HASSO-PLATTNER-INSTITUT
Fachgebiet Computergrafische Systeme

Photorealistisches Rendering atmosphärischer Effekte in geovirtuellen 3D-Umgebungen in Echtzeit

Masterarbeit
zur Erlangung des akademischen Grades
"Master of Science"
(M.Sc.)
im Studiengang IT-Systems Engineering
des Hasso-Plattner-Instituts an der
Universität Potsdam

vorgelegt von
Daniel Müller

Aufgabenstellung und Anleitung:
Prof. Dr. Jürgen Döllner
Juri Engel, M.Sc.

Potsdam,
6. September 2012

Zusammenfassung

Diese Arbeit präsentiert ein System zur Bildsynthese atmosphärischer Effekte in virtuellen 3D Umgebungen. Es ermöglicht echtzeitfähiges Rendering dynamischer, qualitativ hochwertiger Umgebungen für die Hintergrundgestaltung (Hintergrundkulisse) in Videospielen und Simulatoren (z.B. Flugzeug- oder Fahrzeugsimulatoren), aber auch architektonischen und historischen Visualisierungen. Um eine einfache Integration zu gewährleisten, werden ausschließlich dem Hintergrund zugehörige Phänomene berücksichtigt. Dabei wird zum einen ein texturbasierter, und zum anderen ein physikalischer, simulationsbasierter Ansatz verfolgt.

Der texturbasierte Ansatz zielt auf die Imitation beliebiger Umgebungen ab und nutzt bekannte, texturbasierte Verfahren welche kaum gestalterische Einschränkungen mit sich bringen. Er ist vielseitig erweiterbar und ideal für Anwendungen, in denen bezüglich der Rendering Leistung nur geringe Kapazitäten alloziert werden können.

Der physikalische, simulationsbasierte Ansatz hingegen, baut auf astronomische Algorithmen und astrophysikalischen Beobachtungen auf. Er wird im Rahmen der Arbeit auf bodennahe Betrachter auf der Erde begrenzt und verfolgt eine ganzheitliche Darstellung: Für einen Betrachter können Gestirne (Sonne, Mond und Sterne), betrachtet durch eine Atmosphäre mit mehreren Wolkenschichten, raumzeitlich korrekt der Realität angenähert abgebildet werden. Die dazu verwendeten, teils neuartigen Modelle und Verfahren stellen den konzeptionellen Schwerpunkt dieser Arbeit dar.

Beide Ansätze sind optimiert für Rendering dynamischer Tag-Nacht-Zyklen innerhalb eines Rendering-Durchlaufs, und sind exemplarisch in ein großes Renderingsystem (OpenSceneGraph) integriert. Performance-Messungen unter verschiedenen Parametrisierungen belegen dem System eine geringe Auswirkung auf die Rendering-Geschwindigkeit. Es wird untersucht wie eine Kombination beider Ansätze genutzt werden kann um auch in performancekritischen Anwendungen hochwertige Hintergrundkulissen zu simulieren. Vergleiche mit Fotos sprechen den von unserem System erzeugten Ergebnissen einen hohen Realitätsgrad und damit einhergehend, erhöhte Glaubwürdigkeit zu. Ausgewählte Erweiterungsmöglichkeiten werden kurz diskutiert.

Abstract

In this work we present a system for image-synthesis of atmospheric effects in virtual 3d environments. It allows realtime rendering of dynamic, high quality environments for background composition (background scenery) for a variety of applications not only in video-games and simulators (e.g., flight- or vehicle-simulators), but also for architectural and historical visualizations. To ensure ease of integration, only the background associated phenomena are taken into account. Thereby a texture-based, and a simulation-based approach are presented.

The texture-based approach targets the imitation of any environment and uses well-known, texture-based methods with hardly any creative limitations. They are expandable and ideal for applications where, in respect to the rendering performance, only low capacities can be allocated.

The simulation-based, astrophysical approach, in contrast, is based on astronomical algorithms and astrophysical observations. Within the scope of this work it is limited to ground-level observer on Earth and targets a holistic representation: For a viewer, celestial bodies are considered to be seen through an atmosphere with several layers of clouds, and are shown as spatiotemporally correct approximations of real world phenomena. The new models and methods used within this approach, provide the conceptual focus of this work.

Both approaches are optimized for rendering dynamic day-night cycles within a single rendering pass and without subsequent processing. They are integrated in a large representative rendering system (OpenSceneGraph). Detailed performance measurements in different scenarios and quality levels show that the system has little effect on overall rendering-speed. We further examine how a combination of both approaches can be used to simulate high-quality background sceneries in performance-critical applications. Comparisons with results of related methods attest results generated with our system, a high degree of photo-realism and, consequently, increased credibility. In some cases (e.g., close-ups of the moon or certain star constellations at night) our results even pass comparisons with photos. Selected expansion capabilities through to immersive virtual realities are briefly discussed.

Inhaltsverzeichnis

Kapitel 1

Einführung

Echtzeitfähige Bildsynthese mit dem Ziel der photo-realistischen Darstellung ist fortwährend integraler Bestandteil der Computergrafik. In Verbindung mit beständig steigender Leistungsfähigkeit der Grafikhardware findet Bildsynthese zunehmend Einsatz in verschiedensten Verbraucher-Plattformen und -Anwendungen. Zu den Plattformen zählen unter anderen Desktop-PCs, Spielkonsolen, Notebooks, Ultrabooks, Netbooks, Tablets und Tablet-PCs, Smartphones, und zunehmend auch Fernsehgeräte. Bei den Anwendungen ist die Bandbreite mittlerweile unüberschaubar. Erkennbar ist aber, dass die Bildsynthese in der vollen Bandbreite Anwendung findet, nicht zuletzt in der Darstellung virtueller 3D Welten. Sie ermöglicht, eine exponentiell wachsende Vielfalt an einerseits offensichtlichen, andererseits oft nur subtilen Feinheiten der Realität zu imitieren oder zu simulieren.

Videospiele und Simulatoren (z.B. Flugzeug- oder Fahrzeugsimulatoren), aber auch architektonische und historische Visualisierungen mit dem Anspruch raumzeitlicher Korrektheit, profitieren oft durch Darstellungen angemessener Hintergrundkulissen. Diese Arbeit vereint verschiedene Verfahren zu einer Software-Bibliothek (System) zur Anreicherung virtueller 3D Welten um dynamische, photo-realistische Hintergrundkulissen. *Hintergrundkulisse* bezeichnet den Teil synthetisierter Darstellungen virtueller 3D Szenen, der vom Betrachter aus als umschließende Umgebung aller Szenenobjekte wahrgenommen wird. Eine angemessene Hintergrundkulisse erfordert zwischen gefühlten Mehrwert der Darstellung und dem zur Bildsynthese erforderlichen Aufwand abzuwägen. Unter geringen Anforderungen kann bereits eine statische, blaue Hintergrundkulisse genügen. Den hohen Anforderungen in modernen, immersiven, virtuellen 3D Umgebungen genügen derartige Vereinfachungen nicht. Vielmehr bedarf es dynamischer, raumzeitlicher – also dem Datum, der Uhrzeit sowie der Position bewusster – Simulationen einer ganzheitlichen Hintergrundkulisse mit dem Ziel, einer, unseren täglichen Erfahrungen und unbewussten Erwartungen kohärenten Darstellung. Ganz-

Abbildung 1.1: *Auszüge simulierter Hintergrundkulissen einer Tag-Nacht-Tag Transition.*

heitlichkeit erfordert in diesem Fall, die Berücksichtigung verschiedenartiger Faktoren wie beispielsweise Wolken und Wetter, Beleuchtungsverhältnisse und Stimmung, Geräuschkulisse wie Windrauschen, oder die Korrektheit sichtbarer Gestirne. Wir beschränken uns auf die visuellen Aspekte und bieten, je nach Anwendungsszenario, zwei unterschiedliche Ansätze: Zum einen wird ein texturbasierter, zum anderen ein physikalisch simulierter Ansatz zur Erzeugung der Hintergrundkulissen verfolgt.

Der texturbasierte Ansatz zielt auf die Imitation beliebiger Umgebungen ab und greift auf bekannte, texturbasierte Verfahren zurück. Diese bieten höchste gestalterische Flexibilität: Sie erlauben eine Gestaltung unter Anwendung beliebiger Stile aus Kunst, Fotografie, und Bildsynthese. Der Ansatz ist vielseitig erweiterbar und ideal für Anwendungen, in denen bezüglich der Rendering Performance nur geringe Kapazitäten alloziert werden können.

Der physikalisch simulierte Ansatz baut auf astrophysikalischen Modellen und Beobachtungen auf und ist besonders geeignet für geovirtuelle, immersive 3D Umgebungen. Der Gestaltungsspielraum ist hier durch die verschiedenen physikalischen und abstrakten Parameter festgelegt. Alle Modelle sind spezialisiert auf Betrachter nahe der Erdoberfläche und dienen einer ganzheitlichen Darstellung des Himmels: Sonne, Mond, und Sterne, betrachtet durch eine Atmosphäre mit mehreren Wolkenschichten können der Realität angenähert dargestellt und angepasst werden. Die dazu verwendeten, teils neuartigen Modelle und Verfahren stellen den konzeptionellen Schwerpunkt dieser Arbeit, und wurden für die *Vision, Modelling and Visualization 2012* zur Veröffentlichung akzeptiert (Müller et al., 2012).

Beide Ansätze sind ausgelegt für Rendering dynamischer Tag-Nacht-Zyklen (wie in Abbildung 1.1 illustriert) innerhalb eines einzigen, integrierten Rendering-Durchlaufs (*Renderingpass*) ohne nachträglicher Verarbeitung (*Postprocessing*). Das bedeutet das jede Form des Postprocessing innerhalb eines oder in folgenden Passes nicht vorgegeben ist und dem Anwender obliegt. Folglich nimmt die Darstellung einer Hintergrundkulisse an sich keinen direkten Einfluss auf die Darstellung der Szenegeometrie. Die Gesamtdarstellung beeinflussende Effekte wie Tiefenunschärfe, Nebel, Luft- und Farbperspektive, oder High Dynamic Range Rendering, werden bewusst nicht durch unser System vorgenommen, und müssen im jeweiligen Anwendungsfall eigenständig umgesetzt werden. Die entwickelte Software-Bibliothek mit all seinen Modulen und Verfahren ist exemplarisch für das freie Rendering-Framwork OpenSceneGraph (OSG) implementiert und mit dem Namen *osgHimmel* unter der neuen BSD-Lizenz als Open-Source-Software frei verfügbar.

osgHimmel zielt auf einfache Integration durch minimale Anforderungen, Bereitstellung aufgeräumter Schnittstellen, sowie geringer Leistungsbeeinträchtigung ab. Es bietet vollständig vorkonfigurierte, bei Bedarf dynamische Hintergrundkulissen mit vielfältigen Konfigurationsmöglichkeiten. Die qualitativ hochwertigen, photo-realistischen Himmel bieten eine erhöhte Glaubwürdigkeit der Hintergrundkulissen und verbessern damit die Gesamtdarstellung. Speziell die Ergebnisse des astro-physikalischen, simulationsbasierten Ansatzes erlauben einen Vergleich mit entsprechendem Fotomaterial.

osgHimmel dient der Anreicherung von beliebigen, virtuellen 3D Welten um hochwertige Hintergrundkulissen mit besonderem Fokus auf die Darstellung eines Himmels mit dynamischen Tag-Nacht-Zyklen bei minimalen Leistungseinbußen.

Kapitel 2

Systemanforderung

Dieser Arbeit ist eine Seminararbeit vorangegangen, welche die von Kegel (2006) vorgestellten Verfahren zum Rendering von Wolken und Atmosphäre des Virtual Rendering System (VRS) nach OpenSceneGraph[1] (OSG) portiert. Während einer ersten funktionierenden Implementierung in OSG kamen schrittweise neue Anforderungen dazu, da inzwischen modernere Verfahren existierten. Schließlich wurde die Implementierung verworfen und mit der Entwicklung von *osgHimmel* begonnen.

In diesem Kapitel werden die Systemanforderungen für *osgHimmel* herausgearbeitet. Die Motivation dieser Arbeit ergibt sich aus der Abgrenzung zu ähnlichen, bereits existierenden Systemen. Aber auch in der Entwicklung neuer Modelle an sich, z.B. zur Visualisierung und Simulation von Mond und Sternen. Verschiedene Anwendungsszenarien werden benannt und dienen zur ungefähren Einordnung des Systems sowie dessen Leistungsumfang. Die wichtigsten Anforderungen bezüglich des Gesamtsystems leiten sich aus den Anwendungsszenarien ab. Dabei wird der Schwerpunkt der Arbeit auf den simulierten Ansatz gelegt (Kapitel 4). Bemerkungen zu qualitativen Anforderungen einzelner Bestandteile der Simulation und Bildsynthese weisen auf Anforderungen an einzelne Verfahren sowie deren Aggregation zu einer Gesamtdarstellung hin. Im Überblick über verwandte Verfahren wird auf deren Schwachpunkte eingegangen und auf offene Probleme hingewiesen. Gleichzeitig werden die vom System simulierten Phänomene im Einzelnen angesprochen und die Schwerpunkte dieser Arbeit begründet. So sind beispielsweise astronomische Algorithmen als Konsequenz der Anforderungen existenziell für das zu implementierende System. Sie stehen aber nicht im Fokus dieser Arbeit, da diesbezüglich fast ausschließlich auf Grundlagenwissen der Astronomie zurückgegriffen werden kann.

2.1 Motivation und Abgrenzung

Hochwertige Hintergrundkulissen vermögen die Glaubwürdigkeit photo-realistischer Visualisierungen vielfältiger Anwendungsbereiche essentiell zu verbessern. Viele der heutigen Anwendung besitzen eine zeitabhängige Dynamik, da in ihnen entweder eine eigene, virtuelle Zeit abläuft, oder zu einer realen Zeit Bezug genommen wird. Eine dynamische, der Zeit bewusste Hintergrundkulisse für nahtlose Tag-Nacht-Übergänge ist daher für viele Anwendungen relevant. Bezüglich einer Simulation erfordert dies die Komposition verschiedener Himmelsphänomene zu einer einheitlichen, kohärenten Darstellung.

[1] http://openscenegraph.org/

Für offline Rendering existieren viele kommerzielle und nicht-kommerzielle Lösungen die unter hohen Rechenaufwand und vergleichsweise unbegrenzten Renderingzeiten, photo-realistische Ergebnisse erzielen können. Hier sei als Beispiel Terragen™2 von Planetside Software[2] erwähnt dessen Visualisierung von Atmosphäre und Wolken die Grenzen zur Realität überwindet. Dies zeigt, dass das Wissen um Modelle und Verfahren zur realitätsgetreuen Bildsynthese des Himmels bereits sehr ausgereift ist. Für die Anwendung in Echtzeit genügen diese Modelle zur Zeit jedoch nur im Ansatz. So wird das Rendering des Himmels oder allgemeiner, von Hintergrundkulissen, in Echtzeitanwendungen nur sehr zielgerichtet, mit großen Einschränkungen bzw. groben Annäherungen eingesetzt. Bei einer angestrebten Bildwiederholrate von 60 Bildern pro Sekunde in heutigen Anwendungen, erlaubt schließlich selbst die stärkste Hardware keine umfangreichen Berechnungen für Hintergrundkulissen. Zumal der Fokus der Bildsynthese nur selten auf die Hintergrundkulisse allein beschränkt ist.

Das Virtual Terrain Project[3] bietet als umfangreiche Sammlung verschiedener Ressourcen einen Einstiegspunkt, Verfahren und Systeme zum Thema zu finden. Zum Zeitpunkt der Arbeit sind uns nur wenige nicht-kommerzielle Systeme mit einem ähnlichen Fokus wie *osgHimmel* bekannt. In Videospielen und deren oft proprietären Werkzeuge finden sich nur sehr spezielle, auf individuelle Szenarien hoch optimierte Implementierungen. Es lassen sich Demoanwendungen zum Rendering von Wolken finden, welche meist auch ein Atmosphärenmodell einschließen. So haben zum Beispiel Harris & Lastra (2001) und Bruneton & Neyret (2008) ihre Implementierungen veröffentlicht. Speziell für OpenSceneGraph exitiert osgEphemeris[4], welches eine gute astronomische Grundlage bietet, jedoch seit 2007 nicht mehr weiterentwickelt wird. Neuere Systeme wie Silverlining von Sundog Software[5] sind meist ausgelegt für dynamische Darstellung von Tagesszenen, bieten jedoch keine überzeugende Nachtszenen.

Mit *osgHimmel* möchten wir ein offenes, nicht-kommerzielles, echtzeitfähiges System zur Erzeugung von Hintergrundkulissen, mit dem Schwerpunkt photo-realistischer, ganzheitlicher Tag-Nacht-Zyklen, unter Berücksichtigung heutiger Leistungsanforderungen und Verwendung neuster Verfahren zur Verfügung stellen. Im Gegensatz zu vorhandenen Systemen, bietet *osgHimmel* neben der Komposition von Gestirnen, Atmosphäre, und mehreren Wolkenschichten auch raumzeitliche und astronomische Korrektheit. Des weiteren werden keine zusätzlichen Abhängigkeiten und Anforderungen an die Rendering Pipeline gestellt, sodass dass System potentiell für eine große Bandbreite an Anwendungsszenarien einsetzbar ist und auf verschiedenen Plattformen funktioniert: eine schrittweise Reduzierung von umfangreicher Simulation bis hin zur vorberechneten Hintergrundtextur ist möglich. Zusammen mit dem System, stehen mit mehreren Textur-Sammlungen für texturbasierte Anwendungen oder dem erweiterbaren Sternenkatalog viele zusätzliche, freie Materialien zur Verfügung. Auch die unzähligen freien und kommerziellen Umgebungstexturen im Internet werden vom System in den gängigsten Projektionsarten unterstützt. Eine Orientierung an minimalen Demoanwendungen sowie der mit Codebeispielen angereicherten Online Dokumentation ermöglichen schließlich eine schnelle und unkomplizierte Einbindung.

[2] http://planetside.co.uk
[3] http://vterrain.org/
[4] http://andesengineering.com/Projects/OsgEphemeris
[5] http://sundog-soft.com/sds/

2.2 Anwendungsszenarien

Die ersten Anwendungsszenarien zu *osgHimmel* entstammen der Notwendigkeit eines Himmels für Präsentationen von prototypischen, echtzeitfähigen Geovisualisierungen. Dabei sollte der Aufwand zur Einbindung des Himmels, bzw. einer Hintergrundkulisse minimal sein. Daraus entstand das allgemeinere Szenario eines Systems zur Anwendung in beliebigen Visualisierungen mit geringem Fokus auf Hintergrundkulissen, die jedoch im Ergebnis stark von ihnen profitieren könnten. Das System darf folglich nur geringe technische und strukturelle Anforderung stellen. Gerade für eine Nutzung in komplexen Renderingpipelines wiegen bei diesbezüglichen Entscheidungsprozessen, erforderliche Anpassungen mehr als die evtl. nur marginal empfundene Verbesserung der Darstellung. Ähnliches gilt auch für Postprocessing in einem oder mehreren Rendering-Durchläufe (Postprocessing). Indem wir die Verfahren unseres Systems auf *Single-Pass*, also einem einzigen Renderingpass, beschränken, erweitern wir die Menge möglicher Anwendungsfälle. Jedes Postprocessing baut auf zuvor erzeugte Puffer, die oft sehr durchdachte, systemspezifische Anforderungen erfüllen müssen, um effizient gespeichert und zur Realisierung mehrerer Effekte verwendet zu werden. Ein Eingriff in diese Anforderungen kann kritische Veränderungen verlangen und im Extremfall, bereits implementierte Techniken unbrauchbar machen, bzw. die Entscheidung gegen die Verwendung unseres Systems bedeuten.

Speziell für den simulierten Ansatz, ist es hilfreich, ein Standardszenario zu identifizieren, an dem die einzelnen Verfahren vorkonfiguriert und Ergebnisse bewertet werden können: Eine dynamische virtuelle 3D Welt, in der während der Interaktion virtuelle Zeit vergeht, folglich ein dynamischer Tag-Nacht-Zyklus wünschenswert ist. Die Geschwindigkeit der virtuellen Zeit, also das Verhältnis virtueller Sekunden zu realen Sekunden, kann dabei zwischen wenigen Sekunden bis zu ganzen Tagen pro realer Sekunde variieren (Zeitraffer) und bei Bedarf auch rückwärts ablaufen. Der Betrachter befindet sich auf unserer Erde, möglichst bodennahe, also innerhalb der Atmosphäre, und im Falle von Wolken unterhalb der untersten Wolkenschicht. Bei Angabe einer genauen geografischen Lage und eines Datums, wird eine korrekte Darstellung der Gestirne gewünscht. Erwartet werden alle wesentlichen Phänomene des Tag- und Nachthimmels in einer über alle Phasen hinweg konsistenten, hochwertigen, aber auch unaufdringlichen Darstellung, um nicht von wesentlichen Objekten der jeweiligen Visualisierung abzulenken. Im Rahmen der Integrität und Ganzheitlichkeit wird erwartet, die Hintergrundkulisse auch für Reflektionen und globale Beleuchtung bereitzustellen.

Ein weiteres Szenario bezüglich des texturbasierten Ansatzes wurde im vorhergehenden Abschnitt bereits angesprochen: Für die Hintergrundkulisse steht beispielsweise bereits bevorzugtes Bildmaterial zur Verfügung oder die Umgebung ist vorgegeben (z.B. eigene Fotos am gewünschten Ort). Das Bildmaterial wiederum kann in verschiedenen Projektionen vorliegen, und es wird erwartet, dass die gängigsten Varianten direkt vom System unterstützt werden.

Diese Szenarien sind neben der Visualisierung geovirtueller 3D Welten auch für Anwendungen in Bereichen der Bildung und Aufklärung (z.B. Museen und Planetarien), Ausbildung (z.B. Trainingssimulationen), Planung, Edutainment, Unterhaltung und Videospielen hinreichend. Zur Evaluation der Verfahren und Manipulation interner Parameter wird eine grafische Oberfläche verwendet: ein, dem System zugehöriger Editor.

2.3 Anforderungen

Entsprechend der zuvor vorgestellten Anwendungsszenarien (Abschnitt 2.2) ließen sich für die API des Systems funktionale und nicht-funktionale Anforderungen ableiten. Diese dienten als Orientierung zur Entwicklung des Systems und sind wie spezifiziert, oder darüber hinaus, implementiert. Die folgenden Auflistungen dienen daher nur der Benennung wesentlicher Systemfähigkeiten und -Eigenschaften, und sind nicht streng formal formuliert.

2.3.1 Funktionale Anforderungen

- Zeitangaben können gleichwertig entweder als Fließkommazahl im Bereich $[0, 1]$ mit 0 für 0:00 Uhr und 1 für 24:00 Uhr sowie in UTC vorgenommen werden.
- Die Geschwindigkeit der Zeit wird als Verhältnis von virtueller Sekunden pro realer Sekunde angegeben: 3600 definiert einen Tag-Nacht-Zyklus von 24 realen Sekunden.
- Das System implementiert die vier bekanntesten Textur Projektionen zur Umgebungsdarstellung: Polar-, Cube-, Sphere- und Paraboloid-Projektion.
- Dynamische Übergänge mittels Verblendung zweier texturierter Hintergrundkulissen sind durch eine Übergangsdauer an vordefinierten Zeitpunkten bestimmt.
- Unter Angabe der geografischen Lage, eines Datums, sowie eines Zeitpunktes, simuliert das System die Position der Gestirne in einer für eine glaubhaften Darstellung ausreichenden Genauigkeit.
- Entwurf und Umsetzung benötigter Generatoren (z.B. Noise für Wolken) sowie mathematischer Funktionen (z.B. Interpolation) stehen unabhängig von OSG zur Verfügung.
- Die einzelnen, simulierten Phänomene sind unabhängig voneinander, in Modulen gekapselt, und können beliebig aggregiert werden (z.B. Himmel ohne Mond und Sterne).
- Es ist exemplarisch ein Verfahren zur Wolkendarstellung, mit dynamischer Translation und wolkeneigener Transformation, implementiert.
- Das System bietet die Möglichkeit, Hintergrundkulissen direkt in eine Umgebungstextur (in diesem Falle einer cubemap) zu rendern, um anderen Verfahren, beispielsweise Reflektion, globale Beleuchtung, oder Export von Bildmaterial, zur Verfügung zu stehen.

2.3.2 Nicht-Funktionale Anforderungen

- Das System führt keine weiteren Abhängigkeiten zu anderen Bibliotheken ein, und nutzt ausschließlich Kernfunktionalitäten von OSG.
- Alle Schnittstellen, über die Bildmaterial oder sonstige Ressourcen zugewiesen werden, müssen über OSG interne Datenstrukturen, unabhängig von deren jeweiligen Datenformaten, erfolgen.

- Die Verwendung des Systems, erfordert keine Umstrukturierung des Szenengraphs und ist vollständig, nicht-invasiv in einer Node (OSG Knoten) gekapselt.
- Die Verwendung des Systems, führt keine zusätzlichen Anforderungen (wie beispielsweise weitere Rendering-Durchläufe oder Kameras) an die Rendering-Pipeline ein.
- Alle implementierten Verfahren kommen ohne Postprocessing aus, funkionieren also vollwertig innerhalb eines Renderingpasses.
- Alle Verfahren beider Ansätze erzielen ein gutes Verhältnis zwischen Bildqualität und Leistungsverbrauch für die benannten Anwendungsszenarien.
- Die verwendeten astronomischen Algorithmen sind vollständig durch Tests auf Korrektheit abgedeckt, und bieten zuverlässige Ergebnisse.
- Das System ist mit dem Namen *osgHimmel* unter der neuen BSD-Lizenz veröffentlicht, und frei verfügbar.
- Die Weiterentwicklung des Systems durch Dritte wird ermöglicht (öffentliches Projekthosting, Online-Dokumentation, etc.).
- Alle, zur minimalen Funktionsfähigkeit des Systems, notwendigen Ressourcen (Bildmaterial, Sternenkataloge, etc.) sind öffentlich und frei verfügbar.

2.3.3 Anmerkungen zur Qualität

Entgegen klassischen Ansätzen, setzen sich neuere Verfahren, speziell zur Himmelsdarstellung, aus einer Vielzahl einzelner, teils unabhängiger aber auch sich wechselseitig beeinflussender Komponenten zusammen. Dabei ist es für die Glaubwürdigkeit einer Darstellung enorm wichtig alle Komponenten konsistent aufeinander und auf realen Sinneserfahrungen abzustimmen. Astronomische Korrektheit kann numerisch verifiziert werden, und feine Abweichungen fallen üblicherweise nicht auf. Da der Mensch jedoch eine erhebliche Prägung durch alltägliche Sinneserfahrung mit den simulierten Phänomenen hat, fallen kleinste visuelle Unstimmigkeiten auf. Die wahrgenommene Genauigkeit in der Farbdarstellung hängt von einer Vielzahl von Einflüssen ab. Unter anderem tragen variierende Auflösungen und Pixeldichten sowie die Farbkalibrierung des Ausgabegeräts wesentlich zur subjektiven, durch individuelle Erwartungen und Erfahrungen begründete Glaubwürdigkeit bei.

Frühzeitige Vereinfachung bereits auf Modelebene, senken stets die maximal zu erzielende Genauigkeit der darauf aufbauenden Verfahren. Dies ist bei der Wahl und Implementierung der Verfahren zu berücksichtigen, um diesbezüglich unnötigen Aufwand zu vermeiden. Kompromisse bei der Genauigkeit der Berechnungen zugunsten der Leistung sind jeweils im Rahmen der vorgesehenen Anwendungsszenarien zu rechtfertigen. Die in dieser Arbeit vorgestellten Verfahren zielen zwar stets auf astrophysikalische Exaktheit ab, beanspruchen diese jedoch nicht notwendigerweise.

2.4 Übersicht verwandter Verfahren

Texurbasierte Hintergrundkulissen stellen bis heute den in den meisten Anwendungsszenarien verwendeten Ansatz. Diese sind sehr einfach zu implementieren und zeichnen sich durch minimale Rechenleistung aus. Die wichtigsten Gründe, diese meist älteren Verfahren auch in *osgHimmel* zu unterstützen, sind die Einfachheit und Bekanntheit des Konzepts, sowie die große Menge an bereits vorhandenem Bildmaterial im Internet. Mit der Polarkoordinaten Projektion (*Polar-Mapping*) stellten Blinn & Newell (1976) eines der ersten Verfahren zur Umgebungsdarstellung mit dem Schwerpunkt auf Reflektionen vor. Dabei wurde in einer zweidimensionalen Textur die vollständige, dreidimensionale Umgebung über in Polarkoordinaten transformiert und abgebildet. Darauf folgten Miller & Hoffman (1984) mit sphärischen Projektion (*Sphere-Mapping*), bei dem die Umgebung in das innere einer Halbkugel projiziert und als Textur abgelegt wird. Mit der kubischen Projektion (*Cube-Mapping*) präsentierten Greene (1986) das wohl am häufigsten verwendete Verfahren. Dabei wird die Umgebung in die Innenflächen eines Würfels projiziert. Mit der paraboloide Projektion (*Paraboloid-Mapping*) folgte eine weitere Technik von Heidrich & Seidel (1998), welches ähnlich wie die sphärische Projektion, jedoch nur eine Hälfte der Umgebung abbildet, und somit für eine vollständige 3D Umgebung zwei Texturen benötigt.

Die Aspekte der Simulation von Hintergrundkulissen zu Tag und Nacht wurden bisher immer isoliert betrachtet: Eine realistische Darstellung der Atmosphäre bei Tag hat in den letzten Jahren große Aufmerksamkeit erfahren. Eine aktuelle, sehr interessante, jedoch weniger technische Auseinandersetzung mit diesem Thema bietet Hoeppe (2007). Bezüglich der echtzeitfähigen Verfahren, ist die Entwicklung von einfachen Farbverläufen zu genauen Streuungsmodellen vorangeschritten. Lässt man die Farb- und Luftperspektive außen vor, erfüllen alle Verfahren unsere Anforderung, in nur einem Renderingpass zu erfolgen. Um einen guten Überblick zu erhalten, haben wir uns mit den folgenden Arbeiten auseinander gesetzt und deren Verfahren teilweise implementiert: Preetham et al. (1999); Dobashi et al. (2002); Stokholm Nielsen (2003); Josth (2005); Haber et al. (2005); Wang et al. (2007). Letztlich haben wir uns für das Verfahren von Bruneton & Neyret (2008) entschieden. Es zeichnet sich durch eine genaue Annäherung an real gemessenen Leuchtdichtewerten aus (Zotti et al., 2007), und bietet korrekte Farben bei Abend- und Morgendämmerung, einschließlich des Erdschattenbogens. Rechenaufwändige Simulationsschritte, wie der des Lichttransports, werden vorberechnet und entsprechend effizient für die Grafikkarte aufbereitet.

Ein ähnlich großes Interesse wie bei der Atmosphärendarstellung, kommt der Wolkendarstellung entgegen. Hier unterscheiden sich die Modelle und Verfahren jedoch wesentlich stärker voneinander. Wir entscheiden uns gegen die Verwendung eines 3D Verfahrens, wegen der zusätzlichen Handhabung (Bouthors et al., 2008) und der notwendigen, meist manuellen Modellierung (Wang, 2003; Schpok et al., 2003) von Geometrie. Stattdessen verwenden wir zweidimensionale, prozedurale Ansätze inspiriert durch die Arbeiten von Roden & Parberry (2005) und Hasan et al. (2005). Mittels grober Annäherungen des Streuungsverhaltens des Lichts, werden dynamische, quasi-dreidimensionale Wolken und Wolkendecken vorgetäuscht.

Für die astronomisch simulierte, Echtzeit-Visualisierung des Mondes sind uns zum Zeitpunkt der Erstellung dieser Arbeit keine Publikationen bekannt. Hier findet man im Regelfall nur eine statische Darstellung des Mondes vor. Dabei wird der Mond gemeinhin als separate oder aggregierte, statische Textur mit festgelegter Phase, Orientierung, Farbe und Intensität

modeliert. Ein Simulation wurde von Jensen et al. (2001a) vorgestellt. Diese generiert eine raumzeitbezogene, photo-realistische Darstellung des Mondes in korrekter Größe, Position, Farbe, Ausrichtung, und Schattierung mittels angenäherter Oberflächenstreuung des Mondes. Zur Abbildung von Mondfinsternissen, schlagen Yapo & Cutler (2009) vor, mittels physikalisch simuliertem Photonen-Tracing die Färbung des Mondes während der Finsternis anzunähern. Dieser Ansatz ist jedoch nicht echtzeitfähig. Zur Bildsynthese einer Sonnenfinsternis konnten wir keine Arbeiten finden.

Für die Bildsynthese astrophysikalisch simulierter Sterne sind uns ebenfalls keine umfassenden, publizierten Verfahren bekannt. Bisher wird hier auf statisches Rauschen wie in Roden & Parberry (2005) oder hochauflösende Sternkarten mit tatsächlichen Positionen und Farben wie bei Jensen et al. (2001a) zurückgegriffen. Diese Ansätze sind unzureichend, da entweder die Ergebnisse unglaubwürdig erscheinen oder Probleme beim Textursampling auftreten, wie beispielsweise verschwommene Sterne. Neben der korrekten Positionierung der Sonne und des Mondes ist auch die der Sterne für eine glaubhafte Darstellung der Gestirne relevant. Nadeau et al. (2000) schlagen vor die einzelnen Sterne als Geometrie mit einer Gaussian-Point-Spread Funktion texturiert, und über ihre Entfernung abgeschwächten Intensität und Größe zu rendern. Dieser Ansatz überwindet zumindest die Nachteile der rein texturbasierten Verfahren. Bezüglich der genauen Farb- und Intensitätsannäherung einzelner Sterne, enthalten Krystek (1985); Hunt (1987); Olson (1998) alle benötigten Informationen. Wie kürzlich von Magnor et al. (2010) bemerkt, existieren keine Lösungen um Streueinflüsse und Funkeln (*Szintillationen*) von Sternen zu simulieren. Für Szintillationen, nutzen Video-Spiele häufig einen Trick, bei dem statische Sterntexturen mit sich bewegenden Rauschen (*Noise-Map*) maskiert werden.

Die Komposition der erwähnten Phänomene zu einem dynamischen Tag-Nacht-Zyklus ist schließlich ein weiteres, bisher unbeachtetes Problem in der echtzeitfähigen Bildsynthese. Es scheint als wäre das von Jensen et al. (2001a,b) vorgestellte System dazu in der Lage, jedoch sind keine genauen Informationen bezüglich der Komposition sowie des Transitions- und Intensitäts-Managements bekannt. In Braun & Cohen (2011) wird eine lineare Helligkeitsanpassung als Funktion über die Tageszeit verwendet. Objekte der Szene sowie beliebige Phänomen des Himmels können dabei zur Nacht entsprechend in ihrer Darstellung verdunkelt werden.

Die Referenzimplementierung aller benötigten astronomischen Algorithmen basiert auf Meeus (1994). Abweichungen hierzu, sowie weitere Referenzen zu den einzelnen, hier vorgestellten Verfahren werden zu Gunsten der Lesbarkeit in den jeweiligen Abschnitten erläutert.

Kapitel 3

Umgebungs-Rendering mittels aufbereiteter Texturen

In der zweiten Hälfte der 90er Jahre, verbreitete sich mehr und mehr die Idee der *Sky Box* zur Bereitstellung einer Hintergrundkulisse, welche seither in allen Bereichen der echtzeitfähigen Visualisierung, eingesetzt. Dabei wird ein virtueller Würfel an die Position einer virtuellen Kamera gebunden, und entsprechendes Bildmaterial blickwinkel-unabhängig in den Würfel projiziert. Das Rendering geschieht ohne Berücksichtigung des Tiefen-Puffers, was den Eindruck einer unendlich weit entfernten Kulisse erweckt. Anstelle des Würfels können auch andere Geometrien und beliebige Projektionsarten genutzt werden. Als wohl wichtigste Projektionsart, gilt die Kubische Projektion (Cube-Mapping), welche in einer Vielzahl von Verfahren wie Beleuchtung, Reflektion, Schatten, oder eben als einfache Hintergrundkulisse (Abbildung 3.1) zum Einsatz kommt.

Abbildung 3.1: *Typische Szene aus dem Spiel Rage von id Software LLC (2011): Eine hochauflösende, statische Hintergrundkulisse vervollständigt die virtuelle 3D Landschaft und erhöht die Glaubwürdigkeit der Darstellung insgesamt.*

Abbildung 3.2: *Beispiel einer typischen Textur für Polar-Mapping. In diesem Fall ein hochauflösendes, für die obere Halbkugel transformiertes Foto.*

osgHimmel bietet die Möglichkeit, die vier meistgenutzten Textur Projektionen zur allgemeinen Umgebungsdarstellung zu verwenden. Im Folgenden werden diese vorgestellt und kurz bewertet. Darüber hinaus werden Erweiterungen um zeitgesteuerte Transitionen, Horizontverblendungen und Rotation um den Zenit erläutert. Minimale Implementierungsbeispiele sind in Abschnitt A.3 aufgeführt.

3.1 Übersicht der Projektionen

Unabhängig von der Projektion, werden alle folgenden Verfahren einheitlich über ein am Bildschirm ausgerichtetes Rechteck (*Screen-Aligned Quad*) im Fragment Shader realisiert. Da das Screen-Aligned Quad auch für die Simulation genutzt wird, ist die genaue Anwendung in Kapitel 5 beschrieben.

3.1.1 Polarkoordinaten Projektion (Polar-Mapping)

Die Polarkoordinaten Projektion wurde von Blinn & Newell (1976) vorgeschlagen und ist auch als *Newell/Blinn-Mapping*, *Sphere-Mapping* oder *Latitude/Longitude-Mapping* bekannt. Um der Verwechselung mit dem Sphere-Mapping vorzubeugen, wird es in *osgHimmel* als Polar-Mapping identifiziert. Dabei wird wahlweise die Oberfläche bzw. Innenfläche einer Kugel (full-mode) oder Halbkugel (half-mode) vollständig in eine Textur abgelegt.

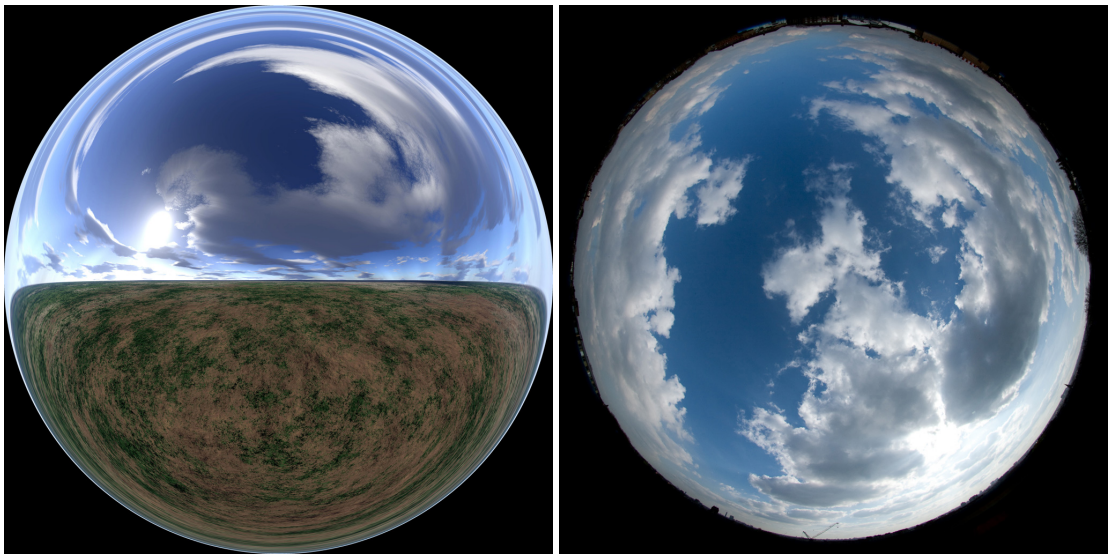
Längengrade werden im Bereich $[0, 2\pi]$ in der Breite und Breitengrade in $[-\pi, \pi]$ bzw. $[0, \pi]$ der Höhe entlang abgetragen (Abbildung 3.2). Beim Rendering ergeben sich für die Richtung r folgende Texturkoordinaten u und v:

$$u = 0.5\pi^{-1}\, atan(r_x, r_y), \tag{3.1}$$

$$v_f = 1.0\pi^{-1}\, acos(-r_z), \tag{3.2}$$

$$v_h = 2.0\pi^{-1}\, asin(+r_z). \tag{3.3}$$

Ein Nachteil sind die Singularitäten an den Polen, welche unnötig viel Detail und damit Speicherplatz, sowie eine starke Textur-Filterung erfordert. Für Bereiche nahe des Horizonts, welche in den meisten Anwendungsszenarien der wesentlichen Blickrichtung entsprechen, liefert sie dafür einen gut kontrollierbaren Detailgrad. Zudem ist Textur-Filterung am horizontalen Übergang von Kante zu Kante hardware-unterstützt möglich. Entsprechendes Bildmaterial kann einfach transformiert werden und ist weit verbreitet.



(a) *Typische, computergenerierte Textur für Sphere-Mapping.* (b) *Textur der oberen Hemisphäre für Paraboloid-Mapping.*

Abbildung 3.3: *Der Vergleich zeigt, das beim Sphere-Mapping im Gegensatz zum Paraboloid-Mapping, mit einer Textur unter zusätzlicher Verzerrung die vollständige Umgebung abgebildet werden kann.*

3.1.2 Sphärische Projektion (Sphere-Mapping)

Bei dieser Projektion von Miller & Hoffman (1984), wird die vollständige Umgebung als Reflektion einer perfekt spiegelnden Kugel (light probe) abgebildet. Dies kann in der Realität z.B. durch Fotografieren einer Weihnachtskugel erzeugt werden, wobei der Fotografierende oder die Kamera dabei im Zentrum der Abbildung zu sehen ist. Als Hintergrundkulisse ist diese Projektion ideal, wenn die Bewegungsrichtung bzw. das wesentliche Blickfeld in Richtung einer Hemisphäre begrenzt ist, oder für einen Himmel, dass Zentrum der inneren Hemisphäre zum Zenit hin ausgerichtet erfolgt.

Ein Vorteil gegenüber dem Polar-Mapping (full-mode) ist, dass nur eine Singularität entlang des Kreisrands vorliegt (Abbildung 3.3(a)). Die äußere Hemisphäre ist enorm verzerrt, und bedarf starker Textur-Filterung. Geringe Texturgrößen oder ungenaues Sampling, verstärken dabei die Sichtbarkeit der Singularität gegenüber der inneren Hemisphäre. Ein weiterer Nachteil ist die für rechteckiges Bildmaterial ungünstige Platznutzung. So ergibt sich eine ungenutzte Fläche von $1 - 0.5^2\pi \approx 0.2146$, also ca. 21% der Textur. Für einen Pixel in Richtung r ergeben sich die Texturkoordinaten u und v wie folgt:

$$u = 0.5 - r_x m^{-1}, \tag{3.4}$$

$$v = 0.5 + r_z m^{-1}, \tag{3.5}$$

$$m = 2.0\sqrt{r_x^2 + r_z^2 + (1.0 - r_y)^2}. \tag{3.6}$$

Die Hintergrundkulisse kann beliebig auf den Anwendungsfall, durch Multiplikation der Sampling-Richtung r mit einer entsprechenden Rotationsmatrix, orientiert (Ausrichtung am Zentrum der inneren Hemisphäre) werden. In *osgHimmel* geschieht dies durch Angabe der Richtung zum Zentrum der inneren Hemisphäre. Diese Projektionsart ist bei der Bildsynthese von Hintergrundkulissen besonders zur Darstellung des Himmels interessant, also bei einer Orientierung in Richtung des lokalen Zenit.

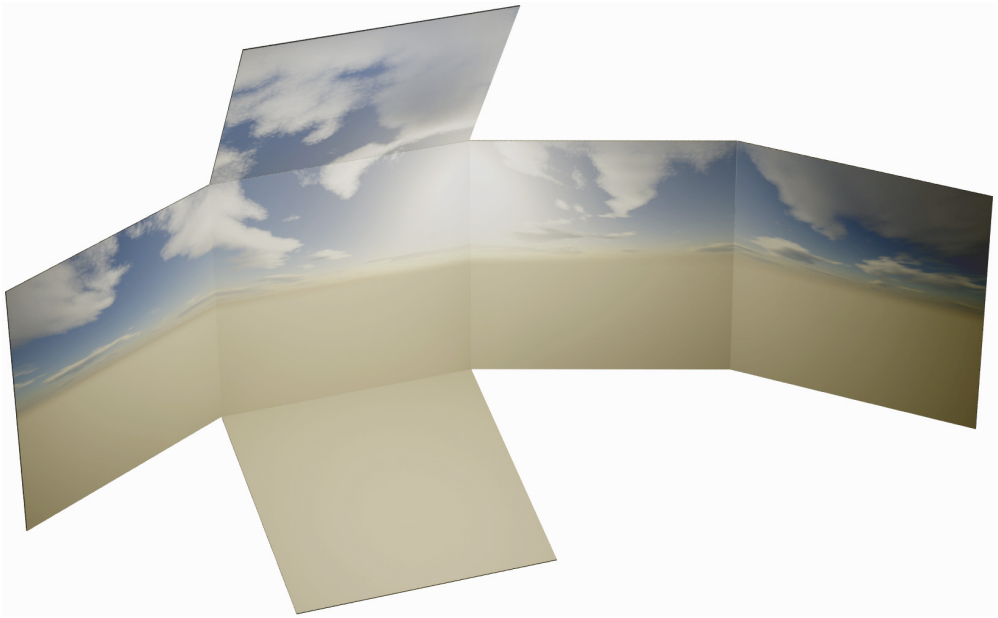


Abbildung 3.4: *Illustration zur Zusammensetzung einer Cube-Map, bestehend aus sechs einzelnen, in diesem Fall computer-generierten Texturen. Zum besseren Verständnis ist der Würfel geöffnet dargestellt.*

3.1.3 Paraboloide Projektion (Paraboloid-Mapping)

Heidrich & Seidel (1998) stellen eine Projektion vor, mit der der Himmel bis zum Horizont ohne Singularitäten in einer Textur abgebildet werden kann (Abbildung 3.3(b)). Die Texturkoordinaten u und v sind wie folgt definiert:

$$u = 0.5 + r_x m^{-1}, \tag{3.7}$$

$$v = 0.5 + r_y m^{-1}, \tag{3.8}$$

$$m = 2.0 + 2.0\, r_z. \tag{3.9}$$

Sie bietet eine ausreichend regelmäßige Detaildichte, und kann mithilfe spezieller Fischaugenobjektive direkt fotografiert werden. Zur vollständigen Abbildung der Umgebung können zwei Texturen der jeweils gegenüberliegenden Hemisphären verwendet werden (Dual-Paraboloid-Mapping). Diese Variante wird allerdings nicht von *osgHimmel* unterstützt, da nur die Anwendung mit Zenit-Ausrichtung, also als Himmel, angedacht ist.

3.1.4 Kubische Projektion (Cube-Mapping)

Die wohl wichtigste aller Projektionen zur Abbildung vollständiger, dreidimensionaler Umgebungen für die Bildsynthese ist die von Greene (1986) vorgestellte kubische Projektion. Sechs einzelne Texturen, je eine pro Würfelinnenseite, bieten eine, im Vergleich zu den zuvor erwähnten Projektionen, kaum wahrnehmbare Verzerrung (Abbildung 3.4). Der wichtigste Vorteil des Cube-Mappings ist die Grafikhardware-Unterstützung zum Speichern und Verarbeiten der Cube-Maps. Anstelle sechs einzelner Texturen, arbeitet man nur mit einem einzigen Texturobjekt. Der Zugriff auf beliebige Pixel der einzelnen Texturen erfolgt direkt über die Sampling-Richtung r an das Texturobjekt.

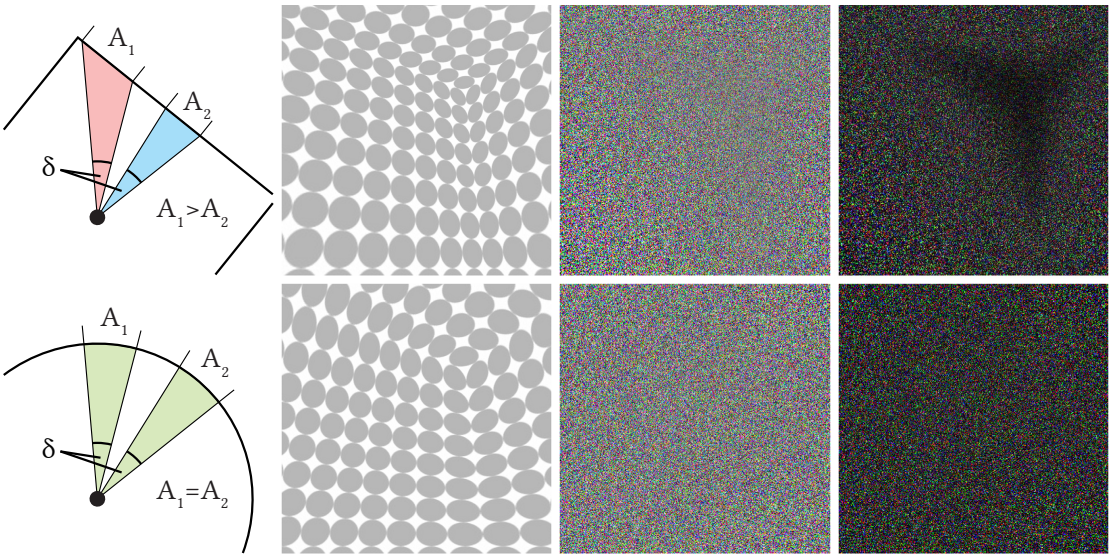


Abbildung 3.5: *Links: Vergleich der Samplingflächen unterschiedlicher Sampling-Richtungen. Mitte-Links: Textur mit Kreismuster zur räumlichen Verdeutlichung der Kanten und speziell der Ecke. Mitte-Rechts: Farbiges Gauß-Rauschen offenbart variierende Texeldichte. Rechts: selbe Abbildung wie zuvor, jedoch mit angepassten Intensitätslevel zur besseren Sichtbarkeit der Texeldichte. Die obere Reihe zeigt unverändertes Cube-Mapping mit starken Schwankungen in der Texeldichte. In der unteren Reihe wurde die Sampling-Richtung wie vorgeschlagen angepasst. Ergebnis: Keinerlei erkennbare Variationen in der Texeldichte.*

Aber auch hier gibt es zwei, zwar weniger auffällige, aber dennoch erwähnenswerte Probleme. Zum einen muss das Filtern über die Kanten hinweg individuell gelöst werden, da dies zur Zeit nicht von der Grafikhardware unterstützt wird. Isidoro & Mitchell (2005) schlagen eine entsprechende Filterung zur nahtlosen Darstellung vor. Zum anderen, ist aufgrund der variierenden Texeldichte die Wahrnehmung der Würfelgeometrie möglich (Abbildung 3.5). Die Texeldichte bezeichnet das Verhältnis zwischen der Anzahl an Texel der Cube-Map pro Pixel der finalen Darstellung. Schnelles Bewegen oder Rotieren innerhalb der Szene führt zur unbewussten Wahrnehmung der Distanzänderung zu den jeweiligen Würfelseiten. Die einzelnen Kanten und Ecken des Würfels sind folglich auszumachen und können störend wirken.

Interpretiert man die normalisierte Sampling-Richtung als Oberflächenpunkt einer Kugel und bildet diesen auf einen Würfel ab, erhält man ein Sampling mit gleichmäßiger Texeldichte. Dies ist die Umkehrung des Mappings eines Würfels auf eine Kugel, jedoch wesentlich komplizierter als dieses. Abschnitt A.2 zeigt die von *osgHimmel* im Fragment Shader angewandten Algorithmus. Um den zusätzlichen Aufwand im Fragment Shader zu vermeiden, kann man die Sampling-Richtung auch durch die Verwendung einer fein unterteilten 3D Geometrie, als Texturkoordinaten per Vertex auf der CPU vorberechnen. Als Nachteil der Korrektur sei erwähnt, dass das vorhandene Bildmaterial nicht mehr korrekt projiziert wird, und folglich neu vorberechnet werden müsste. Die leicht verzerrte Projektion stört aus unserer Sicht jedoch weniger, als eine ungleichmäßige Texeldichte bzw. erkennbare Würfelgeometrie.

3.2 Erweiterungen

osgHimmel bietet zur Verwendung texturbasierter Verfahren grundlegende Dynamik sowie Lösungen zu häufigen Problemen. Im folgenden werden zeit-bedingte Transitionen vorgestellt, mit denen beispielsweise dynamische Tag-Nacht Transitionen möglich sind. Mit Hilfe des Horizontbandes, können auf eine einzelne Hemisphäre beschränkte Texturen auch in Szenen genutzt werden, in denen potentiell der gesamte Himmel beobachtbar ist. Durch eine leichte Rotation der Hintergrundkulisse um den Zenit wird Wind und damit mehr Dynamik suggeriert. Des weiteren wird das Konzept der maskierten Sonne kurz erläutert und Möglichkeiten zur Parallaxe diskutiert.

3.2.1 Zeit-bedingte Transitionen

Um texturbasierte Hintergrundkulissen mit mehr Dynamik zu versehen, können zeitlich bedingte Transitionen zwischen mehreren Texturen helfen. So können in virtuellen 3D Welten mit Zeitbezug, über Sonnenaufgänge, -Untergänge, Wolken und Wetterveränderungen, Nachthimmel, und schließlich dynamische Tag-Nacht-Zyklen auf einfache Weise erzeugt werden. Die Texturen sollten dabei möglichst aufeinander abgestimmt sein, und je mehr Texturen verwendet werden, desto feiner können die Übergänge ablaufen.

Beim Zuweisen einer Textur an den Himmel, wird zusätzlich ein Zeitpunkt, eine sogenannte Stützstelle τ, gefordert, ab welcher die Textur in Erscheinung treten soll. Es können also beliebige Zeitintervalle für die Hintergrundkulisse angegeben werden. Die Stützstellen werden verwendet, um wahlweise sanfte oder abrupte Transitionen zu Texturen der jeweils darauffolgenden Stützstellen zu erzeugen. Alle Stützstellen beziehen sich auf einen geschlossenen Zyklus mit $\tau_i \in [0, 1]$ und $i \in \mathbb{N}$. Eine Transitionsdauer t gibt die relative Dauer des Übergangs zwischen benachbarten Stützstellen an. Ferner, wird der Zeitbezug über die Anzahl realer Sekunden pro Zyklus hergestellt. Inkorrekte Transitionen lassen sich vermeiden, indem t kleiner oder kleiner gleich der geringsten Distanz zwischen zwei Stützstellen gewählt wird. Falls dies nicht gegeben ist, passt *osgHimmel* t mit $t = \min |\tau_i - \tau_j|$ und $i, j \in \mathbb{N} \wedge i \neq j$ entsprechend an. Abschließend muss festgelegt werden, welches Verhalten an einer Stützstelle zu erwarten ist. Soll die dort geforderte Textur bereits vollständig sichtbar sein, sich mitten im Übergang befinden, oder den Übergang einleiten. Für die Umsetzung im System haben wir uns für die zweite Variante entschieden. Die anderen Varianten können durch Einbeziehen eines zeitlichen Versatzes, entsprechend einfach realisiert werden. Ein Versatz von $-0.5\,t$ verursacht dabei das zuerst, $+0.5\,t$ das zuletzt genannte Verhalten.

Ein Beispiel mit zwei Stützstellen (Abbildung 3.6): Die erste Textur wird für den Zeitpunkt $\tau_0 = 0.0$ angegeben. Die zweite mit $\tau_1 = 0.5$. Zum Zeitpunkt τ_0 und τ_1 ist das Ergebnis mit gleich gewichteter Verblendung beider Texturen identisch. Bei einer relativ langen und damit sehr sanften Transitionsdauer von $t = 0.4$, ist die erste Textur für einen Zeitraum von 0.1 zwischen 0.2 und 0.3, die Zweite für einen ebenso langen Zeitraum zwischen 0.7 und 0.8 exklusiv sichtbar. Bei einer Zyklusdauer von beispielsweise 60 Sekunden, dauern die exklusiven Phasen jeweils für 6 Sekunden an. Beim Zeitpunkt 1.0 wird nathlos zum Zeitpunkt 0.0 gesprungen, und der Zyklus läuft somit fortlaufend ab.

Die Auswahl der Texturen für die Verblendung erfolgt zeitabhängig auf der CPU. Dabei wird zwischen Vorder- und Hintergrundtextur unterschieden, und die Transparenz der

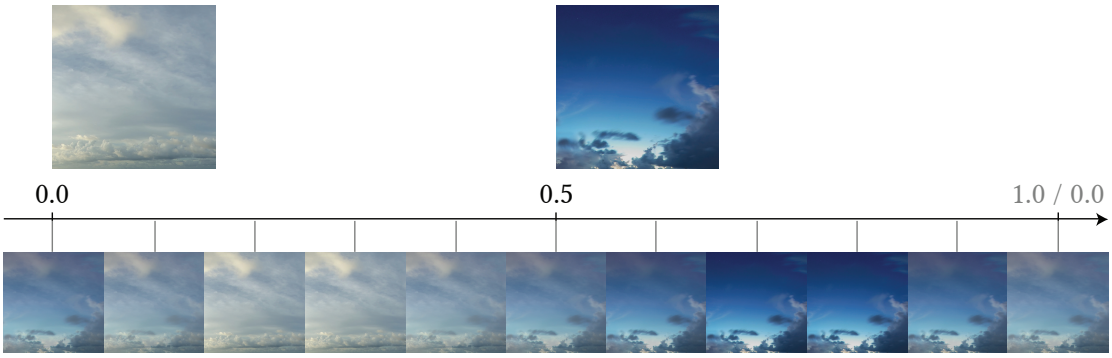


Abbildung 3.6: *Illustration zum Ablauf einer zeit-bedingten Transition: Für den Zeitpunkt* 0.0 *sowie für* 0.5 *wird jeweils eine Textur angegeben. Unter dem Zeitstrahl sind die verblendeten Ergebnisse für eine Transitionsdauer von* $t = 0.4$ *dargestellt.*

Vordergrundtextur bestimmt. Für die zeitliche Abhängigkeit der Transparenz zu den zugehörigen Stützstellen wählten wir eine lineare Abhängigkeit. Andere Interpolationen sind in diesem Fall nicht empfehlenswert. Sie führen meist zu einer ungleichmäßigeren, abschnittsweise hektischeren und damit dem Betrachter überhaupt erst auffälligen Transition, was nicht Ziel der Hintergrundkulisse ist. Zum Zeitpunkt der Fertigstellung dieser Arbeit, sind Transitionen ausschließlich innerhalb der selben Projektion möglich. Dies vereinfacht die jeweiligen Shader erheblich und reduziert die darin benötigten konditionellen Ausführungszweige. Mit einer modularen Zusammensetzung, für jede Transition individueller Shader sowie einem entsprechenden, dynamischen Wechsel derer, wäre auch eine Transition zwischen beliebigen Projektionen realisierbar. Wir empfehlen in diesem Fall stattdessen, alle Texturen für eine Projektion vorzubereiten (es lassen sich alle Projektionen ineinander überführen).

Da alle Hintergrundkulissen von *osgHimmel* einer internen, eigenen Zeit folgen, also nicht zwingend an die Zeit ihrer virtuellen Welt gekoppelt sind, ergeben sich in diesem Fall weitere Anwendungsmöglichkeiten. So können unter geringen Einschränkungen, entsprechender Vorbereitung der Stützstellen und Beeinflussung der Zeit dynamische Wetterübergänge zu beliebigen, unvorhersehbaren Zeitpunkten realisiert werden. Abschließend sein angemerkt, dass die Transitionen ideal zwischen Texturen ohne oder mit gleichbleibender Landschaft, unter leichten Änderung der Atmosphärenfarbe und beliebig starken Änderungen der Bewölkung funktionieren. Das Verschwinden ganzen Landschaften, oder Aufkommen mehrere Sonnen während der Transition wirkt sich negativ auf die Glaubwürdigkeit der Hintergrundkulisse aus.

3.2.2 Horizontband

Einige der vorgestellten Projektionsarten bilden nur die obere Hemisphäre der Umgebung ab (Paraboloid-Mapping und Polar-Mapping im half-mode). So entsteht entweder eine harte Kante zur gewählten Hintergrundfarbe (clear color) oder die Textur wird entsprechend der Texture-Wrapping Einstellungen angezeigt. Sofern der Betrachter nie über den Horizont hinaus schauen kann, fällt dies nicht auf und kann vernachlässigt werden. Ist dies nicht der Fall, so ermöglicht *osgHimmel* die obere Hemisphäre mit der Unteren nahtlos zu verbinden (Abbildung 3.7). Dies geschieht durch Überblendung der Schnittstelle mit einem Farbverlauf,

Abbildung 3.7: *Polar-Mapping ohne (rechts) und mit (links) Horizontband, sowie Annotation aller zur genauen Spezifikation verfügbaren Parameter.*

dem Horizontband. Es wird durch Gesamtbreite, innere Breite, vertikalen Versatz, sowie zwei Farben spezifiziert. Die eine dient für das Band selbst (Bandfarbe), die andere zur Farbgebung unterhalb des Horizonts (Bodenfarbe). Die Gesamtbreite (scale) spezifiziert die Breite des Bandes und wird als vertikaler, normaliserter Öffnungswinkel angegeben. 1.0 steht dabei für 180Grad, und erzeugt ein Horizontband über die gesamte Hintergrundkulisse. Mit der inneren Breite (width) lässt sich die Kernregion, welche vollständig mit der Bandfarbe gefärbt ist, relativ zur Gesamtbreite einstellen. Schließlich wird die Bandfarbe vom inneren Rand bis zum äußeren Rand, nach oben mit der oberen Hemisphäre und nach unten mit der Bodenfarbe, nahtlos verblendet. Zusätzlich kann mit der Angabe eines vertikalen Versatzes (offset) das gesamte Horizontband der vertikalen Achse entlang verschoben werden.

In einem Vorberechnungsschritt, könnte man zu einer gegebenen Hintergrundkulisse die Farben des Bandes und des Bodens extrapolieren. Denkbar ist auch die Bodenfarbe mit einer kachelbaren (tileable) Textur zu ersetzen und somit eine texturierte, mit der Entfernung auslaufende Bodenfläche zu erhalten. Die Größe der Kachel bzw. die Skalierung deren Inhalts beeinflusst dabei die wahrgenommene Höhe des Betrachters über dem Boden. Zur sauberen Darstellung würde jedoch zusätzliche anisotrope Filterung im Fragment Shader benötigt.

3.2.3 Rotation um den Zenit

Zuletzt inspiriert durch die Hintergrundkulissen der Videospiel-Reihe Assassin's Creed von Ubisoft[1], vermag die sanfte Rotation einer Hintergrundkulisse um den Zenit die Glaubwürdigkeit und Dynamik der gesamten Szene zu steigern. *osgHimmel* erlaubt eine Rotation um den Zenit, welche mit einer vom Himmel unabhängigen Geschwindigkeit erfolgt. Sie besteht somit auch bei schnellen Textur Transitionen (z.B. bei Zeitraffern). Die Geschwindigkeit wird in Sekunden per vollständiger Rotation um den Zenit angegeben und die Richtung von Nord über Osten nach Süden, oder über Westen nach Süden kann gewählt werden. Die Angabe erfolgt über Himmelsrichtungen, um Verwirrungen zu vermeiden.

Als weitere Möglichkeit könnte man, unter Angabe einer Translationsrichtung, die Rotationsgeschwindigkeit in Abhängigkeit zum Winkel zwischen der Blickrichtung und Translationsrichtung anpassen: Parallel zur Blickrichtung keine Rotation, orthogonal zur Blickrichtung maximale Rotation mit entsprechender Rotationsrichtung. Bei kleinen Sichtfeldern (field of

[1] http://www.ubi.com/DE/Games/Info.aspx?pId=5918

Abbildung 3.8: *Links ein Ausschnitt einer Hintergrundkulisse ohne zusätzlicher Sonne. Rechts der gleiche Ausschnitt, diesmal mit an den Wolken leicht maskierter Sonne.*

view), könnte dies den Eindruck sich bewegender Wolken verstärken. Der Effekt im Ganzen, ob mit oder ohne Abhängigkeit zur Blickrichtung, funktioniert jedoch nur bei sehr unauffälligen Rotationsgeschwindigkeiten.

3.2.4 Weitere Ideen und Ergänzungen

Maskierte Sonne

Bei dieser Erweiterung wird zusätzlich zur Textur eine dynamische Sonne eingeblendet (Abbildung 3.8). Dieses Verfahren findet unter Anderen in der X-Ray Engine von GSC Game World[2], hier auch in Verbindung mit dynamischen Textur Transitionen, Verwendung. Die Bewegung der Sonne ist über eine beliebige Ellipse in Abhängigkeit zur Zeit möglich. Die Sonne kann die Texturen entweder vollständig überdecken, oder mithilfe einer im Alphakanal der Texturen kodierten Maske erfolgen. Die Sichtbarkeit bei Wolken oder sonstigen Objekten sollte entsprechend eingeschränkt sein und es sollte auch keine weitere Sonne abbildet sein. Das Ergebnis hängt stark von der Qualität der Maske ab, welche sich nur sehr schwer aus vorhandenen Bildern ableiten lassen, und manuell erzeugt werden müssen.

Mehrere Ebenen zur Parallaxe

Eine weitere, bisher allerdings nicht umgesetzte Idee, bezieht sich auf das Konzept der 3D Skybox, welche in der Source Engine[3] von Valve vorgestellt wurde. Dabei wird die Skybox mit zusätzlicher Geometrie angereichter, welche sich abhängig von der Position des Betrachters bewegt, und somit Parallaxe erzeugt. Anstelle von Geometrie, würden bereits weitere, texturierte Schichten, deren Position je nach zugewiesener Entfernung mit der des Bobachters korrelieren, diesen Effekt erzielen. So könnte vor einer geschlossenen, unendlich weit entfernten

[2] http://www.stalkerzone.de/soc_engine.php
[3] http://source.valvesoftware.com/

Schicht, weitere Wolkenschichten oder Berge in endlichen Entfernungen zur Erzeugung der Parallaxe dienen. Dieser Effekt kann mit geringem Aufwand in *osgHimmel* integriert werden.

Kapitel 4

Himmel-Rendering mittels physikalischer Modelle

In diesem Kapitel werden verwandte und neue Verfahren zur Simulation verschiedener atmosphärischer Phänomene zur Darstellung eines dynamischen Himmels vorgestellt. Zu den Phänomenen zählen: Sonne, Mond, Mondfinsternis, Sterne, Sternhaufen, Milchstraße, Atmosphärische Streuung, und dynamische Wolkenschichten. Für die Simulation des Mondes und Mondfinsternissen, sowie der Sterne bzw. Sternhaufen und Milchstraße, stellen wir jeweils neue Verfahren vor (Müller et al., 2012). Bezüglich der Atmosphäre, welche aufgrund der Streuung nach Rayleigh- und Mie-Theorie auch immer die Sonne abbildet, werden vorhandene Verfahren verwendet und erläutert (Abbildung 4.1). Zur Darstellung der Wolken stellen wir eine Abwandlung eines einfachen Verfahrens vor. Da im Rahmen dieser Arbeit keine umfassende Auseinandersetzung mit dem Thema der Bildsynthese von Wolken möglich ist, soll unser Verfahren nur einer stellvertretenden Umsetzung dienen, und hinreichend glaubwürdige Darstellungen schaffen. Im letzten Abschnitt werden die Ideen und Probleme der Komposition aller Phänomene zu einer konsistenten Gesamtdarstellung beschrieben.

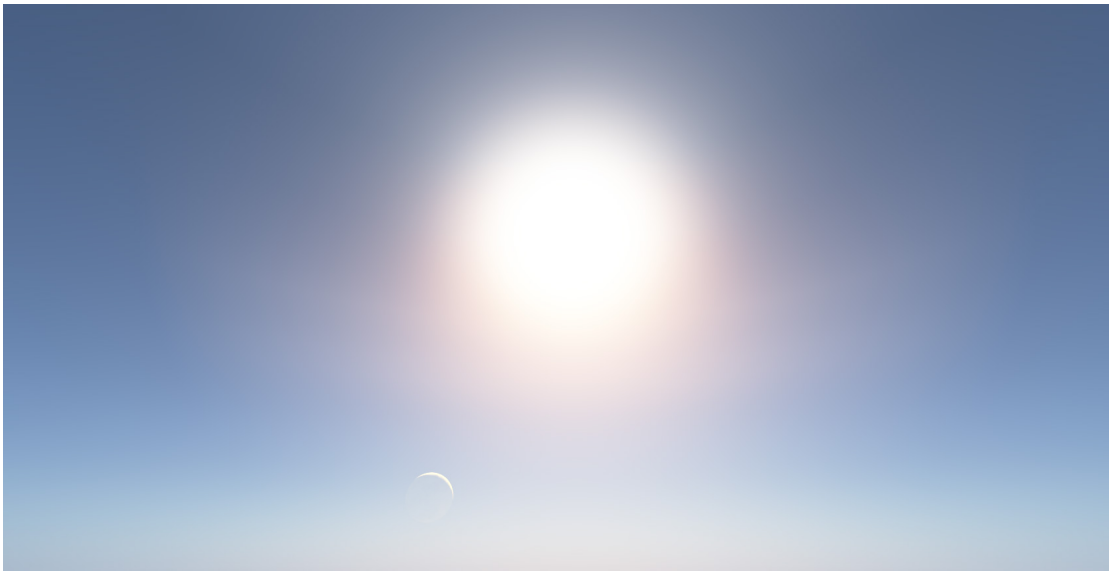


Abbildung 4.1: *Beispiel einer simulierten Hintergrundkulisse ohne Wolken, nach Sonnenaufgang. Mond und Sonne sind 10 fach vergrößert dargestellt.*

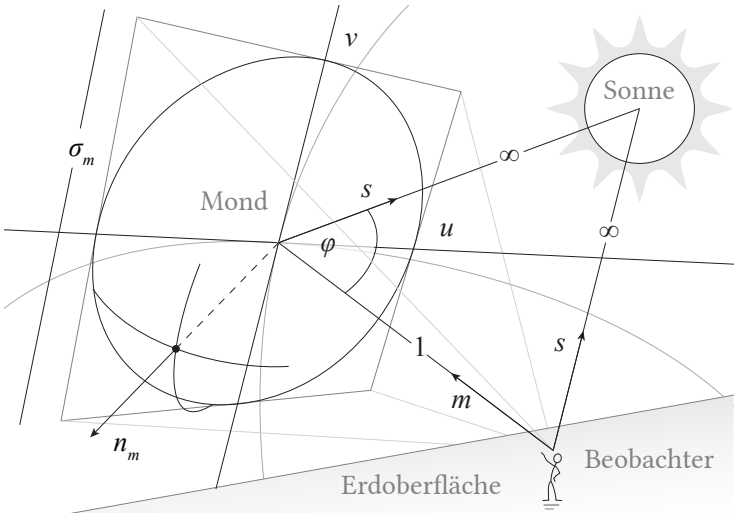


Abbildung 4.2: *Die virtuelle Mondkugel. Für Beobachter und Mond wird die selbe Richtung s zur Sonne angenommen. Die Texturkoordinaten u und v auf der Tangentialebene werden zur Ableitung des Mondfragments mit der Normalen n_m verwendet.*

4.1 Rendering des Mondes

Bildsynthese die auf realistisch wirkende Himmel abzielt, sollte stets die Darstellung des Mondes beinhalten, da dieser die zweit stärkste, extraterrestrische Lichtquelle ist, die auf der Erde wahrgenommen werden kann. Der Mond, sofern sichtbar, ist wesentlich für die Beleuchtung bei Nacht verantwortlich und kann ohne Problem von der Erde aus beobachtet werden. Er emittiert keine eigenes Licht, stattdessen reflektiert er ein Teil des einfallendes Sonnenlicht in Richtung Erde. Trotz einer mittleren scheinbaren Helligkeit (genauere Definition in Abschnitt 4.3) von -12.74mag, dem vierhundert Tausendstel der scheinbaren Helligkeit der Sonne, ist der Mond auch tagsüber klar erkennbar (Williams, 2010). Mit unserem Modell wollen wir alle augenscheinlichen Phänomene des Mondes nachahmen. Da die Darstellung in den meisten Anwendungsszenarien nur innerhalb eines Bruchteils der verfügbaren Darstellungsfläche erfolgt, muss die Glaubwürdigkeit und astrophysikalische Korrektheit der Darstellung am damit verbundenen Rechenaufwand bemessen werden. Das Modell soll vorrangig den beschriebenen Anwendungsszenarien genügen.

Der Mond wird vollständig, als sicht-orientiertes Billboard (McReynolds & Blythe, 2005), auf der GPU modelliert. Genauer: Es wird ein Quadrat auf die Tangentialebene einer Einheitskugel, projiziert und nach oben ausgerichtet. Die Einheitskugel ist am Beobachter zentriert und die Projektion erfolgt in Richtung des Mondes. Auf dem Quadrat wird schließlich eine virtuelle Mondkugel im Bildraum simuliert, sodass keine zusätzliche Geometrie benötigt wird und eine perfekte Kugel abgebildet werden kann. Dieser Ansatz wird häufig zur Bildsynthese von Flüssigkeiten oder Punktwolken verwendet. Die Oberfläche der Mondkugel wird aus den Schnittpunkten der Sichtstrahlen mit dem Quadrat extrapoliert. Dabei hilft die hardware-unterstützte Interpolation der den Eckpunkten des Quadrats zuvor zugewiesenen Texturkoordinaten u und v, welche dem Fragment Shader die selenozentrischen, also auf den Mondmittelpunkt bezogenen, normierten Koordinaten liefert. Mit diesen kann anschließend die selenozentrischen z-Komponente $z^2 = 1 - u^2 - v^2$ ermittelt werden, und damit den

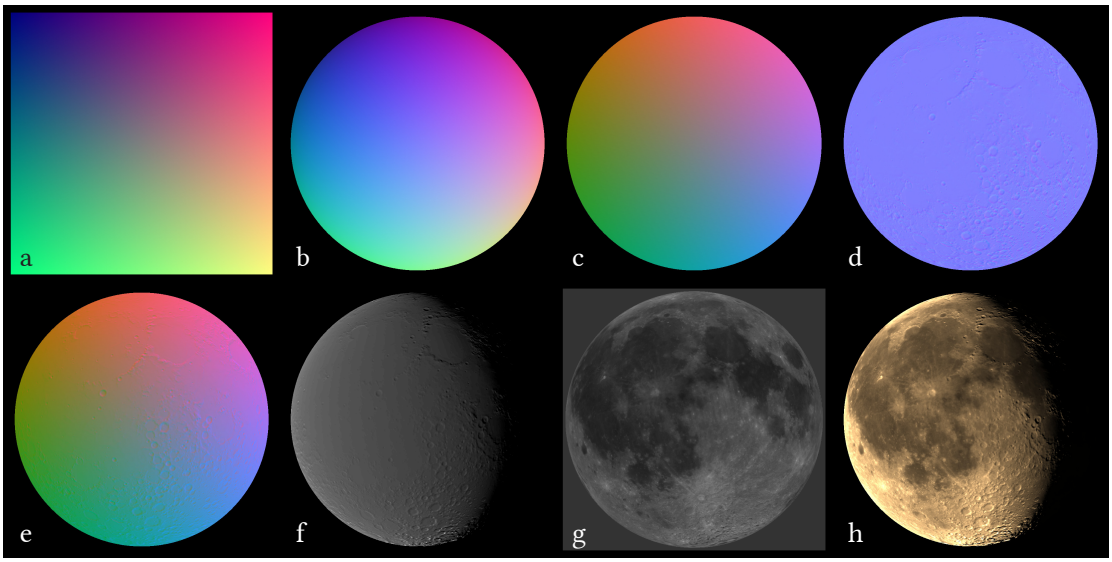


Abbildung 4.3: *Vom Billboard zur virtuellen Mondkugel: a) Billboard mit u und v Koordinaten, b) z-Komponente und verworfene Fragmente ergeben die virtuelle Kugel, c) Mondfragmente sind in horizontale Koordinaten transformiert, d) und g) Normalen bzw. Albedo wurden aus der RGBA Cube-Map gelesen, e) Normalen der Mondfragmente werden mit denen der Normalen-Map nach Bedarf vermischt, f) Hapke-Lommel-Seeliger BRDF wird auf Mondfragmente angewandt, h) Albedo, Schattierung, Erdschein, und Farbkoeffizienten werden zur Darstellung des Mondes zusammengeführt.*

Schnittpunkt des Sichtstrahls mit der virtuellen Mondoberfläche (im Folgenden *Mondfragment* genannt). Jedes Mondfragment ist in seiner Ausrichtung durch die Normale $n_m = (u, v, z)$ und in seiner Position definiert. Fragmente des Quadrats, die außerhalb der Mondkugel liegen, werden verworfen. Zur Vermeidung von Aliasing Artefakten entlang der Kante der sichtbaren Mondscheibe, wird eine sichtfeld- und auflösungs-invariante Kantenglättung verwendet. Unter Berücksichtigung der genauen Entfernung zum Mond kann die Größe des Billboards entsprechend angepasst werden. Das grundlegende Mond-Modell ist in Abbildung 4.2 aufgeführt. Der grobe, schrittweise Ablauf im Fragment Shader ist in Abbildung 4.3 illustriert.

4.1.1 Scheinbare Position, Größe, und Ausrichtung des Mondes

Zur Bestimmung der scheinbaren Position und damit der Projektionsrichtung m des Mondes, werden dessen ekliptische Koordinaten ermittelt, um die Brechung innerhalb der Atmosphäre (*Refraktion*) korrigiert und in horizontale Koordinaten transformiert (Meeus, 1994). Refraktion beeinflusst die wahre Höhe (true altitude) um eine Verschiebung zum Zenit, also eine scheinbare Anhebung vor allem horizontnaher Objekte. Auf Meereshöhe beträgt die Verschiebung, beim Blick in Richtung Horizont, etwa 36 Bogenminuten, was in etwa der scheinbaren Größe des Mondes selbst entspricht. Azimut und Höhe werden schließlich in euklidische Koordinaten transformiert, und normalisiert an die GPU übertragen.

Bei der Refraktionskorrektur vernachlässigen wir die Höhe des Betrachters, da der Einfluss in den meisten Anwendungsfällen kaum wahrnehmbar wäre. Befindet sich der Betrachter beständig in großen Höhen, und schaut nicht unter den Horizont, kann auf die Korrektur verzichtet werden. Für eine sehr genaue Darstellung, speziell wenn der Mond den Horizont

über- und unterläuft, müsste die Refraktion pro Fragment berücksichtigt werden, um Abweichungen von der Kreisform entsprechend der Realität abzubilden. Dazu kann das Mond-Billboard in zwei oder mehrere vertikale Streifen unterteilt werden, und deren refraktionskorrigierte Höhenwinkel ermittelt werden. Anschließend wird im Fragment Shader die v-Koordinate um die Differenz zur linear interpolierten Streifenhöhe korrigiert, und erhält somit einen deformierten Mond. Mit Hilfe einer zweidimensionalen Look-Up Textur könnten die Höhenwinkel der einzelnen Fragmente in Abhängigkeit zum Höhenwinkel der scheinbaren Mondmitte abgetragen werden, und im Fragment Shader eine Deformation pro Fragment erfolgen. Wir haben diese Idee evaluiert (Abbildung A.2), aber zur Zeit noch nicht mit *osgHimmel* zur Verfügung gestellt. Generell ist zu beachten, dass Refraktionskorrektur ausschließlich bei horizontaler Positionierung angewandt wird wird. Alle anderen vom Modell ableitbaren Phänomene wie Schattierung und Mondfinsternis, ergeben sich außerhalb der Atmosphäre.

Entfernung und scheinbare Größe des Mondes

Die Entfernung d der Erde zum Mond wird zwischen deren Mittelpunkten angegeben und variiert innerhalb Perigäum (Erdnähe) und Apogäum (Erdferne) zwischen ungefähr $d_{perigee} = 363\,300$ km und $d_{apogee} = 405\,500$ km. Die genaue geozentrische Position des Betrachters sowie der genaue Erdradius an dieser Stelle werden dabei nicht berücksichtigt. Diese Vereinfachung führt zu einer maximalen Abweichung des scheinbaren Durchmessers des Mondes von 1%. Ist die Entfernung bekannt, kann der scheinbare Durchmesser des Mondes δ_m, welcher zwischen 0.49Grad und 0.56Grad variiert, und dessen sichtbare Größe auf der Erde beschreibt, wie folgt ermittelt werden:

$$\delta_m(d) = 2 \arctan\left(\frac{r_m}{d}\right). \tag{4.1}$$

Dabei ist r_m der mittlere Mondradius mit $r_m = 1\,737.1$km. Der Durchmesser der virtuellen Mondkugel σ_m, also der Seitenlänge des Billboards, wird wie folgt ausgedrückt:

$$\sigma_m(\delta_m) = 2 \tan\left(\frac{c_d\ \delta_m}{2}\right), \tag{4.2}$$

mit c_d als Skalierungsfaktor. Nutzt man korrekte, scheinbare Größen in normalen Sichtfeldern (field of view), wirkt der Mond subjektiv zu klein, was je nach Empfinden eine künstliche Vergrößerung notwendig macht. Wir vermuten, dies liegt zum einen an der geringen Detaildichte des Mondes auf Bildschirmen mit geringer Pixeldichte. Wenn diese der individuellen (vom Sehvermögen abhängigen) Erfahrung und Erwartung entspricht, scheint der Mond glaubwürdiger. Zum anderen, sind wir durch Filme, Fotos, Kunstwerke, etc., einer starken Prägung überzogener Nahaufnahmen des Mondes (Entsprechendes gilt für die Sonne) ausgeliefert, welche den Realitätsabgleich einer Visualisierung zusätzlich verzerrt. Werte zwischen 2 und 3 für c_d führen bei geringerer Pixeldichte zu weniger bewussten Irritationen.

Ausrichtung des Mondes

Der Mond befindet sich mit der Erde als Zentralplaneten in einer gebundenen Rotation. Die Eigenrotation des kleineren Mond hat sich durch die starken Gravitationskräfte der Erde an seine Umlaufzeit angepasst, sodass er sich während eines Umlaufs einmal um die eigene Achse dreht. Einem Betrachter auf der Erde ist folglich jederzeit die gleiche Mondhälfte zugewandt.

Für eine genaue Ausrichtung müssen die oszillierenden Bewegungen, bekannt als *Librationen* oder Taumelbewegung, sowie betrachter-abhängige, parallaktische Winkel mit einbezogen werden. Librationen entstehen (neben weiteren Ursachen), weil entgegen der konstanten Eigenrotation des Mondes, weder seine Umlaufbahn exakt kreisförmig, noch seine Winkelgeschwindigkeit konstant sind. Es wird dabei zwischen optischer Libration und physischer Libration unterschieden. Aufgrund der unbemerkbar leichten Auswirkung der physischen Libration (maximal 0.04Grad) sowie der parallaktischen (täglichen) Libration (etwa 1Grad), werden diese in unserem Modell vernachlässigt. Die verbleibenden Librationen in Länge und Breite verursachen eine zusätzliche Sichtbarkeit der Oberfläche von 9% über die Zeit. Sie werden in selenographischer Breite und Länge, also mit Bezug auf das Zentrum der scheinbaren Mondscheibe entsprechend geografischer Breite und Länge auf der Erde, angenähert (Meeus, 1994). Libration in Breite b, ist der Winkel zwischen dem Nullmeridian der scheinbaren Mondscheibe und dessen Rotationsachse, und enthüllt abwechselnd Nord- und Südpol. Libration in Länge l, ist der Winkel der seitlichen Rotation um das Lot der Mondbahn.

Zur Abbildung der Librationen verwenden wir eine selenozentrische Rotations-Matrix R_m (Jensen et al., 2001a), definiert als:

$$R_m = R_x(b)\, R_y(l)\, R_z(p - a). \tag{4.3}$$

R_x, R_y, und R_z bezeichnen gegen den Uhrzeigersinn und um die Hauptachsen laufenden Rotations-Matrizen, a ist der Positionswinkel und p der parallaktische Winkel. Zusammen beschreiben sie die Ausrichtung um die Achse zwischen Mond und dem, sich auf der rotierenden Erde befindlichen Betrachter – dem Lot der Tangentialebene des Billboards.

4.1.2 Farbe und Schattierung des Mondes

Die Beleuchtung der Mondkugel hängt hauptsächlich von der perspektivischen Lageänderung der Tag-Nacht-Grenze (Terminator) ab und ist durch den beleuchteten Anteil der sichtbaren Scheibe gegeben. Um diesen Anteil zu erhalten, wird normalerweise der Mondphasen Winkel (*Phasenwinkel*) φ berechnet. Dieser gibt die selenozentrische Elongation, also dem vom Mondmittelpunkt aus gesehenen Winkelabstand, der Erde zur Sonne an. Mit ihm kann der Positionswinkel der beleuchteten Seite berechnet werden.

Stattdessen, ist der Phasenwinkel in unserem Modell bereits als Winkel zwischen den euklidischen Richtungsvektoren s und m gegeben. Interpretiert man φ als Winkel zwischen dem einfallenden und reflektierten Licht auf der Mondoberfläche, ergibt sich die korrekte Beleuchtung und damit auch die korrekte Tag-Nacht-Grenze (Jensen et al., 2001a). Durch die Vereinfachung der Sonnenposition s als direktionale Lichtquelle, von der Erde ausgehend, führen wir einen nicht wahrnehmbaren maximalen Fehler des Phasenwinkels von 9 Bogenminuten ein. Annäherung der Schattierung, der Albedo (dem Rückstrahlvermögen), und des Erdscheins werden als Funktionen von φ ausgedrückt. Die resultierende Oberflächenfarbe I_m ergibt sich aus dem Anteil an reflektierten Sonnen- und Erdlicht:

$$I_m(\varphi) = k_\lambda\, a\, \left(F(\theta_i, \theta_r, \varphi) + \beta_e\, E_{em}(\varphi)\right), \tag{4.4}$$

mit a als Oberflächen Albedo, dem Erdschein $\beta_e\ E_{em}$, F als der Hapke-Lommel-Seeliger BRDF (Bidirektionale Reflektanzverteilungsfunktion), einer Annäherung der realen Mond Reflexion

Abbildung 4.4: *Verschiedene Fotos des Mondes (oben) im Vergleich mit unseren Ergebnissen bei entsprechenden Tag und Ort (unten).*

(Hapke, 1966), welche berechnet werden kann durch:

$$F(\theta_i, \theta_r, \varphi) = \frac{2}{3\pi} \cdot \frac{B(\varphi, g)\, S(\varphi)}{1 + \cos\theta_r / \cos\theta_i}. \tag{4.5}$$

Die Retrodirektive Funktion B ist gegeben durch:

$$B(\varphi, g) = \begin{cases} 2 - \frac{\tan\varphi}{2g}\left(1 - e^{-g/\tan\varphi}\right)\left(3 - e^{-g/\tan\varphi}\right) & \text{falls } \varphi < \frac{\pi}{2} \\ 1 & \text{sonst} \end{cases}, \tag{4.6}$$

wobei g die Oberflächendichte beschreibt, und die Schärfe der Tag-Nacht-Grenze bei Neu- und Vollmond beeinflusst (üblicherweise wird $g = 0.6$ verwendet). Für die Streuung S wählen wir anstelle der von Jensen et al. (2001b) verwendeten Approximation (Hapke, 1963), die durch Hapke (1966) verbesserte Approximation, welche dein Einfluss des Oppositionseffektes berücksichtigt:

$$S(\varphi) = \frac{1}{2}\left(1 - \sin\frac{\varphi}{2}\tan\frac{\varphi}{2}\ln\left(\cot\frac{\varphi}{4}\right) + t\,(1 - \cos\varphi)^2\right), \tag{4.7}$$

$$\text{bzw. } S(\varphi) = \frac{1}{2}\left(1 - \sin\frac{\varphi}{2}\tan\frac{\varphi}{2}\log\left(\tan\left(\frac{\pi}{2} - \frac{\varphi}{4}\right)\right) + t\,(1 - \cos\varphi)^2\right). \tag{4.8}$$

Der Faktor t kontrolliert die Stärke der Vorwärtsstreuung und wird mit $t = 0.1$ angegeben. Die Oberfläche des Mondes hat eine geringe Albedo und zeigt eine rückstreuende Reflexion. Letztere sorgt zusammen mit dem Oppositionseffekt dafür, dass der Mond bei Vollmond wesentlich heller erscheint (Hapke, 1971). θ_r beschreibt den Winkel der Normale n_m zum reflektierten Licht, θ_i den zwischen n_m und einfallenden Licht. Dazu wird die Normale des Mondfragments zuvor in das benötigte horizontale Koordinatensystem überführt. Dies gelingt durch Multiplikation der Normale mit der Orthonormal Matrix des Billboards. k_λ wird benutzt, um die resultierende Farbe, wie z.B. der mond-typischen, rötlichen Färbung, und Intensität der Mondoberfläche zu gestalten.

Erdschein

Das bläuliche, schwache Licht der unbeleuchteten Mondseite, wird als Erdschein (oder Erdlicht) bezeichnet, und beschreibt die Reflexion des von der Erde emittierten Lichts. Seine Intensität E_{em} hängt vom Phasenwinkel φ ab. Je weniger reflektiertes Sonnenlicht uns vom Mond erreicht, desto stärker würde ein uns gegenüber liegender Beobachter auf dem Mond die Erde beleuchtet sehen (entsprechend der nicht sichtbaren, stärker zur Sonne gewandten Rückseite des Mondes). Der Erdschein ist folglich bei Neumond am stärksten. Er lässt sich wie von van de Hulst (1980) vorgeschlagen annähern:

$$E_{em}(\varphi) = \frac{0.19}{2}\left[1 - \sin\left(\frac{\pi-\varphi}{2}\right)\tan\left(\frac{\pi-\varphi}{2}\right)\ln\left(\cot\left(\frac{\pi-\varphi}{4}\right)\right)\right]. \tag{4.9}$$

Für unsere Anwendungsszenarien scheint diese Annäherung jedoch unverhältnismäßig genau, da zusätzliche Färbung, Intensitäts-Gestaltung sowie die finale Verblendung mit der Atmosphäre das Ergebnis stark beeinflussen. E_{em} lässt sich vereinfacht mit der folgenden Approximation abbilden:

$$E_{em}(\varphi) = -0.0061\,\varphi^3 + 0.0289\,\varphi^2 - 0.0105\,\sin(\varphi). \tag{4.10}$$

Die maximale Differenz zu Gleichung (4.9) beträgt 3%. Mit einem Farbkoeffizienten β_e wird die Farbe des Erdscheins wie gewünscht gestaltet und eine bläuliche Färbung hervorrufen (wir verwenden $\beta_e = (0.88, 0.96, 1.00)$). E_{em} wird pro Frame auf der CPU berechnet und an den Fragment Shader übergeben. Abbildung 4.4 zeigt im letzten Fall einen deutlich erkennbaren Erdschein (weitere Beispiele: Abbildung 4.1 sowie Abbildung 4.9).

Albedo und Oberflächenstruktur des Mondes

Für die Gestaltung der Mondoberfläche wird das in Abschnitt 3.1.4 erwähnte Cube-Mapping verwendet. Albedo und Oberflächennormalen können aus existierenden Polar-Maps (Celestia, 2011) mithilfe des HDR-Shop[1] transformiert und in einer RGBA Cube-Map kodiert werden (Abbildung 4.3).

Für die Albedo werden die Daten des Clementine Projekts verwendet (NASA, 1994), welche eigentlich nur teilweise als Albedo verwendet werden können: Die Mondoberfläche wurde so aufgenommen, dass die Sonne stets Nahe des selben Längengrads der Kamera war. Dadurch sind zu den Polen hin Schatten erkennbar, welche den Aufnahmen eine unerwünschte, statische Oberflächenstruktur geben. Aggregiertes, zu verschiedenen Vollmonden und an unterschiedlichen Orten der Erde aufgenommenes Bildmaterial wäre für einen Albedodatensatz besser geeignet. Während der Erstellung dieser Arbeit sind wir auf kein derartiges Material gestoßen. Sofern Librationen ignoriert werden, kann die Albedo auch einem projizierten, von der Erde aus aufgenommenen Foto des Mondes bei Vollmond entnommen werden.

Die Darstellung leichter Unregelmäßigkeiten in der Oberfläche sind wichtig im Bereich der Tag-Nacht-Grenze, aber selbst hier, sind sie bei realistischen scheinbaren Mondgrößen und aktuellen Ausgabegeräten kaum erkennbar. Dennoch werden sie je nach gewünschten Strukturintensität, linear mit den Normalen der Mondfragmente verblendet. Die Oberflächennormalen basieren auf den Stereo Aufnahmen der LRO-WAC Camera – Low Reconnaissance

[1] http://www.hdrshop.com/

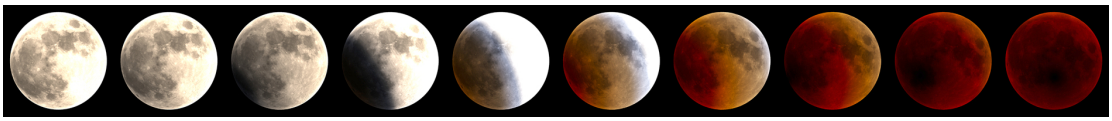


Abbildung 4.5: *Die Ergebnisse des vorgestellten Verfahrens zeigen den halben Ablauf der totalen Mondfinsternis am 15. Juni 2010, wie sie in Mangalore, Indien zu beobachten war. Die einzelnen Auszüge sind mit 18:00 Uhr UTC beginnend in 15 Minuten-Intervallen dargestellt.*

Orbite Wide Angle Camera (NASA, 2011). Texturauflösungen von 256^2 Pixel per Cube-Map Seite bieten ausreichend Detail, auch bei Nahaufnahmen.

Die staubige Oberfläche des Mondes hat eine leicht rötliche Färbung, die durch Anpassen der durchschnittlichen Albedo an ein gemessenes Reflexionsspektrum (Pieters, 1999) simuliert werden kann (Yapo & Cutler, 2009):

$$k_r(\lambda) = -0.27063 + 2.6175 \times 10^{-3}\lambda - 1.2673 \times 10^{-6}\lambda^2. \tag{4.11}$$

Die Verwendung dieses Polynom zweiten Grades liefert für die repräsentativen Wellenlängen $\lambda = (680, 550, 440)$nm (Riley et al., 2004) den Farbkoeffizient $k_\lambda = (0.92, 0.79, 0.64)$. Wie auch für Sterne, sollten die von der Erde aus sichtbaren Mondfragmente durch atmosphärische Streuung beeinflusst werden.

Oppositionseffekt des Mondes

Der Oppositionseffekt (opposition surge oder auch shadow hiding) benennt das Phänomen, wenn bei kleinen Winkel zwischen einfallenden und reflektierten Licht keine Schattierung oder Schatten der Oberfläche mehr erkennbar sind, die Oberfläche also keine sichtbare Tiefenstruktur mehr hat. Allein in den letzten 4Grad vor Vollmond (bei dem der Phasenwinkel gegen Null geht), bewirkt der Oppositionseffekt beim Mond eine Aufhellung der Mondoberfläche von etwa 40%. (Buratti et al., 1996) fanden heraus, dass dieses Aufhellung nicht wie zuvor angenommen allein auf rückstreuende Reflexionen zurückzuführen ist, sondern zum größten Teil auf den Oppositionseffekt.

Da die vom System verwendeten Albedodaten als schattenfrei idealisiert werden (die starken Schatten an den Polen lassen sich auf Kosten der Realitätstreue ausbessern), dies aber nicht sind, sollten Bereiche um die Pole des Mondes aufgehellt werden. Da die verwendete BRDF Gleichung (4.5) eine Annäherung gemessener Daten entstammt, berücksichtigt diese bereits den Oppositionseffekt.

Schatten können durch Ausnutzung der Oberflächennormalen dynamisch erzeugt werden (Blinn, 1978; Max, 1988; Onoue et al., 2004; Green, 2007). Diese würden die Darstellung vor allem an der Tag-Nacht-Grenze, aufgrund der dort sehr langen Schatten, verbessern. Die damit verbundene Verringerung der Intensität der scheinbaren Mondscheibe müsste jedoch wieder ausgeglichen werden. Zudem ist eine Umsetzung in Anbetracht der damit einhergehenden Verbesserung relativ aufwendig, sodass *osgHimmel* keine dynamischen Mondschatten unterstützt.

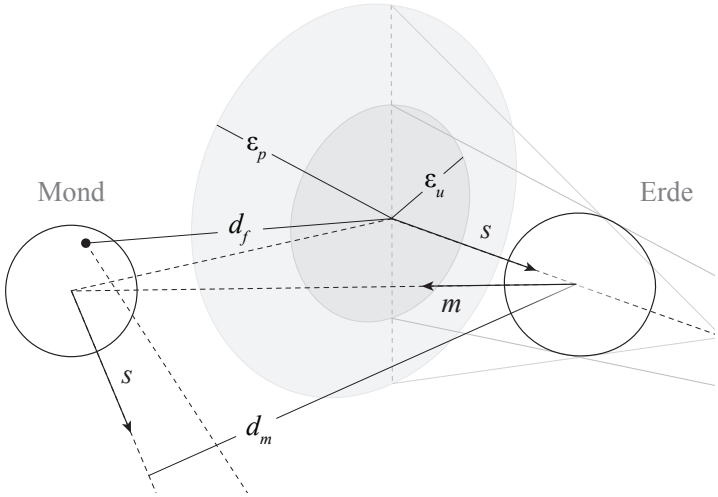


Abbildung 4.6: *Modell zur Abbildung der Mondfinsternisse mit dem Mond bei m, der Finsternissphasen d_m und d_f , der Sonne in Richtung s, der Erde im Ursprung des Models, und den tatsächlichen Schatten Radien ε_u and ε_p.*

4.2 Rendering von Mondfinsternissen

osgHimmel stellt ein neues Verfahren zur echtzeitfähigen Bildsynthese von Mondfinsternissen zur Verfügung, mit dem Ziel photometrischer Ähnlichkeit zur Realität. Abbildung 4.5 zeigt die erste Hälfte einer totalen, vom System zur Laufzeit dargestellten Mondfinsternis. Bei einer Mondfinsternis ist der Phasenwinkel sehr klein, sodass sich Sonne und Mond von der Erde aus gesehen gegenüber stehen. Dabei wirft die Erde einen Halb- und einen Kernschatten auf den Mond. Der Mond kann den Halbschatten (penumbral) oder den Kernschatten anschneiden (partiell), oder sich vollständig im Kernschatten befinden (total). Nur die totale und die partielle Mondfinsternis im Kernschatten führen zu starken Schattengebungen. Seine scheinbare Helligkeit sinkt dabei von −12.74mag auf bis zu +2mag ab, also etwa ein dreihundert Tausendstel. Halbschattenfinsternisse sind oft nicht wahrnehmbar. Da die Umlaufbahn des Mondes leicht gegen die der Erde um die Sonne geneigt ist, läuft der Mond zumeist ober- oder unterhalb des Schattens vorbei. Im Durchschnitt über den Zeitraum von 1999 v.Chr. bis 3000 n.Chr. treten pro Jahr 2.4128 Mondfinsternisse, mit einem Verhältnis zwischen Partiellen zu Totalen von 35 zur 29 Prozent. Den Rest bilden penumbrale, also totale oder partielle Halbschattenfinsternisse.

In diesem Kapitel wird ein einfaches Modell zur Abbildung der Mondfinsternisse vorgestellt und auf Techniken zur Farb- und Helligkeitsbestimmung, sowie auf Probleme der Farbkalibrierung, und Vergleichbarkeit zu anderen Verfahren oder Fotos eingegangen.

4.2.1 Modell zur Abbildung einer Mondfinsternis

Die Magnitude (oder Größe) beschreibt die Eindringtiefe des Mondes während der Mondfinsternis. Farbe und Helligkeit werden als zwei separate, zur Magnitude korrelierende Multiplikatoren ausgedrückt, und können im folgend vorgeschlagenen Modell (Abbildung 4.6) sehr einfach ermittelt werden. Die Trennung von Farbe und Helligkeit dient zur unabhängigen Kontrolle dieser, ähnlich wie Belichtungszeit und Farbfilter in der Fotografie.

Die Durchmesser ε_u und ε_p des Kern- und Halbschatten (Umbra und Penumbra) können als Vielfache des Mondradius angegeben werden. Unter der Annahme mittlerer Mond- und Sonnen-Entfernung können ε_u und ε_p mit 2.65 bzw. 4.65 angenähert werden. Bei genauerer Betrachtung, lassen sich die Durchmesser in Abhängigkeit zur tatsächlichen Mondentfernung D_m und Sonnenentfernung D_s wie folgt bestimmen:

$$\varepsilon_u(D_m, D_s) = \frac{r_e}{r_m} - \frac{(r_s - r_e)\, D_m}{r_m\, D_s}, \tag{4.12}$$

$$\varepsilon_p(D_m, D_s) = \frac{r_e}{r_m} + \frac{(r_s + r_e)\, D_m}{r_m\, D_s}. \tag{4.13}$$

Durch Substitution des mittleren Sonnenradius $r_s = 696\,000$km, mittleren Mondradius $r_m = 1\,737.1$km sowie mittleren Erdradius $r_e = 6\,371$km ergeben sich folgende Abhängigkeiten für die tatsächlichen Schattendurchmesser:

$$\varepsilon_u(D_m, D_s) = 3.6676 - 397.0001\, D_m\, D_s^{-1}, \tag{4.14}$$

$$\varepsilon_p(D_m, D_s) = 3.6676 + 404.3354\, D_m\, D_s^{-1}. \tag{4.15}$$

Zur weiteren Berechnung der Entfernungen d_f und d_m wird das horizontale Erde-Mond-System als normalisiert angenommen, und liefert folglich eine Mondentfernung von 1 (Abbildung 4.6) – der Länge von m. Des weiteren wird die Sonne, wie schon beim Mondmodell in Abschnitt 4.1, als direktionale Lichtquelle betrachtet. Für jedes sichtbare Mondfragment wird die kürzeste Distanz d_f der entsprechenden Mondfragment-Sonnen-Linie g zum Erdmittelpunkt berechnet. Die Position eines Mondfragments a_f ist gegeben durch $a_f = m - \varepsilon_0\, n_m$, mit $\varepsilon_0 = D_m^{-1}$ als normalisierenden Faktor. Mit der Mondfragment-Sonnen-Linie in parametrischer Form $g : x = a_f + x\, s$, ist die kürzeste Entfernung d_f zwischen a_f und g, in Mondradien gemessen und fortan als Finsternisphase (Fragment-Finsternisphase) bezeichnet, definiert durch $d_f = \varepsilon_0 \left| a_f \times s \right|$. Substitution von a_f mit m ergibt die Mond-Finsternisphase d_m.

Im Gegensatz zur Magnitude, lässt sich mit der korrelierenden Finsternisphase ein einfachere Abbildung für die Farbe und Helligkeit vornehmen. Alle Berechnungen sind prinzipiell nur notwendig, wenn der Phasenwinkel φ (Abschnitt 4.1) größer als 175Grad ist. Durch atmosphärische Refraktion können Mond und Sonne, bei einer Mondfinsternis (auch einer Totalen) für Betrachter auf der Erde einen kurzen Zeitraum gleichzeitig sichtbar sein. Dieses Phänomen der horizontalen Finsternis, wird korrekt in *osgHimmel* durch die, für die Darstellung refraktions-korrigierten, horizontalen Positionen abgebildet. Im Modell zur Mondfinsternis darf die Korrektur nicht vorgenommen werden, da alle, die Darstellung beeinflussende Vorgänge unabhängig von der Beobachtung auf der Erde sind. Eine weitere Eigenschaft des Modells ist seine Unabhängigkeit zur gestalterischen Skalierung des Mondes c_d (Abschnitt 4.1.1).

4.2.2 Farbapproximation des Mondes während einer Mondfinsternis

Danjon (1920) hat eine fünf stufige Skala zur Klassifizierung der Färbung und Helligkeit von Mondfinsternissen entwickelt. Da die einzelnen Stufen nur textuell beschrieben sind und in der skalen-eigenen Einheit L (Danjon-Skala), als Durchnummerierung der Stufen angegeben wird, kann diese nicht zur mathematischen Annäherung einer Farbe herangezogen werden.

Stattdessen wird die Farbe als Funktion h über die transformierte Finsternisphase $t_f(d_f)$ modelliert. Da ε_u und ε_p variieren, werden ihnen zur einfacheren Handhabung feste Bereiche

zugewiesen: $0 \leq t_f < 1/2$ für umbrale und $1/2 \leq t_f < 1$ für penumbrale Entfernungen. Die Finsternisphase d_f wird innerhalb dieser Bereiche linear transformiert:

$$t_f(d_f) = \begin{cases} \frac{1}{2} d_f \varepsilon_u^{-1} & \text{falls } d_f < \varepsilon_u \\ \frac{1}{2} + \frac{1}{2}(d_f - \varepsilon_u)(\varepsilon_p - \varepsilon_u)^{-1} & \text{sonst} \end{cases}. \tag{4.16}$$

$h(t_f)$ ergibt einen Multiplikator pro Farbkanal, um die Farbe des Mondfragments anzupassen. Folgende, markante Phänomene werden dabei aggregiert:

- Ein kaum wahrnehmbar Halbschatten, angedeutet durch eine leichte Abdunklung in Richtung des Kernschattens.
- Ein auffälliger Kernschatten, mit einem geglätteten Rand.
- Eine rötlich-orange, zum Zentrum dunkler werdende Färbung des Kernschatten. Diese ist die Folge der wellenlängenabhängigen Brechung des Lichts durch die Erdatmosphäre, bei der kürzere Wellenlängen stärker gebrochen werden. Rotes Licht wird folglich stärker zur Mitte des Kernschattens gebrochen als blaues Licht.
- Eine, sanft zur Kernschattenkante hin, stärker werdende Blaufärbung, durch schwach gebrochenen Lichts.

Die genauen, im System aggregierten Funktionen sind in Abschnitt A.2 gelistet. Abbildung 4.8 zeigt einen Plot von h je Farbkanal. Die Berechnung von h ist für den Fragment Shader sehr aufwendig. Alternativ kann diese vorberechnet und in eine eindimensionale Lookup-Textur über t_f pro Mondfragment, kodiert werden. Anstelle der Spezifikation von Phänomen annähernden Funktionen, kann die Textur auch anders erzeugt werden: Physikalische Simulationen wie die von Yapo & Cutler (2009), künstlerische Gestaltung von Hand, oder Analyse und Mittlung der Verläufe in entsprechenden Fotos sind einige Beispiele.

Helligkeit des Mondes während einer Mondfinsternis

Äquivalent zu h, wird die Helligkeit während der Mondfinsternis als eine Funktion i über t_m beschrieben. Entgegen der pro Fragment transformierten Finsternisphase t_f, korreliert i zur Finsternisphase t_m bezüglich D_m, also zur Mondmitte. Zu Beachten ist hier der benötigte Versatz beim Mapping: Für $t_m = 0.5$ würde sich bereits ein großer Teil der Mondfragmente im Kernschatten, bzw. Halbschatten für $t_m = 1.0$, befinden. Da die Finsternisphase in Mondradien angegeben wird, beträgt der Offset 1 und ergibt mit Gleichung (4.16) folgende Transformation:

$$t_m(d_m) = \begin{cases} d_m (2 + 2\varepsilon_u)^{-1} & \text{falls } d_f < \varepsilon_u + 1 \\ 0.5 - (d_m - \varepsilon_u - 1)\left(2\varepsilon_p - 2\varepsilon_u\right)^{-1} & \text{sonst} \end{cases}. \tag{4.17}$$

Ein Beispiel für eine mögliche Helligkeitsfunktion i ist in Abbildung 4.8 skizziert. Der farbkanal-spezifische Multiplikator für die Farbe eines Mondfragments I_m aus Gleichung (4.4) ergibt sich schließlich aus:

$$C_f(d_f, d_m) = h[t_f(d_f)]\ i[t_m(d_m)]. \tag{4.18}$$

Auszüge resultierender Mondfragment-Farben zu verschiedenen Finsternisphasen sind in Abbildung 4.7 dargestellt.

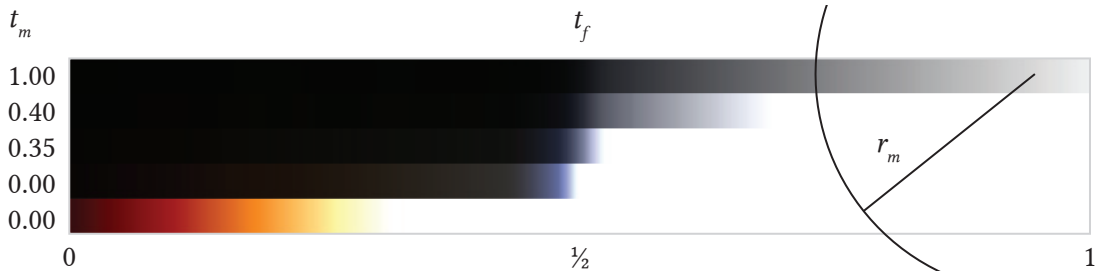


Abbildung 4.7: *Die Fragment-Farben über der Fragment-Finsternisphase t_f angewandt zu ausgewählten Mond-Finsternisphasen t_m. r_m deutet, zur besseren Vorstellung der Farbanwendung bzw. des Größenverhältnis zwischen Mond und Schatten, grob (Halb- und Kernschatten Radien variieren) den Radius des Mondes an. Man beachte dazu, dass die Skala t_f eine Unstetigkeitsstelle an* 0.5 *aufweist.*

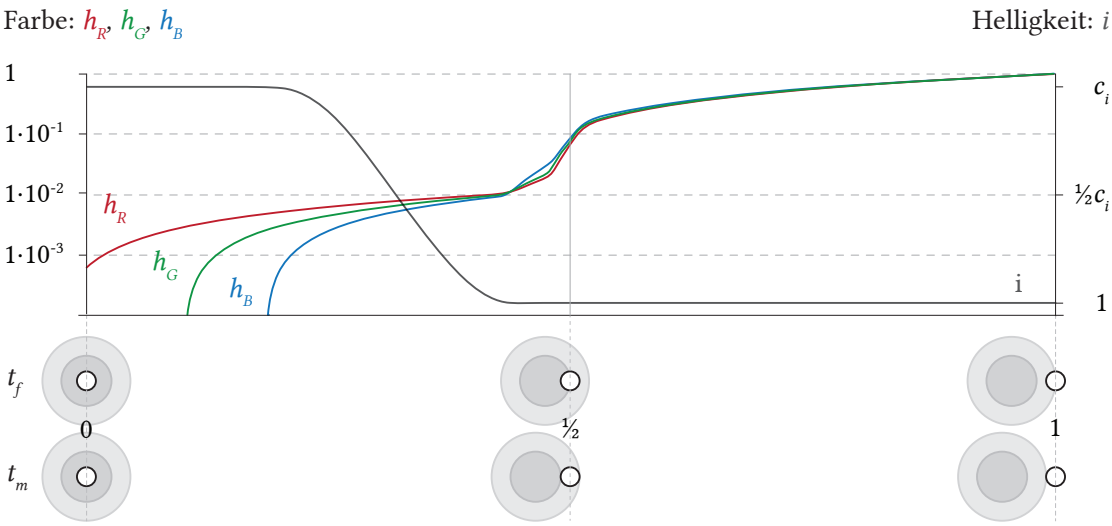


Abbildung 4.8: *Links: Farb-Multiplikator h pro Farbkanal (h_R, h_G, und h_B) dargestellt auf einer logarithmischen Skala. Rechts: Hellighkeits-Multiplikator i auf einer linearen Skala zwischen* 1 *and c_i (mit beispielsweise $c_i = 30$). In Korrelation zur $t_{[f;m]}$-Achse sind die jeweiligen Phasen – den Mondpositionen zum Halb- und Kernschatten – illustriert.*

4.3 Rendering von Sternen

In nahezu allen Systemen zur Darstellung virtueller 3D Welten, selbst in aufwändig produzierten Offline-Sequenzen, unterliegt die Darstellung der Sterne meist schwerer Mängel: Keine erkennbaren Konstellationen, ungewohnte Gleichverteilung, einseitige oder falsche Färbung, oder übertriebenes Funkeln sind nur einige Beispiele.

Die Darstellung von zufällig verteilten Sternen zufälliger Intensitäten, wie beispielsweise in Roden & Parberry (2005), ist nicht glaubwürdig, da Menschen auch hier eine starke Gewöhnung an ihren nächtlichen Himmel, mit seinen typischen Konstellationen und Erscheinungen haben. Wie auch bei den zuvor vorgestellten Phänomenen, ist es nicht Ziel, mit der Hintergrundkulisse vom eigentlich Inhalt und Kontext der virtuellen 3D Welt abzulenken. Die Darstellung der Sterne soll die Glaubwürdigkeit der Gesamtszene unauffällig steigern und bei

Abbildung 4.9: *Nahaufnahmen eines Nachthimmels mit Orion rechts und dem Mond mit Erdschein links. Im Hintergrund ist die Milchstraße ist schwach erkennbar (Farblevel für bessere Erkennbarkeit nachträglich angepasst).*

genauer Betrachtung eine akkurate Abbildung astrophysikalischer Phänomene bieten (Abbildung 4.9). Fotos oder vorberechnete Texturen mit korrekter Sternenplatzierung sind nicht ausreichend, da Sterne durch dass Textursampling oft zu klobig wirken, oder im anderen Extrem, bei leichtesten Kamerabewegungen anfangen zu wackeln. Letzteres entsteht, wenn dass Nyquist-Shannon-Abtasttheorem vernachlässigt wird, und die Abtastrate zu gering gewählt ist (Aliasing). Obgleich hohe Texturauflösungen oder mehrfaches Samplen hier unter zusätzlichen Rechenaufwand Abhilfe schaffen, skalieren sie nicht für steigende Bildschirmauflösungen und -pixeldichten. Die Nutzung von Point-Sprites ist aufgrund der starken Haufenbildung im Zentrum großer Sichtfelder ebenfalls unzureichend. Da Point-Sprites unbeeinflusst von der Verzerrung perspektivischer Kameras sind, nimmt die Sterndichte zum Bildrand hin ab.

osgHimmel bietet eine Simulation tausender, heller, individueller Sterne und deren Darstellung unter Anwendung von Punktspreizfunktionen (PSF). Jeder Stern wird als Punktlichtquelle betrachtet, die in der Darstellung von perspektivischer Verzerrung betroffen ist. Des weiteren, wird eine Abhängigkeit zwischen Öffnungswinkel des Sichtfeldes (*Field of View*) zur dargestellten Sternintensität vorgestellt. Das System skaliert diesbezüglich hervorragend über unterschiedliche Pixeldichten (Pixel Per Inch – ppi) moderner Bildschirme.

Wir berechnen tatsächliche Sternpositionen für korrekte Konstellationen und nutzen, wie schon beim Mond (Abschnitt 4.1), sicht-orientierte Billboards um individuell angepasste PSF zur Darstellung anzuwenden. Eine weitere PSF wird eingeführt um den Dynamikumfang zu erhöhen und physikalische Limitierungen des Ausgabegeräts zu überwinden (wie auch beim HDRR üblich). Die Farbe und Intensität der einzelnen Sterne wird anhand ihrer Temperatur und Entfernung angenähert. Der Ansatz basiert auf den von Maiwald (2009) verfassten Artikel zum hochwertigen Rendering von Sternen.

Diese Idee kann in leicht abgewandelter Form zur Darstellung der Planeten unseres Son-

nensystems genutzt werden, welche aufgrund ihrer höheren Helligkeit wesentlich stärker und länger sichtbar sind als Sterne. Sie können für raumzeitlich korrekte Darstellung des Himmels zum Abend, oder zur Nacht charakteristisch sein, wie beispielsweise zur Jupiter-Venus Konjunktion im März 2012. Planeten sind jedoch keine Punkt-Licht-Quellen, sondern besitzen eine tatsächliche, wahrnehmbare scheinbare Größe. Bei heutigen ppi ist diese Differenzierung allerdings noch nicht nötig, da die Planeten bei korrekter Skalierung meist kleiner als ein Pixel wären – bis zu 66 Bogensekunden (Ausnahmen sind kleine Sichtfelder bzw. Nahaufnahmen). Im Gegensatz zu Sternen, emittieren unsere solaren Planeten kaum eigenes Licht, sondern reflektieren dies der Sonne. Das bedeutet, dass jeder Planet, bei genauer Beobachtung, eine Tag-Nacht-Grenze aufweist. Während die Probleme bei den Sternen eher ihrer enormen Anzahl entspringen, liegen die Probleme hier in der aufwändigeren Positions- und Distanzbestimmung sowie einer physikalisch begründeten Farbgebung. Es ist uns keine, für das Rendering gebräuchliche Beziehung zwischen Phase korrelierender Farbe für solare Planete bekannt. Im Rahmen der Arbeit wird daher nicht weiter auf das Rendering von Planeten eingegangen.

Neben den hellen Sternen, wird für eine korrekte Hintergrundbeleuchtung eine statische Cube-Map, fortan als *Star-Map* bezeichnet, verwendet. Sie aggregiert alle, mit bloßem Auge nicht einzeln erkennbare Sterne und Sternhaufen, und sorgt für einen unauffälligen Hintergrundschein und deutet die Milchstraße an. Ihre Helligkeit wird ebenfalls dynamisch dem Sichtfeld angepasst.

4.3.1 Ansatz zur Darstellung von hellen Sternen

Für einen Betrachter auf der Erde sind unter optimalen Bedingungen bis zu 9500 helle Sterne mit bloßem Auge sichtbar. Diese werden modelliert als Punkte mit vorberechneten Positionen, Farben, und Helligkeit, wobei alle Berechnungen auf Daten des Yale Bright Star Catalogue von Hoffleit & Warren (1995) zurückgreifen. Anschließend werden sie einmalig im GPU Speicher abgelegt, und den raumzeitlichen Anforderungen entsprechend im Vertex Shader rotiert. Dabei werden farb- und helligkeits-beeinflussende, atmosphärische Streuung sowie Szintillationen approximiert. Zuletzt, wird im Geometry Shader ein Billboard pro Stern in zur Helligkeit korrelierender Größe erzeugt und mit Hilfe zweier PSF im Fragment Shader gerendert.

Positionierung von Sternen

Die Berechnung der Vertexkoordinaten basiert auf äquatorialer Rektaszension α und Deklination δ der momentan verwendeten Standardepoche J2000.0, wobei zur weiteren Verwendung in dezimale Stundenwinkel bzw. in dezimale Winkel umgewandelt wurde. Die raumzeitliche Orientierung erfolgt in euklidischen Koordinaten $p = (x, y, z)$, α und δ müssen folglich zuvor noch transformiert werden. Um nicht bei jeder Änderung des Datums, der Uhrzeit oder des Ortes die Positionen aller Sterne aktualisieren zu müssen, werden diese einmalig, initial an die GPU übertragen. Mit der von Jensen et al. (2001a) vorgeschlagenen Transformationsmatrix R_s, werden die Koordinaten in topozentrische, horizontale Koordinaten transformiert. Entsprechend der Zeit T in Julianischen Jahrhunderten, werden mit der Rotationsmatrix P die Einwirkungen von *Präzession* (precission) – also die Richtungsänderung der Achse eines rotierenden Körpers (der Erde) – und *Nutation* – den periodischen Schwankung der Erdachse

Abbildung 4.10: *Abhängigkeit der Sternhelligkeit zur Auflösung des Sichtfeldes: Links eine Ansicht mit Orion im Zentrum, bei normalem Sichtfeld und für $m_A = 5.0$. In der Mitte eine Vergrößerung der markierten Region links. Rechts die gleiche Region, jedoch bei kleinerem Sichtfeld in nativer Auflösung. Aufgrund der angepassten Helligkeit, kommen wesentlich mehr Sterne zum Vorschein (Farblevel für bessere Erkennbarkeit nachträglich angepasst).*

– berücksichtigt. R_s ist gegeben mit:

$$R_s = R_y(lat - \pi/2)\, R_z(-LMST)\, P, \tag{4.19}$$

$$P = R_z(0.01118\, T)\, R_y(-0.00972\, T)\, R_z(0.01118\, T), \tag{4.20}$$

mit lat in rad, dem Breitengrades des Betrachters, und der mittleren lokalen Sternzeit $LMST$ (Meeus, 1994). Die Vertexpositionen werden folglich im Vertex Shader über $p\, R_s$ angepasst. Leider erlaubt dieser Ansatz nicht, die jährlichen Eigenbewegungen der Sterne zu berücksichtigen, da diese bei größeren Zeitsprüngen oder schnellem Zeitablauf eine Neuberechnung der Sternpositionen und damit einhergehende Übertragung an die GPU erfordern. Ähnliches gilt für die Refraktionskorrektur. Diese müsste ebenfalls individuell im Vertex Shader erfolgen, und könnte beispielsweise durch eine Look-Up Textur, mit in Abhängigkeit zum Höhenwinkel vorberechneter Refraktionswerte oder angenähert über die Luftmasse (Abschnitt 4.4) erfolgen. Die Anzahl an Texturzugriffen pro Frame würde dadurch jedoch enorm ansteigen. Sofern die Darstellung der Sterne nur für einen festen, statischen Zeitraum erfolgt, sollten erwähnte Korrekturen angewandt werden und die Positionen bereits CPU-seitig in horizontale Koordinaten transformiert werden. Da die astrophysikalische Simulation in *osgHimmel* priorisiert für dynamische Anwendungszenarien entwickelt wurde, besteht diese Korrekturmöglichkeit noch nicht.

4.3.2 Darstellung und scheinbare Helligkeit der Sterne

Zur Beschreibung der Helligkeit der Gestirne wird in der Astronomie die *scheinbare Helligkeit m* verwendet. Sie ist ein logarithmisches Maß (Reddy et al., 2007) für die Helligkeit eines Himmelskörpers, betrachtet von einem Beobachter auf der Erde, unter Abwesenheit der Atmosphäre. Die scheinbare Helligkeit wird in Magnitude (mag) angegeben: je kleiner die Magnitude, desto heller der Himmelskörper. Ein Stern mit 1mag erscheint hundert mal heller als ein Stern mit 6mag. Dies führt zu einer relativen Helligkeit von $2.512^{m_1 - m_2}$, für zwei Sternen mit den scheinbaren Helligkeiten m_1 and m_2. Für Anwendungen innerhalb der Erdatmosphäre

muss eine Abschwächung durch addieren von 0.4mag erfolgen.

$$\Delta_m(m) = 2.512^{m_a - m} \tag{4.21}$$

wird benutzt um die individuelle Helligkeit eines Sternes bezüglich einer Kontroll-Magnitude m_a zu ermitteln. m_a korreliert zur momentanen Helligkeitsempfindlichkeit des Beobachters, kann aber beliebig definiert werden. Für gewöhnlich erscheinen hellere Sterne auch größer. Allerdings sind bereits die schwächsten und somit kleinsten Sterne bezüglich der Pixeldichte heutiger Ausgabegeräte bereits zu klobig. Ziel ist es, die Sterne in der minimalen Auflösung darzustellen ohne dabei auf einen visuellen Indikator zur Helligkeit zu verzichten. Das System verwendet dazu zwei Punktspreizfunktionen, eine *Kern-PSF* und eine *Glare-PSF*. Die Kern-PSF wird unabhängig von der Billboardgröße des Sterns mit einem festen, minimalen Durchmesser gezeichnet und dient der initialen Helligkeitsangabe des Sterns. Übersteigt die Helligkeit des Sterns das Abbildungsvermögen der Kern-PSF, wird die Glare-PSF addiert. Diese bildet die überschüssige Helligkeit durch Größe und Intensität ab. Das Ergebnis sind nicht-verpixelte Sterne, mit scharfem Kern und aussagekräftigem Schein zur Indikation der scheinbaren Helligkeit. Dies ist photometrisch gesehen, nicht korrekt, erlaubt aber eine gute Annäherung innerhalb eines Renderingpasses, ohne Postprocessing.

Für eine Abbildung radialer Streifen und Anpassung an die Wahrnehmung kann die Glare-PSF auch auf Grundlage des von Spencer et al. (1995) vorgestellten, aufwendigeren Verfahrens generiert werden. Unter Nutzung mehrerer Rendering-Durchläufe können zeitlich dynamische Glares, über eine physikalische Modellierung der menschlichen Helligkeitswahrnehmung, nachträglich im Bildraum generiert werden (Ritschel et al., 2009; Theisgen, 2011).

Der feste Radius der Kern-PSF wird zur Vermeidung von Aliasing-Artefakten mit 2pixel definiert. Dies ist etwas größer als die minimal benötigte Abtastrate (mit einem Radius von $\sqrt{2}$pixel), und ist mit der Wahl der Kern-PSF begründet. Wird zusätzliches Antialiasing angewandt, kann dieser auch kleiner gewählt werden. Die minimale Größe des Billboards ergibt sich entsprechend. Im Bildraum wird dieser Radius q über den vertikalen Öffnungswinkel des Sichtfeldes γ_v in rad und der vertikalen Ausgabeauflösung res_v in pixel auf der CPU ermittelt:

$$q = 2\sqrt{2}\,\tan\left(\frac{\gamma_v}{2}\right) res_v^{-1}. \tag{4.22}$$

Als Kern-PSF wird $T(l) = 2l^3 - 3l^2 + 1$ verwendet, wobei $l \in [0;1]$ die normalisierte Entfernung des Fragments zum Billboardmittelpunkt beschreibt. Die direkte Korrelation zwischen der Helligkeit Δ_m und der Intensität der Kern-PSF wird mittels I_T beschrieben:

$$I_T = V_T^{-1} \Delta_m(m)\, c_q\, q^{-2}, \tag{4.23}$$

mit $c_q \approx 4 \times 10^{-7}$, ermittelt durch Vergleiche der resultierenden Darstellungen bei $m_a = 4$ mit Fotos. Durch q^{-2} ist I_T umgekehrt proportional zu γ_v, so dass bei kleiner werdendem Sichtfeld, mehr und mehr Sterne erkennbar werden (Abbildung 4.10). Stellt man sich die PSF als Rotationskörper vor, kann dessen Volumen V als Intensität interpretiert werden. Integration durch Rotation (disc-integration) ergibt ein Volumen von $V_T = 1.167$ für T. Die resultierende Intensität I_T T ist 1 für $m = m_a$.

Für $m < m_a$ wird zusätzlich die Glare-PSF mit folgender Intensität verwendet:

$$I_G = \Delta_m(m + V_T - 1) - 1. \tag{4.24}$$

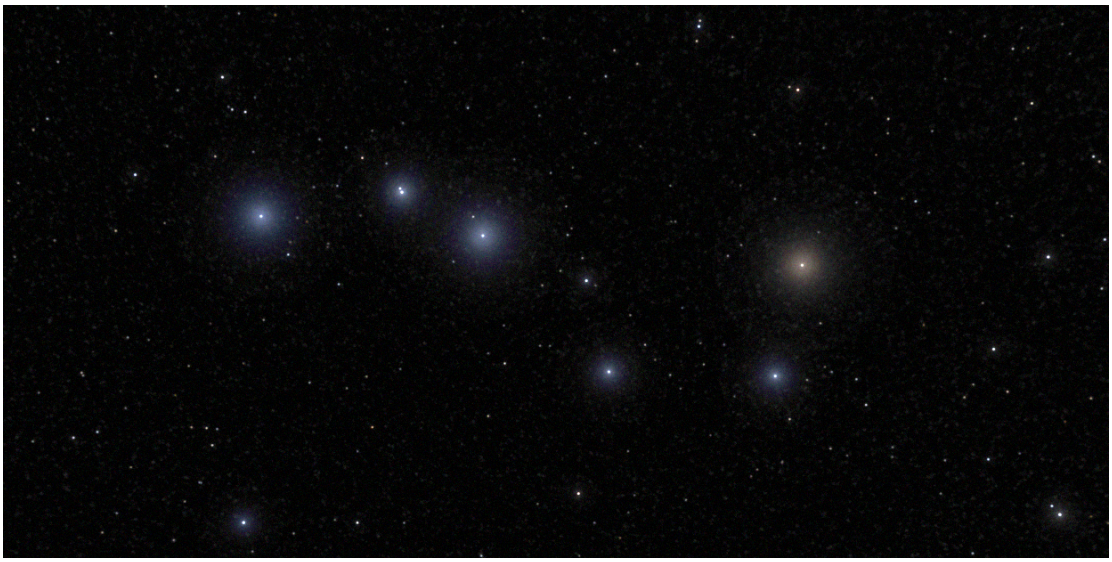


Abbildung 4.11: *Vom System erzeugte Darstellung des Großen Wagen, mit angenäherter Helligkeit und Farbe. An der rechten oberen Ecke des Wagens befindet sich der Stern Dubhe in orange-gelbener Färbung.*

Die Glare-PSF ist mit $G(l) = \sqrt[c^G]{l}$ und $c^G > 8$ gewählt und der Billboard-Radius k ergibt sich wie folgt:

$$k = \max\left(q, c_k \sqrt{I_G}\right). \tag{4.25}$$

Im Gegensatz zur Kern-PSF, wird die Intensität der Glare-PSF nicht in Abhängigkeit ihres Volumens, sondern über den auflösungs-abhängigen Koeffizienten c_k skaliert. Das bedeutet, dass ab $\Delta_m > 1$, also mit Einsetzen der Glare-PSF, keine korrekte Abbildung der Helligkeit in der Darstellung erfolgt. Dies ist unter der Bedingung, die Sterngröße bzw. den Radius Kern-PSF nicht zu beeinflussen und somit klobige Sterne zu verhindern, ein notwendiger Kompromiss.

Bei der Anwendung der Kern-PSF, wird T mit k/q auf den erzielten, konstanten Radius skaliert. Sterne mit einer relativen Helligkeit $\Delta_m < 0.01$ werden für die Darstellung verworfen, um die Füllrate zu verringern.

4.3.3 Farben der Sterne

Die korrekte Färbung der Sterne trägt zur Glaubwürdigkeit der Sternbilder bei. So ist eines der bekanntesten Sternbilder, dem Großen Wagen, welches die sieben hellsten Sterne des Sternbildes Großer Bär bezeichnet, am Nachthimmel sehr einfach auszumachen. So befindet sich der Stern Dubhe (Alpha Ursae Majoris, HR 4301) an der rechten oberen Ecke des Wagens. Er gehört einer anderen Spektralklasse als die anderen Sterne des Großen Wagens an und weist eine orange-gelbene Färbung auf (Abbildung 4.11). Eine individuell korrekte Färbung ist weniger wichtig für die unbewusste Glaubhaftigkeit der Hintergrundkulisse – hier würden zufällige, gewichtete Farbvariation ausreichen – sondern dient der korrekten Darstellung bei genauerer Beobachtung, z.B. bei Nahaufnahmen, also kleinem Field of View.

Die vollständige Herleitung der Farben wird im Folgenden nur grob, mathematisch hinreichend skizziert (Hunt, 1987). Die Bezeichnungen der Parameter beziehen sich auf den während der Entwicklung von *osgHimmel* angefertigten Sternenkatalog (Müller, 2012). Darin katalogisierte *B-V* Werte im Johnson-Morgan-System (Johnson & Morgan, 1953), beschreiben das

Verhältnis des blauen Anteils zum übrigen, wahrnehmbaren emittierten Licht in scheinbarer Helligkeit. Diese gilt es in den RGB-Farbraum zu überführen. Aus dem *B-V* Wert in mag eines Sterns lässt sich nach Jensen et al. (2001a) oder in diesem Fall Olson (1998) dessen Temperatur *TEMP* in K annähern:

$$TEMP = \frac{7000}{B\text{-}V + 0.56}. \tag{4.26}$$

Mit Hilfe eines Polynomfits wird die Temperatur auf die Plancksche Kurve abgebildet, welche die möglichen Farben eines glühenden schwarzen Körpers beschreibt, und Farbartkoordinaten (Chromatizität) PL_x und PL_y ermittelt (Kim et al., 2003):

$$PL_x = \sum_{i=0}^{3} 10^{3\cdot i} c_i^{x_j} \, TEMP^{-i} \qquad \text{mit } j = \begin{cases} 0 & \text{falls } 1667\text{K} \leq T < 4000\text{K} \\ 1 & \text{falls } 4000\text{K} \leq T < 25000\text{K} \end{cases}, \tag{4.27}$$

$$PL_y = \sum_{i=0}^{3} c_i^{y_j} \, PL_x^i \qquad \text{mit } j = \begin{cases} 0 & \text{falls } 1667\text{K} \leq T < 2222\text{K} \\ 1 & \text{falls } 2222\text{K} \leq T < 4000\text{K} \\ 2 & \text{falls } 4000\text{K} \leq T < 25000\text{K} \end{cases}, \tag{4.28}$$

mit $c_i^{x_j}$ und $c_i^{y_j}$ als Koeffizienten der Polynome wie in Tabelle A.1 abgebildet. Die Farbartkoordinaten werden in Tristimuluskoordinaten CIE_x und CIE_z des CIE-Normvalenzsystem transformiert (mit $CIE_y = 1$)

$$CIE_x = PL_y^{-1} \, PL_x, \tag{4.29}$$

$$CIE_z = PL_y^{-1} \left(1 - PL_y - PL_x\right), \tag{4.30}$$

und schließlich in den sRGB-Farbraum abgebildet (Olson, 1998; Krystek, 1985):

$$sRGB_r = +3.240 CIE_x - 0.499 CIE_z - 1.537, \tag{4.31}$$

$$sRGB_g = -0.969 CIE_x + 0.042 CIE_z + 1.876, \tag{4.32}$$

$$sRGB_b = +0.056 CIE_x + 1.057 CIE_z - 0.204. \tag{4.33}$$

Die Farben der Sterne wurden einmalig zur Verwendung vorberechnet und sind direkt in *osgHimmel* als auch in Form eines erweiterten Katalog verfügbar (Müller, 2012). Ein zeitaufwändiges Suchen und Interpolieren über Farbtabellen entfällt.

Theisgen (2011) erklärt, wie die Darstellung der Sterne mittels Tone-Mapping und Eigengrau, also der Farbe die wir in völliger Dunkelheit sehen, wesentlich verbessert werden kann.

Atmosphärische Abschwächung der Sterne

Licht das einen Beobachter auf der Erde bei Nacht erreicht, ist ebenso wie am Tage, durch *Extinktion*, einer atmosphärischen Absorption, und durch Streuung beeinflusst. Bei Nacht ist der Einfluss zwar schwächer als am Tage, jedoch bemerkbar (Abbildung 4.12). Beide Phänomene stehen in direkt proportionaler Abhängigkeit zur Luftmasse. Diese ist ein Maß für die Länge des Weges, den dass Licht eines Sterns durch die irdische Atmosphäre zurücklegt. Sie ist folglich am Zenit am geringsten und wird zum Horizont hin stärker. Die Folge ist, dass Sterne zum Horizont hin durch Rayleigh-Streuung in Abhängigkeit zur Wellenlänge abgeschwächt

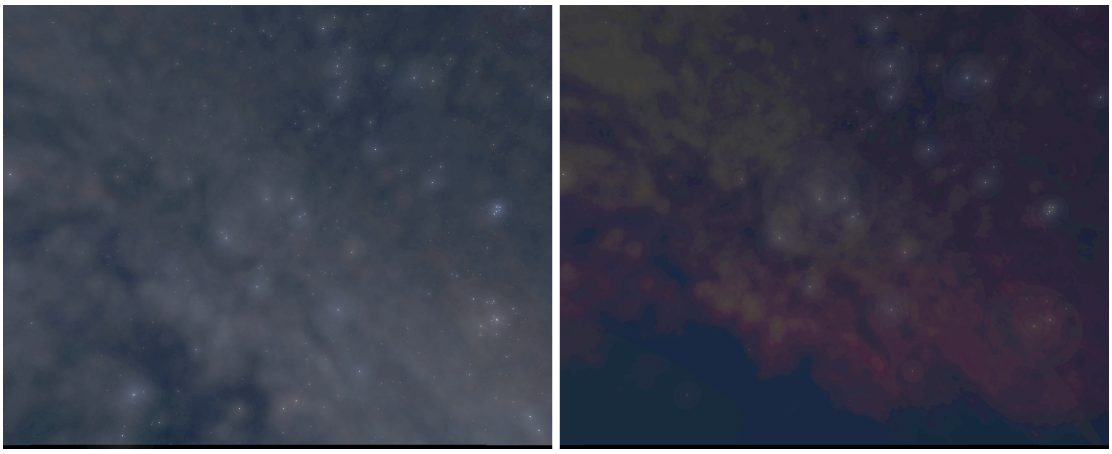


Abbildung 4.12: *Links ein Sternenhimmel knapp über dem Horizont, mit individuellen Sternen und der Star-Map im Hintergrund. Rechts der gleiche Ausschnitt, jedoch mit wellenenlängen-abhängiger, atmosphärischer Abschwächung in Farbe und Intensität (Farblevel für bessere Erkennbarkeit nachträglich angepasst).*

werden oder gar verschwinden (Extinktion). Die Auswirkung auf die Helligkeit des Sterns ist dabei stärker als die Änderung in der Farbe, welcher der Blauanteil stärker entzogen wird (rötliche Färbung).

Wir berechnen die relative Luftmasse Φ für eine nach Bucholtz (1995) vereinfachte Atmosphäre konstanter Dichte (Abschnitt 4.4). Sie gibt das Verhältnis der Luftmasse in einer, über die Zenitdistanz θ definierten Richtung, zur Luftmasse am Zenit t_e an. Mit dem mittleren Erdradius $r_e = 6371\text{km}$, der Höhe des Betrachters über der Erdoberfläche h in km, und θ kann Φ über den Sinussatz wie folgt bestimmt werden:

$$O(\theta) = c_r\, \beta_r\, \Phi(\theta) \qquad \text{mit} \tag{4.34}$$

$$\Phi(\theta) = -\sin\left(\arcsin\left(\frac{r_e + h}{r_e + t_e}\sin\theta\right) - \theta\right)\frac{r_e + t_e}{\sin\theta\,(t_e - h)}, \qquad \text{oder} \tag{4.35}$$

$$\Phi(\theta) = \frac{-(r_e + h)\,\cos\theta + \sqrt{(r_e + t_e)^2 - (r_e + h)^2\,\sin^2\theta}}{(t_e - h)}. \tag{4.36}$$

Farbe und Intensität wird mit O abgeschwächt, wobei c_r mit $c_r \approx 6$ und β_r als Streuungskoeffizient für Luftmoleküle der Rayleigh-Theorie gegeben ist. In diesem Modell beschreibt es die Abhängigkeit zur Wellenlänge mit $\beta_r = (0.52, 1.22, 2.98)$ für $h = 0$ (Bucholtz, 1995). Für t_e werden für Rayleigh-Streuung meist Werte um 8km verwendet (Nishita et al., 1993; Preetham et al., 1999; Bruneton & Neyret, 2008). Sofern die Höhe des Beobachters ignoriert werden kann, und der geozentrische (wahre) Horizont konstant bei $\theta = \pi/2$ liegt, kann auf einfachere Annäherungen wie $\Phi(\theta) \approx (\pi/2 - 0.984\,\theta)^{-1}$ oder der folgenden, von Preetham et al. (1999) vorgeschlagenen und beispielsweise von Yapo & Cutler (2009) verwendeten Annäherung, zurückgegriffen werden:

$$\Phi(\theta) \approx \frac{1.002432\cos^2\theta + 0.148386\cos\theta + 0.0096467}{\cos^3\theta + 0.149864\cos^2\theta + 0.01012963\cos\theta + 0.000303978}. \tag{4.37}$$

Die hier beschriebene Abschwächung der Farbe und Intensität durch atmosphärische

Streuung wird im System sowohl für helle Sterne, als auch für die Star-Map und den Mond verwendet.

Szintillationen

Szintillationen beschreiben das Funkeln der Sterne am Nachthimmel. Atmosphärische Fluktuationen im Millisekunden-Bereich – Schwankungen im Brechungsindex, verursacht durch warme und kalte Luftschichten, auch als Luftunruhe bezeichnet – verursachen feine Änderung in der Streuung und Refraktion Ellison (1952). Da Streuung und Refraktion abhängig von der Wellenlänge des Lichtes sind, wirken sich beide auf die scheinbare Intensität und Farbe des von Punktlichtquellen (Sterne) emittierten Lichts aus. Die Erscheinung des Mondes und der Sonne (sowie Planeten) werden hier durch leichten Verschiebungen und Deformationen der Oberflächen bzw. der Kanten ihrer scheinbaren Scheiben nur unter starker Vergrößerung erkennbar (*Astronomisches Seeing*). Diese Effekte sind unter normalen Bedingungen nicht wahrnehmbar und werden vom System vorerst nicht beachtet.

Genau wie die Streuung, werden Szintillationen zum Horizont hin stärker, und können als Funktion über die Luftmasse Φ beschrieben werden. Dabei wird für jeden Stern eine zufällige Basis $n \in [0;1]$ pro Frame erzeugt, um Fluktuationen auf der kleinsten verfügbaren Zeitskala (Frame) zu ermöglichen. Die direkte Anwendung der Zufallszahlen als Fluktuation der Luftmasse führt zu unnatürlich häufigen, stark auffälligen Funkeln der Sterne. Um die Anzahl starker, simultaner Szintillationen zu reduzieren, wird die Zufallsverteilung durch eine nicht-lineare Abbildung N von n auf $[0;1]$ angepasst und damit die Gesamt-Intensität S aller Szintillationen gesenkt. Für N kann beispielsweise $N(n) = 0.02/n$ gewählt werden und S wird wie folgt definiert:

$$S(\theta, n) = c_s \, \beta_r \, \Phi(\theta) \, N(n), \tag{4.38}$$

mit c_s zur Anpassung der Intensität (wir nutzen $c_s \approx 20$). Schließlich wird die Helligkeit jedes Sterns mit β_R per Farbkanal bzw. repräsentativer Wellenlänge abgeschwächt. In *osgHimmel* ist damit eine unabhängige Kontrolle über Szintillationen und Streuung der Sterne erlaubt. Eine Zusammenführung von O und S als einheitlicher Ausdruck der Extinktion I_{ext} über die Luftmasse ist möglich durch:

$$I_{ext}(\theta, n) = \beta_R \, \Phi(\theta) \, (c_r + c_s \, N(n)) \, . \tag{4.39}$$

Für die Intensität der Kern-PSF ergibt sich mit Gleichung (4.23):

$$I_T = V_T^{-1} \Delta_m(m) \, c_q \, q^{-2} - I_{ext}(\theta, n). \tag{4.40}$$

4.3.4 Star-Map zur Hintergrundbeleuchtung

In klaren Nächten mit geringen künstlichen Lichtquellen (z.B. Stadtlicht) ist die Milchstraße leicht erkennbar (Abbildung 4.12). Ein Screen-Aligned Quad wird zur Darstellung des kosmischen Hintergrunds durch eine Cube-Map, der Star-Map, verwendet. Diese kann aus der computergenerierten Tycho Catalog Skymap[2] heraus in eine Cube-Map transformiert werden. In der Tycho Catalog Skymap sind alle Sterne in äquatorial Koordinaten und mithilfe

[2] http://svs.gsfc.nasa.gov/vis/a000000/a003500/a003572/

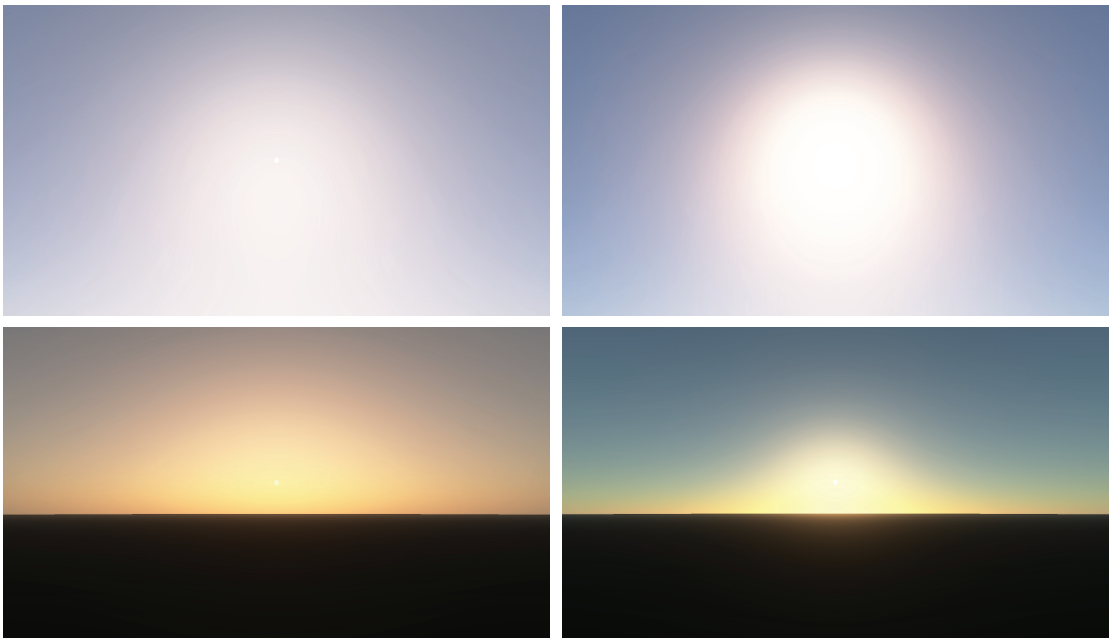


Abbildung 4.13: *Vergleiche des Mie-Streuungs-Verhaltens am Vormittag (oben) und bei Sonnenaufgang (unten) für jeweils $g = 0.6$ (links) und $g = 0.8$ (rechts).*

einer Polarkoordinaten Projektion dargestellt. Die Verfahren zur Farb- und Intensitätsabbildung sowie zur Darstellung der Sterne sind den hier vorgestellten Verfahren ähnlich.

Die Auflösung der Star-Map sollte gering sein und die Milchstraße durch Helligkeitsbereiche statt einzelner Sterne andeuten. Besonders bei höheren Auflösungen mit sichtbaren, einzelnen Sternen müssen diese entfernt werden, um eine doppelte, versetzte Darstellung heller Sterne zu vermeiden. Die raumzeitliche Orientierung durch Gleichung (4.19) wird auf den Textur-Lookup angewand. Die Helligkeit wird wie folgt skaliert:

$$c_s \sqrt{q}^{-1} \, 2.512^{m_a - m}, \tag{4.41}$$

wobei c_s sowohl von der Auflösung als auch der initialen Helligkeit der Star-Map abhängt. Sie sollte für ein repräsentatives Sichtfeld bei $m_a = 4$ oder $m_a = 5$ so angepasst sein, dass die Milchstraße kaum wahrnehmbar ist. Wie auch bei den hellen Sternen, werden die Auswirkungen der Streuung sowie die Abhängigkeit der Helligkeit zum Sichtfeld berücksichtigt. Szintillationen jedoch sind unpassend, da die Sterne nicht mehr als vereinzelte Punktlichtquellen auszumachen sind.

4.4 Rendering der Atmosphäre

Für eine glaubhafte Bildsynthese der Atmosphäre sorgt in *osgHimmel* das von Bruneton & Neyret (2008) vorgestellte Verfahren. Dieses besticht durch die bisher beste, echtzeitfähige Annäherung an den CIE Standard Sky nach Zotti et al. (2007). Zudem wird der *Erdschattenbogen*, also der erd-eigene Schatten der kurz nach Sonnenuntergang bzw. vor Sonnenaufgang auf der, der Sonne gegenüberliegenden Seite zu erkennen ist, berücksichtigt und Mehrfach-Streuung unterstützt (Abbildung 4.15). Das Verfahren greift auf vorberechnete, in Texturen kodierte Streuungswerte zurück, und senkt dadurch den Rechenaufwand zur Laufzeit.

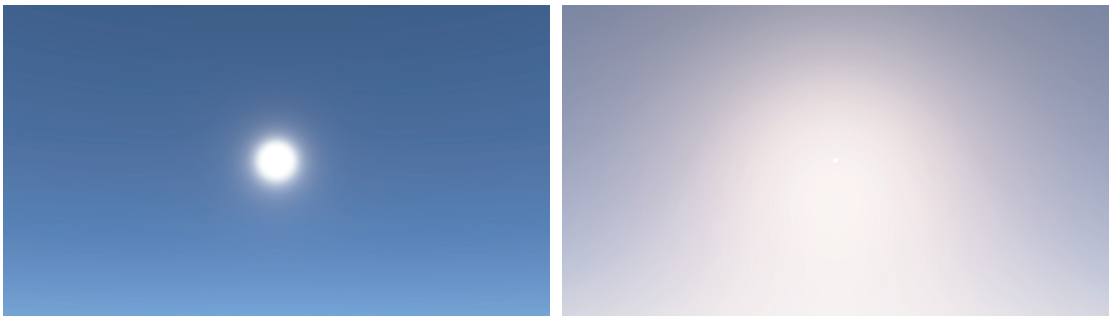


Abbildung 4.14: *Gegenüberstellung einer sauberen Atmosphäre (links) mit* $H_M = 1.2$, $c_M = 8.0$, *und* $g = 0.99$ *zu einer verschmutzten Atmosphäre (rechts) mit* $H_M = 6.0$, $c_M = 20.0$, *und* $g = 0.6$.

Zuerst wird zur Laufzeit die Position der Sonne in horizontalen Koordinaten transformiert (Meeus, 1994). Über die Entfernung zur Sonne lässt sich wie schon beim Mond (Gleichung (4.1)) ihr scheinbarer Durchmesser berechnen. Die Farben beliebiger Fragmente des Himmels innerhalb der Atmosphäre können mit Hilfe des von Preetham et al. (1999) vorgestellten Modells approximiert werden: Auf dem Weg einer elektromagnetischen Welle der Wellenlänge λ durch die Atmosphäre, kann diese an verschiedenen Luftmolekülen gestreut, reflektiert oder absorbiert werden. Die Stärke und Häufigkeit dieser Einflüsse ist abhängig von der Wellenlänge und der Größe der getroffenen Teilchen. Für einen Betrachter, werden diese Streueinflüsse über jedes Fragment bzw. dessen Sichtrichtung, in Abhängigkeit zur Sonnenposition integriert. Die Streuung wird als Produkt eines wellenlängen-abhängigen, luftspezifischen Koeffizienten und einer Winkelphasenfunktion, welche den in eine Richtung θ gestreuten Anteil des Lichtes berechnet, angenähert. Zur Vereinfachung wird die Atmosphäre mit konstanter Dichte und Höhe angenommen und fortan als *Standard-Luft* bezeichnet – trockene Luft mit einem Anteil von 0.03% CO_2 (bei 450ppm) unter normalen Druck von 760mmHg ($\approx$ 1013.25mb) und einer Temperatur von 15°C (Bucholtz, 1995). Zudem müssen für wellenlängen-abhängige Phänomene, wie auch bei der Streuung in Abschnitt 4.3.3, zur Diskretisierung in Farbkanäle repräsentative Wellenlängen gewählt werden. Preetham et al. (1999) verwenden dazu (700, 530, 400)nm. Bruneton & Neyret (2008) nutzen hingegen die von Riley et al. (2004) vorgeschlagenen Koeffizienten (680, 550, 440)nm. Refraktion wird vernachlässigt. Wang et al. (2007) präsentieren ein entsprechendes Verfahren welches diese berücksichtigt.

Das Streuungsverhalten an Moleküle mit einem Durchmesser von 0.1λ bis 10λ wird durch die nach Gustav Mie und Ludvig Lorenz benannte Lorenz-Mie-Streuung, oder meist nur *Mie-Streuung*, als Streuung an sphärischen Teilchen beschrieben. Sie ist der Grund für den Schein der Sonne, dem gräulichen Dunst und Nebel bei feuchter oder verschmutzter Luft, sowie für die Erscheinung von Regenbögen. Die mathematische Beschreibung erfolgt mit Hilfe einer Cornette-Shanks Phasenfunktion P_M basierend auf Henyey & Greenstein (1941) (Thomas & Stamnes, 2002) und dem Absorptionskoeffizienten β_M^s:

$$P_M(\theta, g) = \frac{3}{8\pi} \cdot \frac{1-g^2}{2+g^2} \cdot \frac{1+\cos^2\theta}{(1+g^2-2\,g\cos\theta)^{1.5}}, \tag{4.42}$$

$$\beta_M^s(h,\lambda) = c_M\,\beta_M^s\ (0\text{m},\lambda)\exp^{-\frac{h}{H_M}}\ . \tag{4.43}$$

mit einem konstanten g (gewöhnlich zwischen 0.7 und 0.9 gewählt, Abbildung 4.13) welches

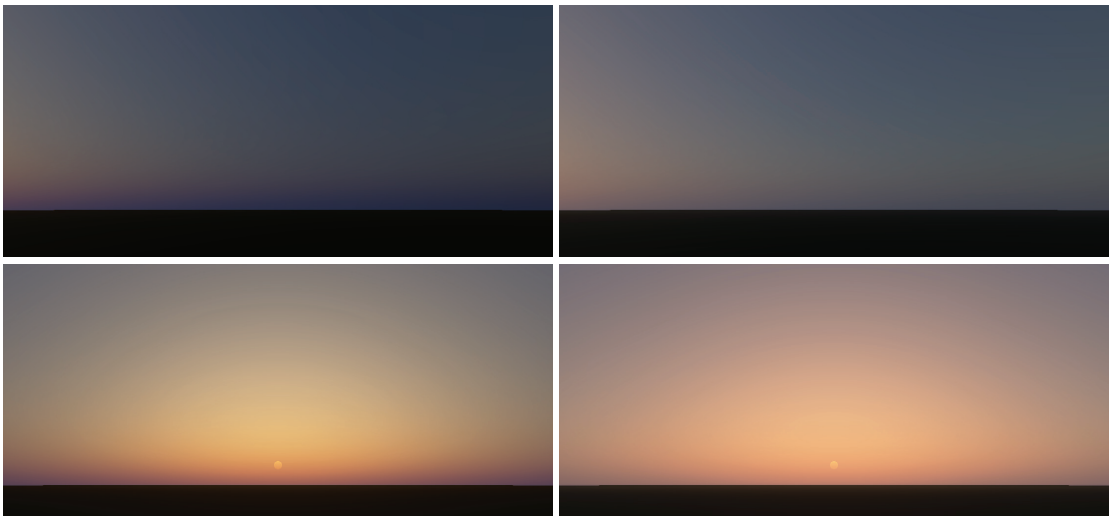


Abbildung 4.15: *Vergleich zwischen Einfach- (links) und Mehrfach-Streuung (rechts) am Beispiel eines Sonnenuntergangs mit der Sonne im Zentrum (unten) und dem Erdschatten nach Sonnenuntergang (oben).*

die Symmetrie der Streuung beeinflusst, θ als Zenitdistanz, h der Höhe des betrachteten Punktes, und $H_M \approx 1.2\text{km}$ als Dicke der Atmosphäre für große Moleküle (Abbildung 4.14). Die Abhängigkeit zur Wellenlänge ist sehr gering und kann vernachlässigt werden. $\beta_M^s\ (0\text{m}, \lambda)$ wird mit der Moleküldichte $K\left(\langle\lambda\rangle\right) \approx 2 \times 10^{-5}$ festgesetzt. Die Moleküldichte K wird beschrieben durch:

$$K(\lambda) = \frac{24\,N_S}{N_S^2\,\left(n^2+1\right)^2} \cdot \frac{\pi^3(n^2-1)^2}{\lambda^4} \cdot \frac{6+3\,\rho_n}{6-7\,\rho_n} \approx \frac{8}{3\,N_S} \cdot \frac{\pi^3\left(n^2-1\right)^2}{\lambda^4}. \tag{4.44}$$

n ist der mittlere Brechungsindex der Standard-Luft mit $n \approx 1.000\,293$ und $N_S = 2.54743 \times 10^{25}\text{m}^{-3}$ deren Teilchenzahldichte. Der letzte Term in Gleichung (4.44) ist der King-Korrektur Faktor (Bucholtz, 1995), welcher die Anisotropie der Luft-Moleküle beschreibt, mit $\rho_n \approx 0.0035$ als Depolarisations-Faktor (Preetham et al., 1999).

Für Moleküle mit einem Durchmesser von unter $0.1\,\lambda$, wird die Mie-Streeung mit der von John William Strutt unter Rayleigh (1871) vorgestellte Theorie, der *Rayleigh-Streuung* verwendet. Die Kernaussage dieser Theorie ist, dass sich der Streuquerschnitt, also die Wahrscheinlichkeit das Licht von einem Molekül absorbiert, gestreut oder reflektiert wird, proportional zur λ^{-4} ist. Blaues Licht (400nm) wird folglich fünf mal stärker gestreut als rotes Licht (700nm), was die Blaufärbung der Atmosphäre erklärt. Dies ist auch der Grund für die leichte Gelb-Färbung der Sonne. Zum Horizont hin wird mehr und mehr grünes Licht gestreut und die Sonne erscheint schließlich orange. Im Gegensatz zur asymmetrischen Mie-Streuung ist die Rayleigh-Streuung nach vorn und hinten symmetrisch ($g = 0$). Phasenfunktion P_R und Koeffizient β_R^s sind spezifiziert mit:

$$P_R(\theta) = P_M(\theta, 0) = \frac{3}{16\pi}\left(1+\cos^2\theta\right), \tag{4.45}$$

$$\beta_R^s(h,\lambda) = K(\lambda)\exp^{-\frac{h}{H_R}}. \tag{4.46}$$

Die molekulare Anisotropie wirkt sich leicht auf die Winkelverteilung, wird in der Gleichung (4.45) jedoch nicht berücksichtigt. Chandrasekhar (1960) bietet dazu folgende Annä-

herung, die bei Bedarf entsprechend für P_R verwendet werden kann:

$$P_R(\theta) = \frac{3}{16\pi(1+2v)}\left((1+3v)+(1-v)\cos^2\theta\right), \tag{4.47}$$

$$v = \frac{\rho_n}{2-\rho_n}. \tag{4.48}$$

Mittels dieser Funktionen lässt sich die *optische Dicke* τ vom Beobachter bis zum oberen Rand der vereinfachten Atmosphäre integrieren. τ gibt an wie gut elektromagnetische Wellen ein physikalisches Medium passieren können (*Transmissivität*). Aufbauend auf diesen Grundlagen, lassen sich die meisten Ansätze zur Atmosphären Darstellung, wie beispielsweise Preetham et al. (1999); O'Neil (2005); Bruneton & Neyret (2008) realisieren. Bruneton & Neyret (2008) kodieren die Streuungseinflüsse für alle Blickrichtungen und Zenitdistanzen über einen Vorberechnungsschritt in eine 3D Textur. Über diese können mit geringen Berechnungen zur Laufzeit die gewünschten Fragmente zu beliebigen Tageszeiten gefärbt werden. Die Verfahren funktionieren auch für Betrachter außerhalb der Atmosphäre, der Übergang von irdisch zu außerirdisch ist jedoch recht abrupt. Zudem ist dies in den system-relevanten Anwendungsfällen vorerst nicht notwendig, und konfligiert mit der Simulation der Wolken, welche zum Zeitpunkt der Erstellung dieser Arbeit nicht von oben betrachtet werden können. Ein weiterer Nachteil ist, dass Änderungen bestimmter Parameter nicht ohne erneute Vorberechnung vorgenommen werden können. Somit muss man bei gleicher Kodierung entweder mit einer Einstellung arbeiten, oder für mehrere die entsprechenden Texturen vorberechnen und zwischen diesen interpolieren.

Beginnend mit dem Sonnenuntergang bzw. endend mit ihrem Aufgang, tritt eine klare, blaue Färbung der Atmosphäre, besonders im Zenit, auf. Diese Färbung tritt nur zu einem Teil durch Rayleigh-Streuung auf. Die Hauptursache liegt in der Absorption des orangen Lichtes in der stratosphärischen Ozonschicht, genauer den Chappuis-Banden. Sie wurde von Hulburt (1953) als *Blaue Stunde des Ozons* benannt. *osgHimmel* widmet sich dieser Erscheinung durch eine zwischen −12Grad und +0Grad gewichteten, zum Zenit hin stärker werdenden Blaufärbung. Die Dauer beträgt abhängig von der Position des Beobachters zwischen 20 und 30 Minuten und wird über die Zenitdistanz der Sonne im benannten Bereich über eine Gaußverteilung gewichtet.

4.5 Rendering von Wolken

Wolken sind ein allgegenwärtiges und sich ständig veränderndes Merkmal jeder Szenerie im Freien. Sie sind ein wesentlicher Bestandteil des Wetters und Indikator für Wetterveränderungen. Wolken sind für jede Bildsynthese von Außenszenen relevant, ihre komplexe Struktur und Dynamik erschweren jedoch eine echtzeit-fähige Simulation. Natürlich aussehende, photo-realistische Wolken auf Consumer-Hardware zu simulieren, erfordert einen Kompromiss zwischen Rechenaufwand und Präzision, bzw. Exaktheit der Darstellung. Eine allgemeine Klassifizierung der unterschiedlichen Wolkentypen ist über die Höhenschichten, in denen diese auftreten möglich. Die ABC-Klassifizierung beschreibt vier Familien von Wolkentypen: A für hohe Wolken über 6km, B für mittlere Wolken zwischen 3km und 6km, C für Wolken unterhalb 3km und schließlich Wolkentürme, welche über den gesamten Höhenbereich auftreten können. Die Modellierung der Wolken muss differenziert, für die einzelnen Wolkenfamilien,

erfolgen. So werden für eher flache A Wolken meist prozedurale, texturbasierte Verfahren wie beispielsweise von Roden & Parberry (2005) und Hasan et al. (2005) verwendet. Dabei entscheidet die Wahl der Noise-Generatoren und Bildfilter über die Glaubwürdigkeit der resultierenden Wolkendarstellung. Für Wolken mit sichtbarer Ausdehnung in die Höhe, werden vorzüglich 3D Ansätze gewählt. Wolkentransformation und -Bewegungen lassen sich sehr einfach durch Textur-Offsets und -Überlagerungen realisieren. Ebert (2003) zeigt prozedurale Verfahren zur Erzeugung sowohl zweidimensionaler als auch dreidimensionaler Wolken. Brickman (2007) und Torchelsen & Musse (2005) stellen jeweils Verfahren zur manuellen, gestalterischen bzw. automatisierten Modellierung dreidimensionaler Wolken vor. Bezüglich der Dynamik schlagen Dobashi et al. (1998) erstmals zelluläre Automaten zur zeit-bedingten Berechnung und Variation der Dichteverteilungen in Volumen Wolken vor (im Ansatz auch für 2D geeignet).

Das vorgestellte System verwendet zwei, einander bezüglich der Wolkenmodellierung ähnliche Verfahren. Zum einen wird ein zweidimensionaler, texturbasierter Ansatz mit einfacher Hintergrundverblendung für hohe A-Wolken verwendet. Für die Darstellung von tiefer liegender B- und C-Wolken werden texturbasierte Wolkenbeschreibungen als Höhenfeld interpretiert und das Streuungsverhalten eines Wolkenfragments mittels Strahlenverfolgung (*Ray-Tracing*) angenähert. Beide Verfahren repräsentieren jeweils eine Wolkenebene und können mehrfach, mit variierenden Parametern, unabhängig zur Abbildung mehrerer Wolkenebenen angewandt werden. Die einzelnen Wolkenebenen werden durch ihre Höhe spezifiziert und werden auf der GPU der Erde entsprechend korrekt gekrümmt (Entgegen der sonst üblichen Vereinfachung zur Ebene). Eine Darstellung von Wolkentürmen ist nicht möglich.

Für eine glaubhafte, leicht bewölkte Szene, seien mindestens zwei verschiedenartige Wolkenschichten mit individueller Dynamik und Bewegungsrichtungen empfohlen. Ziel ist, dass das System mehr als nur die zwei folgend vorgestellten Verfahren zur Verfügung stellt.

4.5.1 Erzeugung prozeduraler, dynamischer Wolken

Die Wolken werden durch Überlagerung mehrere *Perlin-Noise-Tiles* (Perlin, 2002) mit unterschiedlichen Oktaven, Auflösungen und Gewichtungen erzeugt. *Tiles* meint dabei Texturen, die zur Darstellung nahtlos in vertikaler und horizontaler Richtung wiederholt werden können. Dies ist eine wesentliche Voraussetzung für zeit-bedingte Änderungen in Form und Position.

Wichtig ist dabei die Verschiebung der Wolkendecke über die Zeit. Diese suggeriert das Vorhandensein von Wind und erhöht die Immersion der gesamten Szene.

Wolkeninnere Dynamik, also Veränderungen in Form und Struktur, kann durch gewichtete Verschiebungen einzelner Oktaven vor der Komposition erfolgen. Werden hohe Oktaven schneller verschoben als niedrige, unterliegt die grobe Gesamt-Struktur der Wolke nur geringen, bei kurzer Beobachtung kaum wahrnehmbaren Änderungen. Kleinsten Ausprägungen ändern sich hingegen sichtbar schnell. Dies erfordert jedoch, dass die Komposition auf der GPU erfolgt und erhöht die Zahl der Texturzugriffe enorm. Das System erzeugt dazu pro Frame eine hoch-aufgelöste Textur, die *Cloud-Map*. Problematisch ist dabei, dass durch die, bezüglich der gekrümmten Wolkenebene, meist sehr spitzen Beobachtungswinkel nur ein Bruchteil der verfügbaren Auflösung relevant ist. Zum Zenit hin dagegen, ist eine enorme Detaildichte erforderlich. Durch geschickte Wahl der Projektion, beispielsweise der Paraboloid-Projektion

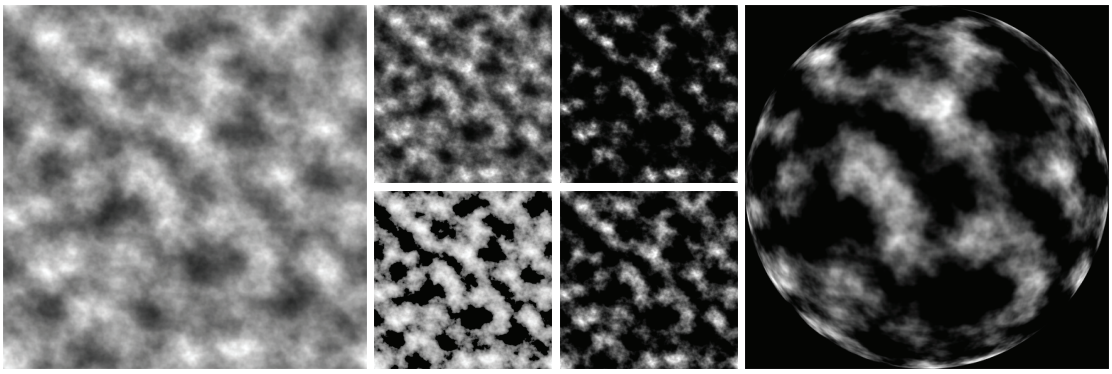


Abbildung 4.16: *Links ist die aus mehreren Oktaven auf der GPU dynamisch zusammengefügte Textur. In der Mitte sind je zwei Varianten mit unterschiedlichen Parametern abgebildet: Oben ist $c = 2/3$ (links) und $c = 1/3$ (rechts). Unten variiert bei gleichem $c = 1/3$ die Härte der Kante mit $s = 0.2$ (links) und $s = 0.6$ (rechts). Rechts ist die Anwendung der Paraboloid-Projektion für eine angemessenere Detaildichte-Verteilung illustriert.*

(Abschnitt 3.1.3), kann die Texturauflösung verringert werden, und die Detaildichte in den relevanten Bereichen beibehalten werden.

Sofern etwas mehr Speicher verfügbar ist, kann einzelne Ausschnitte zu verschiedenen Zeitpunkten in einzelne Schichten einer 3D Textur abgebildet werden. Zeit-bedingte Dynamik erfolgt in diesem Fall über die Wahl der Textur-Schicht, genauer, der Verschiebung der z-Koordinate beim Texturzugriff. Vorteil ist die Hardware unterstützte Interpolation zwischen den Schichten. Dies in Verbindung mit der Paraboloid-Projektion ist für *osgHimmel* geplant, zum Zeitpunkt der Erstellung dieser Arbeit allerdings noch offen.

Ein weiterer, neuartiger Ansatz wäre, die gesamte prozedurale Generierung der Noise-Maps inklusive der Noise-Generatoren vollständig auf die GPU auszulagern. Erkennbare Muster aufgrund von Tiling würden dabei vermieden. Beim Zugriff auf ein Fragment der Wolkenebene kann in Abhängigkeit dessen sichtbarer Fläche, der für das Fragment ausreichende Detailgrad der Wolke bestimmt werden. Die sichtbare Fläche lässt sich über die Projektion des Pixels auf die Ebene ermitteln oder einfacher, über die Differenz der zur oberen und unteren Pixelkante gehörenden Zenitdistanzen. Der Detailgrad lässt sich durch die Anzahl der benötigten Oktaven zur gewichteten Aggregation beschreiben. Offen bleibt vorerst, ob die Anzahl an Texturzugriffen der Noise-Generatoren durch die adaptive Anwendung ausreichend gering gehalten werden kann um messbare Geschwindigkeitsvorteile zu erzielen.

Unabhängig von der gewählten Umsetzung, lassen sich Art und Grad der Bewölkung über eine Level-und-Gamma Korrektur η_c der Noise-Werte η zur Laufzeit beschreiben:

$$\eta_c(\eta, s, c) = 1 - s^{\max(0, \eta - c)}, \qquad \text{oder} \tag{4.49}$$

$$\eta_c(\eta, s, c) = \max\left(0, \frac{c + \eta - 1}{c}\right)^{1-s}. \tag{4.50}$$

Dabei gibt der Parameter $c \in [0; 1]$ jeweils den Grad der Bewölkung an, wobei 0 wolkenfrei und 1 vollständig bewölkt ist. Der Parameter $s \in [0; 1]$ beschreibt die Härte der Wolkenkante. Gleichung (4.49) ist durch Dubé (2005) definiert. In *osgHimmel* findet die Gleichung (4.50) Anwendung, da die Auswirkungen der Parameter intuitiver (entspricht der Level-Korrektur aktueller Bild- und Videobearbeitungs-Werkzeugen – Abbildung 4.16).

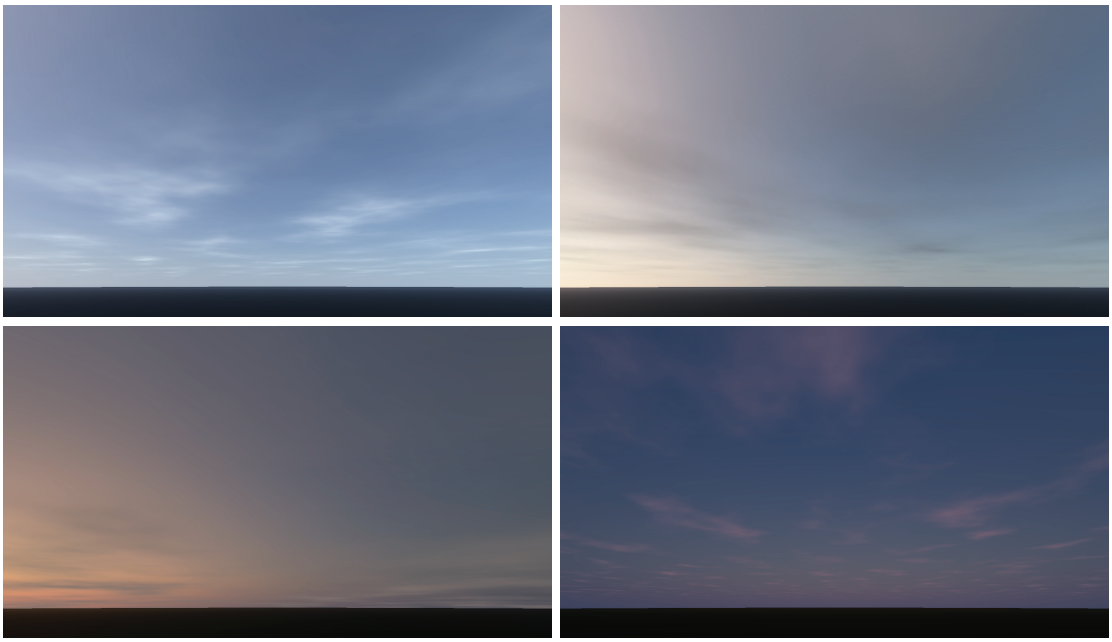


Abbildung 4.17: *Beispiele zweidimensionaler A-Wolken zu unterschiedlichen Tageszeiten, unter verschiedenen Bewölkungsgraden und Färbungen.*

Weitere Parameter der Wolkenschicht sind die Größenskalierung in Nord-Süd und Ost-West Ausrichtung, die Richtung und Geschwindigkeit der Wolkenbewegung (Windrichtung mit entsprechender Länge als Geschwindigkeit), sowie die Geschwindigkeit der wolken-eigenen Dynamik.

4.5.2 Darstellung hoher Wolkenschichten (2D)

Für die Darstellung der zweidimensionalen, höher gelegenen A-Wolken genügt die Färbung und Verblendung der Cloud-Map mit dem Hintergrund. Hierbei handelt es sich meist um dünne, feine oder lichte Wolkenschichten in denen keine komplexen Ausprägungen der Lichtstreuung auftreten. Die Färbung geschieht zur Zeit nicht automatisiert und muss für jeden Anwendungsfall individuell über die Zeit spezifiziert werden. Hauptgrund für die Verwendung dieser Wolken ist die Anreicherung von Variation und Abwechslung in der Atmosphäre. Gerade in Szenen ohne gut ausmodellierter Geometrie, wirkt diese in Verbindung mit einer bis zum Horizont sichtbaren, klaren, wolkenfreien Atmosphäre sehr steril und künstlich (Abbildung 4.17). Sie können mehrfach, in unterschiedlichen Höhen mit jeweils eigenen Cloud-Map Parametern angewendet werden. Dabei sei zu beachten, dass die Wolken-Bewegung in höheren Luftschichten nicht so stark auffällt, da zum einen oft andere Windstärken und -richtungen als in den unteren Schichten vorherschen. Zum anderen würde die Bewegung, aufgrund der größeren, sichtbaren Fläche, auch bei gleichem Windrichtungsvektor weniger stark bemerkbar sein. Beobachtbare Wolkentransformationen wirken in diesen Höhen und Größenordnungen unglaubwürdig und hektisch, und sind meist nur für Zeitraffer relevant.

Während der Morgen- und Abenddämmerung ist es hilfreich, die Wolken orange-rötlich bis flieder oder magenta zu Färben um eine Beleuchtung durch die Sonne, unterhalb der Wolken anzudeuten. Zur physikalischen Korrektheit bei Beobachtungen aus geringen Höhen, soll-

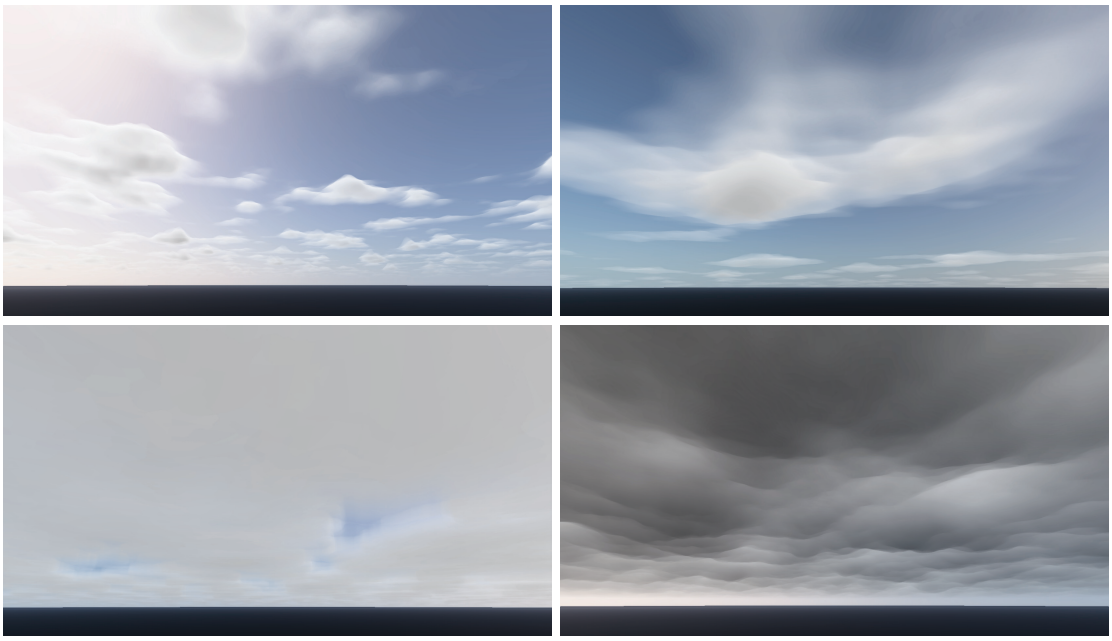


Abbildung 4.18: *Das Verfahren zur Darstellung dreidimensionaler Wolken ist flexibel einsetzbar, und erlaubt kleine Wölkchen, Schleierwolken, dunkle Regenwolken und schließlich vollständige Wolkendecken.*

te dies jedoch nur geschehen, wenn sich die Sonne unterhalb des wahren Horizonts befindet, also nicht sichtbar ist. Zudem sollten sie aufgrund ihrer geringen Dichte, während des Tages nie dunkel sein, also keine Regenwolken repräsentieren, und den Himmel nie vollständig bewölken ($c << 1.0$).

4.5.3 Darstellung niedriger Wolkenschichten (3D)

Zur Darstellung dreidimensionaler Wolken, interpretieren wir die Cloud-Map als vertikales Dichtefeld (Dubé, 2005). Für die Beschreibung einer 3D Wolkenschicht wird neben der Höhenlage α folglich noch eine Höhe der Schicht selbst, sowie ein optionaler Versatz o (Abbildung 4.19) benötigt. Die typisch weiße Erscheinung der Wolken entsteht durch anisotropische Lichtstreuung an Wasserpartikeln innerhalb der Wolke, und ist im Ergebnis der Streuung in transluzenten Objekten ähnlich. Je länger der Weg eines Sonnenstrahls durch die Wolke, desto mehr Licht wird (wellenlängen-unabhängig) ausgestreut. Um dieses Verhalten zu simulieren, werden in Blickrichtung und Sonnenrichtung Schnittpunkte mit dem Dichtefeld der Cloud-Map gezählt. Je höher die Anzahl der Schnittpunkte, desto stärker die Ausstreuung am sichtbaren Fragment der Wolke.

Unter Annahme parallel verlaufender Sonnenstrahlen, werden die Schnittpunkte des Strahls in Blickrichtung mit der unteren (t_0) und oberen (t_1) Grenze der Wolkenschicht ermittelt. Entlang des resultierenden Liniensegments wird in n regelmäßigen Schritten die relative Höhe h ermittelt, und überprüft ob sich diese innerhalb des vertikalen Dichtefeldes befindet (*Probe*). Ist dem so, werden von dieser Probe aus, weitere Probes in regelmäßigen Abständen und in Richtung Sonne entsendet. Gezählt werden nur die Probes, die sich im Dichtefeld befinden (Abbildung 4.19).

Mit diesem Verfahren können quasi-dreidimensionale, glaubhafte Wolken und Wolken-

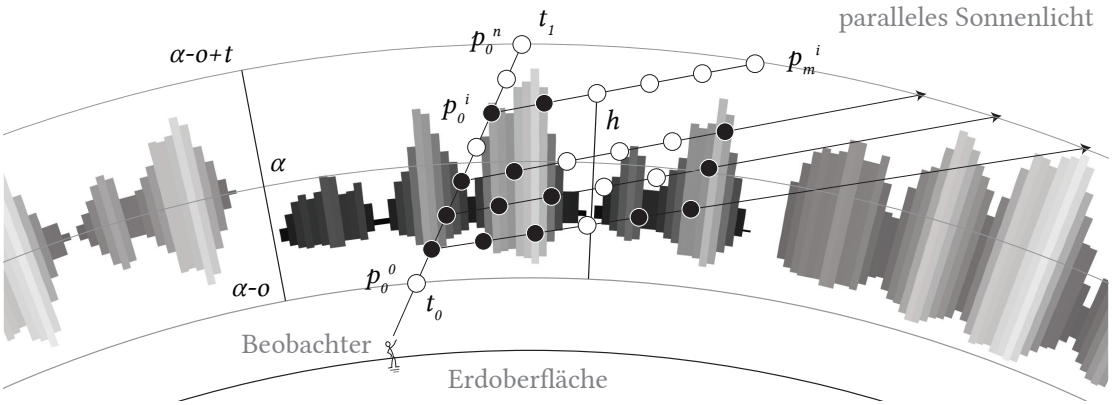


Abbildung 4.19: *Streuungs Approximation im Dichtefeld der Cloud-Map: Zwischen t_0 und t_1 bzw. p_0^0 und p_0^n werden entlang des Sichstrahls Schnittpunkte mit dem vertikalen Dichtefeld gezählt. Schwarze Punkte annotieren Probes die sich im Dichtefeld befinden, Weiße liegen außerhalb. Von jeder im Dichtefeld befindlichen Probe des Sichtstrahls, werden m weitere Probes in kurzen Abständen zur Sonne entstand. Über den vertikalen Versatz, der Bewölkung und Kantenhärte kann die Wolkenform variiert werden. Links ist das Dichtefeld zentriert, also die Cloud-Map Intensität gleichmäßig nach und oben gewichtet ($o = 0$). In der Mitte ist der Offset $o < 0$ negativ gewählt, und sorgt für die wolkentypische, flache Unterseite. Rechts ist das Dichtefeld durch $s > 0$ vertikal nicht linear und erzeugt kräftigere, voluminösere Wolken.*

decken dargestellt werden. Im Vergleich zu Dubé (2005), welcher nur vom sichtbaren Wolkenfragment in Sonnenrichtung abtastet und dadurch sehr harte Wolkenkanten erzeugt, erlaubt die Erweiterung entlang des Sichtstrahls weiche, transparente Wolkenkanten und ermöglicht die Darstellung vieler unterschiedlicher Wolkentypen und -beschaffenheiten (Abbildung 4.18). Abbildung A.1 zeigt verschiedene Ergebnisse bei unterschiedlichen Abtast-Ansätzen und -Abständen.

Ein großes Problem stellt die Menge der benötigten Probes dar. Da aufgrund der Dynamik der Cloud-Map, der variablen Höhenlage der Wolkenschicht sowie der Höhe des Beobachters, ist keine Vorberechnung der Schnittpunkt mit den Sichtstrahlen möglich. Für eine gute Abtastung muss also stets vollständig entlang des Sichtstrahls, und jeweils vollständig in Richtung Sonne erfolgen (Abbildung A.1). Bei größeren Probe-Abständen zur Sonne können gleichzeitig Schatten durch entferntere Wolkenteile berücksichtigt werden (mit exponentieller Steigung der Probe-Abstände). Sehr hohe Wolken oder Wolkentürme können wegen zu geringer Vielfalt an den Wolkenwänden nicht abgebildet werden. Zudem führt die für eine glaubwürdige Darstellung benötigte und unerwartet hohe Anzahl an Texturzugriffen zu starken Leistungseinbrüchen. Da der größere Teil der Probes üblicherweise nicht im Dichtefeld liegt, wäre eine adaptive Anpassung über die Zenitdistanz nur geringfügig vorteilhaft.

Weiterhin müsste für Sonnenpositionen unterhalb des Horizonts mit eventueller rötlicher Beleuchtung der Wolkenunterseiten sowie Okklusion durch die Erde mit berücksichtigt werden. Ein weiteres Problem: Je kleiner der Winkel zwischen Blickrichtung und Sonne ist, desto dichter liegen die Probes beieinander. Hier sollte adaptiv die Anzahl der Probes verringert werden, indem die Abstände in Blickrichtung mit kleiner werdenden Winkel zur Sonnenrichtung zunehmen.

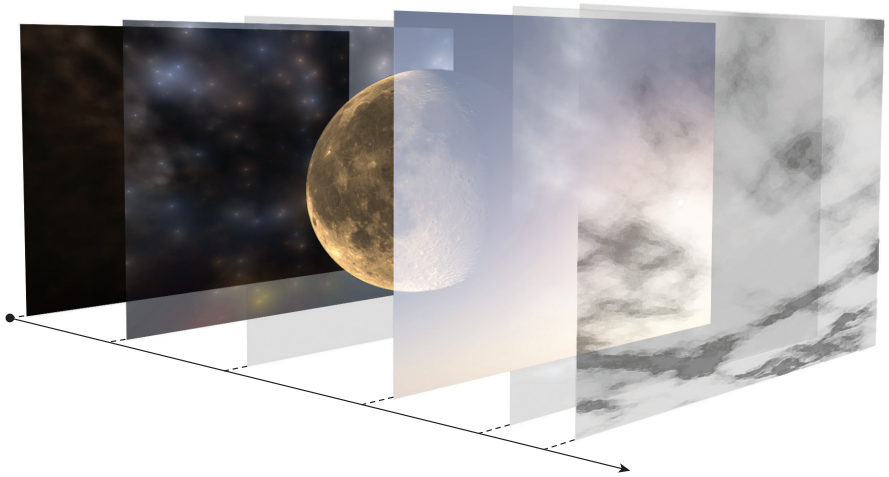


Abbildung 4.20: *Exemplarische Illustration der Kompositions Reihenfolge. Beginnend von hinten links: Star-Map, Helle Sterne, der Mond, Atmosphäre mit Sonne, und darüber beliebige Wolkenschichten.*

4.6 Komposition zu dynamischen Tag-Nacht-Zyklen

Um alle Phänomene innerhalb eines Rendering-Passes zu vereinen (komponieren), müssen diese in korrekter Reihenfolge zusammengeführt werden. Diese ergibt sich aus der Realen Entfernung der Phänomene selbst (Abbildung 4.20). Zu erst wird die opake Star-Map gerendert bzw. eine schwarzer Hintergrund, sofern die Star-Map nicht gewünscht ist. Helle Sterne werden darüber geblendet, wobei die Transparenz als Funktion über die Intensitäten pro Pixel bestimmt wird. Es folgt der Mond. Dieser ist undurchsichtig, und zwar auch wenn er nicht von der Sonne beleuchtet wird. Dies scheint offensichtlich, wird aber in vielen Anwendungen und Visualisierungen oft nicht berücksichtigt (Abbildung A.3). Schließlich wird die Atmosphäre über aller vorherigen Phänomene geblendet, sodass Sterne und Mond durch ihre Farbe beeinflusst werden. Wolken werden anschließend mit, nach der Atmosphäre gerendert und verblendet. Einige Beispiele unter unterschiedlichen Bedingungen sind in Abbildung 4.21 abgebildet.

Die Helligkeitsunterschiede innerhalb eines Tag-Nacht-Zyklus' über mehrere Magnituden, mehreren Größenordnungen (1×10^9) in der scheinbaren Helligkeit, führen zu nicht sichtbaren Sternen und stark attenuierten Mond. Da die Helligkeit des Mondes sowie die der Atmosphäre nicht über die scheinbare Helligkeit abgebildet ist, kann keine einheitliche Helligkeitskontrolle über die scheinbare Helligkeit erfolgen. Zudem wird die Atmosphäre in HDR gerendert, Sterne und Mond hingegen nicht.

osgHimmel skaliert die jeweiligen Phänomene mit Hilfe eines Butterworth-Tiefpassfilters B erster Ordnung (Butterworth, 1930):

$$B = a\sqrt{\frac{1}{1 + (s_z + \omega)^b}} + c, \tag{4.51}$$

mit s_z als normalisierte euklidische Altitude der Sonne (1 für Zenit am Tag, -1 entsprechend in der Nacht und 0 bei Sonnenuntergang und -aufgang), und a, ω, b, c als phänomenspezifische, durch Vergleiche mit Fotos zu entsprechenden Zeiten ermittelte Koeffizienten. a

Abbildung 4.21: *Kompositionen zu unterschiedlichen Tageszeiten und Bewölkungsgraden.*

und c legen den Helligkeitsbereich sowie die Mindesthelligkeit fest. Mit ω wird der Helligkeitsübergang auf den gewünschten Zeitraum verschoben. b legt schließlich die Dauer und damit die Heftigkeit der Transition fest. Für Sterne wird wurden $a = 1$, $\omega = 1.14$, $b = 32$, und $c = 0$gewählt. Für den Mond $a = 1/2$, $\omega = 1.05$, $b = 32$, und $c = 1/3$ gewählt.

Mittels dieser Methode lassen sich weitere Phänomene wie beispielsweise das Zodiakallicht oder Polarlichter (Magnor et al., 2010). Zu beachten ist dabei, dass aufgrund der Definition der Hintergrundkulisse und der daraus resultierenden Komposition keines der Phänomene die eigentliche Szene zu überlagern vermag. So können z.B. Wolken nicht mit großen Gebirgsketten oder Wolkenkratzern interagieren.

Bis auf die Atmosphären Simulation, werden alle Phänomene am wahren Horizont abgeschnitten. Für helle Sterne gilt dies nur bezüglich der einzelnen Vertices, nicht jedoch für die verwendeten PSF. Helle Sterne können folglich über den Horizont scheinen.

Kapitel 5

Umsetzung

Das System, *osgHimmel*, ist in Form einer Software-Bibliothek in C++ umgesetzt und unter der neuen BSD-Lizenz als Open-Source-Software frei verfügbar. Zur exemplarischen Einbindung in ein Rendering Sysetm wurde OpenSceneGraph 3 als bekanntes und komplexes Framework verwendet. Für den Editor (*skybox*) wurde Qt[1] 4 sowie den QtPropertyBrowser[2] verwendet. Das Software-Projekt ist plattformunabhängig mit CMake[3] aufgesetzt und über Google code[4] in einem SVN-Repository versioniert (Abschnitt A.1). Der Projektumfang beläuft sich zum Zeitpunkt der Fertigstellung dieser Arbeit auf ~ 20 000 Zeilen Code für die Bibliothek, bzw. weitere 6 000 unter Berücksichtigung der Beispiele und des Editors.

Im folgenden wird ein knapper Überblick der wichtigsten Klassen des Systems gegeben, ein typischer Szenengraph besprochen und auf spezielle Probleme bei der Umsetzung als Szenengraph diskutiert.

5.1 System Überblick

Eine Übersicht über die wichtigsten Bestandteile des Systems ist in Abbildung 5.1 gegeben. `AbstractHimmel` ist von `MatrixTransform` ableitet und verwaltet eine `TimeF` Zeitinstanz und sorgt für die korrekte Platzierung der Node nahe der Kamera (Abschnitt 5.2). `Himmel` und `AbstractMappedHimmel` sind von `AbstractHimmel` abgeleitet. `Himmel` dient der Aggregation der einzelnen simulationsbasierten, wahlweise aktivierten Phänomene und regelt deren Darstellungsreihenfolge. Wolkenschichten können dabei prinzipiell mehrfach eingebunden werden (noch nicht umgesetzt). Hier gilt es noch, den Phänomenen ein einheitliches Interface zu geben, und eine generische Aggregation zukünftiger Himmel Objekte zu erlauben. Die jeweiligen Phänomene leiten jeweils von `osg::Geode` bzw. `osg::Group` im Falle der Wolkenschichten ab.

`TimeF` bietet ein Interface zur Zeitsteuerung und muss für dynamische Hintergrundkulissen, dem Himmel zugewiesen werden. Die Steuerung erfolgt entweder automatisiert oder manuell, von anderen system-fremden Komponenten.

`AbstractAstronomie` trennt über eine Schnittstelle, allen benötigten astrophysikalischen Berechnungen von ihren Implementierungen. So sind in *osgHimmel* zum Zeitpunkt der Fertigstellung der Arbeit zwei unterschiedliche Funktionssammlungen verfügbar. `Astronomy`

[1] http://qt.nokia.com/products/
[2] http://qt.gitorious.org/qt-solutions/qt-solutions/trees/master/qtpropertybrowser
[3] http://www.cmake.org/
[4] http://code.google.com/

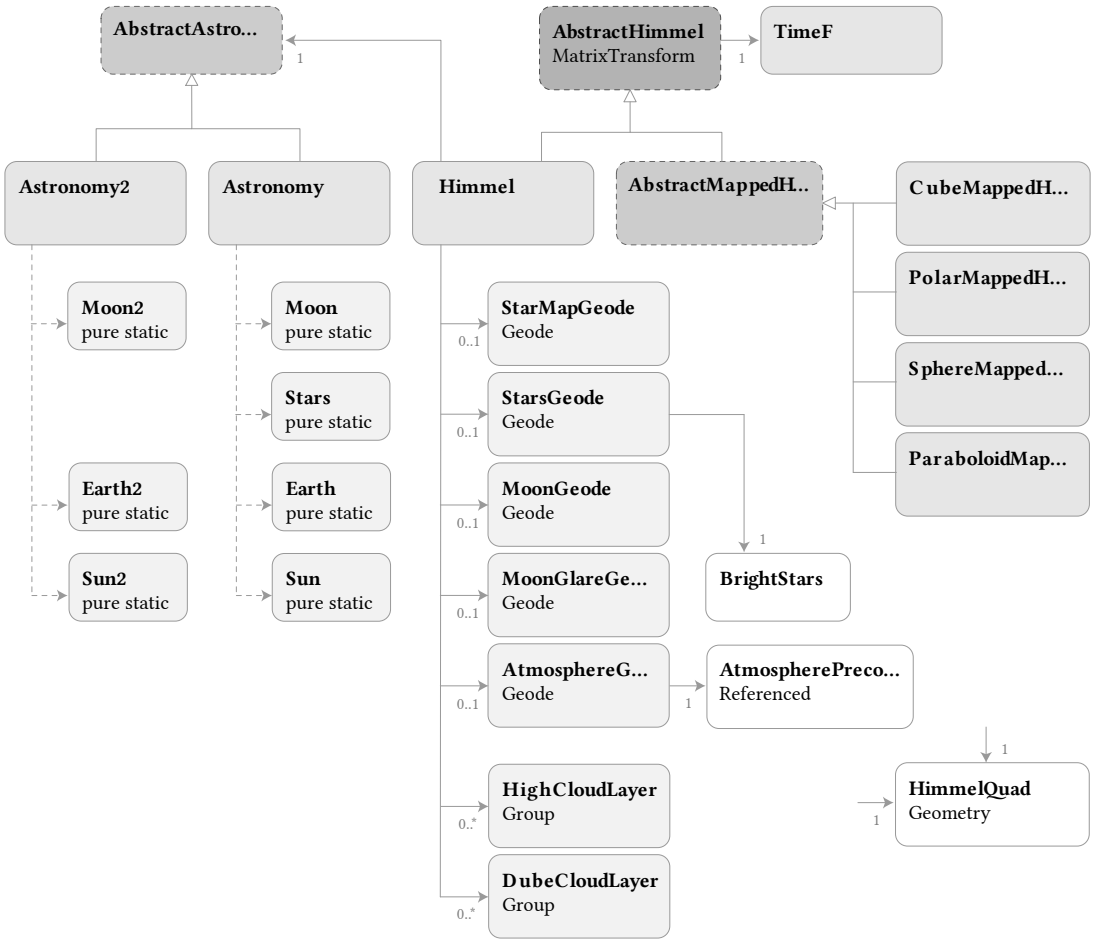


Abbildung 5.1: *Klassendiagramm der Kernbestandteile in* osgHimmel.

und `Astronomy2`. Sie unterscheiden sich ausschließlich in der Genauigkeit der Berechnungen, und dem damit verbundenen Rechenaufwand. Sofern keine weniger genaue Berechnung bekannt ist, greift `Astronomy2` auf `Astronomy` zurück. Wenigstens zwei weitere Astronomie Klassen sind angedacht: Zum einen soll die vollständige Kontrolle der Berechnungen geboten werden, indem die Ergebnisse zur Laufzeit beliebig definiert werden können (`ManualAstronomy`). Dies ist wichtig, um die Vorteile der simulationsbasierten Darstellung zu nutzen, ohne dabei ganzheitliche, astrophysikalische Korrektheit zu fordern. Zum anderen sollen die meisten der berechneten Ergebnisse zwischengespeichert werden, und wiederholte Berechnungen pro Frame zu vermeiden (`CachedAstronomy`). Die jeweiligen Funktionssammlungen (`Moon`, `Stars`, `Earth`, `Sun`, etc., ...) sind als Schnittstellen mit ausschließlich statischen Funktionen definiert. Die Klasse `BrightStars` enthält die benötigten Daten des erweiterten Yale Bright Star Catalogue (Müller, 2012), und ermöglicht diesen im binärer Form zu serialisieren und deserialisieren. `AtmospherePrecompute` dient der Vorberechnung der Atmosphären Texturen und basiert auf den frei verfügbaren Quellcode von Bruneton & Neyret (2008).

`AbstractMappedHimmel` stellt eine Schnittstelle zur zeit-bedingten Zuweisung einzelner Texturen für dynamische Transitionen zur Verfügung. Zudem übernimmt sie die Rotation um den Zenit und erlaubt eine maskierte Sonne einzubinden. Die jeweiligen Projektions-

Arten sind über die abgeleiteten Klassen `CubeMappedHimmel`, `PolarMappedHimmel`, `SphereMappedHimmel`, und `ParaboloidMappedHimmel` verfügbar. Diese vier Klassen, und nahezuh alle auf `Geode` endenden Klassen des simulationsbasierten Ansatzes, verwenden jeweils einen `HimmelQuad` als Projektionsfläche. Das genaue Zusammenspiel zwischen diesem Rechteck und `AbstractHimmel` wird in Abschnitt 5.2 beschrieben.

5.2 Besonderheiten bei der Umsetzung mit OpenSceneGraph

Die Anbindung an OpenSceneGraph war gedacht, um einer möglichst großen Zielgruppe, ein für sich abgeschlossenes, einfach einzubindendes System zur Erzeugung und Darstellung von Hintergrundkulissen zu bieten. Im Nachhinein scheint es, wäre eine Umsetzung in reinem OpenGL, ohne Szenengraphen, in vielerlei Hinsicht einfacher gewesen. So bietet OpenSceneGraph beispielsweise keine direkte, bzw. für einige Fälle sehr umständliche Kontrolle über zeitliche Abläufe (z.B. Wechselnde Zuweisung von Rendering Targets während des Renderings). Dieses und weitere Probleme sowie deren Lösungen bezüglich *osgHimmel* werden im folgenden kurz beschrieben.

Integration im Szenengraph

Um die Hintergrundkulissen von *osgHimmel* in OSG Szenen zu verwenden, genügt es, eine Instanz, wahlweise vom simulationsbasierten `Himmel` oder eine der texturbasierten Varianten `PolarMappedHimmel`, `SphereMappedHimmel`, `ParaboloidMappedHimmel`, und `CubeMappedHimmel` neben die eigentliche Szene zu hängen. Die Struktur des Himmel-Knotens (Abbildung 5.2) ist nahezu identisch mit der der Klassenhierarchie (Abbildung 5.1).

Culling beim Rendering des Screen-Aligned-Quads

Ein Screen-Aligned-Quad ist ein Rechteck, welche verwendet wird, um eine zweidimensionale Projektionsfläche im dreidimensionalen Raum zu erhalten. Üblicherweise wird diese zur Darstellung zuvor gerenderter Texturen benötigt, oder wie im Falle von *osgHimmel*, dazu verwendet, den Bildschirm vollständig abzutasten. Ein Problem bezüglich der Integration in OSG stellt die mehrfache Traversierung des Szenengraphen und die dabei vorgenommene, automatische Anpassung der *Near-* und *Far-Plane* – also der Sichtebenen der am dichtest- und am weitest-entfernten Objekte in der Szene. Die automatische Anpassung erlaubt eine optimale Ausnutzung des Tiefen-Puffers. Alle Objekte außerhalb der beiden Sichtebenen werden nicht in der Zeichnen-Traversierung (*Draw-Traversal*) berücksichtigt und die Near- und Far-Plane der Projektionsmatrix entsprechend angepasst. Zur Darstellung von Hintergrundkulissen bedeutet dies jedoch ein Problem für die in *osgHimmel* verwendete(n) Rechtecke, den *HimmelQuad*(s): befinden sich alle Eckpunkte eines HimmelQuads bzw. dessen umhüllende Box außerhalb der Sichtebenen, wird dieses nicht weiter verarbeitet. Der Hinweis an OSG, das Rechteck während des Draw-Traversal trotzallem zu berücksichtigen reicht nicht, da anschließend die Grafikhardware alle Vertices entsprechend aussortiert. Aus diesem Grund, wird die gesamte Node, und damit alle unter ihr befindlichen Rechtecke, während der Cull-Traversierung (*Cull-Traversal*) für die Erzeugung der Bounding-Box zur Kamera

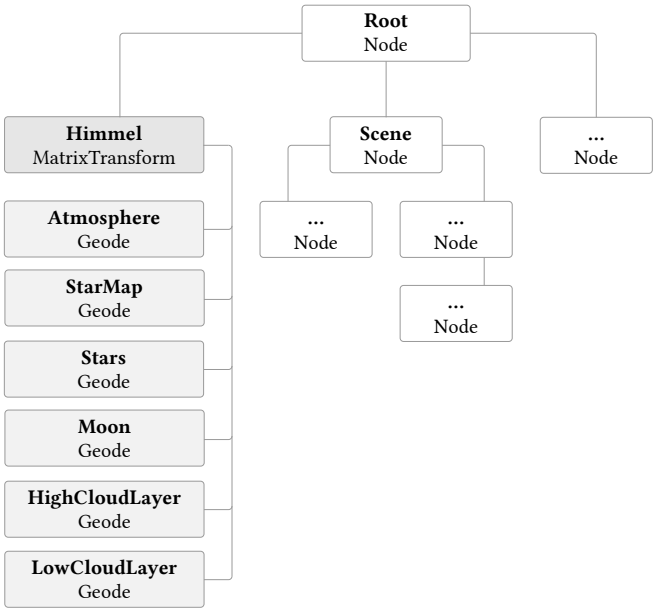


Abbildung 5.2: *Exemplarischer Szenengraph. Die* `MatrixTransform` *Node Himmel sollte neben, nicht unter der Scene Node eingehangen werden. Anstelle der Himmel Node können auch von* `AbstractMappedHimmel` *abgeleitete Instanzen eingesetzt werden (*`PolarMappedHimmel`*,* `SphereMappedHimmel`*,* `ParaboloidMappedHimmel`*,* `CubeMappedHimmel`*).*

hin verschoben, bzw. zum Kamerasichtpunkt (Abschnitt A.2). Für eine Nutzung als Projektionsfläche werden die einzelnen Eckpunkte in den jeweiligen Vertex-Shadern als Koordinaten des Bildschirm-Koordinaten-Raums interpretiert (Abschnitt A.2)und zur Ableitung des Sichtstrahls verwendet. Dieser Ablauf ist in Abbildung 5.3 grob skizziert.

Kontrolle durch Zeit und explizite Aktualisierung

Die Zeit jeder Hintergrundkulisse kann auf verschiedene Arten manipuliert und verwendet werden. Für raumzeitlich korrekte Visualisierungen können Zeitpunkte durch Uhrzeit, Datum und Zeitzone definiert werden. Zusammen mit der Vorgabe der Anzahl an Sekunden pro realer Sekunden, kann die Zeit entsprechend automatisch bei jedem Frame oder manuell aktualisert werden. Ist eine korrekte Abbildung der Zeit nicht relevant, oder deren Einstellung über Sekunden, Minuten, und Stunden unerwünscht, kann die Zeit als Fließkommazahl in $[0; 1]$ definiert werden. Dabei steht 0.0 bzw. 1.0 für Mitternacht und 0.5 für Mittag. Diese Angabe ist beispielsweise angemessen für texturbasierte Ansätze, da die einzelnen Texturen unabhängig von realen Tageszeiten definiert werden können. Bei automatischer Zeitaktualisierung, kann diese jeder Zeit pausiert oder zurückgesetzt werden, und bei Bedarf rückwärts ablaufen.

Atmosphären Rendering

Die Kodierung atmosphärischer Parameter in mehreren Texturen zur Atmosphärendarstellung nach Bruneton & Neyret (2008) is in einem Szenengraph nicht auf gleiche Weise wie in einer reinen OpenGL Implementierung vorzunehmen. Während des mehrstufigen Renderns

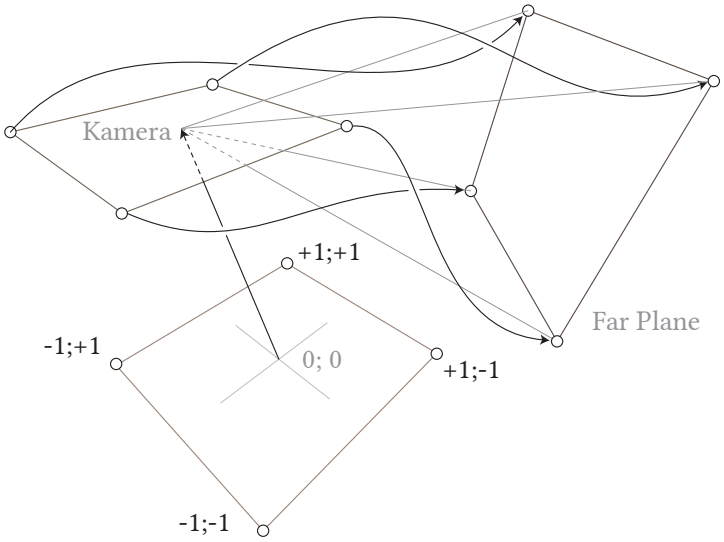


Abbildung 5.3: *Transformationsprozess vom Rechteck im Ursprung (unten), zum zur Kameraposition verschobenen Rechteck (links), hin zum Screen-Aligned-Quad auf der Far Plane der Kamera (rechts).*

der einzelnen Texturen wird häufig zwischen zwei Texturen als Rendering-Target dynamisch gewechselt. In OSG lässt sich dieses, als *Texture-Ping-Ponging* bekannte Verfahren, nicht direkt umsetzen, da jedes Render-Target fest an eine Kamera gebunden ist, und keine zeitliche Kontrolle währen des Draw-Traversals möglich ist. Im System ist daher das Texture-Ping-Ponging in Form eines Multi-Pass-Renderings abgekapselt und wird über mehrere Kameras abgewickelt. Eine exakte zeitliche Zuweisung der Rendering-Targets und Kontrolle des Renderingablaufs dabei möglich (`AtmospherePrecompute`).

Kapitel 6

Ergebnisse und Diskussion

osgHimmel wurde mit dem Fokus auf einfache Integrierbarkeit unter geringen Leistungseinbußen entwickelt. Im Folgenden werden die Hardwareanforderungen und Messungen der Performance-Auswirkungen vorgestellt. Es werden ausgewählte Vergleiche mit verwandten Arbeiten und Fotos gezeigt und bewertet. Die Problematik der stark variierenden Genauigkeiten einzelner Teilergebnisse werden kurz besprochen. Abschließend sind weitere, mögliche Anwendungsszenarien aufgeführt und die Benutzeroberfläche des system-internen Editors beschrieben.

6.1 Hardwareanforderungen und Leistungs-Messungen

Für das gesamte System wird OpenGL in der Version 2.1 empfohlen[1]. Eine Portierung für mobile Anwendungen mit OpenGL ES 2.0 bzw. WebGL Standard ist möglich [2]. Die texturbasierten Verfahren laufen selbst auf grafik-schwachen System, auch mit älteren integrierten Grafikkernen sehr schnell (mit Anpassungen auf Mobile Intel GM45 Express Chipset oder direkt auf Intel HD Graphics 3000/2000). Die Performance-Auswirkungen des simulationsbasierte Ansatzes sind ohne Wolken ebenfalls sehr gering. Die Darstellung der Wolken hingegen benötigt viele Texturzugriffe und hat selbst auf aktuellen Grafikkarten einen spürbaren Einfluss auf die Rendering-Geschwindigkeit der gesamten Szene. Die Vorberechnung der zur Atmosphärendarstellung benötigten Texturen erfolgt üblicherweise innerhalb einer Sekunde auf normalen Desktop-PCs (gemessen auf einem Intel Core2Duo 8400 mit einer NVidia GeForce 9800 GT), ist jedoch abhängig von der gewünschten Qualität der Streuungs-Annäherungen (Einfach- oder Mehrfachstreuung, sowie die jeweiligen Texturauflösungen). Änderungen bestimmter atmosphärischer Parameter wie beispielsweise die Dicke der Atmosphären oder wellenlängen-abhängige Streuungskoeffizienten zur Laufzeit sind zwar möglich, führen während der Interaktion jedoch zu kurzen, aber spürbaren Unterbrechungen.

Die Darstellung der hellen Sterne nutzt zur Zeit Geometrie-Shader zur Erzeugung der Billboards. Dies kann bei Bedarf auf der CPU erfolgen. Da die Vertices nur einmal an die GPU übertragen werden, erhöht sich nur der Speicherbedarf auf der GPU (da je vier statt ein Vertex benötigt werden), die Renderingzeiten hingegen nur unmerklich.

Für die folgende Leistungs-Messung verwendeten wir die genauen astronomischen Berechnungen ohne Zwischenspeicherung von Ergebnissen, sowie unoptimierten Shader Code. Bei einer Szene die aufgrund der komplexen Geometrie und umfangreichen Texturierung eine

[1] http://www.opengl.org/registry/doc/glspec21.20061201.pdf
[2] http://www.khronos.org/registry/gles/specs/2.0/es_cm_spec_2.0.24.pdf

Bildwiederholrate von knapp 60 Bilern pro Sekunde erreicht, beläuft sich der durchschnittliche Leistungs-Einfluss einer dynamischen, simulationsbasierten, wolkenfreien Hintergrundkulisse auf ungefähr 5%. Tabelle 6.1 listet die Renderingzeiten der einzelnen Phänomene auf.

Phänomen	Konfiguration	# Vertices	Zeit in μs
Star-Map	$m_a = 6.0$	4	223
Star-Map	$m_a = 8.0$	4	222
Helle Sterne	$m_a = 6.0$	$4 \times 9\,129$	61
Helle Sterne	$m_a = 8.0$	$4 \times 9\,129$	447
Mond	$c_d = 2.0$	4	4
Mond	$c_d = 100.0$	4	167
Atmosphäre		4	593
Atmosphäre	mit Dithering	4	728
Hohe Wolken (2D)	$c = 0.5, s = 0.5$	4	23 032
Niedrige Wolken (3D)	$n = 128, m = 32, c = 0.33, s = 0.5$	4	48 047

Tabelle 6.1: *Aufgelistet sind die durchschnittlichen Zeitdifferenzen pro Frame bezüglich des Renderings einer leeren Szene, gemessen über eine Minute. Gemessen mit einem Intel Core2 Duo E8400 mit 3.0GHz, 8.0GB Arbeitspeicher und einer NVidia GeForce GTX 460 mit 1.0GB Speicher.*

Der Unterschied zwischen der Renderingzeit der Star-Map zu der heller Sterne resultiert aus der Menge an verworfenen Sternen in der Geometriestufe – die Menge an zu prozessierenden Fragmenten liegt dabei in den meisten Fällen unter der gesamt Anazahl an verfügbaren Bildschirmfragmenten. Den stärksten Einfluss auf die Renderingzeit der hellen Sterne hat folglich die Kontroll-Magnitude m_a, da diese die Größe der Glare-PSF und damit die Anzahl der zur verarbeitenden Fragmente pro Stern stark beinflusst.

6.2 Genauigkeit der Algorithmen

Ein wesentliches Problem, welches sich über den gesamten simulationsbasierten Ansatz in dieser Arbeit zieht, ist die teilweise stark variierende Genauigkeit in den einzelnen Berechnungen und der resultierenden Darstellung. Für atmosphärische Berechnungen haben wir uns für mittlere, standardisierte Eigenschaften der Luft entschieden (Abschnitt 4.4). Zudem ist die Wahl der repräsentativen Wellenlängen für die drei Farbkanäle willkürlich. Bucholtz (1995) und andere Arbeiten, zeigen wie sich Veränderungen der vielfältigen Eigenschaften der Luft (wie beispielsweise Dichte und Feuchtigkeit) sich auf die Streuung auswirken. Da die resultierende Darstellung jedoch einem beliebigen HDR-Mapping, einer beliebigen Attenuation, sowie bei der Darstellung auf einem Ausgabegerät schließlich benutzerdefinierter Farb- und Gammakorrektur unterliegt, ist die in *osgHimmel* verwendete Genauigkeit der Farbberechnungen hinreichend. Ähnliches gilt für astronomische Berechnungen. Hier sind selbst die Berechnungen der weniger genauen `Astronomy2` Funktionssammlung für eine Darstellung auf modernen Bildschirmen mit hohen PPI-Dichten noch zu genau, bzw. die Ungenauigkeiten, Abweichungen und Fehler nicht wahrnehmbar. So sind die optischen Librationen des Mondes, sowie die Feinheiten der Mondoberfläche und der des Tag-Nacht-Terminators bei typischen

Abbildung 6.1: *Variationen in der Darstellung astrophysikalischer Phänomene am Beispiel der totalen Mondfinsternis am 21. Dezember 2010, beobachtet aus New York um ungefähr 7:40 Uhr.. Yapo & Cutler (2009) (a), zwei Fotos mit unterschiedlicher Blende und Belichtung (b und c), sowie das Ergebnis der vorgestellten Methode (d). Zu erkennen ist, dass durch die vielen Parameter bei der Aufnahme sowie der Komplexität unserer Wahrnehmung keine einheitliche Wahrnehmung der Phänomene gegeben ist. Eine eineindeutige, korrekte Abbildung astrophysikalischer Ereignisse im Bildraum scheint also nicht möglich und muss stets an subjektiven Erwartungen gemessen und ausgerichtet werden.*

Größenskalierungen nicht bemerkbar. Erst bei hochauflösenden Nahaufnahmen werden diese, teils nur im Zeitraffer, erkennbar. Auch die durch atmosphärische Refraktion eingeführte Farbveränderungen sind selbst im direkten Vergleich, unter Abwesenheit jeglicher Szenengeometrie kaum erkennbar.

osgHimmel versucht daher, alle Phänomene möglichst konsistent abzubilden und legt letztlich bei der Komposition weniger Gewicht auf astrophysikalische Korrektheit, als vielmehr auf eine angenehme, mit subjektiven Erfahrungen verträgliche Darstellung.

Die synthetischen Ergebnisse von *osgHimmel* sind für die spezifizierten Anwendungsfälle ausreichend realistisch, und mit ausgewählten Fotos vergleichbar (Abbildung 4.4, Abbildung 6.1). Der Realitätsgrad obliegt schließlich stets dem Anwender und die Glaubwürdigkeit einer Szene ist hauptsächlich von der kohärenten Einbettung der Hintergrundkulissen in die gesamte Szene abhängig.

6.3 Editor zum System - Skybox

Der während der Entwicklung des Systems entstandene Editor (*skybox*) erlaubt alle Parameter der jeweiligen `AbstractHimmel` Instanzen bzw. die untergeordneter Geoden zur Laufzeit zu manipulieren. Zudem bietet er die Möglichkeit zur genauen Kontrolle der raumzeitlichen Bedingungen wie beispielsweise Uhrzeit, Zeitzone, Ort, und Altitude. Das wichtigste Werkzeug, stellt der Shader-Editor dar, der es erlaubt, alle aktiven Shader nicht nur zu lesen, sondern auch zur Laufzeit umzuschreiben und anschließend rekompilieren. In Abbildung 6.2 ist eine typische Konfiguration der Benutzeroberfläche zu sehen.

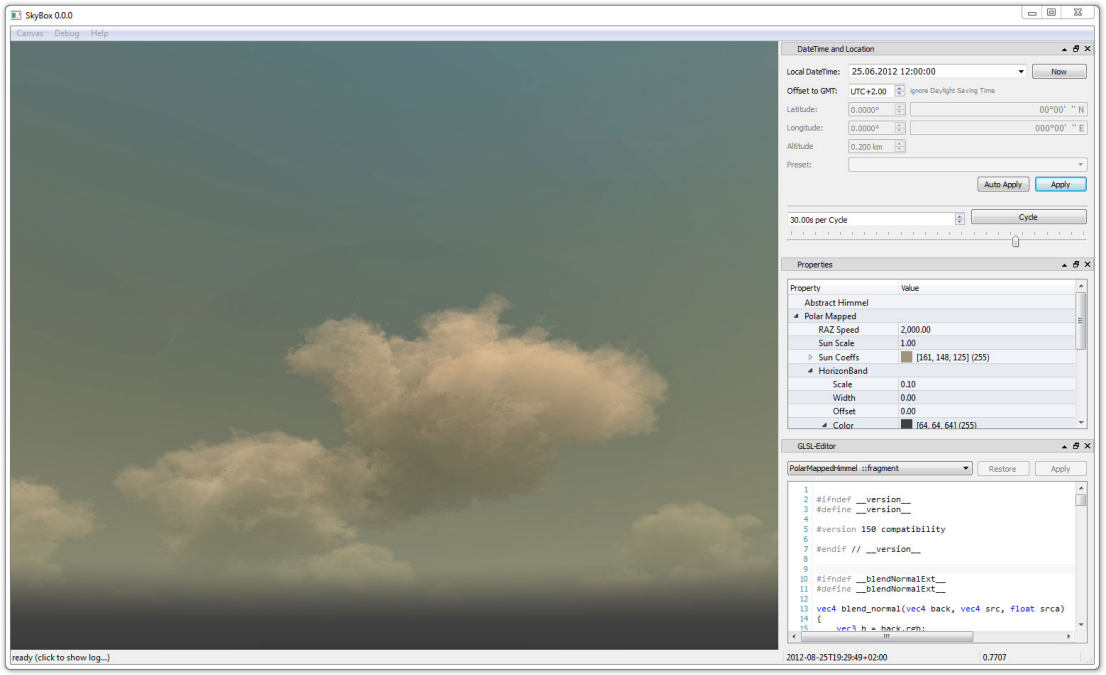


Abbildung 6.2: *Skybox: Benutzeroberfläche des Editors für* osgHimmel

Kapitel 7

Fazit und Ausblick

In dieser Arbeit wurde das System *osgHimmel* zur Erzeugung und Darstellung von Hintergrundkulissen vorgestellt. Dabei wurde ein texturbasierter und ein simulationsbasierter Ansatz verfolgt. Beide Ansätze bieten zeitbedingte Dynamik für vielfältige Anwendungsszenarien. *osgHimmel* ist als Software-Bibliothek offen und mit allen benötigten Referenzen und Ressourcen frei verfügbar. Es erlaubt durch ihre übersichtliche Architektur und der nichtinvasiven Integration des Szenengraphen, eine leichte Handhabung und Integration in existierenden, auf OSG basierenden Systemen. Hintergrundkulissen beider Ansätze können über enthaltene Hilfsfunktionen in Form einer beliebig hoch auflösenden Cube-Map zur weiteren Verarbeitung wie globaler Beleuchtung oder Reflektion genutzt werden. Im folgenden werden die wesentlichen, vorgestellten Merkmale beider Ansätze stichpunktartig aufgeführt:

Vorteile des texturbasierten Ansatzes

- Sehr geringer Rechen- bzw. Renderingaufwand.
- Bis auf das Screen-Aligned-Quad ist keine weitere Szenengeometrie nötig.
- Hoher gestalterischen Freiraum.
- Häufig genutzte Projektionsarten werden unterstützt: Polar-Mapping, Shere-Mapping, Paraboloid-Mapping, und Cube-Mapping (mit optionaler, optimierter Texeldichte-Verteilung).
- Zeitbedingte Textur-Transitionen.
- Mehr Dynamik durch Rotation um den Zenit, Horizontband, und maskierte Sonne.
- Vielfältige Sammlung an Bildmaterial (ausschließlich aus frei verfügbaren Quellen).

Vorteile des simulationsbasierten Ansatzes

- Auf astronomoischen Berechnungen basierte, dynamische Darstellung von Sterne, Sonne und Mond.
- Glaubwürdige Atmosphäre mittels Mie- und Rayleigh Streuung.
- Limitiert für bodennahe Beobachter auf der Erde (bei Bewölkung unterhalb der ersten Wolkenschicht).

- Raumzeitliche (Datum, Uhrzeit, Ort, Höhe über Meeresspiegel) Korrektheit bezüglich astronomischer Positionen und Konstellationen.
- Helle Sterne mit großem Spielraum zur gewünschten Darstellung.
- Berücksichtigung von Streuung, Refraktion, und der Blauen Stunde des Ozons.
- Neues Verfahren zur Bildsynthese von Mondfinsternissen.
- Fokus auf vollständige Tag-Nach-Zyklen.
- Gut geeignet für Postprocessing.
- Vollständig prozedural erzeugte 2D und 3D Wolkenschichten mit angenäherter Streuung im 3D Fall.

Probleme und Ideen

Bezüglich der Komposition des simulationsbasierten Ansatzes wäre eine einheitliche Abbildung der Intensitäten aller Phänomene zur Verblendung und Weiterverarbeitung enorm hilfreich. Hier wäre die scheinbare Helligkeit der außer-atmosphärischen Phänomene oder Leuchtdichte und Leuchtstärke sinnvoll (Theisgen, 2011; Debevec et al., 2004). Zudem sollte, vor allem in der Nacht ein Tone-Mapping zur Berücksichtigung der veränderten Helligkeits- und Farbwahrnehmung (z.B.: Skotopisches Stäbchen Sehen) vorgenommen werden. Um Farbbändereffekte zu vermeiden, wird in *osgHimmel* optional ein leichtes Dithering verwendet.

Die bisher weniger berücksichtigten Wolken erlauben umfangreiche Verbesserung, wie beispielsweise automatische Farbgebung, bessere Streuungsannäherung, und Generation und Transformation (z.B.: Fraktale, Strömungsimulationen) sowie einer Erweiterung hin zu Wolkentürmen. Für die Simulation von Luft- und Farbperspektive wären weitere Shader zur Berechnung der Streuung über die Entfernung sinnvoll. Diese könnten aus den vorhandenen, vorberechneten Texturdaten stammen oder vereinfacht angenähert werden.

Zur Beleuchtung der sonstigen Szene auf Grundlage der Erscheinung der Hintergrundkulisse wurde in dieser Arbeit nicht eingegangen. Die Positionen der Leuchtquellen im simulationsbasierten Ansatz sind verfügbar, nicht jedoch deren Farben und Intensitäten. Wolken sollten zudem einen Schatten auf die Szene werfen (bei den vorgestellten Ansätzen wäre dies durch Projektion der Noise-Map bzw. des Dichtefelds möglich) und mit Objekte der Szene interferieren können. Die Farben könnten aus der Hintergrundkulisse extrapoliert oder über astrophysikalische Annäherungen ermittelt werden.

Weitere Ideen bestehen in der Erweiterung der Darstellung der hellen Sterne um Planeten sowie Satelliten, Wetterballons und Kometen, da auch diese von erdnahen Beobachtern mit bloßem Auge gesichtet werden können. Des weiteren erlauben die vorgestellten Ansätze zur Wolkensimulation und die Parametrisierung der Atmosphäre und Komposition eine wetterabhängige Anpassung mittels entsprechender Voreinstellungen (Presets). Darauf aufbauend könnte schließlich das Wetter und damit die Darstellung der gesamten Hintergrundkulisse anhand lokaler (Echtzeit-)Wetter-Daten (Head, 2011) angepasst werden.

Alle Probleme, Fehler, Verbesserungsvorschläge und Ideen werden im Versionierungs-System dokumentiert (Abschnitt A.1). Es sei jedoch kurz auf die wichtigsten Probleme, offene Fragen und Erweiterungsmöglichkeiten hingewiesen.

Schlussworte

Mit *osgHimmel* steht der OSG Gemeinschaft prinzipiell ein flexibles System zur Erzeugung vielfältiger, hochwertiger und glaubwürdiger Hintergrundkulissen bereit. Über diese Arbeit hinaus gilt es, dass System zu pflegen und die vorhandenen Probleme soweit wie möglich zu beseitigen. Für ein Fortbestehen des Systems ist es allem voran wichtig, echte Anwendungsfälle und Nutzer zu finden und damit die Fähigkeiten des Systems zu validieren und an den entstehenden, zukünftigen Anforderungen zu wachsen.

Anhang

A.1 Ergänzende Abbildungen, Tabellen und Ressourcen

Koefzienten der Polynome zur Bestimmung der Farbartkoordinaten

i	3	2	1	0
$c_i^{x_0}$	−0.26612390	−0.23435800	+0.87769560	+0.17991000
$c_i^{x_1}$	−0.30258469	+0.21070379	+0.22263470	+0.24039000
$c_i^{y_0}$	−1.10638140	−1.34811020	+2.18555832	−0.20219683
$c_i^{y_1}$	−0.95494760	−1.37418593	+2.09137015	−0.16748867
$c_i^{y_2}$	+3.08175800	−5.87338670	+3.75112997	−0.37001483

Tabelle A.1: *Koeffizieten zur Bestimmung der Farbartkoordinaten.*

Tief liegende 3D Wolkenschicht bei variierenden Parametern

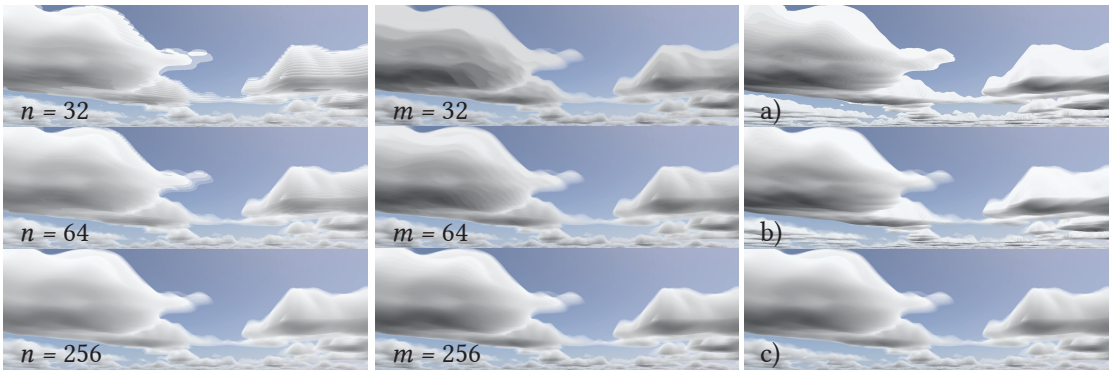


Abbildung A.1: *3D Wolkenschicht bei variierenden Parametern. Links und Mitte: Unterschiedliche Probe Anzahl in den jeweiligen Abtastrichtungen. Rechts: a): Dubé variante, nur für die erste Probe, die innerhalb des Dichtefeldes ist, werden weitere Probes in Richtung Sonne vorgenommen. b): Wie a), nur werden weitere Probes in Blickrichtung gezählt. c): Probes in Blickrichtung und für jede Probe innerhalb des Dichtefeldes auch in Richtung Sonne.*

Refraktionsbasierte Mond Deformation

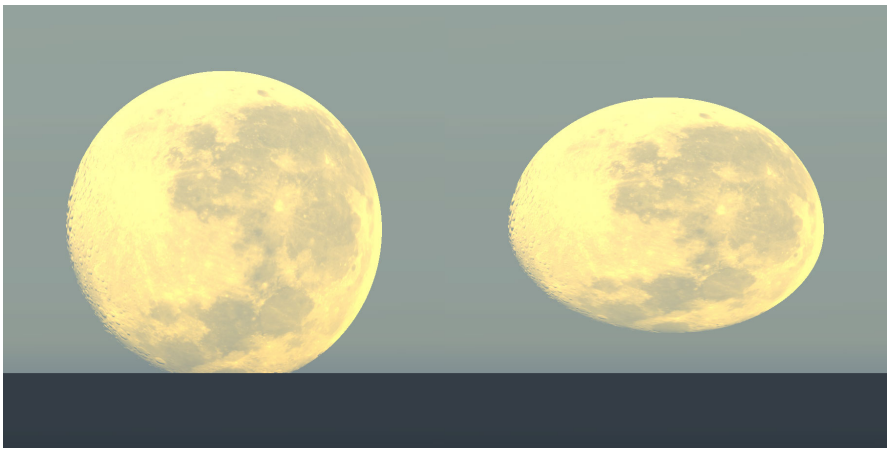


Abbildung A.2: *Nahaufnahme einer Per-Fragment Deformation der Mondscheibe. Links: Kreisrunder Mond. Rechts: Test des Deformations-Ansatzes bei hoher Refraktion nahe des Horizonts.*

Beispiel für fehlerhaftes Verblenden des Mondes

Abbildung A.3: *Dieses ältere Thema von Dreamworks zeigt einen Mond, dessen dunkle Seite durchsichtig ist und die dahinterliegenden Sterne zeigt. Dies ist ein sehr häufiger Fehler, der selbst in aufwändigen Film-Produktionen auftaucht. Dreamworks hat dies inzwischen korrigiert und den Mond vollständig undurchsichtig dargestellt.*

Projekthosting und Resourcen

osgHimmel ist auf Google code versioniert und verfügbar unter `http://code.google.com/p/osghimmel/`. Diese und weiterführende Arbeiten und Referenzen, Videos und Abbildungen zu den einzelnen Fähigkeiten des Systems, ein Wiki zur Dokumentation des Build-Prozess sowie einzelner Verfahren und deren Verwendung (Wiki), und schließlich einem Issue-Tracker (Issues) sind ebenfalls angelegt. Links zu ähnlichen kommerziellen oder freien Systemen sind ebenfalls aufgeführt. Alle verwendeten und darüber hinaus brauchbaren Resourcen wie Texturen, Daten, oder 3D Modelle stehen als Download zur Verfügung (Downloads).

A.2 Quellcode Ausschnitte

Algorithmus zur Abbildung einer Kugel auf einen Würfel

Der folgende Algorithmus zur Abbildung einer Kugel auf einen Würfel basiert auf den von Peterson (2010) vorgestellten, nicht trivialen Algorithmus. Für die Verwendung in *osgHimmel* wurde der Algorithmus stark optimiert.

```
const float isqrt2 = 0.70710678118654752440084436210485;

vec3 cubify(const in vec3 s)
{
    float xx2 = s.x * s.x * 2.0;
    float yy2 = s.y * s.y * 2.0;

    vec2 v = vec2(xx2 - yy2, yy2 - xx2);

    float ii = v.y - 3.0;
    ii *= ii;

    float isqrt = -sqrt(ii - 12.0 * xx2) + 3.0;

    v = sqrt(v + isqrt);
    v *= isqrt2;

    return sign(s) * vec3(v, 1.0);
}

vec3 sphere2cube(const in vec3 sphere)
{
    vec3 f = abs(sphere);

    bool a = f.y >= f.x && f.y >= f.z;
    bool b = f.x >= f.z;

    return a ? cubify(sphere.xzy).xzy : b ? cubify(sphere.yzx).zxy : cubify(sphere);
}
```

Listing A.1: *Quellcode der sphere2cube Funktion, welche im Fragment-Shader des Cube-Mappings zur Gleichverteilung der Texeldichte verwendet wird.*

Vorberechnung der Farbmanipulation für Mondfinsternisse

```
// generate lunar eclipse 2d-texture

const int sizeS = 128;

float *map = new float[sizeS * 3];

const osg::Vec3 le0 = osg::Vec3(1.0, 1.0, 1.0) * 0.900f;
const osg::Vec3 le1 = osg::Vec3(1.0, 1.0, 1.0) * 0.088f;
const osg::Vec3 le2 = osg::Vec3(0.4, 0.7, 1.0) * 0.023f;
const osg::Vec3 le3 = osg::Vec3(0.3, 0.5, 1.0) * 0.040f;
```

```
11
12 const float s_u = 0.05;
13
14 for(int s = 0; s < sizeS; ++s)
15 {
16     const float fs = static_cast<float>(s) / sizeS;
17     const unsigned int i = s * 3;
18
19     osg::Vec3 l = osg::Vec3(1, 1, 1);
20
21     // remove the penumbral soft shadow from the moons coloring
22     l -= le0 * (1.0 - _clamp(0.0, 1.0, 2 * fs - 1));
23     // remove the umbral hard shadow from the moons coloring
24     l -= le1 * (1 - _smoothstep_ext(fs, 0.5 * (1.0 - s_u), 0.5 * (1.0 + s_u)));
25     // add reddish darkening towards umbral center from atmosphere scattering,
26     // linear scaling within the umbral distance of e1
27     l -= le2 * (fs < 0.5 ? _clamp(0, 1, - 2.0 * fs + 1) : 0);
28     // account for blue scattered light visible at the umbrals outer edge
29     l += le3 * (_smoothstep_ext(fs, 0.5 * (1 - 4 * s_u), 0.5 * (1 + s_u)))
30              * (1 - _smoothstep_ext(fs, 0.5, 1.0));
31
32     map[i + 0] = l[0] < 0 ? 0 : l[0];
33     map[i + 1] = l[1] < 0 ? 0 : l[1];
34     map[i + 2] = l[2] < 0 ? 0 : l[2];
35 }
```

Listing A.2: *Quellcode zur Vorberechnung der Farbmultiplikatoren $h(s_f)$.*

Umgehen des Cullings für das Screen-Aligned-Quad

Hinweis: In OSG macht es kaum einen Unterschied, ob das Rechteck zur Kameraposition oder deren Sichtposition verschoben wird, da die meisten Navigationen, den Sichtpunkt normalisieren und dieser somit nahe der Kameraposition liegt.

```
1  void AbstractHimmel::traverse(osg::NodeVisitor &nv)
2  {
3      osgUtil::CullVisitor* cv = dynamic_cast<osgUtil::CullVisitor*>(&nv);
4      if(!cv)
5          return;
6
7      const osg::CullSettings::ComputeNearFarMode cnfm(
8          cv->getCurrentCamera()->getComputeNearFarMode());
9      cv->getCurrentCamera()->setComputeNearFarMode(
10         osg::Camera::DO_NOT_COMPUTE_NEAR_FAR);
11
12     osg::MatrixTransform::traverse(nv);
13
14     cv->getCurrentCamera()->setComputeNearFarMode(cnfm);
15 }
16
17
18 bool AbstractHimmel::computeLocalToWorldMatrix(
19     osg::Matrix& matrix, osg::NodeVisitor* nv) const
```

```
{
    osgUtil::CullVisitor* cv = dynamic_cast<osgUtil::CullVisitor*>(nv);
    if(!cv)
        return false;

    const osg::Vec3 t(cv->getEyePoint());
    matrix.preMultTranslate(t);

    return true;
}


bool AbstractHimmel::computeWorldToLocalMatrix(
    osg::Matrix& matrix, osg::NodeVisitor* nv) const
{
    osgUtil::CullVisitor* cv = dynamic_cast<osgUtil::CullVisitor*>(nv);
    if(!cv)
        return false;

    const osg::Vec3 t(cv->getEyePoint());
    matrix.preMultTranslate(-t);

    return true;
}


void AbstractHimmel::setupNode(osg::StateSet* stateSet)
{
    setCullingActive(false);

    ...
}
```

Listing A.3: *Quellcode des Anti-Cullings der* `AbstractHimmel` *Node.*

Screen-Aligned-Quad als zweidimensionale Projektionsfläche

Die Koordinaten der Eckpunkte des Rechtecks werden direkt als Bildschirmkoordinaten verwendet. Der Sichtstrahl eines jeden Pixels, ist aus der Interpolation der Sichtstrahlen der Eckpunkte gegeben.

```
...

vec4 quadRetrieveRay()
{
    return gl_ProjectionMatrixInverse * gl_Vertex * gl_ModelViewMatrix;
}

void quadTransform()
{
    gl_Position = gl_Vertex;
}

```

```
...

out vec4 m_ray;

void main(void)
{
    gl_TexCoord[0] = gl_Vertex * 0.5 + 0.5;

    m_ray = quadRetrieveRay();
    quadTransform();
}

...
```

Listing A.4: *Quellcode zur Nutzung des Screen-Aligned-Quads im Vertex-Shaders.*

A.3 Anwendungsbeisiele

Beispiel zur Verwendung von Textur Transitionen

```
osg::ref_ptr<PolarMappedHimmel> himmel
    = new PolarMappedHimmel(PolarMappedHimmel::MM_Half);

himmel->assignTime(timef);
himmel->setTransitionDuration(0.4f);

himmel->getOrCreateTexture2D(0)->setImage(
    osgDB::readImageFile("resources/polar_half_art_1.jpg"));
himmel->getOrCreateTexture2D(1)->setImage(
    osgDB::readImageFile("resources/polar_half_art_2.jpg"));

himmel->pushTextureUnit(0, 0.0f);
himmel->pushTextureUnit(1, 0.5f);
```

Listing A.5: *Erzeugen einer Textur-Transition mit den Texturen `polar_half_art_1.jpg` und `polar_half_art_2.jpg`.*

Beispiel zur Verwendung des Horizont Bands

```
// The second parameter activates the horizon band.
osg::ref_ptr<PolarMappedHimmel> himmel
    = new PolarMappedHimmel(PolarMappedHimmel::MM_Half, true);

himmel->assignTime(timef);

himmel->hBand()->setBottomColor(osg::Vec4(0.50f, 0.50f, 0.50f, 1.00f));
himmel->hBand()->setColor(      osg::Vec4(0.74f, 0.80f, 0.83f, 1.00f));

himmel->hBand()->setOffset(0.1f);
himmel->hBand()->setScale( 0.7f);
himmel->hBand()->setWidth( 0.2f);
```

Listing A.6: *Exemplarische Konfiguration des Horizontbandes.*

Beispiel zur Verwendung der Rotation um den Zenit

```
osg::ref_ptr<PolarMappedHimmel> himmel
    = new PolarMappedHimmel(PolarMappedHimmel::MM_Half);

himmel->assignTime(timef);

// reasonable values should be around 2000+
himmel->setSecondsPerRAZ(2000.f);
himmel->setRazDirection(AbstractMappedHimmel::RD_NorthWestSouthEast);
```

Listing A.7: *Nutzung der Schnitstelle zur Beschreibung der Rotation um den Zenit.*

Abbildungsverzeichnis

Literaturverzeichnis

Blinn, J. F. (1978). Simulation of wrinkled surfaces. In *Proceedings of the 5th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '78 (pp. 286–292). New York, NY, USA: ACM.

Blinn, J. F. & Newell, M. E. (1976). Texture and reflection in computer generated images. *Commun. ACM*, 19, 542–547.

Bouthors, A., Neyret, F., Max, N., Bruneton, E., & Crassin, C. (2008). Interactive multiple anisotropic scattering in clouds. In *ACM Symposium on Interactive 3D Graphics and Games (I3D)*.

Braun, H. & Cohen, M. (2011). A simple model for real time sky rendering. Ubisoft/PUCRS game dev. specialization, http://www.henrybraun.info/files/skyrendering.pdf.

Brickman, N. (2007). Shapes in the clouds: Interactive, artist-guided, cloud simulation.

Bruneton, E. & Neyret, F. (2008). Precomputed atmospheric scattering. *Comput. Graph. Forum*, 27(4), 1079–1086. Special Issue: Proceedings of the 19th Eurographics Symposium on Rendering 2008.

Bucholtz, A. (1995). Rayleigh-scattering calculations for the terrestrial atmosphere. *Applied Optics*, 34, 2765–2773.

Buratti, B. J., Hillier, J. K., & Wang, M. (1996). The lunar opposition surge: Observations by clementine. *icarus*, 124, 490–499.

Butterworth, S. (1930). On the theory of filter amplifiers. *Wireless Engineer*, 7.

Celestia (2011). The celestia motherlode. http://www.celestiamotherlode.net/catalog/moon.php.

Chandrasekhar, S. (1960). *Radiative Transfer*. Dover Books on Physics. Dover Publications.

Danjon, A.-L. (1920). Comptes rendues.

Debevec, P., Reinhard, E., Ward, G., & Pattanaik, S. (2004). High dynamic range imaging. In *ACM SIGGRAPH 2004 Course Notes*, SIGGRAPH '04 New York, NY, USA: ACM.

Dobashi, Y., Nishita, T., & Okita, T. (1998). Animation of clouds using cellular automaton. In *Proc. of Computer Graphics and Imaging '98* (pp. 251–256).

Dobashi, Y., Yamamoto, T., & Nishita, T. (2002). Interactive rendering of atmospheric scattering effects using graphics hardware.

Dubé, J.-F. (2005). Realistic cloud rendering on modern gpus. In K. Pallister (Ed.), *Game Programming Gems 5* (pp. 499–505). Charles River Media.

Ebert, D. (2003). *Texturing & Modeling: A Procedural Approach*. Morgan Kaufmann Series in Computer Graphics and Geometric Modeling. Morgan Kaufmann.

Ellison, M. W. (1952). Why do stars twinkle? *Irish Astronomical Journal*, 2, 5.

Green, C. (2007). Efficient self-shadowed radiosity normal mapping. In *ACM SIGGRAPH 2007 courses*, SIGGRAPH '07 (pp. 1–8). New York, NY, USA: ACM.

Greene, N. (1986). Environment mapping and other applications of world projections. *IEEE Computer Graphics and Applications*, 6, 21–29.

Haber, J., Magnor, M., & Seidel, H.-P. (2005). Physically-based simulation of twilight phenomena. *ACM Trans. Graph.*, 24(4), 1353–1373.

Hapke, B. (1963). A theoretical photometric function of the lunar surface. *Journal of Geophysical Research*, 68, 4571–4586.

Hapke, B. (1966). An improved theoretical lunar photometric function. *Astronomical Journal*, 71, 333.

Hapke, B. (1971). Optical properties of the lunar surface. In *Physics and Astronomy of the Moon*: Academic Press.

Harris, M. J. & Lastra, A. (2001). Real-time cloud rendering. In *MUH*: Eurographics Association.

Hasan, M. M., Karim, M. S., & Ahmed, E. (2005). Generating and rendering procedural clouds in real time on programmable 3d graphics hardware. In *9th International Multitopic Conference, IEEE INMIC 2005*: IEEE.

Head, A. (2011). 3d weather: towards a real-time 3d simulation of localised weather. In *Proceedings of the 2011 international conference on Electronic Visualisation and the Arts*, EVA'11 (pp. 35–41). Swinton, UK, UK: British Computer Society.

Heidrich, W. & Seidel, H.-P. (1998). View-independent environment maps. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS workshop on Graphics hardware*, HWWS '98 (pp. 39–ff.). New York, NY, USA: ACM.

Henyey, L. G. & Greenstein, J. L. (1941). Diffuse radiation in the galaxy. *Astrophysical Journal*, 93, 70–83.

Hoeppe, G. (2007). *Why the sky is blue: discovering the color of life*. Princeton University Press.

Hoffleit, D. & Warren, Jr., W. H. (1995). Bright star catalogue, 5th revised ed. (hoffleit+, 1991). *VizieR Online Data Catalog*, 5050.

Hulburt, E. O. (1953). Explanation of the brightness and color of the sky, particularly the twilight sky. *Optical Society of America*, 43(2).

Hunt, R. W. G. (1987). *Measuring Colour*. Ellis Horwood Ltd.

Isidoro, J. R. & Mitchell, J. L. (2005). Angular extent filtering with edge fixup for seamless cubemap filtering. In *ACM SIGGRAPH 2005 Sketches*, SIGGRAPH '05 New York, NY, USA: ACM.

Jensen, H. W., Durand, F., Dorsey, J., Stark, M. M., Shirley, P., & Premože, S. (2001a). A physically-based night sky model. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '01 (pp. 399–408). New York, NY, USA: ACM.

Jensen, H. W., Premoze, S., Shirley, P., Thompson, W. B., Ferwerda, J. A., & Stark, M. M. (2001b). Night rendering.

Johnson, H. L. & Morgan, W. W. (1953). Fundamental stellar photometry for standards of spectral type on the revised system of the yerkes spectral atlas. *Astrophysical Journal*, 117.

Josth, R. (2005). Real time atmosphere rendering for the space simulators.

Kegel, A. (2006). Echtzeitfähige rendering-verfahren für die umgebungsdarstellung virtueller 3d-landschaften.

Kim, Y.-S., Cho, B.-h., Kang, B.-s., & Hong, D.-I. (2003). Color temperature conversion system and method using the same.

Krystek, M. P. (1985). An algorithm to calculate correlated color temperature. *Color Research and Application*, 10.

Magnor, M., Sen, P., Kniss, J., Angel, E., & Wenger, S. (2010). Progress in rendering and modeling for digital planetariums. In *Proceedings of Eurographics 2010*.

Maiwald, C. (2009). Hochwertiges rendern von sternen 2.0. http://zfx.info/viewtopic.php?f=11&t=8.

Max, N. L. (1988). Horizon mapping: shadows for bump-mapped surfaces. *The Visual Computer*, 4(2), 109–117.

McReynolds, T. & Blythe, D. (2005). *Advanced Graphics Programming Using OpenGL*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Meeus, J. (1994). *Astronomische Algorithmen*. Barth.

Miller, G. S. & Hoffman, R. C. (1984). Illumination and reflection maps: Simulated objects in simulated and real environments. *SIGGRAPH '84*.

Müller, D. (2012). Extended bright star catalogue. http://osghimmel.googlecode.com/files/resources_extended_bright_star_catalogue.zip.

Müller, D., Engel, J., & Döllner, J. (2012). Single-pass rendering of day and night sky phenomena. In *VMV to appear*.

Nadeau, D. R., Genetti, J. D., Napear, S., Pailthorpe, B., Emmart, C., Wesselak, E., & Davidson, D. (2000). Visualizing stars and emission nebulae.

NASA (1994). Clementine project information. http://nssdc.gsfc.nasa.gov/planetary/clementine.html.

NASA (2011). Low reconnaissance orbiter. http://www.nasa.gov/mission_pages/LRO/main/index.html.

Nishita, T., Sirai, T., Tadamura, K., & Nakamae, E. (1993). Display of the earth taking into account atmospheric scattering. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '93 (pp. 175–182). New York, NY, USA: ACM.

Olson, T. (1998). The colors of the stars. In *In IST/SID 6th Color Imaging Conf.*

O'Neil, S. (2005). Accurate atmospheric scattering. In M. Pharr (Ed.), *GPU Gems 2* (pp. 253–268). Addison-Wesley.

Onoue, K., Max, N., & Nishita, T. (2004). Real-time rendering of bumpmap shadows taking account of surface curvature. In *Proceedings of the 2004 International Conference on Cyberworlds*, CW '04 (pp. 312–318). Washington, DC, USA: IEEE Computer Society.

Perlin, K. (2002). Improving noise. *ACM Trans. Graph.*, 21(3), 681–682.

Peterson, A. (2010). Sphere to cube mapping. http://petrocket.blogspot.de/2010/04/sphere-to-cube-mapping.html.

Pieters, C. M. (1999). The moon as a spectral calibration standard enabled by lunar samples: The clementine example. In *Workshop on New Views of the Moon II: Understanding the Moon Through the Integration of Diverse Datasets* (pp. 8025).

Preetham, A. J., Shirley, P., & Smits, B. (1999). A practical analytic model for daylight. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '99 (pp. 91–100). New York, NY, USA: ACM Press/Addison-Wesley Publishing Co.

Rayleigh, L. (1871). On the scattering of light by small particles. *Philosophical Magazine 41*, (pp. 447–451).

Reddy, V., Snedegar, K., & Balasubramanian, R. K. (2007). Scaling the magnitude: the fall and rise of n. r. pogson. *Journal of the British Astronomical Association, vol.117, no.5, p.237-245 (JBAA Homepage).*

Riley, K., Ebert, D. S., Kraus, M., Tessendorf, J., & Hansen, C. (2004). Efficient rendering of atmospheric phenomena. In *Proceedings of Eurographics Symposium on Rendering 2004* (pp. 375–386).

Ritschel, T., Ihrke, M., Frisvad, J. R., Coppens, J., Myszkowski, K., & Seidel, H.-P. (2009). Temporal glare: Real-time dynamic simulation of the scattering in the human eye. In *Proceedings Eurographics 2009*.

Roden, T. & Parberry, I. (2005). Clouds and stars: Efficient real-time procedural sky rendering using 3D hardware. In *Proceedings of the 2005 ACM SIGCHI International Conference on Advances in Computer Entertainment Technology* (pp. 434–437).

Schpok, J., Simons, J., Ebert, D. S., & Hansen, C. (2003). A real-time cloud modeling, rendering, and animation system. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, SCA '03 (pp. 160–166). Aire-la-Ville, Switzerland, Switzerland: Eurographics Association.

Spencer, G., Shirley, P., Zimmerman, K., Greenberg, D. P., & Inc, T. (1995). Physically-based glare effects for digital images. In *SIGGRAPH 95 Conference Proceedings*, Annual Conference Series (pp. 325–334).: Addison Wesley.

Stokholm Nielsen, R. (2003). Real time rendering of atmospheric scattering effects for flight simulators. Supervisor: Niels Jørgen Christensen.

Theisgen, S. (2011). Hdr rendering -physikalische sinnvolle werte? http://zfx.info/viewtopic.php?f=11&t=8.

Thomas, G. & Stamnes, K. (2002). *Radiative Transfer in the Atmosphere and Ocean*. Cambridge Atmospheric and Space Science Series. Cambridge University Press.

Torchelsen, R. P. & Musse, S. R. (2005). Modeling and animating clouds in real-time using billboards.

van de Hulst, H. C. (1980). *Multiple Light Scattering: Tables, Formulas, and Applications*. Academic Press.

Wang, C., Wang, Z., & Peng, Q. (2007). Real-time rendering of sky scene considering scattering and refraction. *Comput. Animat. Virtual Worlds*, 18, 539–548.

Wang, N. (2003). Realistic and fast cloud rendering in computer games. In *ACM SIGGRAPH 2003 Sketches & Applications*, SIGGRAPH '03 (pp. 1–1). New York, NY, USA: ACM.

Williams, D. R. (2010). Moon fact sheet. http://nssdc.gsfc.nasa.gov/planetary/factsheet/moonfact.html. [Online; accessed 26-April-2012].

Yapo, T. C. & Cutler, B. (2009). Rendering lunar eclipses. In *Proc. Graphics Interface* (pp. 63–69).

Zotti, G., Wilkie, A., & Purgathofer, W. (2007). A critical review of the preetham skylight model. In V. Skala (Ed.), *WSCG ' 2007 Short Communications Proceedings I* (pp. 23–30).: University of West Bohemia.

Eidesstattliche Erklärung

Ich erkläre hiermit an Eides statt, dass ich die vorliegende Arbeit ohne unzulässige Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe; die aus anderen Quellen direkt oder indirekt übernommenen Daten und Konzepte sind unter Angabe des Literaturzitats gekennzeichnet. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

(Ort, Datum) (Unterschrift)