.: *OpenGL Programming Guide*. 8. Addison Wesley, 2013

[SFL+09] SHOWMAN, A. P. ; FORTNEY, J. J. ; LIAN, Y. ; MARLEY, M. S. ; FREEDMAN, R. S. ; KNUTSON, H. A. ; CHARBONNEAU, D.: Atmospheric Circulation of Hot Jupiters: Coupled Radiative-Dynamical General Circulation Model Simulations of HD 189733b and HD 209458b. In: *The Astrophysical Journal* 699 (2009), Juli, S. 564–584. http://dx.doi.org/10.1088/0004-637X/699/1/564. – DOI 10.1088/0004–637X/699/1/564

[Sou] SOURCEFORGE, OPEN-SOURCE PROJECT: *PyOpenGL Dokumentation*. http://pyopengl.sourceforge.net/documentation/

Danksagung

Prof. Dr. Peter Hauschildt danke ich für die Vergabe der Bachelorarbeit und die Themenauswahl. Für die Unterstützung und das Korrekturlesen meiner Bachelorarbeit bedanke ich mich ganz herzlich bei Dr. Sören Witte und Dr. Andreas Schweitzer.

Eidesstattliche Versicherung

Hiermit bestätige ich, dass die vorliegende Arbeit von mir selbständig verfasst wurde und ich keine anderen als die angegebenen Hilfsmittel - insbesondere keine im Quellenverzeichnis nicht benannten Internet-Quellen - benutzt habe und die Arbeit von mir vorher nicht einem anderen Prüfungsverfahren eingereicht wurde. Die eingereichte schriftliche Fassung entspricht der auf dem elektronischen Speichermedium.
Ich bin damit einverstanden, dass die Bachelorarbeit veröffentlicht wird.

Hamburg, den 5. Juni 2015

(Fiona Dorothea Elisabeth Prodöhl)