Realistische Echtzeitdarstellung von Wolkenformationen in virtuellen Umgebungen

DIPLOMARBEIT

zur Erlangung des akademischen Grades
eines Diplom-Ingenieurs der Informatik

verfaßt von

Paul Heinzlreiter

angefertigt an der
Abteilung für Graphische und Parallele Datenverarbeitung
des Instituts für Technische Informatik und Telematik
der Johannes Kepler Universität Linz

eingereicht bei
o. Univ.-Prof. Dr. Jens Volkert

mitbetreuender Assistent
Dipl.-Ing. Gerhard Kurka

Linz, April 2001

Kurzfassung

Die realistische Darstellung von Wolkenformationen findet Anwendung in verschiedenen Bereichen wie beispielsweise Geländevisualisierung oder Computerspielen.
Es wurden schon von mehreren Autoren Konzepte zur realistischen Darstellung von Wolkenformationen entwickelt, die gute Ergebnisse liefern. Diese Ansätze beschränken sich allerdings auf die Erzeugung statischer Bilder. In dieser Arbeit werden die vorhandenen Ansätze dahingehend erweitert, daß die Wolkendarstellung in einer interaktiven, dreidimensionalen Szenerie möglich wird.
Es wird hierzu eine Applikation entwickelt, die es dem Benutzer ermöglicht, sich in Echtzeit durch eine gegebene Szenerie zu bewegen. Die in dieser Implementierung umgesetzte Methode der Bilderzeugung folgt im wesentlichen einem der bereits publizierten Bilderzeugungsverfahren. Es werden die verschiedenen Schritte des Verfahrens als unabhängige Programme implementiert, wobei die interaktive Szeneriedarstellung unter Verwendung der Standardgrafikbibliothek OpenGL erfolgt, um die Plattformunabhängigkeit der Implementierung sicherzustellen.

Danksagung

Ich möchte mich bei all jenen bedanken, die direkt oder indirekt zur Entstehung dieser Diplomarbeit in Form fachlicher oder anderweitiger Unterstützung beigetragen haben.

Besonderer Dank gebührt dabei meinen Eltern, die mir durch ihre finanzielle und persönliche Unterstützung dieses Studium ermöglicht haben.

Weiters bedanke ich mich bei Herrn o. Univ.-Prof. Dr. Jens Volkert sowie Herrn Dipl.-Ing. Gerhard Kurka und Herrn Dipl.-Ing. Dr. Dieter Kranzlmüller für die Betreuung meiner Arbeit.
Ihre zahlreichen wissenschaftlichen Ratschläge waren für mich sehr wertvoll und haben stets zur Verbesserung dieser Arbeit beigetragen.

Inhaltsverzeichnis

Kapitel 1

Einführung

1.1 Motivation

Ein Hauptziel der Computergrafik war und ist einen möglichst hohen Realitätsgrad der Darstellungen zu erzielen. Für den Fall von klar begrenzten Objekten wurden in der Vergangenheit bereits sehr gute und schnelle Verfahren entwickelt, die Aufgaben wie Beleuchtung, Schattierung oder Verdeckungsberechnung realisieren. Weiters werden diese Verfahren bereits sehr gut von moderner Grafikhardware unterstützt.
Auch für die Darstellung von Outdoor-Szenarien gibt es bereits gute Ansätze, die die Beeinflußung des Lichts durch die Atmosphäre berücksichtigen, wie die Verfahren die in [Klass87], [PrSS99] und [Irwi96] vorgestellt werden.
Ein nach wie vor aktuelles Problem bei der Darstellung solcher Szenarien stellen gewöhnliche Wolken dar. Aus den verschiedensten Gründen sind Wolken nach wie vor eine interessante Herausforderung im Bereich der Computergrafik:

- Die vorhandenen Konzepte der Computergrafik sind oft nicht für die Darstellung natürlicher Phänomene geeignet.
- Eine Wolke ist kein klar begrenztes Gebilde sondern hat eine amorphe Struktur.
- Eine Wolke beeinflußt das durch sie hindurchgehende Licht in vielfacher Weise.
- Eine Simulation dieser Lichtbeeinflußung ist sehr rechenintensiv.
- Eine Wolke ist ein aus dem täglichen Leben wohlbekanntes Objekt, daher ist es leicht Fehler in der Darstellung zu erkennen.

Es wurden in den letzten Jahren auf dem Gebiet der Wolkendarstellung große Fortschritte erzielt, wie unter anderem die Papers [DKYO00], [Gard85], [NiDN96] und [NiDo99] beziehungsweise [Taxe99] zeigen. Mit Ausnahme von [DKYO00] haben diese Ansätze aber zum Ziel statische Bilder zu erzeugen.
Im Zuge dieser Arbeit wurde ein System entwickelt, um solche statischen Bilder in einer dreidimensionalen, interaktiven Umgebung für eine realistische Wolkendarstellung einzusetzen.

1.2 Gliederung der Arbeit

Folgende Inhalte werden in dieser Arbeit behandelt:

- Kapitel 1:
  Motivation und Gliederung der Arbeit

- Kapitel 2:
  Hier wird ein Überblick über meteorologische Grundlagen und das optische Verhalten von Atmosphäre und Wolken gegeben. Dies umfasst unter anderem Wolkenenstehung und Wolkenklassifikation. Weiters werden einige im weiteren Verlauf der Arbeit wichtige physikalische Vorgänge wie Absorption und Streuung erläutert.

- Kapitel 3:
  In diesem Kapitel wird die grundsätzliche computergraphische Handhabung von Wolken beschrieben. Es werden verschiedene Ansätze für ihre Modellierung beschrieben. Weiters wird darauf eingegangen, wie man in Abhängigkeit von den im Modellierungsschritt erzeugten Daten die Bilderzeugung durchführen kann.

- Kapitel 4:
  Dieses Kapitel beschreibt verschiedene Möglichkeiten Wolkendarstellung in der dreidimensionalen Computergrafik für interaktive Anwendungen einzusetzen. Es werden Problematik und Lösungsansätze diskutiert.

- Kapitel 5:
  In diesem Kapitel wird die vorliegende Implementierung im Bezug auf Architektur und eingesetzte Algorithmen beschrieben. Weiters erfolgt eine Präsentation der Ergebnisse und eine Betrachtung der Programmleistung.

- Kapitel 6:
  Hier werden die Vor- und Nachteile der Implementierung im Vergleich zu anderen Verfahren betrachtet, und mögliche Erweiterungen der Implementierung diskutiert.

- Kapitel 7:
  Zusammenfassung der Arbeit

Kapitel 2

Meteorologie und atmosphärische Optik

In diesem Kapitel soll ein Überblick über die Zusammensetzung von Atmosphäre und Wolken sowie die durch sie beeinflußten optischen Vorgänge gegeben werden, die unser Bild des Himmels entstehen lassen.
Dabei wird auch auf die elementaren physikalischen Vorgänge und ihre mathematische Behandlung eingegangen.

2.1 Atmosphärische Physik

2.1.1 Trockene Luft

Der Zustand eines Gases läßt sich durch drei Parameter beschreiben:

1. Druck
2. Dichte
3. Temperatur

Änderungen dieser Parameter ergeben sich laut [Hess61] durch Zuführung von Wärme zu einem Luftvolumen oder durch vertikale Bewegung. Das erstere führt zu Änderungen der Temperatur, das zweitere zu Änderungen des Drucks.
Die Beziehungen zwischen diesen Größen werden durch mehrere Gesetze beschrieben:

Gesetz von Boyle und Mariott:

$$pV = p_0 V_0 \tag{2.1}$$

Bei konstanter Temperatur bleibt das Produkt von Druck p und Volumen V konstant.

Gesetz von Avogadro:

$$\frac{\rho_1}{\rho_2} = \frac{m_1}{m_2} \tag{2.2}$$

Bei gleichem Druck und Temperatur verhalten sich die Dichten zweier Gase wie die relativen Molekülmassen.

Ideale Gasgleichung:

$$pV = \frac{p}{\rho} = R^* T \tag{2.3}$$

Dabei bezeichnet p den Druck, V das Volumen, ρ die Dichte, T die Temperatur und R^* die universelle Gaskonstante. Für jedes Gas gibt es eine spezifische Konstante, die die physikalischen Eigenschaften des Gases wiederspiegelt. Für den Fall von Luft wird

$$R_L = 287 \frac{J}{kg\,K}$$

für R^* eingesetzt.

2.1.2 Wasserdampf

Luft ist ein Gemisch aus verschiedenen Gasen mit den Hauptbestandteilen Stickstoff und Sauerstoff. Im Vergleich dazu ist der Anteil an Wasserdampf gering, er schwankt stark zwischen einem und vier Volumenprozent. Man kann ihn aber trotzdem als das meteorologisch wichtigste Gas in der Atmosphäre ansehen, da er den Strahlengang des Lichts in der Atmosphäre stark beeinflußt und außerdem für Wolken und Niederschlag sorgt. [Möll73a]
Mischungen von Gasen können generell mit dem Dalton'schen Gesetz beschrieben werden. Danach nimmt jedes Gas in der Mischung entweder das seinem prozentuellen Anteil entsprechende Volumen oder den entsprechenden Druck ein, gemäß Gleichung 2.1. Der Gesamtdruck des Gasgemisches ist demzufolge

$$p_L = \sum p_i = \frac{p}{V} \sum V_i \tag{2.4}$$

wobei p_i und V_i die Teildrücke beziehungsweise die Teilvolumina bezeichnen. Der Anteil eines Gases am Gemisch ergibt sich daher zu

$$\mu_i = \frac{\rho_i}{\rho_L} \tag{2.5}$$

Im Fall des Wasserdampfes bezeichnet man den Partialdruck als Dampfdruck. Er wird mit dem Buchstaben e bezeichnet und in Millibar angegeben. Da die individuelle Gaskonstante für Wasserdampf einen Wert von

$$R_W = 461,5 \frac{J}{kg\,K}$$

hat, ergibt sich daraus die Dampfdichte zu

$$\rho_W = \frac{e}{R_W\,T} \tag{2.6}$$

Das Massenmischungsverhältnis Wasserdampf zu trockener Luft ergibt sich nach Gleichung 2.3 und Gleichung 2.6 beziehungsweise Gleichung 2.5 zu

$$\mu_W = \frac{e\,R_L}{p_L\,R_W} = \frac{\rho_W}{\rho_L} \tag{2.7}$$

Es gibt nun verschiedene Möglichkeiten den Gehalt der Atmosphäre an Luftfeuchtigkeit anzugeben [Taxe99]:

- Partialdruck des Wasserdampfes e, der den durch Wassermoleküle in einer Volumeneinheit ausgeübten Druck beschreibt.
- Dichte des Wasserdampfes oder absolute Feuchtigkeit ρ_W
- Massenmischungsverhältnis μ_W, beschreibt das Massenverhältnis zwischen Wasserdampf und der trockenen Luft.
- Spezifische Feuchtigkeit

  $$s = \frac{R_L}{R_W}\,\frac{e}{p + e\,\frac{R_L}{R_W - 1}} \tag{2.8}$$

  Die spezifische Feuchtigkeit beschreibt das Massenmischungsverhältnis zwischen Wasserdampf und der feuchten Luft.
- Relative Feuchtigkeit

  $$f = \frac{e}{E(T)} \tag{2.9}$$

  Die relative Feuchtigkeit beschreibt den Anteil des aktuellen Dampfdruckes am Sättigungsdampfdruck und wird oft in Prozent angegeben.
- Virtuelle Temperatur

  $$T_V = T\,(1 + 0,608\,s) \tag{2.10}$$

  Die virtuelle Temperatur ist diejenige Temperatur, die trockene Luft aufweisen muß, damit sie unter dem selben Druck dieselbe Dichte hat wie feuchte Luft mit der spezifischen Feuchtigkeit s [Möll73a].

2.2 Wolkenentstehung

Zur Ausbildung von Wolken kommt es durch Kondensation von Wasserdampf in der Atmosphäre. Der Kondensation gehen zwei notwendige Bedingungen voraus:

1. Erreichen des Sättigungsdampfdruckes
2. Vorhandensein von Kondensationskernen

Der Sättigungsdampfdruck gibt jenen Druck in einem Volumenbereich an, bei dem bei einer gegebenen Temperatur ein Gleichgewicht zwischen Verdampfung und Kondensation herrscht. Das heißt es verdampft pro Zeiteinheit genausoviel Wasser wie kondensiert. Die Abhängigkeit des Sättigungsdampfdruckes E von der Temperatur wird von folgender empirischer Formel von H. G. Magnus beschrieben [Möll73a]:

$$E(t) = 6,107 \cdot 10 \cdot \frac{7,5t}{237+t}\, mb \tag{2.11}$$

t gibt hierbei die Temperatur in °Celsius an.
Bei einem Druck im gegebenen Volumenbereich von e kann man nun folgende Fälle unterscheiden:

1. $e = E(t)$: Der Wasserdampf ist gesättigt und der Sättigungsdampfdruck erreicht.
2. $e < E(t)$: Der Wasserdampf ist ungesättigt oder überhitzt.
3. $e > E(t)$: Der Wasserdampf ist übersättigt.

Zum Überwiegen der Kondensation im Vergleich zur Verdunstung und damit zur Ausbildung von Wolken kommt es, wenn die beiden oben genannten notwendigen Bedingungen erfüllt sind. Typische Arten von Kondensationskernen sind

- Staub
- Industrielle Emissionen
- Rußpartikel

Die Größe solcher Kondensationskerne bewegt sich im Bereich zwischen 10^{-3} und 10 Mikrometern.
Eine Möglichkeit der Kondensation ohne Kondensationskerne tritt dann ein, wenn der Wasserdampf genügend übersättigt ist. Dabei werden die Kondensationskerne durch Wassermoleküle ersetzt, die nach Kollisionen embrionale Wassertropfen bilden. Dazu ist es allerdings notwendig, daß die relative Feuchtigkeit f mehrere hundert Prozent erreicht.

2.3 Wolkenklassifikation

Die heute übliche Klassifikation der Wolkentypen erfolgt nach phänomenologischen Gesichtspunkten und wurde von L. Howard 1804 erstmals vorgeschlagen. Es wird hierbei folgende Einteilung getroffen:

Internationale Bezeichnung	Deutsche Bezeichnung
Cirrus	Federwolken
Stratus	Schichtwolken
Cumulus	Haufenwolken

Dies sind die drei Hauptwolkentypen, zwischen denen es allerdings noch zahlreiche Übergangsformen wie Cirrocumulus, Cirrostratus oder Stratocumulus gibt. Weiters gibt es noch die Unterscheidungen nach Höhe und Wassersättigung der Wolken.
Wichtige Wolkentypen sind nach [Möll73a]:

- *Cirrus, Cirrocumulus, Cirrostratus:*
  Dies sind reine Eiswolken, die aus Schnee- oder Eiskristallen bestehen. Ihr Erscheinungsbild ist zart, weiß, haar- oder faserförmig oder seidig glänzend. Es gibt keine dunklen Schattenstellen und meist ist der Himmel durch die Wolke sichtbar. Cirrus besteht aus getrennten Fasern, Cirrostratus aus einer gleichförmigen Schicht, Cirrocumulus ist in einzelne Ballen aufgelöst.

- *Altocumulus:*
  Die am besten passende deutsche Beschreibung für diesen Wolkentyp ist Schäfchenwolke. Dieser Wolkentyp ist aufgelöst in kleine Ballen oder Walzen, deren scheinbare Größe 5° [1] nicht überschreitet. Der Unterschied zu Cirrocumulus sind stellenweise Schatten auf den Wolken.

- *Altostratus:*
  Dieser Wolkentyp ist eine fast strukturlos erscheinende, weißliche oder graue Wolkenschicht, durch die man die Sonne nicht oder nur sehr undeutlich erkennen kann.

- *Nimbostratus:*
  Dies ist die Bezeichnung für eine tiefe graue oder dunkle Wolkenschicht, aus der Niederschlag fällt. Der Ort der Sonne ist nicht zu erkennen.

- *Stratocumulus:*
  Gröbere und deshalb tiefere, in Ballen oder Walzen aufgelöste Schichtwolken, wobei die einzelnen Ballen durchwegs mehr als 5° Durchmesser haben.

[1] die Größenangabe in Winkelform hat den Vorteil, daß sie die Höhe der Wolke über dem Boden miteinbezieht

- *Stratus:*
  gleichmäßig graue, fast strukturlose Wolkenschicht, aus der höchstens Sprühregen, aber kein großtropfiger Regen fallen kann. Wenn die Sonne erkennbar ist, wird sie als scharfe, runde Scheibe sichtbar.

- *Cumulus:*
  einzelne haufen- oder ballenförmige Wolken mit einer glatten Basis und einer Oberseite mit einer knolligen Struktur.

- *Cumulonimbus:*
  mächtig aufgetürmte Cumuluswolken, unter denen es sehr dunkel werden kann, deren Oberseiten wenigstens stellenweise abgeflacht oder ausgefasert erscheinen, in der typischsten Form wie ein Amboß nach beiden Seiten ausladend. Wie der zweite Teil des Namens sagt, sind es Wolken, aus denen Niederschlag fällt, in Form von Schauern oder Gewittern.

Diese Klassifikation reicht noch nicht aus, um die gesamte Vielfalt der Wolkenformationen zu erfassen, daher kann man die Wolkentypen noch weiter in Arten und Unterarten gliedern, siehe [Möll73a]. Dies würde hier allerdings zu weit führen.

2.4 Grundlagen der atmosphärischen Optik

Die atmosphärische Optik umfasst alle Einflüsse, die die Atmosphäre auf den visuellen Eindruck unserer Welt nimmt.
Beispiele dafür sind die Abnahme des Kontrast mit der Entfernung oder auch Veränderungen in der Farbe durch die Atmosphäre, denen entfernte Objekte stärker unterliegen als nahe. Diese Farbveränderung läßt sich beispielsweise an Bergketten beobachten, die umso bläulicher erscheinen, je weiter sie entfernt sind.
Die Berücksichtigung der Atmosphäre ist im besonderen für Computergrafikanwendungen interessant, da die Berücksichtigung der Atmosphäre bei der Erzeugung von Bildern von Outdoor Szenerien beeindruckende Ergebnisse erzielt. Dies wird als *atmosphärische Perspektive* bezeichnet.

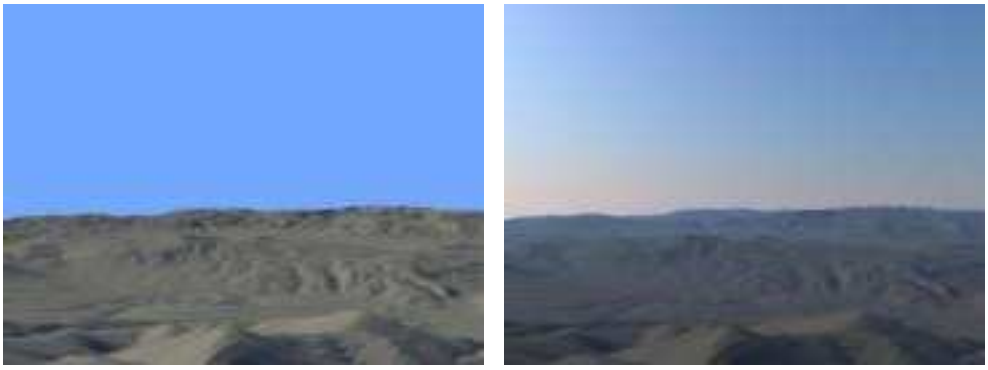


Abbildung 2.1: Links ohne, rechts mit atmosphärischer Perspektive

Im wesentlichen handelt es sich dabei um die Beeinflußung des Lichts durch alle Elemente der Atmosphäre wie Moleküle und Partikel. Diese sind hierbei als größere Ansammlungen von Molekülen zu verstehen.
Die dominante Lichtquelle ist die direkte Sonneneinstrahlung, deren Licht hauptsächlich durch Streuung beeinflußt wird. Genauere Ausführungen dazu finden sich in [Klass87], [PrSS99] und [Bohr95].

2.4.1 Lichtstreuung

Die Streuung des einfallenden Sonnenlichts ist die am stärksten ausgeprägte Beeinflußung des Sonnenlichts durch die Atmosphäre. Hierbei handelt es sich um die Verteilung des einfallenden Lichts durch ein oder mehrere Moleküle, die sich in der Bahn des Lichtes befinden.
Durch atmosphärische Partikel kommt es zu elastischer Streuung, was bedeutet, daß durch den Streuvorgang keine Energie verlorengeht. Die Verteilung des einfallenden Lichtes durch ein Partikel ist von folgenden Faktoren abhängig:

- Größe des Partikels
- Form des Partikels
- Wellenlänge des einfallenden Lichtes

Der wesentliche Faktor, der die Art der Lichtstreuung beeinflußt ist das Verhältnis zwischen der Größe des Partikels und der Wellenlänge des einfallenden Lichtes λ.
Für den Fall der Streuung an Molekülen, wie sie in der Atmosphäre bei klarem Wetter fast ausschließlich vorkommen, dominiert die Rayleigh Streuung. Diese ist nach Lord Rayleigh benannt, der sie als erstes theoretisch untersucht hatte.

2.4.2 Rayleigh Streuung

Wenn der Durchmesser der streuenden Partikel $0{,}1\lambda$ nicht übersteigt, wird das einfallende Licht nach der von Lord Rayleigh entdeckten Gesetzmäßigkeit gestreut. Der Streuungskoeffizient berechnet sich dabei laut [Möll73b] und [Taxe99] nach folgender Formel:

$$\frac{I_0 - I}{I_0} = a_{R\lambda} = \frac{8\pi^3(n_\lambda^2 - 1)^2}{3\lambda^4 N} \tag{2.12}$$

Dabei bezeichnet I_0 die Intensität des einfallenden Lichtes, I die Intensität des Restlichtes nach der Streuung, $a_{R\lambda}$ den Streuungskoeffizienten, n_λ den lichtwellenlängenabhängigen Brechungsindex der Luft und N die Anzahl der Moleküle pro cm^3.
Da die Wellenlänge des eingestreuten Lichtes mit λ^{-4} in die Formel eingeht, wird das blaue Licht mit einer Wellenlänge von $4 \cdot 10^{-7}$ Metern im Vergleich zum roten Licht mit $7 \cdot 10^{-7}$ Metern rund 10 mal so stark gestreut, woraus der optische Eindruck des blauen Himmels entsteht. Genauso läßt sich die Rotfärbung des Himmels am Morgen und am Abend erklären, da durch die niedrigstehende Sonne die Lichstrahlen einen längeren Weg durch die Atmsphäre zurücklegen müssen und dabei ein größerer Anteil der Lichtwellen, deren Frequenz im Lichspektrum der Farbe Blau entspricht, aus der Richtung des Lichts gestreut wird.
Im Fall der Rayleigh Streuung erfolgt die winkelabhängige Verteilung des gestreuten Lichtes laut [NiDN96] nach der Beziehung

$$I_\varphi = \frac{3\,(1 + cos^2(\varphi))}{4} \tag{2.13}$$

wobei φ den Winkel zwischen der Einfallsrichtung und der Emissionsrichtung des Lichtes angibt. Diese Beziehung zwischen dem Winkel φ und der Streuungsintensität I_φ wird als Phasenfunktion bezeichnet.

2.4.3 Mie Streuung

Die Verursacher der Mie Streuung sind im Gegensatz zur Rayleigh Streuung Partikel deren Durchmesser $0{,}1\lambda$ übersteigt. Dies sind zum Beispiel Wassertröpfchen, aus denen sich Wolken hauptsächlich zusammensetzen. Deshalb ist auch die Mie Streuung die für diese Arbeit interessantere. Es gibt wesentliche Unterschied zwischen der Rayleigh Streuung un der Mie Streuung:

- Die Streuungsintensität der Mie Streuung ist nicht von der Wellenlänge des einfallenden Lichtes abhängig.
- Die Richtungsverteilung der gestreuten Strahlung ist bei der Mie Streuung viel asymmetrischer.

Die üblicherweise verwendete Phasenfunktion zur Angabe der richtungsabhängigen Verteilung des gestreuten Lichtes im Falle der Mie Streuung ist die Henyey-Greenstein Funktion:

$$I_\varphi = \frac{1 - g^2}{\sqrt{(1 + g^2 - 2g\,cos(\varphi))^3}} \tag{2.14}$$

g bezeichnet hierbei einen Asymmetriefaktor, der den Grad der Abweichung von der Rayleigh Streuung unter Berücksichtigung des Verhältnisses zwischen der Wellenlänge des einfallenden Lichtes und der Partikelgröße angibt.

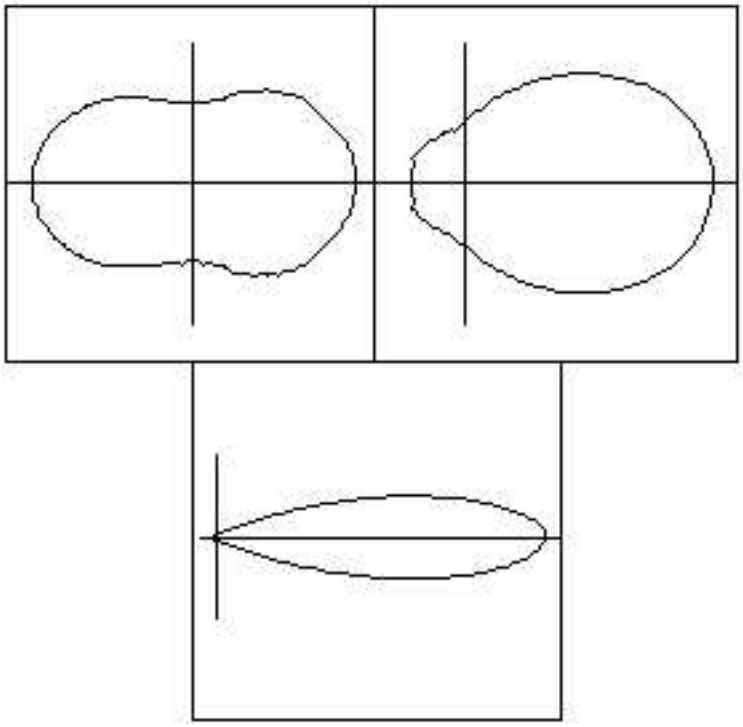


Abbildung 2.2: Diagramme verschiedener Phasenfunktionen

Abbildung 2.2 zeigt die Phasenfunktionen von Wassertropfen:

links oben	Partikeldurchmesser 0,062λ
rechts oben	Partikeldurchmesser 0,316λ
unten	Partikeldurchmesser 1,91λ

Bei der obigen Art der Darstellung in Form eines Polardiagramms kommt die winkelabhängige Verteilung der gestreuten Intensität besonders gut zur Geltung.

Die Streuungsintensität, die einem Winkel zugeordnet ist läßt sich leicht dadurch feststellen, das man vom Ursprung des Koordinatensystems ausgehend eine Gerade im gefragten Winkel mit der Kurve schneidet. Der Abstand des Schnittpunktes zum Ursprung gibt die anteilsmäßige Menge des in diese Richtung gestreuten Lichtes an.
Im Fall von d=0,062λ liegt Rayleigh Streuung vor, daher wird das Licht gleichmäßig nach vorne und hinten gestreut. Bei ansteigendem Durchmesser überwiegt die Mie Streuung immer mehr und der prozentuelle Anteil des nach vorne gestreuten Lichts nimmt zu.
In der Atmosphäre kommen Rayleigh und Mie Streuung immer nur gemeinsam vor, wobei die die Anteile der vorhandenen Partikelgrößen festlegen, welche Art der der Streuung im gefragten Bereich der Atmosphäre überwiegt.

2.4.4 Absorption

Die Absorption beschreibt die Umwandlung von sichtbarem Licht in Wärmeenergie durch atmosphärische Gase. Dabei wird die vom Licht in Form von elektromagnetischer Strahlung übertragene Energie in Bewegungsenergie der Atmosphärenteilchen umgesetzt. Sie ist neben der Streuung ein weiterer wesentlicher Beitrag der Atmosphäre bei der der Modifikation des durch sie hindurchgehenden Lichtes.
Die restliche Lichtintensität nach der der Absorbtion beträgt laut [NiDN96]

$$I_p(\lambda) = I_0(\lambda)\, e^{-\tau(S,\lambda)} \tag{2.15}$$

wobei I_0 die Lichtintensität vor dem Absorptionsvorgang bezeichnet.
Welcher Anteil des Lichtes beim Durchgang durch einen Teil der Atmosphäre verschluckt wird, wird durch den Absorptionskoeffizienten β angegeben. Zur Berechnung der sogenannten optischen Länge τ eines Atmosphärenabschnitts der Länge S wird nach [NiDN96] folgende Beziehung verwendet:

$$\tau(S,\lambda) = \int_0^S \beta\rho(s)\, ds \tag{2.16}$$

Die Dichte der Atmosphäre wird dabei durch ρ angegeben. Der Wert des Absorptionskoeffizienten β beträgt beispielsweise für eine durchschnittliche Wolke nach [Taxe99] 16,33 pro Kilometer.

Kapitel 3

Modellierung und Rendering von Wolken

In diesem Kapitel sollen Möglichkeiten und Verfahren beschrieben werden Wolkendarstellung mit Hilfe der Computergrafik zu realisieren. Dabei wird auf verschiedene Modellierungsmöglichkeiten, die durch sie erzeugten Datenformate und die daraus resultierenden Visualisierungsmöglichkeiten eingegangen.

3.1 Möglichkeiten der Wolkenmodellierung

Das Problem der Wolkenmodellierung ist vielfältig, da es wie oben schon erläutert wurde zahlreiche Wolkentypen mit verschiedenen Erscheinungsformen gibt. Die Modellierung einer Cirruswolke stellt beispielsweise ganz andere Anforderungen als die einer Cumuluswolke.
Die Modellierung von Wolken für Computergrafikapplikationen ist ein interessantes Problem, da Wolken von Natur aus eine amorphe Struktur haben und daher mit klassischen computergrafischen Methoden nicht gut zu modellieren sind, da diese hauptsächlich mit oberflächenbasierenden Verfahren arbeiten.
Wenn man in einen bewölkten Himmel blickt, sieht man beispielsweise bei einer Cirrusbewölkung die fein untergegliederte Struktur, wohingegen Cumuluswolken eine kompakte Struktur aufweisen.
Vor allem die Selbstähnlichkeit der Wolkenstrukturen macht es bei Wolken wie auch bei vielen anderen natürlichen Phänomenen so schwierig, sie zu modellieren. Es wurden hierzu allerdings bereits einige interessante Lösungsansätze entwickelt, unter anderem in [Gard85, Taxe99, NiDN96].
In der Folge werden die Probleme der Wolkenmodellierung beleuchtet und Lösungsansätze diskutiert.
Grundsätzlich unterscheidet man zwei wesentliche Fälle der Wolkenmodellie-

rung:

1. Modellierung mit zweidimensionalen Ergebnisdaten
2. Modellierung mit dreidimensionalen Ergebnisdaten

3.2 Zweidimensionale Modellierung

Die zweidimensionale Wolkenmodellierung ist für Cirrus- oder Stratuswolken geeignet, die von Natur aus eher eine zweidimensionale Struktur haben. Ein Nachteil dieser Art der Modellierung ist allerdings, daß die Position des Betrachters eingeschränkt ist, da bei dieser Art der Darstellung der Realismus nur dadurch gewährleistet weden kann, daß sich der Betrachter immer in einem ausreichenden Abstand zur Wolkenformation befindet, da sich beim Einsatz von zweidimensionalen Modellierungstechniken die dreidimensionale Struktur der Wolken nicht mehr direkt darstellen läßt.
Das bedeutet für den Benutzer, daß er sich in der Regel am Boden aufhalten muß, und die Wolkenformationen aus einem ausreichenden Abstand betrachtet sodaß die zweidimensionale Struktur des dargestellten Bildes den visuellen Eindruck der Wolken nicht stört. Diese Problematik ist natürlich bei einer Wolkendarstellung die nicht nur auf reine Bilderzeugung sondern auf Interaktivität abzielt besonders wichtig, da bei einer wechselnden Betrachterposition auch das Bild der Wolke angepasst werden muß, um einen realistischen Eindruck zu hinterlassen.

3.2.1 Fraktale

Wenn es um die Darstellung natürlicher Phänomene wie Landschaft, Pflanzen oder auch Wolken geht eignen sich klassische computergraphische Verfahren nicht so gut um die oft unregelmäßige Struktur darzustellen. Einen möglichen Ansatz dieses Problem anzugehen, stellt die Verwendung von Fraktalen dar. Die fraktale Geometrie unterscheidet sich in vielerlei Hinsicht von der klassischen euklidschen Geometrie [Volk98]:

1. Die Darstellung ist skaleninvariant und selbstähnlich
2. Fraktale sind nicht rektifizierbar
3. Eignung für natürliche Objekte
4. Beschreibung durch rekursive Algorithmen

Punkt eins besagt dabei, das sich das beschriebene System invariant oder symmetrisch im Bezug auf eine Maßstabsänderung verhält [Schr94]. Dies bedeutet, wenn man einen bestimmten Ausschnitt einer fraktalen Geometrie vergrößert, zeigt sich wieder das Ausgangsbild. Fraktale sind nicht rektifzierbar, das heißt nicht durch Geraden annäherbar, da sie ansonsten die Eigenschaft der Selbstähnlichkeit verlieren würden. Die oben bereits erwähnte Eignung für natürliche Objekte läßt sich ebenfalls mit der Eigenschaft der Selbstähnlichkeit erklären. Wenn man sich beispielsweise die Rinde eines Baumes oder auch die Oberfläche einer Wolke ansieht, erkennt man in der Vergrößerung die selben Strukturen wie in der Gesamtansicht.

Koch Kurve

Ein anschauliches Beispiel für eine solche fraktale Struktur ist die sogenannte Koch Kurve die das Aussehen einer Schneeflocke hat. Sie wurde 1904 vom schwedischen Mathematiker Helge Koch erfunden.

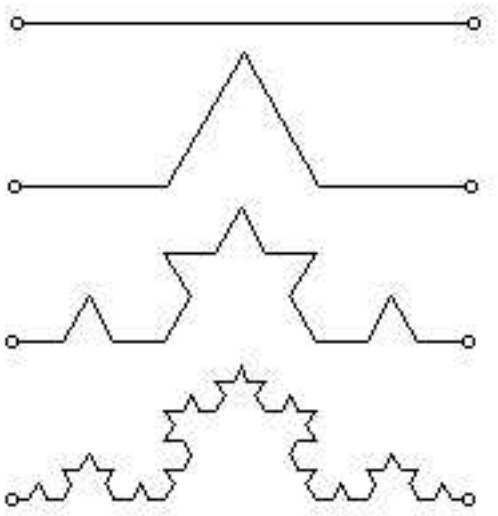


Abbildung 3.1: Koch Kurve

Die Koch Kurve wird durch einen iterierten Prozess erzeugt. Das Ganze beginnt mit dem *Initiator*, der in der obersten Zeile der Abbildung 3.1 gezeigt wird. Darunter befindet sich der *Generator*, der die Zeichenvorschrift für die Erstellung der Kochkurve liefert. In jeder Iteration wird nun auf jedes Liniensegment der Generator angewandt. Die Bilder, die sich nach den folgenden Iterationsschritten ergeben, werden ebenfalls in Abbildung 3.1 gezeigt.
Die Koch Kurve weißt interessante Eigenschaften auf:

- Ihre Länge ist unendlich, da die Länge des Generators $\frac{4}{3}$ der Länge des Initiators beträgt und die Kurve damit bei jeder Iteration länger wird.
- Die Kurve ist zwar überall stetig, aber nirgends differenzierbar.

Plasma Fraktale

Plasma Fraktale [ArRP99] sind Fraktale die sich zur Darstellung von natürliche Phänomenen wie Landschaftstrukturen oder zweidimensionalen Wolkendarstellungen besonders gut eignen. Mögliche Anwendungsgebiete in der Wolkenmodellierung sind der Einsatz zur Erstellung einer Textur zur Darstellung von Stratuswolken, die dann einfach auf eine Ebene abgebildet wird oder auch zur Texturierung von Ellisoiden in der dreidimensionalen Modellierung.

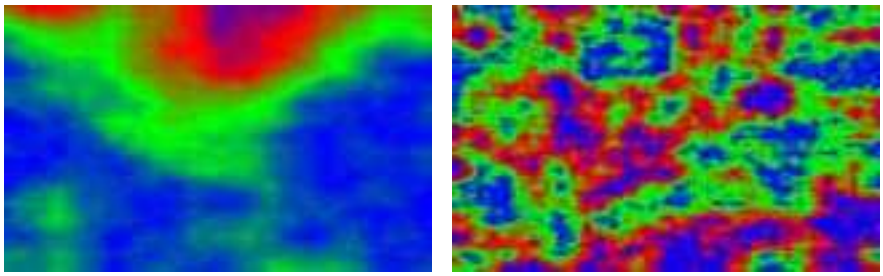


Abbildung 3.2: Plasma Fraktale, links mit einem niedrigen Rauigkeitswert, rechts mit einem hohen

Der Algorithmus zur Erstellung von Plasma Fraktalen läuft folgendermaßen ab:

1. Die Farbwerte für die Ecken des quadratischen Bildes E_1, E_2, E_3, E_4, werden zufällig bestimmt.
2. Der Wert für den Mittelpunkt des Bildes wird durch folgende Formel bestimmt:
$$M = \frac{E_1 + E_2 + E_3 + E_4}{4} + r \cdot R$$
Dabei bezeichnet r einen gegebenen Rauigkeitswert und R eine Zufallszahl.
3. Weiters werden die Werte für die Mittelpunkte der Seiten durch
$$S = \frac{E_A + E_E}{2} + r \cdot R$$

berechnet, wobei E_A den Farbwert am Anfang der Kante bezeichnet und E_E den Wert am Ende.

4. Durch Verwendung der berechneten Werte kann man das Quadrat in vier Teilquadrate unterteilen und die Schritte 2. bis 4. für jedes Teilquadrat wiederholen, bis das Gesamtquadrat ausgefüllt ist.

Fraktale Strukturen eignen sich aufgrund ihrer Selbstähnlichkeit zur Modellierung von Wolken, da genau dieser Aspekt in der Struktur von Wolken ansonsten sehr schwierig zu modellieren ist. Daher wurde auch in dem bereits oben beschriebenen Ansatz aus [NiDN96] eine fraktale Methode zur Generierung der Wolke verwendet.

3.2.2 Rauschfunktionen

Eine Rauschfunktion dient zu dem Zweck zufällige Werte einzubringen, und findet in Bereichen wie der Kryptographie und der Computergraphik Anwendung. Für den Fall der Wolkenmodellierung eignet sich eine spezielle Rauschfunktion, das Perlin Rauschen [EMPP98, Elia00] besonders gut.

Perlin Rauschen

Die Perlin Rauschfunktion kann man im ein-, zwei- oder dreidimensionalen Fall verwenden. Sie basiert auf einem Zufallszahlengenerator und setzt sich aus folgenden Schritten zusammen:

1. Berechnung einer Zufallszahlenfolge
2. Interpolation der Zufallszahlenfolge zu einer stetigen Funktion
3. Wiederholung der Schritte 1 und 2 für beliebig viele Funktionen, wobei die Amplitude jeder neuen Funktion im Vergleich zum Vorgänger halbiert wird, und die Frequenz verdoppelt
4. Aufsummieren der Funktionen
5. Glätten der Rauschfunktion falls notwendig

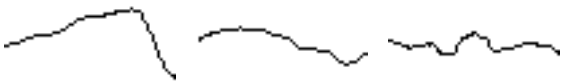


Abbildung 3.3: Teilfunktionen der Perlin Rauschfunktion

Bei den in Abbildung 3.3 gezeigten sechs Teilfunktionen verdoppelt sich die Frequenz zeilenweise gelesen, und die Amplitude halbiert sich mit jedem Bild. Dies ist der Regelfall, der üblicherweise eingesetzt wird, es ist allerdings auch möglich, andere Teilfunktionen zu wählen.
Dieses Verfahren kann ohne Probleme auf den zwei- beziehungsweise dreidimensionalen Fall erweitert werden, um bei der Generierung von Texturen oder Volumenmodellen für Wolken eingesetzt zu werden.

Abbildung 3.4: Perlin Rauschfunktion nach Addition der Teilfunktionen

Interpolation

Ein interessanter Berechnungsschritt bei der Berechnung der Perlinrauschfunktion stellt die Interpolation der Zufallszahlen dar, die aus der Relation $\mathbf{N} \times \mathbf{R}$ eine Funktion $\mathbf{R} \times \mathbf{R}$ ableitet. Hierzu werden verschiedene Methoden verwendet:

1. Lineare Interpolation
2. Cosinus Interpolation
3. Kubische Interpolation

Die Qualität der Interpolation und der Berechnungsaufwand steigen dabei von Punkt 1 bis 3 an.

Lineare Interpolation:

```
float Linear_Interpolate(int a, int b, float x)
{
```

```
  return a*(1-x)+b*x;
}
```

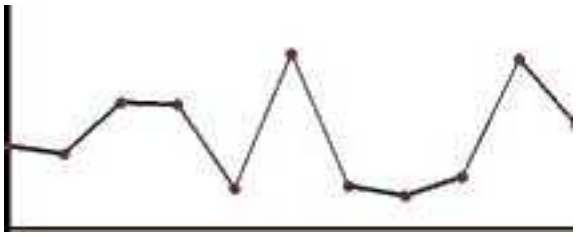


Abbildung 3.5: Lineare Interpolation

Cosinus Interpolation:

```
float Cosinus_Interpolate(int a, int b, float x)
{
  float ft, f;

  ft=x*3.1415927;
  f=(1-cos(ft))*0.5;
  return a*(1-f)+b*f;
}
```

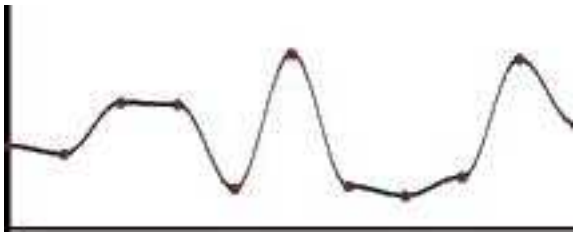


Abbildung 3.6: Cosinus Interpolation

Kubische Interpolation:

```
float Cubic_Interpolate(int v0, int v1, int v2, int v3, float x)
{
  int P, Q, R, S;

  P=(v3-v2)-(v0-v1);
```

```
  Q=(v0-v1)-P;
  R=v2-v0;
  S=v1;
  return (float) (P*pow(x, 3)+Q*pow(x, 2)+R*x+S);
}
```

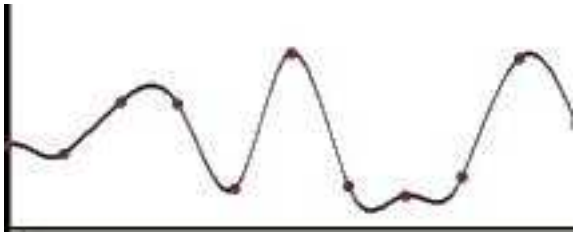


Abbildung 3.7: Kubische Interpolation

Dabei bezeichnen die Parameter a und b die Werte an den beiden benachbarten Punkten und x den Abstand zu Punkt a, beziehungsweise v0 und v1 die beiden Punkte davor und v2 und v3 die beiden Punkte danach.

Glätten der Rauschfunktion

Das Glätten der Rauschfunktion kann notwendig werden, falls die resultierende Perlin Funktion zu stark ausschlägt. Dies wird im eindimensionalen Fall einfach dadurch gelöst, das der neue Wert an der Stelle durch Mittelung mit den beiden Nachbarwerten berechnet wird:

$$R_G(x) = \frac{R(x-1)}{4} + \frac{R(x)}{2} + \frac{R(x+1)}{4}$$

Für den zwei- beziehungsweise dreidimensionalen Fall müssen auch alle weiteren Nachbarwerte mit entsprechender Gewichtung in die Berechnung einbezogen werden.

3.3 Dreidimensionale Modellierung

Im Fall der dreidimensionalen Ergebnisdaten handelt es sich dabei meist um um eine dreidimensionale Dichtefunktion, die stetig ist. Jedem Punkt im Raum, der im Volumenbereich der Wolke liegt, wird hierbei ein Dichtewert zugeordnet, der der Dichte des Wasserdampfes an dieser Stelle entspricht.

$$R^S_{(R_3)}, \mathbf{R^3} \Rightarrow \mathbf{R} \tag{3.1}$$

Da sich eine solche stetige Funktion nicht für eine Simulation der physikalischen Vorgänge im Rechner einsetzten läßt, ist erst zuerst notwendig, diese zu diskretisieren. Dies erfolgt durch ein Voxelmodell, wobei jedem Voxel eine konstante Dichte zugeordnet wird wie beispielsweise in [NiDN96].

3.3.1 Dichtekugeln

Ein Ansatz um eine dreidimensionale Dichtefunktion zu realisieren, ist der in [NiDN96] und [NIDN97] beschriebene Ansatz der Dichtekugeln: Dabei werden die Wolken aus Kugeln zusammengesetzt, von denen jede durch folgende Parameter spezifiziert wird:

- Position des Mittelpunktes
- Radius
- Dichte im Mittelpunkt

Bei der Modellierung wird zuerst die Grobstruktur der Wolke festgelegt. In weiterer Folge wird die Struktur der Wolke im Grenzbereich durch folgende fraktale Technik verfeinert:

1. Es wird die Oberfläche, die die gegebene Kugelmenge umschließt mit Hilfe des Marching Cube Algorithmus [Volk98, SpPr99] ermittelt und trianguliert.
2. Es werden innerhalb jedes Dreiecks, das ermittelt wurde neue Kugeln generiert, wobei Größen und Positionen zufällig bestimmt werden.
3. Diese beiden Schritte werden solange wiederholt, bis die Wolke durch eine ausreichend fein aufgelöste Oberfläche abgegrenzt ist.

Die Ergebnisdaten dieser Modellierung werden dann als die oben genannte dreidimensionale Dichtefunktion abgelegt.

3.3.2 Ellipsoide

Eine andere Möglichkeit die dreidimensionale Struktur der Wolken zu modellieren ist die Verwendung von Ellipsoiden, da sie unter den klassischen geometrischen Körpern der Form von Wolken am nächsten kommen und sich speziell für die Modellierung von Cumuluswolken sehr gut eignen. Es gibt mehrere Arbeiten wie zum Beispiel [Gard85] und [Taxe99] die die Modellierung von Wolken durch Ellipsoide einsetzen.

Bei [Gard85] wird dabei als erster Ansatz die Verwendung eines einzelnen Ellipsoids das durch neun Parameter, die Größe, Form, Position und Ausrichtung beschreiben, spezifiziert ist, vorgeschlagen. Auf dieses Ellipsoid wird dann in weiterer Folge noch eine Textur zur Darstellung der Feinstruktur aufgebracht. Als Erweiterung davon werden komplexere Wolkenstrukturen durch Zusammenfügen von einzelnen Ellipsoiden erzeugt. Diese werden dann als Cluster oder in der Steigerung als Makrocluster, als Cluster von Clustern bezeichnet. Diese Makrocluster eignen sich besonders gut um einen mit zahlreichen Wolken überzogenen Himmel zu simulieren.
Für die Modelierung komplexerer Wolkenformationen wird in [Taxe99] ein interaktiver Wolkeneditor beschrieben, der es dem Benutzer ermöglicht, die Makrostruktur der Wolke selbst festzulegen.

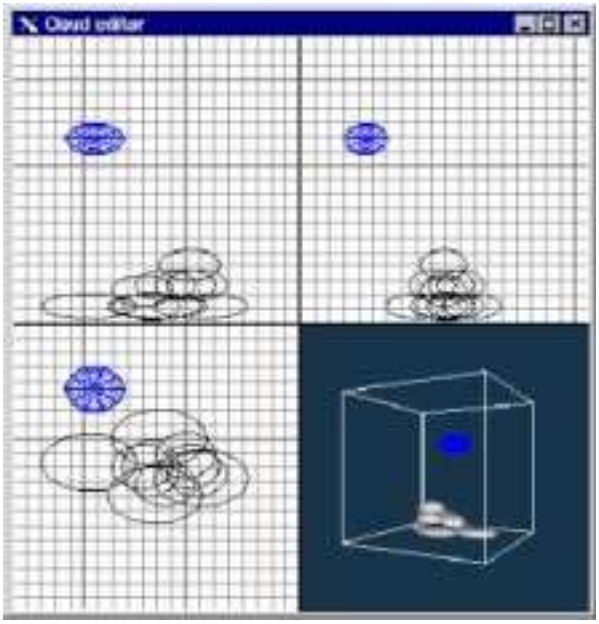


Abbildung 3.8: Interaktiver Wolkenstrukturediteor

3.4 Klassifikation der Modellierungsansätze

Man kann die verschiedenen Möglichkeiten der Wolkenmodellierung nicht nur wie oben beschrieben durch die Dimensionsanzahl der Ausgangsdaten klassifizieren. Eine weitere Möglichkeit ist die Klassifikation nach dem Modellierungsansatz. Es gibt hierbei zwei Möglichkeiten:

- Ontogenetische Modellierung

- Physikbasierende Modellierung

Beim ontogenetischen Modellierungsansatz wird das Ziel verfolgt ein möglichst realistisches Bild zu erzeugen, wobei die physikalischen Abläufe nicht berücksichtigt werden, um Berechnungszeit einzusparen.
Bei der physikbasierenden Modellierung wird eine höhere Berechnungszeit in Kauf genommen, um möglichst realistische Bilder zu erzeugen, wobei die Beeinflußung des Lichtes durch den Wasserdampf der Wolke durch die Berechnung simuliert wird.
Es folgt nun eine Einteilung der vorgestellten Modellierungsansätze:

	zweidimensional	dreidimensional
Ontogenetisch	Fraktale	Ellipsoide
Physikbasierend	-	Dichtefunktion

3.5 Rendering

Rendering oder Bilderzeugung läßt sich nach [LeMS91] und [Volk99] als das Umsetzen einer Szene in einem Bild beschreiben.
Die Bilderzeugnung aus einem dreidimensionalen Modell wird heute üblicherweise in Hardware realisiert. Jedes Objekt durchläuft dabei die Grafikpipeline, die sich laut [Volk99] in zwei Hauptstufen unterteilt:

1. Geometrieverarbeitung
2. Rasterbilderzeugung

Die Geometrieverarbeitung operiert auf dreidimensionalen Objekten und setzt sich aus folgenden Teiloperationen zusammen:

1. Transformationen
2. perspektivische Projektion
3. Rückseitenentfernung
4. Klipping
5. Beleuchtung

Die folgende Rasterverarbeitung operiert im Gegensatz zur Geometrieverarbeitung auf Pixeldaten beziehungsweise werden diese erzeugt. Die in diesem Teil durchzuführenden Schritte umfassen

1. Rasterung
2. Schattierung
3. Texturierung
4. Sichtbarkeitsberechnung
5. Antialiasing

Es ist allerdings nicht in allen Fällen notwendig, sämtliche Schritte zu durchlaufen. Der Texturierungsschritt beispielsweise kann je nach Anwendung oder Objekt entfallen.
Der hier beschriebene Renderingablauf beschreibt die Bilderzeugung aus oberflächenbasierenden Modellbeschreibungen, die auf der Basis von Polygonen arbeiten, was heute in der Computergrafik den Standardfall darstellt.
Wenn das zu visualisierende Modell eine andere Struktur aufweist, wie beispielsweise im Fall des Volume Rendering, wo mit dreidimensionalen Ausgangsdaten gearbeitet wird, kann der Renderingalgorithmus stark von dem beschriebenen Verfahren abweichen, was meist zu deutlich längeren Renderzeiten führt, da andere Verfahren oft nicht in der Grafikhardware implementiert sind.
Der notwendige Rendervorgang ist dabei sehr stark abhängig von der Struktur der zu visualisierenden Daten. Mögliche Varianten sind dabei:

- Drahtgittermodelle
- Oberflächenmodelle
- Volumenmodelle

Je einfacher das Modell der Eingangsdaten ist umso einfacher und schneller ist auch der Vorgang der Bilderzeugung, allerdings ist die Qualität des gerenderten Bildes bei einer höheren Modelldatenkomplexität wesentlich besser. Dies ist vor allem bei Vergleich zwischen Drahtgittermodellen und oberflächenbasierenden Darstellungen offensichtlich.
Der höhere Berechnungsaufwand ergibt sich dabei allerdings aus zusätzlichen Aufgaben wie beispielsweise Verdeckungsberechnung.
Eine weitere Steigerung im Bezug auf die Berechnungszeit ergibt sich bei der Verwendung von Volumenmodellen, unter anderem auch deshalb, da die hardwaretechnische Unterstützung der Volumensvisualisierung noch nicht so ausgereift ist wie im Fall der oberflächenbasierten Verfahren. In der Folge wird nun auf die Darstellung von Volumenmodellen genauer eingegangen, da sie für die die Darstellung von Wolkenformationen große Bedeutung haben.

3.5.1 Volume Rendering

Das Ziel des Volume Rendering ist die Darstellung von multidimensionalen Daten. Einsatzgebiet ist vor allem die wissenschaftliche Visualisierung, im besonderen der medizinische Bereich.
Folgende Anforderungen werden laut [Elvi92] an eine Volume Rendering Applikation gestellt:

- verständliche Datenrepräsentation
- schnelle Datenmanipulation
- möglichst schnelles Rendering

Es gibt zwei grundsätzlich unterschiedliche Verfahren der Volumenvisualisierung:

1. Direktes Volume Rendering
2. Verwendung geometrischer Primitive

Beim direkten Volume Rendering wird aus den dreidimensionalen Daten direkt ein Bild erzeugt, wohingegen beim zweiten Ansatz der Weg über die Erzeugung von Oberflächen führt. Folgende Verfahren können zur Gewinnung der Oberflächen eingesetzt werden [Kauf91]:

- Iso-Konturen
- Iso-Flächen
- Oberflächenextrahierung
- Oberflächenverfolgung
- Grenzverfolgung

Der Vorteil des oberflächenbasierenden Ansatzes liegt in der Geschwindigkeit der Darstellung, da nach der Erzeugung der Polygone Echtzeitdarstellung möglich wird. Dies wird dadurch erreicht, daß eine meist vorhandene hardwarebeschleunigte Polygondarstellung zum Einsatz gebracht werden kann. Allerdings schränkt der Algorithmus zur Oberflächenerzeugung möglicherweise die Aussagekraft der erzeugten Darstellung ein.
Im Gegensatz dazu läßt sich direktes Volume Rendering besonders gut für die Darstellung von Daten mit einer amorphen Struktur wie Gase, Flüssigkeit oder Wolken einsetzen. Für die Implementierung des direkten Volume Rendering kann man verschiedene Algorithmen einsetzen [Elvi92].
In der Folge soll der Ray-Casting Ansatz näher beschrieben werden.

Ray-Casting

Der Ray-Casting Algorithmus wird gerne eingesetzt, da es damit möglich ist, das gesamte Volumen in einem Bild zu erfassen. Der Nachteil ist der hohe Berechnungsaufwand für ein Bild, daher ist es kaum möglich, diesen Algorithmus für interaktives Volume Rendering einzusetzen. Der Grund hierfür liegt in der Notwendigkeit für jede Bilderzeugung den gesamten Datenbestand zu traversieren. Die Qualität der erzeugten Bilder ist allerdings ein gutes Argument für diesen Algorithmus.
Es wird für jedes Pixel des zu erzeugenden Bildes ein Strahl durch das Volumen geschickt und das Pixel entsprechend der entlang des Strahls angetroffenen Volumenstruktur eingefärbt.
Dabei werden die Farbe und die Opazität des Volumens in den angetroffenen Voxeln aufsummiert, bis der Strahl das Volumen wieder verläßt oder sich der summierte Zwischenwert nicht mehr ändert.
Es werden allerdings bei diesem Verfahren weder Schatten noch Reflexionen berücksichtigt, da der Strahl an reflektierenden Flächen nicht weiterverfolgt wird und auch kein Algorithmus zur Schattenberechnung (siehe Abschnitt 4.1.2) angewendet wird.

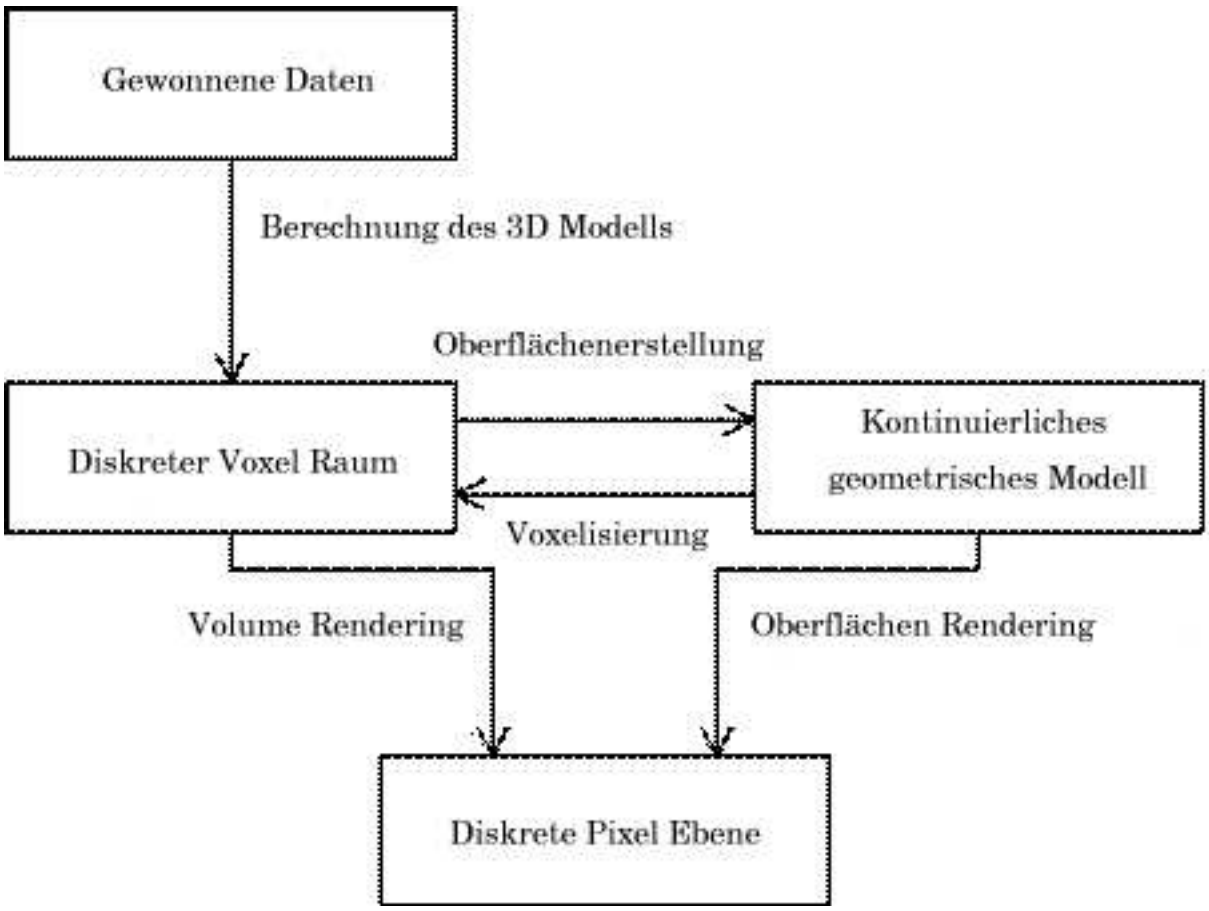


Abbildung 3.9: Schritte der Volumenvisualisierung

Ein Problem der Volumenvisualisierung stellt die Menge der Eingabedaten dar: Sie umfassen meist mehrere Megabytes. Die Gewinnung der Eingabedaten erfolgt durch Messung oder Simulation.
Es lassen sich viele Datenformate darstellen:

- Skalardaten
- Vektordaten
- Tensorfelder

Ein Tensorfeld der Dimension null entspricht dabei einem Feld von Skalardaten, ein Tensorfeld der Dimension eins einem Vektorfeld.
Es gibt verschiedenste Arten der Darstellung von Volumendaten:

- Punkte

- Kurven
- Vektoren
- Linien
- Konturen
- polygonale Netze
- gekrümmte Oberflächen
- Iso-Flächen
- Volumenzellen
- Voxels

Im Bereich der medizinischen Visualisierung wird oft die Anforderung der Darstellung verschiedenster Materialien wie Knochen und Gewebe gestellt. Damit ergibt sich das Problem der Datenklassifikation, da für eine übersichtliche Darstellung die verschiedenen Materialien mit unterschiedlichen Farben dargestellt werden sollen, um die Aussagekraft der Darstellung zu erhöhen.
Im Gegensatz dazu sind die Eingabedaten für den Fall des Renderings von Wolkenstrukturen homogen. Die Eingabe ist in diesem Fall eine dreidimensionale Dichtefunktion des Wasserdampfes, die als Ergebnis des Modellierungsschrittes zur Verfügung steht.
Eine übliche Art der Datenrepräsentation ist die Voxeldarstellung, da sie die notwendige Diskretisierung der Daten realisiert. Die Voxeldaten werden dabei im Modellierungsschritt erzeugt. Dabei ist der Wert innerhalb eines Voxels konstant. Daraus ergibt sich zwangsläufig, daß die Qualität der Bilder mit der Anzahl der Voxel zunimmt, da bei einer höheren Auflösung der Daten bessere Bilder erzeugt werden können. Allerdings bedeuten mehr Voxel einen höheren Speicherbedarf und eine längere Laufzeit.

3.5.2 Projektion

Unter Projektion wird laut [Fell92] im allgemeinen die Abbildung eines Vektors in einem n-dimensionalem Vektorraum X auf Vektor im m-dimensionalen Unterraum U für $m < n$ verstanden. Für die Computergrafik interessant ist in der Regel der Fall $n = 3$ und $m = 2$. Dieser Fall wird in der Folge betrachtet.
Eine Projektion wird durch die Projektionsebene und Projektionszentrum festgelegt. Es gibt verschiedene Arten der Projektion:

- Zentralprojektion
- Parallelprojektion

Der Unterschied zwischen der Zentral- und Parallelprojektion besteht in der Lage des Projektionszentrums Z, das im Falle der Parallelprojektion im Unendlichen liegt. Da das Projektionszentrum der Position des Betrachters im dreidimensionalen Raum entspricht, bewirkt nur die Zentralprojektion eine verkleinerte Darstellung weiter entfernter Objekte. Dies entspricht dem alltäglichen Erleben, und daher wird für eine realistische Szeneriedarstellung in der Regel die Zentralprojektion verwendet.

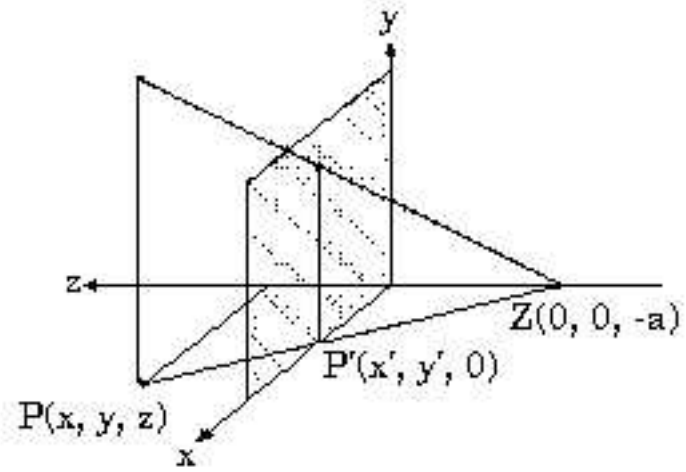


Abbildung 3.10: Zentralprojektion

Die programmtechnische Realisierung dieser Projektion hängt vom Zweck der Applikation ab. Für den Fall der Erzeugung zweidimensionaler Bilder wird gerne ein Ray-Casting beziehungsweise ein Ray-Tracing Algorithmus (siehe [Glas89]) eingesetzt.
Die Projektion läuft dabei so ab, daß für jedes Pixel ein Strahl vom Projektionszentrum durch den entsprechenden Punkt der Bildebene gelegt wird. Das Pixel wird dann im Fall des Ray-Casting in Abhängigkeit vom angetroffenen Volumentyp eingefärbt. Im Fall des Ray-Tracing wird der Strahl je nach der Oberflächenbeschaffenheit des Objekts eventuell weiterverfolgt.

Ein interessanter Aspekt ist dabei die Bestimmung des Punktes in den dreidimensionalen Weltkoordinaten, der einem gegebenen zweidimensionalen Punkt auf der

Bildebene entspricht. Zur Berechnung dieser Koordinaten sind folgende Parameter erforderlich:

- das Projektionszentrum in Weltkoordinaten
- die Blickrichtung
- der Abstand zwischen dem Projektionszentrum und der Bildebene
- der Richtungsvektor, der nach oben zeigt (Up-Vektor)
- die zweidimensionalen Punktkoordinaten auf der Bildebene

Die Blickrichtung entspricht dabei dem Normalvektor der Bildebene. Gesucht werden zwei dreidimensionale Richtungsvektoren, die den Koordinatenachsen im zweidimensionalen Koordinatensystem der Bildebene entsprechen.
Der Richtungsvektor für die x-Richtung ergibt sich aus dem Kreuzprodukt des Blickrichtungsvektors und des Up-Vektors. Der Vektor für die y-Richtung läßt sich in der Folge als das Kreuzprodukt von x-Richtungsvektor und Blickrichtungsvektor bestimmen.

3.5.3 Beschleunigung der Rendering Algorithmen

Ein generelles Problem beim Rendering von Bildern mit Verwendung von physikalischer Simulation bei der Bilderzeugung oder im besonderen von interaktiven Szenerien ist die Reduktion des Berechnungsaufwandes. Dafür gibt es verschiedene Ansätze wie beispielsweise

1. Einschränkungen der Realitätstreue der Simulation
2. Reduktion der Datenmenge
3. Optimierung der Rechenzeit des eingesetzten Algorithmus

Die ersten beiden genannten Punkte sind relativ einfach umzusetzen. Speziell der erste Punkt, da er meist mit einer Vereinfachung des eingesetzten Rendering Algorithmus einhergeht. Das Problem dabei ist allerdings, daß darunter fast immer die Darstellungsqualität leidet.
Eine Möglichkeit die unter Punkt eins genannte Methode einzusetzen ohne das die Bildqualität signifikannt darunter leidet, ist die Vereinfachung der physikalischen Simulation in Bereichen, in welchen sie sich kaum auf das Ergebnis auswirkt. Das Entdecken solcher Optimierungspotentiale ist allerdings eine schwierige Aufgabe, die oft nur durch ausgiebiges Testen des Algorithmus oder durch gute Kenntnis der Problemdomäne erreicht werden kann.

Der dritte Punkt ist sicher die beste Methode den Renderingprozeß zu beschleunigen, da sie meist nicht mit einer Reduktion der Bildqualität einhergeht. Allerdings sind algorithmische Verbesserungen meist nur mit genauer Kenntnis der Problemdomäne und der möglichen Verfahren machbar und stellen somit die schwierigste Methode der Performanzverbesserung dar. Verschiedene Möglichkeiten solcher Algorithmusverbesserungen werden im Abschnitt 4.3.2 beschrieben.
Für den Bereich der Wolkenmodellierung gibt es zahlreiche Möglichkeiten die Berechnungsgeschwindigkeit zu erhöhen, allerdings führt dies meist zu einer Reduktion der erzeugten Bildqualität.

- Reduktion der Voxelmenge bei volumenbasierten Ansätzen
- Vereinfachung der Berechnung der mehrfachen Lichtstreuung
- Reduktion der berücksichtigten Lichtquellen
- Keine Berücksichtigung von atmosphärischen Effekten
- Einsatz von stochastischen Methoden

Die Reduktion der Voxelmenge ist ein sehr gutes Beispiel für eine leicht umzusetzende Methode der Performanzverbesserung, die allerdings auch eine direkte negative Auswirkung auf die Bildqualität hat. Dies gilt vor allem für den Fall des direkten Volume Renderings. Der Grund wieso eine größere Anzahl von Voxeln sich positiv auf die Bildqualität auswirkt liegt auf der Hand:
Wenn die Materialstruktur beziehungsweise Dichte innerhalb eines Voxels konstant bleibt, hat man bei einer größeren Voxelanzahl eine höherere Auflösung des Ausgangsdatenmaterials für die Bilderzeugung.

Einen weiteren wichtigen Aspekt bei der Wahl der Auflösung des Voxelmodells stellt der Speicherbedarf dar, der oft zusätzlich zu den Anforderungen an die Laufzeit des Algorithmus die Anzahl der Voxel beschränkt. Man kann dem Problem des erhöhten Speicherbedarfs jedoch dadurch abhelfen, daß man ein geeignetes Datenformat für den Inhalt der Voxel wählt, beispielsweise einen numerischen Datentyp der weniger Bits für die Speicherung eines Wertes benötigt. Es ist hierbei die Frage zu klären welche Genauigkeit für die Werte, die in den Voxeln abgelegt werden, benötigt wird.
Ein interessanter Ansatz in diese Richtung wird in [DKYO00] verwendet um die Wolkenvolumendaten während der Simulation der Wolkenbewegung zu speichern. Es werden hierbei pro Voxel nur drei Bit benötigt. Um die gewünschte Bildqualität zu erreichen werden die Daten der Voxel vor der Bilderzeugung interpoliert.

Die Reduktion der Berechnung der mehrfachen Lichtstreuung ist ein sehr effektives Verfahren zur Verbesserung der Performanz von physikbasierenden Simulationen. Dieser Punkt wird in Abschnitt 4.3.2 genauer beschrieben.

Bei der Reduktion der berücksichtigten Lichtquellen wird das Ziel verfolgt den Berechnungsaufwand durch eine Verminderung der Realitätstreue bei der Simulation der Szenenbeleuchtung zu erreichen. Mögliche Ansätze dabei sind

- Vernachlässigung der ambienten Beleuchtung
- Vernachlässigung des von der Oberfläche des Geländes reflektierten Lichtes

Ein gutes Beispiel für den Einsatz einer stochastischen Methode für das Rendering von Wolken wird in [NiDN96] verwendet:
Um den Aufwand bei der Berechnung der mehrfachen Streuung zu reduzieren werden nicht alle möglichen Pfade des Lichts berechnet sondern nur zufällig ausgewählte innerhalb der Pfade mit einem hohen Beitrag zur Lichintensität. Ihr Beitrag wird dann mit ihrer Anzahl im Bezug auf die Gesamtzahl der Voxel gewichtet:

$$I = I_{calc} \cdot \frac{rn_2}{n_1} \tag{3.2}$$

Dabei bezeichnet n_1 die Zahl der berechneten Pfade, n_2 die Anzahl der Pfade mit einem hohen Beitrag zur Lichtintensität und r den prozentuellen Beitrag der n_2 Pfade zur Gesamtintensität. Das Verfahren wurde mit einem Wert von $r = 0,8$ berechnet. Es werden jene Lichtpfade, deren Beitrag höher ist als r zur Berechnung herangezogen.

Kapitel 4

Interaktive Wolkendarstellung

In diesem Kapitel werden Möglichkeiten beschrieben, Wolkendarstellungen in interaktiven, dreidimensionalen Szenarien einzusetzen, und die dabei auftretenden Probleme und mögliche Lösungsansätze diskutiert.
Im besonderen werden Verfahren beschrieben, die zu diesem Zweck eingesetzt werden können.
Außerdem wird der in der Implementierung umgesetzte Lösungsansatz skizziert.

4.1 Problembeschreibung und Lösungsansätze

Die bisher beschriebenen Ansätze der Wolkenmodellierung und Bilderzeugung hatten zum Ziel, statische Bilder zu erzeugen. Dafür wurden schon einige gute Verfahren veröffentlicht ([Gard85], [NiDN96], [Taxe99]). Es ist allerdings all diesen Verfahren gemeinsam, das eine gute Bildqualität auch zu einer entsprechend längeren Laufzeit der Bildberechnung führt.
Daher sind diese Verfahren auch für Erstellung statischer Bilder gedacht. Es ist allerdings wünschenswert diese Darstellungsqualität auch für interaktive Anwendungen zu erreichen. Eine weitere Anforderung die sich stellt, wenn die Darstellung von Wolkenformationen in den dreidimensionalen Raum übertragen wird, ist die Beeinflußung des Raums beziehungsweise der Landschaft durch die Wolken zu modellieren. Dies bezieht sich vor allem auf die Darstellung des Schattenwurfs der Wolkenformationen.
Um nun die widersprüchlichen Anforderungen einer realistischen Darstellung und einer ausreichenden Darstellungsgeschwindigkeit erfüllen zu können, gibt es folgende Möglichkeiten:

- Ontogenetische Bilderzeugung
- Einsatz von bekannten Computergrafikverfahren zur Beschleunigung

Die beiden Ansätze werden oft kombiniert eingesetzt, so beispielsweise in [Gard85]. In diesem Artikel wird nicht versucht, die Einflußnahme der Wolken auf die Darstellung der Szenerie auf physikalischer Ebene zu simulieren, sondern es wird das Ziel verfolgt, realistisch aussehende Bilder mit wenig Rechenaufwand zu erzeugen. Es werden dabei zahlreiche Varianten von Texturen eingesetzt um den Realismus der Darstellung bei vertretbarer Berechnungsgeschwindindigkeit zu erreichen.
In der Folge werden verschiedene Varianten von Texturen und Schattendarstellungen beschrieben.

4.1.1 Texturen

Der Einsatz von Texturen stellt ein Standardverfahren der Computergrafik zur Verbesserung der Qualität von dreidimensionalen Szenerien dar. Die grundlegende Motivation für den Einsatz von Texturen ist es, den Realismus in der Darstellung von komplexen Oberflächenstrukturen zu verbessern, ohne dabei die Komplexität des gegebenen polygonalen Modells zu erhöhen.
Ohne den Einsatz von Texturen wäre es beispielsweise für die Darstellung einer Ziegelwand notwendig, jeden Ziegel seperat zu modellieren. Dabei ist es aber noch nicht einmal garantiert, daß sich die Oberfläche der einzelnen Ziegel voneinander unterscheidet, da ja meißt dasselbe Modell für alle Ziegel verwendet wird, um den Modellierungsaufwand zu senken. Weiters führt ein solch komplexes polygonales Modell zu Leistungseinbußen bei interaktiven Applikationen.
Aus diesem Beispiel läßt sich ableiten, daß es drei prinzipiell widersprüchliche Zielsetzungen gibt, die beim Einsatz von Texturen alle bis zu einem gewissen Grad erfüllt werden können:

1. hoher Realismus der Darstellung
2. gute Leistung bei interaktiven Applikationen
3. geringer Modellierungsaufwand

Die Funktion einer Textur ist es nun ein komplexes polygonales Modell durch ein einfacheres zu ersetzen und die gewünschte Oberflächenstruktur als Bild auf das Polygon aufzubringen.
Der hohe Realismus der Darstellung läßt sich dabei leicht durch den Einsatz eines beliebig komplexen Bildes realisieren, da dieses im Prinzip nichts anderes darstellt als eine zweidimensionale Punktmenge, deren Verarbeitungsgeschwindigkeit durch die Grafikhardware unabhängig von ihrem Aufbau ist. Hiermit ist bereits Forderung Nummer zwei erfüllt. Der geringe Modellierungsaufwand ergibt sich wiederum daraus, daß die geforderte Detailliertheit der Darstellung bereits

durch das Bild abgedeckt wird, und man daher das zugrundeliegende geometrische Modell sehr einfach halten kann.

Es gibt nach [Watt89] verschiedene Attribute des darzustellenden Objektes, die durch das Aufbringen einer Textur verändert werden können:

- Farbe der Oberfläche
  Dies ist das am öftesten modifizierte Oberflächenattribut. Die Modifikation der Farbe erfolgt meist durch das Aufbringen von Bildern auf das Polygon. Dabei gibt es die Möglichkeiten einer vollständigen Farbersetzung durch die Textur oder einer Mischung der Texturfarbe mit der Grundfarbe des Polygons.

- Abbildung der Umgebung (*environment mapping*)
  Diese besondere Art von Textur dient zur Simulation von reflektierenden Gegenständen. Dabei wird ein Bild der Umgebung des Gegenstandes als Textur aufgebracht.

Abbildung 4.1: Kelch mit Umgebungsabbildung

- Deformation der Normalvektoren (*bump mapping*)
  Um die Form einer Oberfläche zu modifizieren können die Normalvektoren dieser Oberfläche verändert werden. Dies führt zu Verformungen, wie in Abbildung 4.2 gezeigt. Wenn man solche Verformungen durch konventionelle Modellierung realisieren müßte, wären der Aufwand und die benötigte Polygonmenge sehr hoch.

Abbildung 4.2: Kugel mit verteilten Normalvektoren

- Reflexivität der Oberfläche (*specularity*)
  Hierbei repräsentiert die Textur die variable Reflexivität einer Oberfläche. Allerdings wird diese Art von Textur selten verwendet.
- Transparenz der Oberfläche (*transparency*)
  Bei dieser Art des Einsatzes von Texturen handelt es sich um eine für die Wolkenmodellierung besonders interessante. Solche Texturen werden beispielsweise in [Gard85] eingesetzt um eine Schicht von Stratuswolken zu simulieren.

Abbildung 4.3: Transparenztextur als Stratus-Wolkenschicht

Texturabbildung

Die Texturabbildung beschreibt die Art und Weise wie eine gegebene Textur auf das darzustellende Objekt gelegt wird, das heißt die Zuordnung zwischen Texturkoordinaten und Objektkoordinaten.
Man unterscheidet nun nach der Anzahl der Texturkoordinaten:

- eindimensionale Texturen
- zweidimensionale Texturen
- dreidimensionale Texturen

Es werden allerdings hauptsächlich zweidimensionale Texturen eingesetzt, daher beschränken sich die folgenden Erläuterungen auf diesen Fall. Dabei haben die Texturkoordinaten zwei Dimensionen, u und v, wie aus Abbildung 4.4 ersichtlich ist.

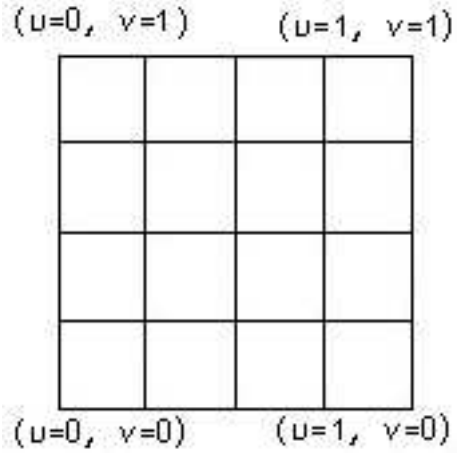


Abbildung 4.4: zweidimensionale Texturkoordinaten

Es ergeben sich nun bei der Abbildung einer zweidimensionalen Textur auf ein dreidimensionales Objekt folgende Probleme:

- Verzerrungen
- Streckungen und Stauchungen
- Wiederholungen von Texturen

Die Aufgabe der Abbildung ist es nun, für einen Punkt in Objektkoordinaten den passenden Punkt in Texturkoordinaten zu finden. Es gibt für die Abbildung von Texturen auf dreidimensionale polygonale Objekte zwei empirische Verfahren:

- Auffalten von benachbarten Polygonen in eine Ebene
- Zweiteilige Abbildung

Beim Auffalten des Objekts wird ein Polygon n solange um die Achse zwischen den Polygonen n und $n-1$ gedreht, bis die beiden Polygone in einer Ebene liegen. Dieser Vorgang wird für jedes Polygon des Objekts durchgeführt. Vor dem Auffalten des Polygons ist es allerdings notwendig, Informationen über die Nachbarschaftsbeziehungen zwischen den einzelnen Polygonen des Objekts zu speichern, da das Auffalten des Objekts nicht eindeutig ist.
In der Folge kann die Textur direkt auf die Ebene projiziert werden und somit kann jedem Polygoneckpunkt eine Texturkoordinate zugeordnet werden. Bei Anwendung dieses Verfahrens kann es allerdings zu Diskontinuitäten der Texturdarstellung an den Polygonkanten kommen.

Die zweiteilige Abbildung erfolgt in zwei Schritten:

1. Abbildung der zweidimensionalen Texturkoordinaten auf ein einfaches dreidimensionales Objekt, beispielsweise ein Zylinder.

   $$T(u,v) \rightarrow T'(x_i, y_i, z_i)$$

   Diese Abbildung wird als S-Abbildung bezeichnet.

2. Abbildung der dreidimensionalen Textur auf die Objektoberfläche

   $$T'(x_i, y_i, z_i) \rightarrow O(x,y,z)$$

   Diese Abbildung wird als O-Abbildung bezeichnet.

Als Zielkörper der S-Abbildung können mehrere Objekte dienen:

- Ebene
- Zylinder
- Würfel
- Kugel

Die S-Abbildung für den Fall des Zylinders läßt sich durch eine Formel folgendermaßen ausdrücken:

$$S_{Zylinder} : (u,v) \rightarrow (\frac{c}{r}\, u + \theta_0, dv + h_0) \tag{4.1}$$

c und d bezeichnen hierbei Skalierungsfaktoren, θ_0 und h_0 die Position der Textur auf dem Zylinder mit dem Radius r. Die inverse Abbildung sieht folgendermaßen aus:

$$S^{-1}_{Zylinder} : (\theta, h) \rightarrow (\frac{r}{c}\,(\theta - \theta_0), \frac{1}{d}\,(h - h_0)) \tag{4.2}$$

Es gibt verschiedene Möglichkeiten, die O-Abbildung zu realisieren. Hierbei wird jeweils der Farbwert am Schnittpunkt verwendet:

- Der Schnitt des vom Objekt reflektierten Sichtstrahls mit der Zylinderoberfläche. Dieses Verfahren wird bei der Abbildung der Umgebung eingesetzt.
- Der Schnitt der Oberflächennormale des zu texturierenden Objekts mit der Zylinderoberfläche.
- Der Schnitt einer Geraden durch den Objektmittelpunkt und den zu texturierenden Punkt an der Oberfläche mit der Zylinderoberfläche.
- Der Schnitt einer Geraden von dem zu texturierenden Punkt am Objekt ausgehend in Richtung des Normalvektors des Zylinders.

Es sind allerdings nicht alle möglichen Kombinationen von S und O-Abbildungen sinnvoll.

4.1.2 Schatten

Die Darstellung von Schatten in der Computergrafik ist ein wesentlicher Beitrag um den Realismus einer Darstellung zu erhöhen.
Der Bereich einer Szenerie, der im Schatten liegt, wird durch eines oder mehrere Objekte von der direkten Lichteinstrahlung abgeschnitten. Die Art der in einer Szenerie auftauchenden Schatten ist vom Typ und der Anzahl der verwendeten Lichtquellen abhängig. Sie können klare oder verschwommene Grenzen haben beziehungsweise sich in einen Kern- und einen Halbschattenbereich unterteilen. Der Kernschattenbereich ist dabei der Bereich der durch das schattenwerfende Objekt vollständig von der Lichtquelle abgeschnitten ist. Im Halbschattenbereich dagegen erreicht ein gewisser Anteil des Lichtes den schattierten Bereich. Hier ist ein gradueller Übergang der Intensität vom Kernschattenbereich her zu beobachten.
In der Computergrafik wird allerdings oft die vereinfachende Annahme einer Punktlichtquelle in unendlicher Entfernung angenommen, was für die Schattenberechnung bedeutet, das alle Lichtstrahlen parallel auf die Szenerie einfallen, wodurch keine Halbschattenbereiche entstehen.

Es gibt einige interessante Aspekte der Beeinflußung des Schattenwurfs durch Lichtquellen, die sich zur Beschleunigung der Bilderzeugung einsetzen lassen:

- In dem Fall, daß die Position des Betrachters der Position der Lichtquelle enspricht und die Lichtquelle die einzige in der Szenerie ist, sind keine Schatten sichtbar.
- Wenn in der Szenerie nur Punktlichtquellen vorkommen, gibt es keine Halbschatten.

- Wenn sich die relativen Positionen von Lichtquellen und Objekten nicht ändern, bleibt auch der Schattenwurf in der Szenerie derselbe.

Algorithmen zur Schattenerzeugung

Es gibt nach [Watt89] verschiedene Verfahren zur Schattenberechnung, die sich in Qualität und Berechnungsaufwand unterscheiden:

- Einfache Schatten auf einer Ebene
  Dies ist eine sehr einfache Methode, den Schattenwurf zu simulieren. Dabei wird das schattenwerfende Objekt auf die Ebene projiziert. Man kann dabei den zu schattierenden Bereich dadurch bestimmen, daß man Geraden in der Richtung des Lichteinfalls durch die Eckpunkte des Objekts legt und die Schnittpunkte dieser Geraden mit der Ebene berechnet. Die konvexe Hülle dieser Punkte begrenzt dann den schattierten Bereich.

- Erweiterter Scan-Line Algorithmus
  Dieser Algorithmus zur Schattenberechnung ist eine Erweiterung des in [Fell92] beschriebenen Scan-Line Algorithmus zur Verdeckungsberechnung von Polygonen. Daher läßt er sich auch sehr gut in Kombination mit diesem einsetzen.
  Der Algorithmus besteht aus zwei Schritten:

  1. Für jedes Polygon wird die Menge an Polygonen bestimmt, von denen es schattiert werden kann
  2. Bestimmung der Schattierung des gerade aktiven Scan-Line Segments

  Für den ersten Schritt werden alle Polygone auf die Oberfläche einer Kugel, in deren Mittelpunkt sich die Lichtquelle befindet, projiziert und alle Paare von Polygonen, die sich gegenseitig nicht schattieren können, aussortiert. Dieser Schritt ist notwendig, da bei n Polygonen ansonsten $n \cdot (n-1)$ Polygonpaare zu überprüfen wären.
  Der zweite Schritt des Schattierungsalgorithmus läuft gleichzeitig mit dem Scan-Line Algorithmus zur Verdeckungsberechnung ab. Es wird dabei für das gerade aktive Scan-Line Segment überprüft, ob es von einem anderen Polygon schattiert wird. Es gibt drei mögliche Ergebnisse dieser Überprüfung:

  1. Das Scan-Line Segment wird von keinem anderen Polygon schattiert.
  2. Die Schattierung betrifft das gesamte Scan-Line Segment. Der Algorithmus läuft normal weiter, aber die Intensität des schattierten Polygons wird herabgesetzt.

3. Das Scan-Line Segment wird teilweise schattiert. In diesem Fall wird das Segment unterteilt und die Teile rekursiv überprüft.

- Schattenvolumen
  Ein Schattenvolumen bezeichnet den Teil des Raumes, der von einem Polygon schattiert wird. Die Intensität eines Punktes wird verändert, wen er innerhalb dieses Volumenbereichs liegt, das heißt das schattierende Polygon liegt zwischen dem fraglichen Punkt und der Lichtquelle.
  Das Schattenvolumen wird von Geraden begrenzt, die von der Lichtquelle ausgehen und durch die Eckpunkte des Polygons gelegt werden. Man bestimmt den relevanten Teil des Schattenvolumens durch den Schnitt mit der Blickpyramide. Dieses Schnittvolumen wird wiederum durch Polygone begrenzt, die zwar unsichtbar sind, aber zur Überprüfung der Schattierung der sichtbaren Polygone verwendet werden.
  Dieser Ansatz der Schattenberechnung läßt sich beispielsweise in die Z-Buffer Verdeckungsberechnung integrieren, indem man die Datenstruktur für den Z-Buffer dahingehend erweitert, daß sie zur Überprüfung der Lage eines Polygons im Bezug auf die Grenzflächen des Schattenraumes herangezogen werden kann.

- Ableitung von Schattenpolygonen von Lichtquellentransformationen
  Dieser Algorithmus arbeitet im Objektraum und basiert auf der Entfernung verdeckter Polygone aus der Sicht der Lichtquelle. Folgende Schritte sind hierzu notwendig:

  1. Die Ansicht der Szenerie wird so transformiert, daß sich der Betrachter an der Position der Lichtquelle befindet.
  2. Die in der dieser Ansicht nicht sichtbaren Teile der Szenerie liegen im Schatten und werden als schattiert abgespeichert.

  Da die zu schattierenden Polygone von der Position des Betrachters unabhängig sind, läßt sich dieser Ansatz besonders gut für animierte Szenerien einsetzen.

- Schatten Z-Buffer
  Bei diesem Algorithmus wird ein seperater Z-Buffer für die Schattenberechnung einer Lichtquelle eingesetzt. Zuerst wird dieser Schatten-Z-Buffer gefüllt, indem die Szene aus der Sicht der Lichtquelle betrachtet wird. In einem zweiten Schritt wird die Szene mit dem normalen Z-Buffer gerendert. Wenn ein Punkt sichtbar ist, wird er in die Koordinaten für den Schatten-Z-Buffer transformiert und dann sein z-Wert mit dem aus dem Schatten-Z-Buffer verglichen. Falls der Wert des Punktes größer ist als der im Schatten-Z-Buffer liegt der Punkt im Schatten, ansonsten wird er normal dargestellt.

4.2 Dreidimensionale Wolkenanimation

Einen bereits publizierten Ansatz zur Darstellung von Wolkenformationen in dreidimensionalen Umgebungen stellt [DKYO00] dar. Es geht hierbei um eine effiziente Methode der Wolkenanimation. Dabei ist es möglich, das Enstehen, die Bewegung und die Auflösung von Wolken sowie Lichtstrahlen durch die Wolken zu simulieren.
Das beschriebene Verfahren setzt sich aus zwei Schritten zusammen:

1. Simulation
2. Rendering

Diese beiden Schritte müssen für jedes Bild der Animation durchgeführt werden. Der simulierte Bereich der Atmosphäre wird durch ein Voxelvolumen repräsentiert. Zur Speicherung der Daten reichen drei Bits pro Voxel aus. Die Bits erfüllen dabei folgende Funktionen:

- Bit *cld*: In diesem Voxel existiert eine Wolke
- Bit *hum*: In diesem Voxel gibt es genug Wasserdampf für die Wolkenbildung
- Bit *act*: Das Voxel ist bereit für den Phasenübergang zwischen Wasserdampf und Wasser.

Die Voxel werden nun im Simulationsschritt als ein zellulärer Automat betrachtet. Der Werte der Boolean-Variablen in den Wolkenvoxeln zum Zeitpunkt t lassen sich durch festgelegte Überführungsfunktionen aus den Werten zum Zeitpunkt $t-1$ berechnen. Auf diese Weise können die Wolkenenstehung, Wolkenbewegung und Wolkenauflösung simuliert werden.
Wenn nun die Simulation für ein Bild abgeschlossen ist, wird das entsprechende Bild unter Verwendung von OpenGL Funktionen gerendert. Davor werden noch die binären Werte der Voxel interpoliert um eine passende Dichteverteilung zu errechnen.
Aus dieser Dichtefunktion werden Dichtekugeln extrahiert, die für die Bilderzeugung verwendet werden. Es wird nun für jede Dichtekugel ein Polygon verwendet, dessen Normalvektor auf den Betrachter zeigt. Nachdem der Rest der Szene ohne die Wolken dargestellt wurde, werden nun die Texturen für jede Dichtekugel in den Framebuffer addiert.
Dieser Ansatz ist vor allem dann schnell, wenn Hardwarebeschleunigung für OpenGL Befehle zur Verfügung steht. Mit heutiger Hardware ist es dabei möglich ein Bild innerhalb einer Minute zu berechnen. Diese Methode geht allerdings von einem statischen Betrachter aus.

4.3 Lösungsansatz

4.3.1 Grundidee

Im Gegensatz zu dem in [DKYO00] beschriebenen Verfahren ermöglicht es der im Zuge dieser Arbeit implementierte Ansatz, den Betrachter beliebig durch eine gegebene Landschaft zu navigieren. Dabei wird die Ansicht der vorhandenen Wolkenformationen laufend seiner Position angepasst. Diese Anpassung geschieht mit einer Geschwindigkeit, die bei Einsatz einer hardwarebeschleunigten OpenGL Implementierung eine Navigation durch die Szenerie in Echtzeit ohne Probleme möglich macht.
Gleichzeitig wird das Ziel verfolgt, die Wolkendarstellung durch Simulation der physikalischen Vorgänge so realistisch wie möglich zu gestalten. Diese beiden widersprüchlichen Ziele können durch den Einsatz von Texturen erreicht werden. Dabei werden zuerst für jede Wolke Ansichten aus allen Richtungen berechnet, die dann als Texturen verwendet werden.
Für die Bildberechnung wird im wesentlichen der in [NiDN96] beschriebene Algorithmus eingesetzt, der sehr gute Ansichten der Wolken liefert. Der Algorithmus berücksichtigt die Absorption und Streuung des einfallenden Lichts durch die Wolke.
Die Wolke ist als diskrete dreidimensionale Dichtefunktion in Form von Voxeln gegeben. Jedes Voxel hat dabei eine konstante Dichte. Der Algorithmus berechnet die Bilder in mehreren Schritten:

1. Lichtabsorption
2. Lichtstreuung
3. Bilderzeugung

Diese Schritte werden sequentiell auf das zu berechnende Voxelvolumen angewandt. Im ersten Schritt wird bestimmt, wieviel Lichtintensität in jedem Voxel nach der Lichtabsorption auf der bisher passierten Strecke der Wolke noch vorhanden ist.
Im zweiten Schritt wird die Streuung dieser Intensitäten berechnet, das heißt es wird das von anderen Voxeln in das fragliche Voxel eingestreute Licht auf Basis der nicht absorbierten Intensitäten und der Phasenfunktion berechnet.
Der dritte Schritt erzeugt das Bild unter Verwendung von Volume Rendering. Hierbei wird das Ray-Casting Verfahren [Elvi92] angewandt.

Als erstes wird die Lichtabsorption durch die Wassertropfen der Wolke berech-

net:

$$\tau(S) = \int_0^S \beta\rho(s)ds \tag{4.3}$$

S bezeichnet dabei die Länge der Strecke, die der Lichtstrahl in der Wolke zurücklegt, β den Absorptionskoeffizienten und ρ die Dichte der Wolke im gerade traversierten Teil der Wolke. Der Absorptionskoeffizient β hat dabei einen Wert von $16,33/km$ [Taxe99] für eine typische Wolke.
Die Lichtintensität nach der Absorption berechnet sich dann zu

$$I_p = I_c \cdot exp(-\tau(\overline{PP_c})) \tag{4.4}$$

I_c bezeichnet hierbei die Lichtintensität vor dem Absorptionsvorgang, I_p diejenige danach.
Die zweite wichtige Art der Beeinflußung des Lichts durch die Wolke ist die Lichtstreuung an den Wassertropfen in der Wolke. In der Realität kommt es hierbei zu einer vielfachen Streuung, die laut [Bohr95] aufgrund der hohen optischen Dichte von Wolkenformationen den Haupteinflußfaktor für ihr Aussehen darstellt.
Daher ist es unerläßlich, die mehrfache Lichtstreuung durch die Wassertropfen in der Wolke in der Bildberechnung zu berücksichtigen. Im Falle der vorliegenden Implementierung wurde die Berechnung der Lichtstreuung aufgrund von Effizienzüberlegungen jedoch nur bis zur dritten Ordnung durchgeführt, da der prozentuelle Beitrag bei höheren Ordnungen der Streuung immer geringer wird.
Da die Streuung an Wassertropfen nach den Gesetzmäßigkeiten der Mie Streuung erfolgt ist sie stark anisotrop und muß durch eine Phasenfunktion beschrieben werden. Die Phasenfunktion besagt, welcher Anteil des einfallenden Lichtes in eine Richtung weggestreut wird, wobei dies vom Winkel zwischen Einfallsrichtung und Ausfallsrichtung des Lichtes abhängt. Der Realismus dieser Berechnung läßt sich noch dadurch verbessern, daß man die inhomogene Zusammensetzung der Wolken durch Verwendung mehrerer Phasenfunktionen berücksichtigt, die durch das Mischungsverhältnis der Partikel in der Wolke gewichtet werden. Hierbei kann auch die durch Luftmoleküle verursachte Raleigh Streuung berücksichtigt werden.
Die schließlich in der Berechnung verwendete Phasenfunktion ist daher eine gewichtete Summe aus Teilfunktionen:

$$F(\theta) = \sum_{i=0}^{K} w_i F_i(\theta) \tag{4.5}$$

Dabei ist K die Anzahl der Teilfunktionen und w_i der Gewichtungsfaktor für die Teilfunktion i. Im Falle der vorliegenden Implementierung wurde die Mie Streuung mit 95 Prozent und die Rayleigh Streuung mit 5 Prozent gewichtet.

Bei dem vorgestellten Algorithmus wird folgende Phasenfunktion für die Mie Streuung eingesetzt:

$$F(\theta, g) = \frac{3(1-g^2)}{2(2+g^2)} \frac{1+cos^2\theta}{\sqrt{(1+g^2-2g\,cos\,\theta)^3}} \tag{4.6}$$

g ist dabei ein Asymetriefaktor, der vom Zustand der Wolke und der Wellenlänge des Lichts abhängt. Die Streuung des einfallenden Lichts wird im Falle der Mie Streuung allerdings kaum von der Wellenlänge des Lichtes beeinflußt, weshalb Wolken in Weiß- beziehungsweise in Grautönen sichtbar sind. Für den Fall $g = 0$ entspricht die Phasenfunktion der der Rayleigh Streuung.
Nun beschränkt sich die Berechung der gestreuten Lichtintensitäten nicht nur auf die einfache Streuung, da das einfallende Licht von vielen Partikeln gestreut wird bevor es den Betrachter erreicht.
Die Formel für die Gesamtintensität des Lichtes, das aus einer Richtung kommt sieht unter Berücksichtigung von Absorption und Streuung folgendermaßen aus:

$$\begin{aligned} I(P,\omega) = {} & I(P_S,\omega)\, exp(-\tau(\overline{PP_S})) + \\ & + \int_{s=0}^{S} (\beta\rho(s)\, exp(-\tau(\overline{P(s)P})) \cdot \frac{1}{4\pi} \int F(\theta) I(P,\omega')d\omega')ds \end{aligned} \tag{4.7}$$

Der Teil in der ersten Zeile beschreibt dabei die Lichtabsorption durch die Wolkenpartikel, die zweite Zeile das eingestreute Licht, das seinerseits wieder durch Absorption gedämpft wird. Für die Bildberechnung interessant ist jenes Licht, das in die Richtung des Betrachters gestreut wird. Zu seiner Berechnung wird die Phasenfunktion herangezogen, die den Anteil des gestreuten Lichtes für einen gegebenen Winkel liefert. Zur Bestimmung des eingestreuten Lichts werden die Intensitäten die aus den verschiedenen Richtungen auf das Voxel gestreut werden mit der Phasenfunktion für den entsprechenden Winkel, den die Lichtrichtung mit der Blickrichtung des Betrachters einschließt, multipliziert und über die möglichen Einfallsrichtungen integriert.
Dies wäre für eine physikalisch exakte Simulation bei jedem Voxel durchzuführen, was aber aufgrund des Rechenaufwandes nicht realisierbar ist, da vor allem die Simulation der mehrfachen Streuung extrem rechenaufwendig ist. Es müssen daher einige Optimierungen in der Berechnung vorgenommen werden, um die Bildberechnung auf einem einem gewöhnlichen Desktop-PC in vernünftiger Zeit durchführen zu können.

4.3.2 Optimierungen

Es besteht folgende Abhängigkeit zwischen der Anzahl der Auswertungen der Phasenfunktion $n_{F(\theta)}$, der Anzahl der Voxel N und der Ordnung der Lichtstreuung

O:

$$n_{F(\theta)} = O \cdot N \cdot (N-1)^{O-1} \tag{4.8}$$

Daraus ist ersichtlich, daß die Anzahl der Auswertungen der Phasenfunktion exponentiell abhängig von der Ordnung der berechneten Lichstreuung ist. Weiters kommt hinzu, wie aus Gleichung 4.6 ersichtlich ist, daß die Berechnung der Phasenfunktion vor allem aufgrund der enthaltenen Winkelfunktionen sehr zeitaufwendig ist. Daher und auch weil der Einfluß der Lichtstreuung auf das erzeugte Bild mit steigender Ordnung immer mehr abnimmt, ist es notwendig die Streuungberechnung auf eine geringe Anzahl von Ordnungen zu beschränken. In [NiDN96] wird als maximale Ordnung der berechneten Streuung drei vorgeschlagen. Dieser Wert führt zu guten Ergebnisbildern. Da diese Optimierung aber lange noch nicht ausreicht, um den Algorithmus in einer angemessenen Zeit terminieren zu lassen, ist es notwendig, noch weitere Optimierungen vorzunehmen.
Hierzu gibt es verschiedene Möglichkeiten:

- Bei der Streuungsberechnung werden nicht alle Voxel berücksichtigt
- Tabellierung der Phasenfunktion

Es wurde bei den Tests der vorliegenden Implementierung mit einer Größe des Wolkenvolumens von 100 x 100 x 100 Voxeln gearbeitet.
Um die Berechnungszeit für die Lichtstreuung auf ein vernünftiges Maß zu reduzieren ist es notwendig, bei Berechnung des in ein Voxel eingestreuten Lichtes nur die Voxel in seiner näheren Umgebung zu betrachten, da weiter entfernte Voxel nur einen sehr geringen Beitrag zur eingestreuten Intensität liefern. Die Größe des in der Implementierung verwendeten Bereichs beträgt 10 x 10 x 10 Voxel. Bei konstanter Sonneneinstrahlungsrichtung und konstanter Betrachterposition ist die prozentuelle Verteilung der eingestreuten Intensitäten aus den benachbarten Voxeln unabhängig vom gerade betrachteten Voxel konstant. Sie braucht daher nur einmal berechnet zu werden und wird in einem Referenzmuster gespeichert.

Die Tabellierung der Phasenfunktion stellt die zweite Möglichkeit der Optimierung der Streuungsberechnung dar. Sie ist aus mehreren Gründen ein wichtiger Optimierungsschritt:

- Die Berechnung der Phasenfunktion ist sehr zeitaufwendig
- Die Berechnung wird oft benötigt
- Die Tabellierung benötigt nicht viel Speicher

Man kann die Funktion gut tabellieren, da sie nur einen Parameter hat, den Winkel θ. Die in der vorliegenden Implementierung gewählte Schrittweite beträgt 1°, was

zu einer Tabelle mit 360 Einträgen führt. Die Tabelle wird nun vor dem Start des eigentlichen Bilderzeugungsalgorithmus mit den korrekten Werten gefüllt, und während der Streuungsberechnung steht der Wert der Phasenfunktion für den entsprechenden Winkel in der Tabelle.

Kapitel 5

Implementierung

In diesem Kapitel wird ein Überblick über die implementierte Applikation zur interaktiven Wolkendarstellung gegeben.
Außerdem wird eine Beurteilung der Implementierung im Bezug auf Laufzeit und Speicherbedarf durchgeführt.

5.1 Überblick über die Implementierung

Die Grundidee ist in einem ersten Schritt die Bilderzeugung mit Hilfe physikbasierender Simulation zu realisieren und dann in weiterer Folge die erzeugten Bilder als Texturen einzusetzen, die auf Trägerpolygone abgebildet werden, um eine für interaktive Anwendungen ausreichende Geschwindigkeit zu erzielen.
Die Implementierung erfolgte in der Programmiersprache C unter Verwendung der Grafikbibliothek OpenGL.

5.1.1 Die Architektur

Das Gesamtsystem setzt sich aus mehreren voneinander unabhängigen Teilprogrammen zusammen, die über Textdateien Daten austauschen. Diese Gliederung entspricht der logischen Struktur des Systems und erhöht überdies die Übersichtlichkeit und Testbarkeit des Systems.
Folgende Teilprogramme wurden implementiert:

- Wolkengenerator
- Wolkenrenderer
- interaktive Szeneriedarstellung

Die folgende Darstellung gibt einen Überblick über die Architektur des Gesamtsystems:

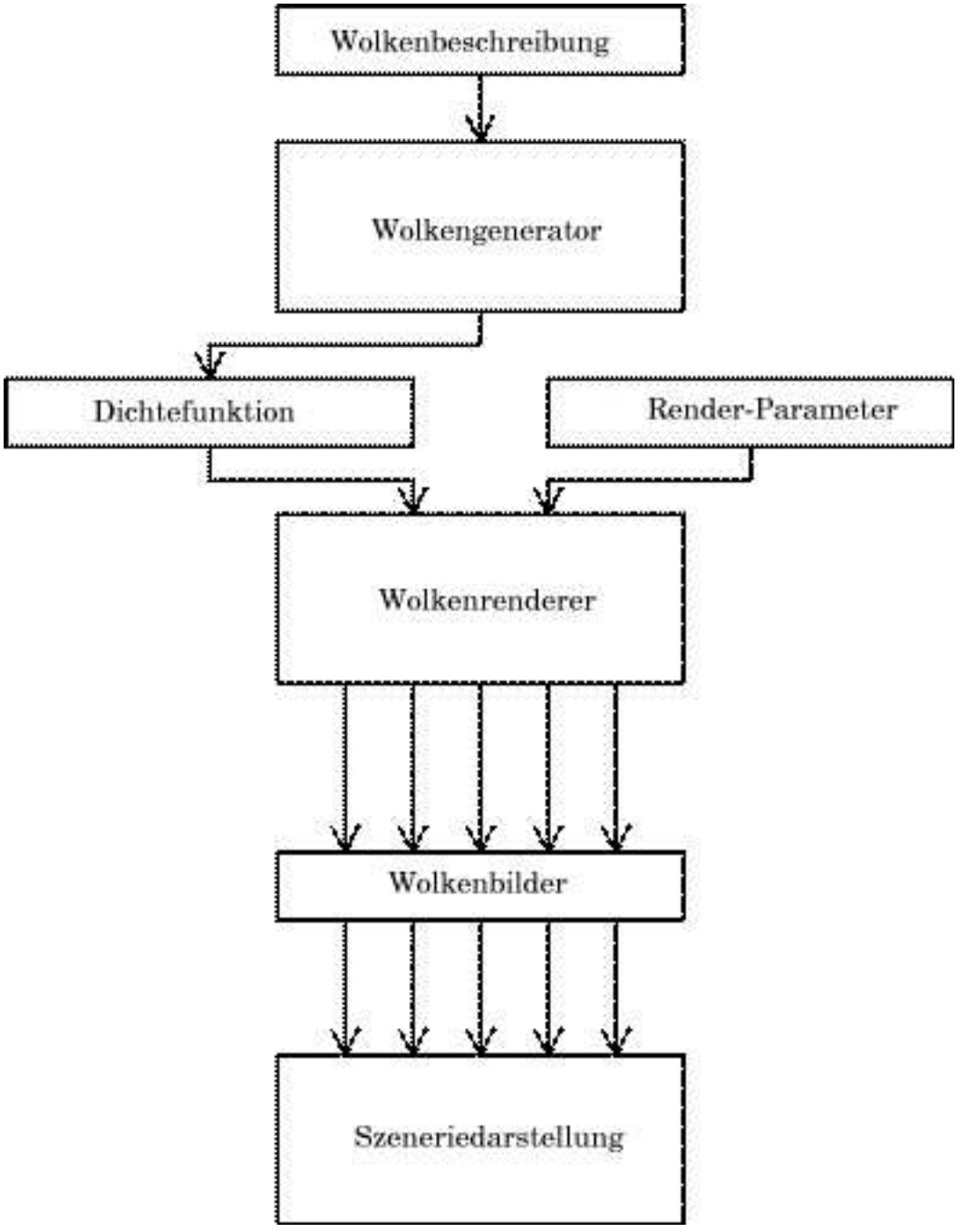


Abbildung 5.1: Systemarchitektur

Die Aufgabe des Wolkengenerators ist es mit einer Wolkenbeschreibungsdatei als Eingabe eine Dichtefunktion zu generieren, die die Wolke beschreibt.
Der Wolkenrenderer erzeugt aus der dreidimensionalen Dichtefunktion ein Bild der Wolke, das der Ansicht von einem gegebenen Betrachterstandpunkt aus entspricht.
In der interaktiven Szeneriedarstellung werden die Bilder der Wolke von verschiedenen Betrachterpositionen aus gesehen, die mit dem Renderer vorab berechnet wurden, als Texturen verwendet, um eine realistische Darstellung zu erzielen.

5.1.2 Der Wolkengenerator

Die Aufgabe des Wolkengenerators ist es aus einer modellhaften Beschreibung einer Wolke die entsprechende dreidimensionale Dichtefunktion in Form einer Voxelrepräsentation zu generieren.
Die Grundelemente im Aufbau einer Wolke sind dabei Ellipsoide, von denen beliebig viele zu einer Wolke vereinigt werden können.

Eingabedaten

Die Eingabedaten werden werden aus einer Wolkenbeschreibungsdatei ausgelesen und umfassen folgende Teile:

- Wertebereich der Objektkoordinaten
- Rauschfaktor für die gesamte Wolke
- Teilellipsoide der Wolke

Die Objektkoordinaten werden zur relativen Positionierung der Teilellipsoide benötigt.
Der Rauschfaktor bezieht sich auf die Anwendung eines in Abschnitt 3.2.2 beschriebenen Perlin-Rauschens auf die Dichtefunktion der gesamten Wolke.
Folgende Daten werden zur Spezifikation der Teilellipsoide verwendet:

- Position des Mittelpunkts in Objektkoordinaten
- Dichte im Mittelpunkt
- Dichte am Rand
- Längen der drei Halbachsen
- Rauschfaktor

Die Position des Ellipsoidmittelpunkts dient der Positionierung der Ellipsoide innerhalb der Wolke. Die Dichteverteilung innerhalb eines Teilellipsoids verläuft linear abnehmend von seinem Zentrum bis zur Oberfläche, deren Lage durch die Längen der drei Halbachsen und die Ellipsoidgleichung

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} + \frac{z^2}{c^2} = 1 \tag{5.1}$$

für den Fall, daß der Mittelpunkt des Ellipsoids im Ursprung des Objektkoordinatensystems liegt, gegeben ist. Dies kann durch eine eine einfache Translation des Teilellipsoids erreicht werden.
a, b und c bezeichnen hierbei die Längen der Halbachsen und x, y und z die Koordinaten des fraglichen Punktes. Falls der Punkt die Gleichung erfüllt ist er Element der Oberfläche des Ellisoids.
Der Rauschfaktor bezieht sich wiederum auf den Einsatz eines Perlin-Rauschens.

Die EBNF-Grammatik für die Wolkenbeschreibungsdatei ist in Anhang B.1 zu finden.

Dichteberechnung

Nachdem die Wolkenbeschreibungsdatei durch einen Parser eingelesen wurde, wird die Dichteverteilung der Gesamtwolke errechnet. Dazu wird zuerst der gesamte Voxelraum der Wolke mit einer Dichte von 0 initialisiert. Danach wird die Dichte der einzelnen Teilellipsoide zum Voxelraum hinzuaddiert.
Die Dichte eines Teilellipsoids in einem gesuchten Voxel ergibt sich aus der Gleichung 5.1 und dem Abstand des fraglichen Voxels vom Mittelpunkt des Ellipsoids. Vor der Berechnung wird zuerst der Ursprung des Koordinatensystems ins Zentrum des Ellipsoids verschoben um die Ellipsoidgleichung in der in 5.1 gegebenen Form verwenden zu können.
Als erster Schritt der Berechnung wird die Gerade durch das fragliche Voxel und den Koordinatensystemmittelpunkt mit dem Ellipsoid geschnitten. Daraus ergeben sich in jedem Fall zwei Schnittpunkte, da der Ursprung des Koordinatensystems mit dem Zentrum des Ellipsoids zusammenfällt.
In einem weiteren Schritt wird der Schnittpunkt bestimmt, dessen Abstand zum fraglichen Voxel geringer ist. Wenn nun der Vektor vom Voxel zum Schnittpunkt in allen Komponenten die selben Vorzeichen wie der Vektor vom Mittelpunkt zum Schnittpunkt aufweist, liegt das fragliche Voxel innerhalb des Ellipsoids, ansonsten außerhalb.
Für den Fall eines außerhalb liegenden Voxels ist der Dichtebeitrag des aktuellen Teilellipsoids 0, falls das Voxel innerhalb des Teilellipsoids liegt wird der Dichtebeitrag durch lineare Interpolation zwischen den Dichten im Ellipsoidzentrum

und an der Ellipsoidgrenze errechnet. Als Gewichtungsfaktor dient dabei der Abstand des Voxels vom Ellipsoidzentrum.
Auf diese Weise wird wird der Dichtebeitrag des aktuellen Teilellipsoids für alle Voxel der Dichtefunktion bestimmt. Dieser Schritt wird für alle Teilellipsoide wiederholt und die Dichten aufsummiert. Dabei wird darauf geachtet, daß die Dichte eines Voxels den Maximalwert von 1 nicht übersteigt.

Ausgabe

Die Ausgabe der Dichteverteilung erfolgt in eine Textdatei mit folgendem Format: In den ersten drei Zeilen der Datei stehen die Dimensionen des Voxelvolumens als Integer-Werte. Daran anschließend werden die Werte für die einzelnen Voxel zeilenweise im Float-Format im Wertebereich von 0 bis 1 aufgelistet.

5.1.3 Der Wolkenrenderer

Der Wolkenrenderer ist das größte und komplexeste Programm der Implementierung. Er erfüllt den Zweck der Bildgenerierung aus einem vom Wolkengenerator erzeugten Voxelmodell, das die Dichteverteilung der Wolke repräsentiert. Als weitere Eingabedatei werden noch die Renderparameter benötigt.

Die Renderparameter-Datei

In der Renderparameter-Datei werden folgende Daten abgelegt:

- Datendatei mit Dichtefunktion und Position der Wolke
- Position des Betrachters
- Blickrichtung des Betrachters
- Einfallsrichtung des Lichts
- Position der Projektionsebene
- Größe und Auflösung der Projektionsebene

In einem Durchlauf des Wolkenrenderers wird ein Bild einer Wolke aus einer festgelegten Betrachterposition erzeugt. Die Datendatei der Wolke beeinhaltet dabei die vom Wolkengenerator erzeugte Dichtefunktion. Die Position der Wolke wird dazu benötigt, um die Objektkoordinaten des Voxelmodells auf die Weltkoordinaten abzubilden, die für die Bilderzeugung benötigt werden.
Die Position und die Blickrichtung des Betrachters werden für die Projektion des

dreidimensionalen Raumes auf die Projektionsebene benötigt.
Die Einfallsrichtung des Lichts ist vor allem für die Berechnung der Lichstreuung notwendig. Das in der Implementierung verwendete Modell beschränkt sich auf die Betrachtung der direkten Sonneneinstrahlung wobei die Lichtquelle in unendlicher Entfernung angenommen wird und daher die Lichtstrahlen parallel auf das Voxelvolumen einfallen.
Die Position der Projektionsebene wird für die Abbildung des Voxelvolumnes auf die Bildebene benötigt, die mit Hilfe des Ray-Casting Verfahrens erfolgt.
Die Größe und Auflösung der Projektionsebene werden ebenfalls für diesen Schritt benötigt, da davon die Anzahl und die Richtung der zu verfolgenden Strahlen abhängt.

Das Format der Renderparameter-Datei ist aus Anhang B.2 ersichtlich.

Der Ablauf des Rendering-Algorithmus

Der verwendete Bilderzeugungs-Algorithmus folgt im wesentlichen dem in [NiDN96] publizierten Verfahren und weist folgende Schritte auf:

- Berechnung der Restlichtintensität nach der Lichtabsorption pro Voxel
- Berechnung der gestreuten Intensitäten
- Summierung der Intensitäten durch Ray-Casting

Bevor mit der Traversierung des Voxelvolumens begonnen wird, werden folgende Initialisierungsschritte vorgenommen:

1. Tabellierung der Werte der Phasenfunktion für die Lichtstreuung
2. Berechnung des Referenzmusters für die mehrfache Lichstreuung

Im ersten Schritt werden die Werte der Phasenfunktion der Lichtstreuung (Gleichung 4.6) für den Bereich von 0 bis 359° mit einer Schrittweite von 1° tabelliert. Damit kann während der Berechnung der Lichtstreuung der benötigte Funktionswert direkt aus dem Speicher geholt werden. Diese Maßnahme führt wie bereits in Abschnitt 4.3.2 erläutert zu einer massiven Beschleunigung der Streuungsberechnung. Dabei wird bei der Berechnung für den Asymetriefaktor g der Wert 0,75 verwendet, was einer starken Vorwärtsstreuung entspricht und die natürlichen Gegebenheiten gut annähert. Es ist allerdings notwendig die Funktionsergebnisse zu normieren, um den Inhalt der Tabelle als prozentuelle Werte verwenden zu können.
Als nächster Schritt wird das Referenzmuster für die Lichtstreuung initialisiert.

Da die Berechnung der mehrfachen Lichstreuung wie bereits in Abschnitt 4.3.2 dargelegt, äußerst rechenzeitaufwendig ist, wird auch in diesem Fall eine Vorabberechnung durchgeführt.
Dies ist möglich, da die prozentuelle Verteilung des gestreuuten Lichts nur von der Lichteinfallsrichtung und der Blickrichtung des Betrachters abhängig ist und diese Daten zu Beginn der Berechnung als Eingabe vorliegen und für das berechnete Bild konstant bleiben.
Das in der Implementierung eingesetzte Referenzmuster hat eine Größe von 10 x 10 x 10 Voxel. Bei der Vorabberechnung wird der Anteil am eingestreuten Licht durch die Summation der Beiträge des betreffenden Voxels aufgrund der Lichstreuung bis zur dritten Ordnung bestimmt. Im Referenzmuster werden dabei die Beiträge aufgrund der Streuung der zweiten und dritten Ordnung gespeichert. Diese Angaben beziehen sich auf die Anteile der Lichtintensität des Quellvoxels, die am fraglichen Voxel in die Richtung des Betrachters gestreut werden. Weiters wird dabei die Lichtabsorption auf den durch die Streuung entstandenen Pfaden berücksichtigt.

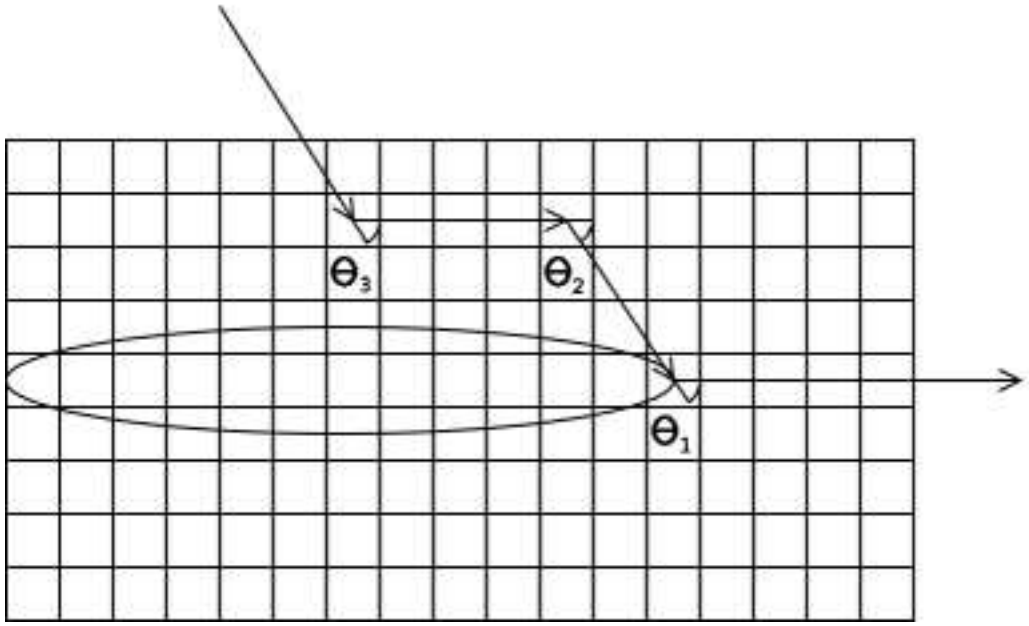


Abbildung 5.2: mehrfache Streuung

Die folgende Aufstellung zeigt, für welche Winkel für eine bestimmte Ordnung der Streuung die Phasenfunktion ausgewertet werden muß:

Ordnung	Winkel
1.	θ_1
2.	θ_1 und θ_2
3.	θ_1, θ_2 und θ_3

Die erste Ordnung der Streuung bezeichnet die direkte Lichteinstrahlung, die zweite Ordnung die Streuung an einem Voxel und die dritte Ordnung die Streuung über zwei Voxel.
Den richtigen Wert des Anteils eines Quellvoxels an der gesamten eingestreuten Intensität erhält man durch multiplikative Verknüpfung der Werte der Phasenfunktion für die betreffenden Winkel.

Nachdem die Vorabberechnungen abgeschlossen wurden wird als erste Operation auf dem Voxelvolumen die Absorptionsberechnung durchgeführt. Dabei wird eine wichtige Teilfunktion eingesetzt, die die Traversierung eines Voxels durch einen Strahl berechnet:

```
void travVoxel(Tripple dir, Tripple enter,
               Tripple* nextEnter, Tripple* nextVoxelDir,
               float* length);
```

Als Eingabeparameter erhält die Funktion den Eintrittspunkt des Strahls in das Voxel und den Richtungsvektor des Strahls und liefert drei Ausgabeparameter:

- den Austrittspunkt
- die relative Lage des nächsten zu betretenden Voxels im Bezug auf das aktuelle Voxel
- Die Länge der Strecke, die der Strahl in dem Voxel zurückgelegt hat.

Diese Funktion wird sowohl in der Absorptionsberechnung als auch im Ray-Casting Teil verwendet.

Bei der Absorptionsberechnung werden die Voxel die die Dichtefunktion der Wolke beinhalten entlang der Dimensionen des dreidimensionalen Arrays traversiert. Dabei wird in jedem Voxel ein rekursiver Algorithmus angewandt. Jedes Voxel hat während des Ablaufes des Algorithmus einen von drei Zuständen:

- bereits berechnet
- noch nicht berechnet
- außerhalb des Wolkenvolumens

Bevor mit der eigentlichen rekursiven Berechnung begonnen wird, wird das gesamte Volumen mit dem Zustand *noch nicht berechnet* initialisiert. Da die Lichtintensität in einem Voxel nach der Absorption von der Lichtintensität des zuvor traversierten Voxels abhängt, muß zuerst dieses Voxel berechnet werden. Dies geschieht dadurch, daß der Vektor der Lichtrichtung mit -1 multipliziert wird und dann die Voxeltraversierungsprozedur zur Bestimmung des Vorgängervoxels eingesetzt wird, da aufgrund der Verwendung der Gegenrichtung das Nachfolgervoxel dem Vorgänger entspricht.
Falls die Intensität des Vorgängervoxels bereits berechnet wurde, wird die Intensität des aktuellen Voxels durch Verwendung der Formel 4.4 mit Hilfe der Dichte und der Weglänge im aktuellen Voxel bestimmt.
Falls das Vorgängervoxel außerhalb der Wolke liegt, wird seine Intensität auf den Maximalwert 1 gesetzt und so wie im Fall ober verfahren.
Für den Fall des noch nicht berechneten Vorgängervoxels wird die Funktion für das Vorgängervoxel rekursiv aufgerufen.
Durch die Verwendung dieses rekursiven Algorithmus umgeht man das Problem der Bestimmung der Eintrittsvoxel der Lichtstrahlen in das Wolkenvolumen.

Als nächster Schritt wird die Streuungsberechnung durchgeführt, wobei das vorab berechnete Referenzmuster eingesetzt wird. Das eingestreute Licht wird durch die Multiplikation der nach der Absorption vorhandenen Restlichtintensität mit den Werten des Referenzmusters berechnet. Es wird wieder das gesamte Voxelvolumen traversiert, wobei das zentrale Voxel des Referenzmusters über dem aktuellen Voxel liegt. Die für den abschließenden Schritt verwendete Voxelintensität entspricht der Summe der von den Nachbarvoxeln, die von dem Referenzmuster überdeckt werden, eingestreuten Intensitäten.
Die Erzeugung des Resultatbildes erfolgt durch Verwendung des in Abschnitt 3.5.1 beschriebenen Ray-Casting Algorithmus. Die dabei eingesetzte Variante der Projektion wird in Abschnitt 3.5.2 erläutert.

5.1.4 Die Parser

Die Parser für beide Beschreibungsdateien, Wolkenbeschreibungsdatei und Renderparameterdatei sind nach dem Prinzip des rekursiven Abstiegs [Möss99] implementierte Parser.
Die in Anhang B.1 beziehungsweise B.2 angegebenen Grammatiken sind bewußt einfach gehalten, sind aber für ihren Zweck vollkommen ausreichend.
Zuerst wird der Inhalt der Dateien in den Hauptspeicher eingelesen, wo sie in der Folge von den Parsern einer Syntaxanalyse unterzogen werden und der Inhalt in den entsprechenden Datenstrukturen für die weitere Verarbeitung abgelegt wird.

5.1.5 Die interaktive Szenerie

Die Szenendarstellung ist eine einfache dreidimensionale Szenerie durch die sich der Betrachter in Echtzeit bewegen kann. Sie setzt sich aus einer einfachen Landschaft mit Himmels- und Wolkendarstellung zusammen.

Das Terrain

Das in der Szenerie verwendete Terrain ist ein einfaches reguläres Gitter für dessen Kreuzungspunkte Höhenwerte nach folgender Formel ermittelt werden:

$$h = \frac{1}{2} \cdot sin(\frac{i}{2}) \cdot cos(\frac{j}{2}) \cdot (\frac{z}{4} + 1) \tag{5.2}$$

Dabei bezeichnen i und j die Position des Punktes im zweidimensionalen Gitter und z eine Integer-Zufallszahl im Bereich zwischen 0 und 4. Der Zufallszahlengenerator wird dabei mit einem konstanten Wert initialisiert. Diese Funktion erzeugt ein leicht welliges Terrain, das für die hier benötigten Zwecke völlig ausreicht.
Die Schattierung des Terrains erfolgt nach dem Modell der Gouraud-Schattierung [Fell92]. Dafür müssen die Normalvektoren der Punkte des Gitters bestimmt werden. Dies erfolgt in mehreren Schritten:

1. Bestimmung der Normalvektoren der einzelnen Dreiecke
2. Berechnung der Normalvektoren der Punkte aus den Dreiecksnormalvektoren

Die Normalvektoren der Dreiecke werden als das Kreuzprodukt der durch die drei Eckpunkte aufgespannten Vektoren errechnet. Dabei ist noch darauf zu achten das die y-Komponente der Normalvektoren einen positiven Wert aufweist. Die Berechnung der Normalvektoren auf die Punkte erfolgt durch Mittelung der Normalvektoren der Dreiecke, die den fraglichen Punkt als Eckpunkt haben.

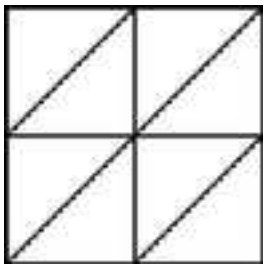


Abbildung 5.3: Dreiecke für die Mittelung der Normalvektoren

Bei der Mittelung wird auf die Gewichtung der Normalvektoren geachtet, je nachdem ob der betreffende Eckpunkt im Dreieck den rechten Winkel beeinhaltet. Abbildung 5.3 zeigt die Situation bei der Berechnung des Normalvektors für den mittleren Punkt. Dabei werden die Normalvektoren der links oben beziehungsweise rechts unten angrenzenden Dreiecke mit der doppelten Gewichtung versehen, da in diesen Dreiecken am fraglichen Punkt der rechte Winkel eingeschlossen wird.

Der Himmel

Die Darstellung des Himmels erfolgt durch einen einfachen Farbverlauf von unten nach oben. Dies dient der Vermeidung eines einfärbigen Hintergrunds, der bei dieser Art der Applikation aufgrund des Schwerpunkts auf der Wolkendarstellung nicht angebracht wäre. Andererseits erspart es dieser Ansatz atmosphärische Effekte auf physikalischer Basis zu simulieren, da dies den Rahmen dieser Arbeit übersteigt.
Der Farbverlauf wird in einer OpenGL RGBA Textur gespeichert, die an der Innenseite einer Box angebracht wird, die das Gelände und den Betrachter umschließt.

Betrachterbewegung und Texturanpassung

Die Navigation des Betrachters spielt eine sehr wichtige Rolle in der Implementierung. Die Position des Betrachters kann mit Hilfe der Cursor-Tasten verändert werden. Dabei wird die y-Koordinate der Betrachterposition automatisch der Geländestruktur angepasst.
Die Darstellung der Wolken erfolgt durch die Aufbringung einer Luminanz-Alpha Textur auf ein Polygon, das sich an der Position der Wolke befindet. Die Verwendung einer Luminanz-Alpha Textur ergibt sich aus den Anforderungen, die an die Darstellung der Wolkenbilder gestellt werden:

- Darstellung der Grauwerte aus den Wolkenabbildungen
- Möglichkeit eines transparenten Hintergrundes

Zur Abspeicherung der Grauwerte der Wolkenbilder reicht ein Luminanzwert aus. Besonders wichtig ist allerdings die Transparenz des Hintergrundes, da sonst die Form des Trägerpolygons sichtbar wäre, was im Fall der verwendeten quadratischen Polygonform sehr störend ist. Der Hintergrund der vom Wolkenrenderer erzeugten Bilder ist schwarz. Es stellt kein Problem dar, dem vom Wolkenrenderer erzeugten schwarzen Hintergrund der Bilder in der Textur den korrekten Alphawert zuzuweisen, es ist dabei allerdings notwendig, einen Grenzwert festzulegen,

ab welchem numerischen Farbwert die Transparenz wirksam werden soll, da im Bild keine scharfe Grenze zum Hintergrund vorliegt.
Die Auflösung der Bilder, die verwendet wurden, beläuft sich auf 256 x 256 Pixel. Da für eine standardkonforme OpenGL Implementierung nur die Verarbeitung von Texturen bis zu einer Größe von 64 x 64 Pixel garantiert sein muß [WNDS99], wurden die Bilder kachelförmig auf mehrere Texturen pro Bild aufgeteilt.
Das Problem, das sich bei der Verwendung statischer Bilder stellt, ist das die vorab berechneten Ansichten der Wolke jeweils nur für einen Standort des Betrachters gültig sind. Daher müssen mehrere Ansichten der betreffenden Wolke erstellt werden, jeweils von einer variierenden Betrachterposition.
Die Grundidee ist nun, das Polygon immer so auszurichten, daß der Betrachter frontal darauf blickt, um keine visuellen Artefakte entstehen zu lassen, da das Bild keine Tiefeninformation beeinhaltet. Gleichzeitig muß die auf dem Polygon dargestellte Textur ausgetauscht werden, wenn sich der Betrachter zu weit vom Standpunkt, der bei der Bilderzeugung angenommen wurde, entfernt. Um die Position des Betrachters in einer für die Erfüllung dieser Aufgaben geeigneten Form abzulegen, wurde eine Repräsentation in Polarkoordinaten gewählt, wobei sich die Wolkenposition im Zentrum des Koordinatensystems befindet.

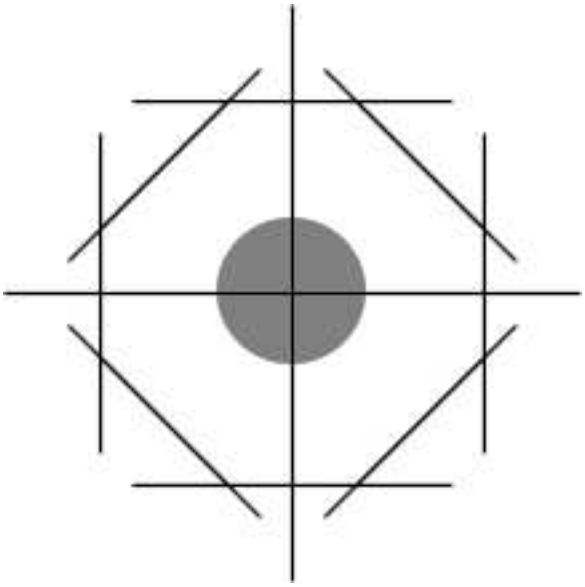


Abbildung 5.4: beispielhafte Positionen des texturierten Polygons

Abbildung 5.4 zeigt mögliche Ausrichtungen des Trägerpolygons aus der Vogelperspektive. Die gedachte Wolkenposition befindet sich dabei, wie durch den grauen Kreis angedeutet, im Ursprung des Koordinatensystems.

Die Ausrichtung des Polygons beschränkt sich dabei allerdings nicht nur auf den Breitenwinkel, sondern sie wird auch im Höhenwinkel an die Betrachterposition angepaßt. Dies wird vor allem bei Betrachterpositionen unterhalb der Wolke notwendig, da sich ansonsten störende Effekte durch die fehlende Tiefe der Darstellung nicht mehr verbergen lassen.
Für den Fall einer Betrachterposition unter der Wolke ist es außerdem notwendig, eine andere Textur zu verwenden, die mit einem Betrachterstandpunkt direkt unter der Wolke erzeugt wurde.

Ein Problem blieb bisher noch unberücksichtigt: Wenn ein Austausch von Texturen notwendig wird, erfolgt dieser Austausch ruckartig, was zu visuellen Artefakten führt und sich sehr störend in der Darstellung auswirkt.
Um dieses Problem zu umgehen, wird jeweils nicht nur ein texturiertes Polygon angezeigt, sondern vier gleichzeitig:

1. die aufgrund der aktuellen Betrachterposition korrekte Textur
2. die links benachbarte Textur
3. die rechts benachbarte Textur
4. die Textur mit der Wolkenansicht von unten

Die Polygone, auf die diese Texturen aufgebracht sind, liegen knapp hintereinander und werden durch Überblendung visuell überlagert. Dies geschieht dadurch, daß bei der Implementierung die Grundfarbe der Polygone durch ein RGBA Quadrupel als weiß mit einem variablen Alpha-Wert festgelegt wird, und auf diese Polygone die Farbe der Textur aufmoduliert wird. Die Ausblendung des schwarzen Hintergrundes der vom Wolkenrenderer erzeugten Bilder erfolgt durch einen empirisch festgelegten Grenzwert, der dahingehend interpretiert wird, daß alle dunkleren Werte als transparent gesetzt werden.
Die visuelle Gewichtung der Texturen erfolgt durch dynamische Anpassung des Alpha-Wertes der Texturen und falls notwendig, ihrer Umsortierung, da bei der Verwendung von transparenten Polygonen sich ihre Lage zueinander im Bezug auf den Betrachter auf die Darstellung auswirkt.
Daher wird die für das vorderste Polygon zu verwendende Textur in Abhängigkeit vom Breitenwinkel folgendermaßen bestimmt:

- Die Textur, die für den Winkelbereich, in dem sich der Betrachter befindet, gültig ist, ist immer die vordere Textur.
- Diejenige Nachbartextur, deren Grenze näher beim Betrachter liegt, wird dahinter angeordnet.

- Die verbleibende dritte Textur ist die hinterste.

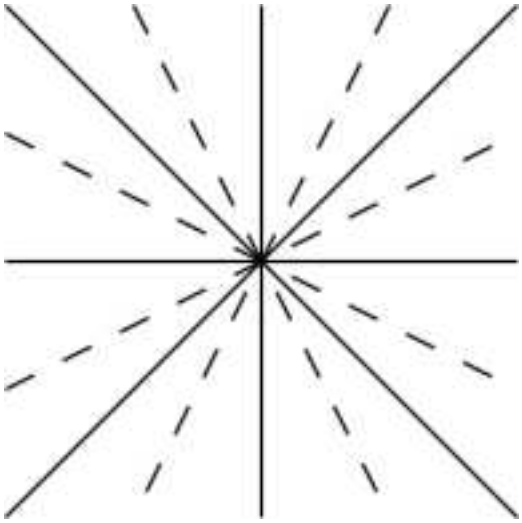


Abbildung 5.5: Grenzwinkel für den Texturaustausch

In Abbildung 5.5 sind die Grenzfälle für den Breitenwinkel mit strichlierten Linien eingezeichnet. Passiert der Betrachter während seiner Bewegung eine dieser gedachten Linien oder durchquert er den Mittelpunkt des Koordinatensystems wird ein Texturaustausch vorgenommen.
Die Anpassung der Alpha-Werte und der Sortierung der dargestellten Texturen wird bei jeder Positions- beziehungsweise Blickrichtungsänderung vorgenommen. Die Bestimmung dieser Werte läßt sich am besten anhand der Abbildung 5.5 erklären. Die durchgezogenen Linien bezeichnen jeweils einen Normalvektor auf eine Textur. Die Gewichtung der Texturen erfolgt nun in Abhängigkeit der Winkeldifferenz zwischen der ensprechenden durchgezogenen Linie und dem Breitenwinkel des Betrachters. Je geringer der Absolutbetrag der Winkeldifferenz ist, desto stärker wird die aktuelle Textur im Vergleich zu ihren beiden Nachbartexturen gewichtet. Die Gewichtung der beiden Nachbartexturen untereinander und damit die Sortierung ihrer Trägerpolygone ergibt sich aus dem Vorzeichen der Winkeldifferenz.
Diese Berechnungsvorschrift muß natürlich für den Spezialfall der Textur, deren Normalvektor den Breitenwinkel 0 einnimmt, adaptiert werden.

Schattenwurf und Wolkenbewegung

Ein weiterer wichtiger Punkt um den Realismus der Darstellung zu erhöhen ist die Berücksichtigung des Schattenwurfs der Wolken. Im vorliegenden Fall wurde

dies durch den Einsatz einer Luminanztextur erreicht.
Dazu wird vom Bilderzeugungsalgorithmus eine Ansicht der Wolke generiert, bei der die Position des Betrachters am Schnittpunkt des Terrains mit einer gedachten Gerade durch die Mitte des Voxelvolumens mit dem Richtungsvektor der Lichteinstrahlung liegt. Diese Luminanztextur wird in der Folge auf das Terrain aufmoduliert. Es ergeben sich dabei zwei Probleme:

- Korrekte Größe der Schattendarstellung
- Schattenwurf mehrerer Wolken

Das Problem der korrekten Größendarstellung ergibt sich dadurch, daß bei direkter Aufbringung der Schattentextur auf das gesamte Geländemodell der Schattenwurf ein Vielfaches seiner natürlichen Größe einnimmt. Hier wurde ein experienteller Lösungsansatz gewählt, wobei der korrekte Skalierungsfaktor für die Schattentextur in mehreren Textläufen bestimmt wurde. Da auch in der endgültigen Version die Schattentextur auf das gesamte Geländemodell abgebildet wird, war es notwendig die Größenanpassung durch eine direkte Veränderung der Pixeldaten der Textur zu realisieren. Dies geschieht dadurch, daß nur ein Teil der Pixel in die veränderte Textur übertragen wird.

Ein weiteres interessantes Problem beim Einsatz der Schattentextur ist die Darstellung des Schattenwurfs mehrerer Wolken. Da in der vorliegenden Implementierung aus Kompatibilitätsgründen nicht mit Mehrfachtexturen gearbeitet wurde, war es notwendig, den Schattenwurf aller vorhandenen Wolken in einer Textur abzulegen. Aus diesem Grund ist es auch unerläßlich, daß diese Textur auf das gesamte Terrain abgebildet wird. Die Vereinigung aller Schattentexturen zu einer Gesamtschattentextur geschieht in einem Vorverarbeitungsschritt, bevor diese Textur auf das Gelände abgebildet wird.
Dabei werden die Intensitätswerte der erzeugten Bilder aller Wolken nach ihrer Skalierung an der Position der Wolke von der Gesamtschattentextur, die zu Beginn mit der maximalen Intensität, was vollkommener Transparenz entspricht, initialisiert wurde, abgezogen.

Die Implementierung der Wolkenbewegung erfolgte als benutzergesteuerte Programmfunktion. Dabei werden einfach als Reaktion auf eine entsprechende Tastatureingabe die Trägerpolygone der Wolken in die entsprechende Richtung verschoben und der Schattenwurf der Wolken durch eine Verschiebung der Texturkoordinaten angepasst.

Navigation und Tastaturbelegung

Abschließend soll hier noch eine Übersicht über die Navigationsmöglickeiten des Betrachters in der interaktiven Szenerie gegeben werden:

Cursortasten oben/unten	Bewegung in/entgegen der Blickrichtung
Cursortasten links/rechts	Veränderung der Blickrichtung
Page Up/Down	Blick nach oben/unten
Begin/End	Rotation des Betrachterstandpunktes
F1/F2/F3/F4	Veränderung der Wolkenposition

5.2 Ergebnisse

Hier sollen einige Ergebnisbilder der Implementierung gezeigt werden, um eine Vorstellung von den erzielten Ergebnissen zu vermitteln.

5.2.1 Wolkengenerator

In Abbildung 5.6 wird eine vom Wolkengenerator erzeugte Dichtefunktion gezeigt. Die Darstellung wurde zum besseren Verständnis der Ausrichtung der Wolke im Raum in einen umschließenden Drahtmodellwürfel eingebettet. Die Darstellung des Dichtemodells erfolgte mit Hilfe von Grauwerten, wobei ein helleres Voxel eine höhere Dichte anzeigt. Daraus ergibt sich zwangsläufig eine relativ dunkle Färbung der Voxeldarstellung da sich im äußeren Bereich der Wolke die Voxel mit der geringeren Dichte des Wasserdampfs befinden. Die Abbildung zeigt das Dichtemodell aus verschiedenen Betrachtungswinkeln.
Man sieht in diesen Darstellungen die durch die Ellipsoidbausteile vorgegebene Grundstruktur deutlich, welche im Modellierungsschritt erzeugt wurde.

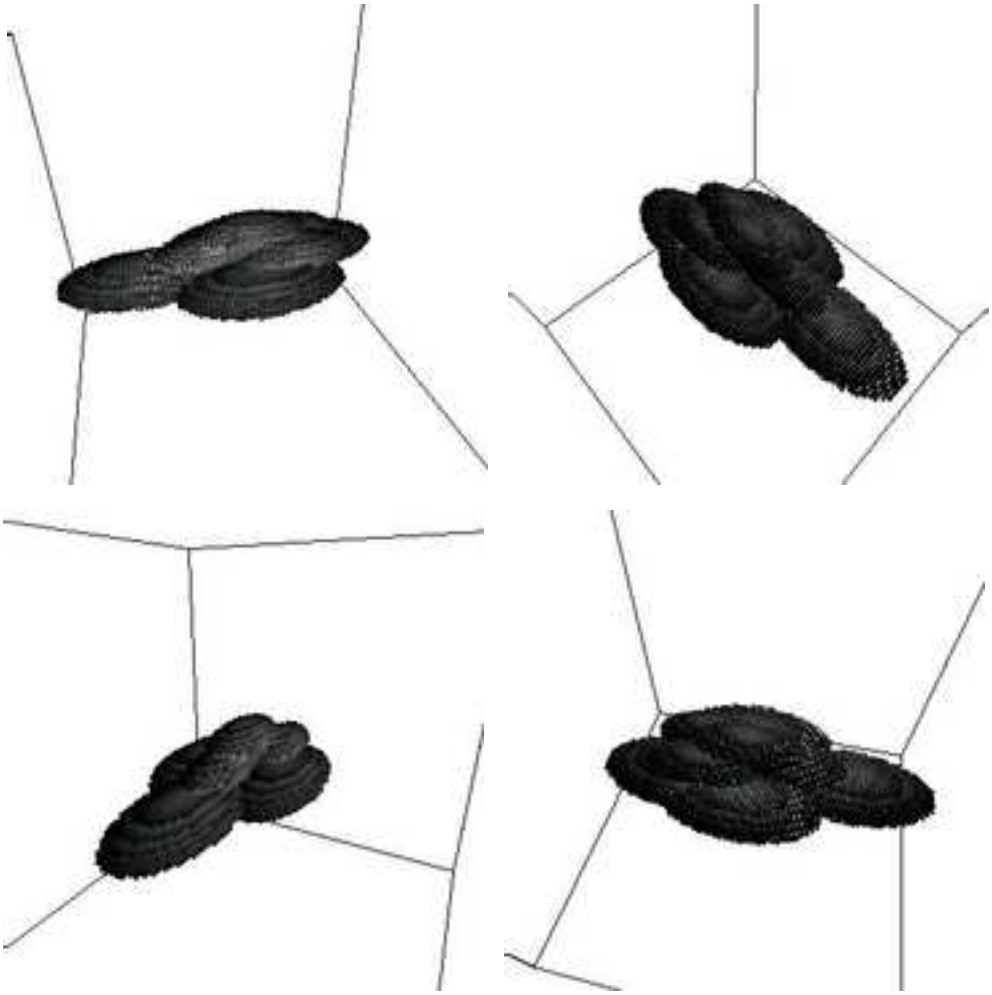


Abbildung 5.6: Dichtemodell einer Wolke

5.2.2 Wolkenrenderer

Die in Abbildung 5.7 dargestellten Bilder sind Ergebnisse des Wolkenrenderers, die aus dem oben gezeigten Dichtemodell erzeugt wurden.
Dabei wurde der Standort des Betrachters relativ zur Wolke für jedes Bild anders gewählt. Man sieht deutlich die längliche Form der modellierten Wolke, wobei sie in der Darstellung rechts unten direkt von der Seite betrachtet wird. Das Bild links unten zeigt eine Darstellung von vorne. Die beiden Bilder in der oberen Reihe wurden mit einer Betrachterposition diagonal zur Wolke erzeugt.

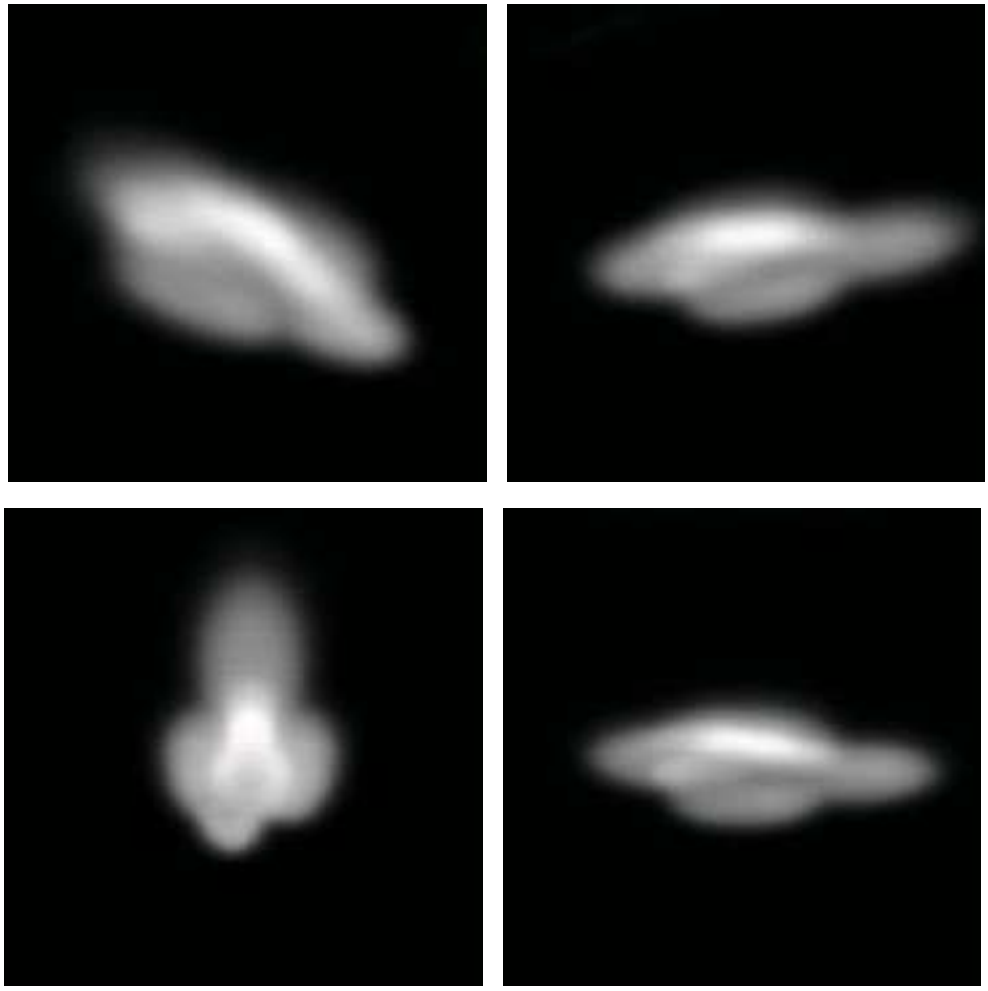
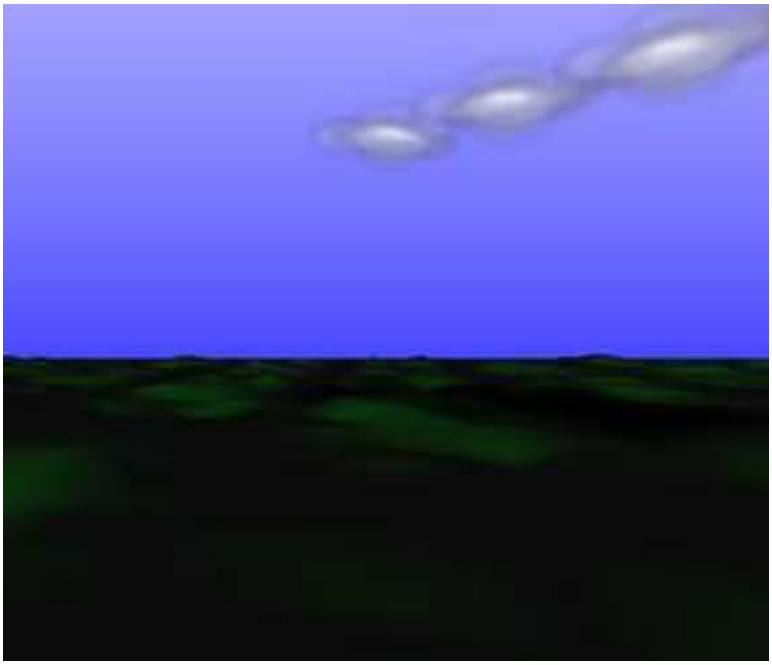


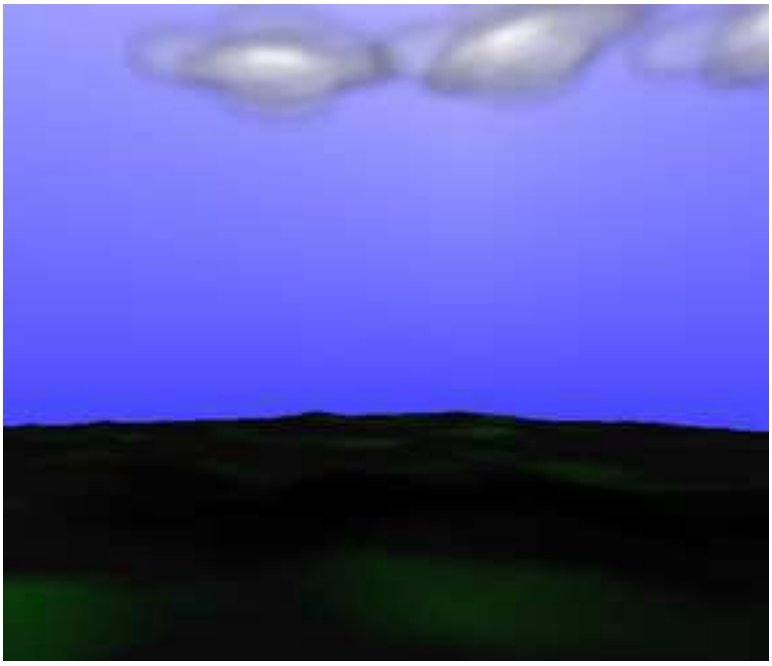
Abbildung 5.7: gerenderte Bilder

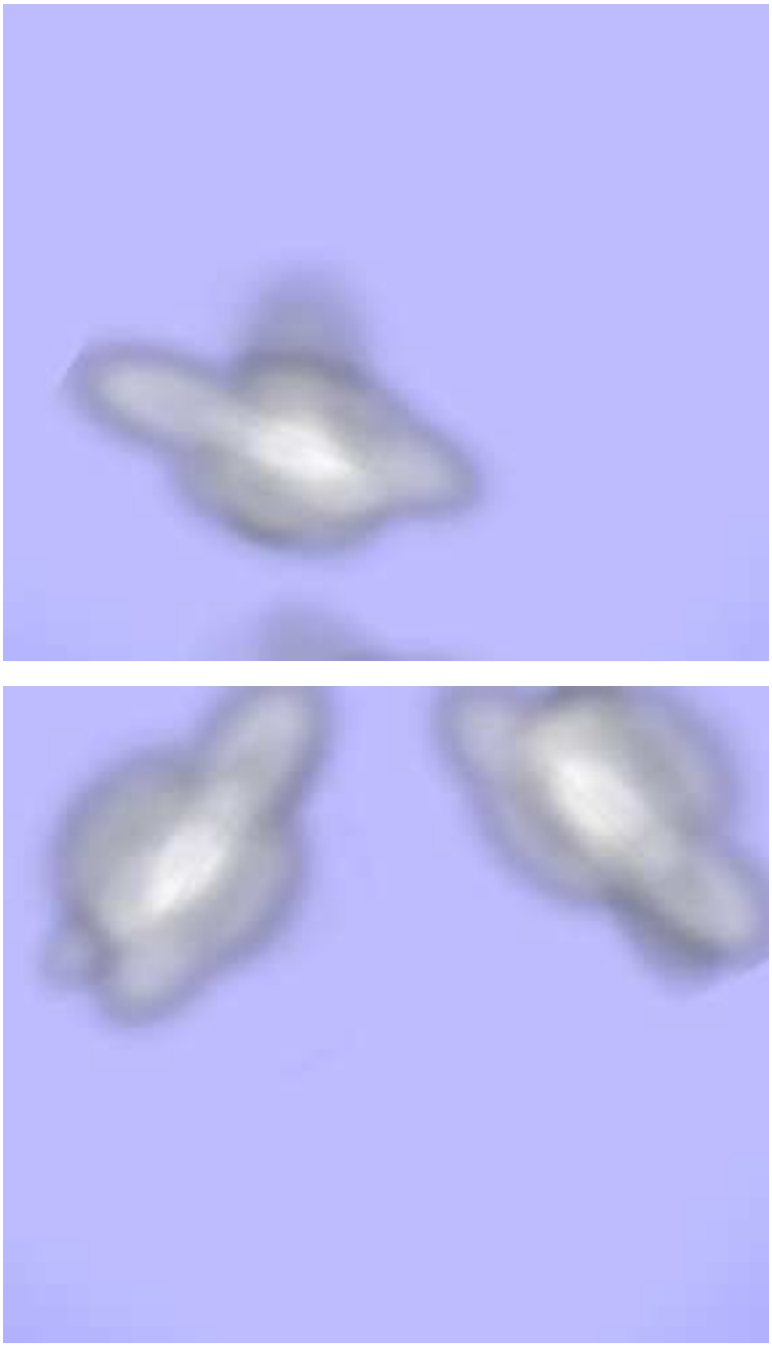
5.2.3 Szeneriedarstellung

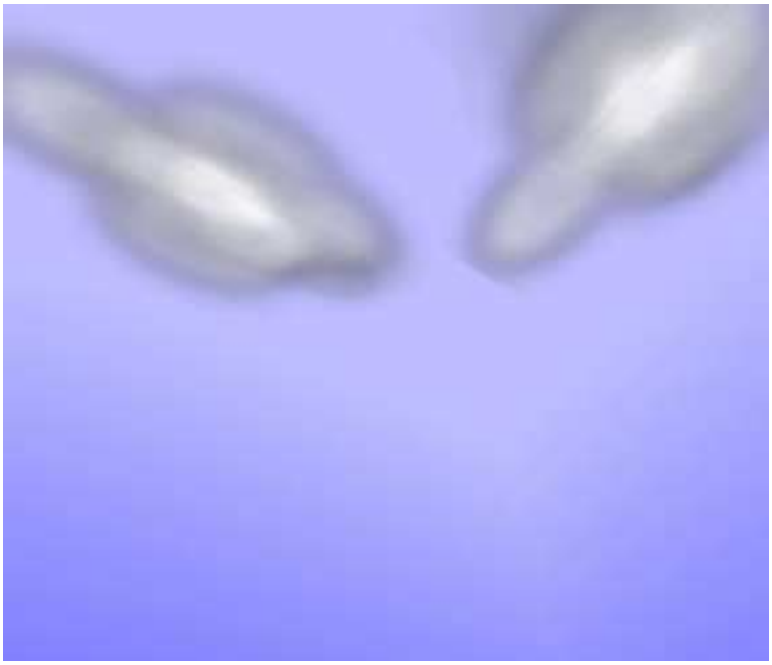
Hier werden nun einige Bilder aus der interaktiven Szenerieapplikation gezeigt, mit dem Einsatz der erzeugten Bilder als Texturen. Die ersten drei Bilder sind Gesamtansichten der Szene mit Wolkendarstellung und Schattenwurf. Dazu ist zu bemerken, daß der Schattenwurf der Wolken aufgrund der Blickrichtung des Betrachters nicht gut zu erkennen ist, da die Ausleuchtung des Terrains dazu etwas zu gering ist. Die am Boden sichtbaren Schattierungen werden durch die Struktur des Geländes verursacht.
Bei den folgenden vier Bildern handelt es sich um Detailansichten der Wolken, die in dieser Szene dargestellt werden. Die ersten drei Bilder sind dabei Darstellun-

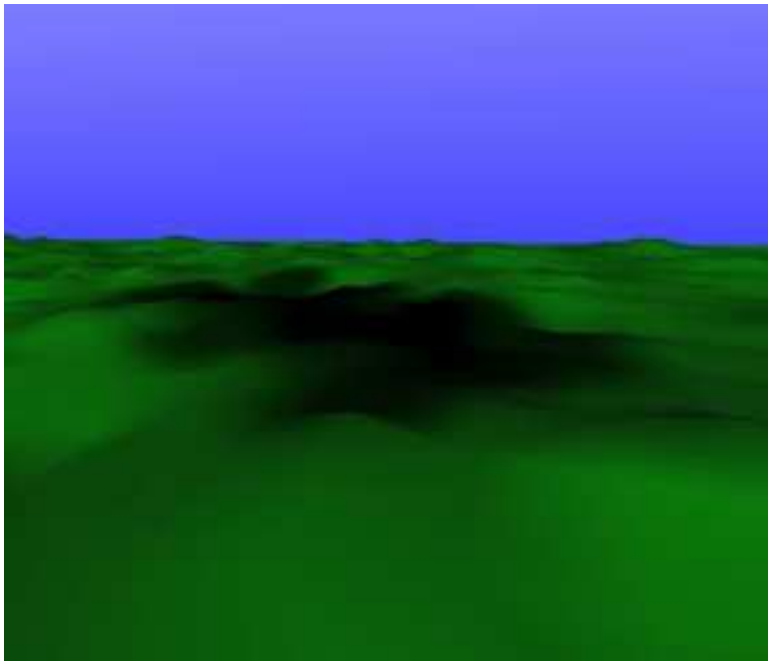
gen, die die Wolken von einem Betrachterstandpunkt unterhalb zeigen, wohingegen die vierte Darstellung die Wolke mehr von der Seite und aus größerer Distanz zeigt.
Bei diesem Bild ist weiters zu bemerken, daß hierbei die Überlagerung der drei Texturen gut sichtbar wird, da die für die Texturen verwendeten Bilder gewisse durch den Standort des Betrachters bedingte Unterschiede aufweisen. Diese Unterschiede sind auch in den drei Bildern mit der Gesamtszeneriedarstellung zu sehen.
Die letzten beiden Bilder zeigen den Schattenwurf der Wolken im Detail, wobei die Kombination der einzelnen Wolkenschatten zu einem Gesamtschatten gut zu sehen ist.

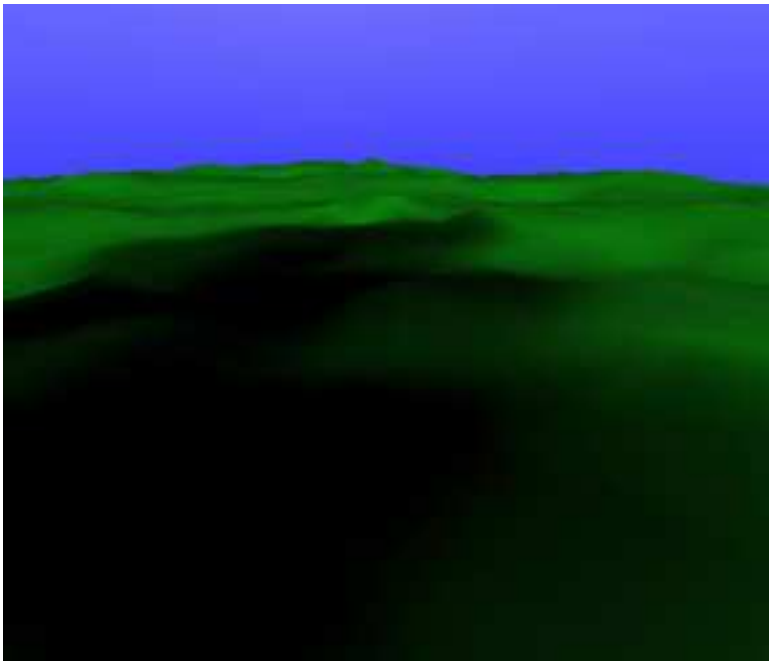












5.3 Leistungsbetrachtung

In der Folge wird ein Überblick über die Performanzdaten der implementierten Programmteile gegeben.
Die Zeitmessung wurde auf einem Rechner mit folgender Systemkonfiguration durchgeführt:

Prozessor	AMD K6-II 400 MHz
Arbeitsspeicher	128 MB
Betriebssystem	SuSE Linux 7.0
Kernelversion	2.2.16

Weiters wurden die Leistungsmessungen mit folgenden Datenmengen durchgeführt:

Voxelanzahl	100x100x100
Auflösung der Projektionsebene	256x256

5.3.1 Wolkengenerator

Laufzeit

Die Laufzeit des Wolkengenerators hängt von der Anzahl der Ellipsoide ab, die die zu erzeugende Wolke bilden. Weiters hängt die Berechnungszeit davon ab, ob eine Geräuschfunktion eingesetzt wird.

	Laufzeit/Ellipsoid	Laufzeit/Ellipsoid
ohne Geräuschfunktion	18 Sekunden	$\approx$ 0,3 Minuten
mit Geräuschfunktion	156 Sekunden	$\approx$ 2,6 Minuten

Speicherbedarf

Der Speicherbedarf des Wolkengenerators ist nicht hoch, da nur die Dichtefunktion zwischenzuspeichern ist.

	Datentyp	Größe	Speicherbedarf
Dichtefunktion	float	100x100x100	3,8 MB

5.3.2 Wolkenrenderer

Laufzeit

Die Laufzeit des Wolkenrenderer-Algorithmus teilt sich auf folgende Teile auf:

1. Berechnung des Referenzmusters für die Lichtstreuung

2. Berechnung der Restlichtintensität nach der Absorption
3. Berechnung der gestreuten Intensitäten
4. Bildberechnung

Dabei ergaben sich folgende Werte für die Laufzeiten der Berechnungen:

Referenzmuster	32 Sekunden	$\approx$ 0,53 Minuten
Absorption	44 Sekunden	$\approx$ 0,73 Minuten
gestreute Intensitäten	328 Sekunden	$\approx$ 5,47 Minuten
Bildberechnung	334 Sekunden	$\approx$ 5,57 Minuten
Gesamt	738 Sekunden	$\approx$ 12,30 Minuten

Speicherbedarf

Der Speicherbedarf des Wolkenrenderers wird vor allem durch den Platz den die Speicherung von Zwischenergebnissen während des Rendering-Algorithmus benötigt bestimmt. Dies umfaßt das Voxelvolumen, das mehrfach benötigt wird, und die Projektionsfläche.

	Datentyp	Größe	Speicherbedarf
Referenzmuster	float	11x11x11	5,2 KB
Dichtefunktion	unsigned long	100x100x100	3,8 MB
nicht absorbierte Intensitäten	unsigned long	100x100x100	3,8 MB
gestreute Intensitäten	float	100x100x100	3,8 MB
Bildinformation	float	256x256	256 KB
Gesamt			11,7 MB

Generell kann man zur Beurteilung der Leistung des Programmsystems sagen, daß sich die Berechnungen auf einem heutigen Standard-PC ohne Performanzprobleme durchführen lassen. Das war auch eines der Ziele, das mit der Implementierung verfolgt wurde.

Kapitel 6

Bewertung

Das Ziel dieser Arbeit war es, bereits existierende physikbasierende Verfahren der statischen Wolkenbilderzeugung für eine interaktive dreidimensionale Szenerie einzusetzen. Die wesentliche Neuerung hierbei stellt die Verbindung von bereits bekannten Verfahren dar, deren kombinierter Einsatz die Echtzeitdarstellung von Wolkenformationen möglich macht:

- physikbasierende Bilderzeugungsalgorithmen
- Texturen

6.1 Vergleich mit anderen Ansätzen

Wie schon weiter oben erläutert wurden schon viele Möglichkeiten publiziert eine realistische Wolkendarstellung für statische Bilder zu erreichen [NiDN96, Gard85, NiDo99, Eber97, EMPP98]. In der Folge soll nun der implementierte Ansatz mit den bereits vorliegenden Methoden verglichen werden.
Dazu muß einleitend gesagt werden, daß die erzeugte Bildqualität im wesentlichen von zwei Faktoren abhängig ist:

- Qualität der Modellierung
- Qualität der Bilderzeugung

Die folgenden Ausführungen zur Bilderzeugung schließen den Modellierungsschritt nicht mit ein, da kein eigener Programmteil zur Wolkenmodellierung entwickelt wurde und es dem Benutzer selbst überlassen bleibt, die Struktur und Form der Wolken mit Hilfe der Wolkenbeschreibungsdatei festzulegen.

6.1.1 Bilderzeugung

Im Vergleich zu den in [Gard85] vorgestellten Verfahren läßt sich sagen, daß die Qualität der erzeugten Bilder vergleichbar ist. Beide Algorithmen produzieren überzeugende Bilder, wenn auch mit einem unterschiedlichen Ansatz. Außerdem ist zu sagen, das sich die in [Gard85] vorgestellten Verfahren zur Darstellung von Cirrus- bzw. Stratus-Wolkenformationen besser eignen, da das hier vorgestellte Verfahren in der Wolkenmodellierung auf die Verwendung von Ellipsoiden beschränkt ist, die sich für die Darstellung von Cumulus-Wolken besonders gut eignen.

Der Vergleich mit dem Verfahren von [NiDN96] ist besonders interessant, da dieses Verfahren mit kleinen Änderungen nachimplementiert wurde. Hierbei läßt sich sagen, das obwohl die Berücksichtigung des Himmelslichtes nicht implementiert wurde und bei der Streuungsberechnung einige Vereinfachungen vorgenommen wurden, die Bilder eine ähnliche Qualität erreichen.

6.1.2 Szeneriedarstellung

Die Beurteilung der Szeneriedarstellung durch Vergleich mit anderen Ansätzen gestaltet sich schwierig, da meines Wissens kein anderer Ansatz für die interaktive Echtzeitdarstellung von Wolken existiert.
Das in [DKYO00] vorgestellte Verfahren zielt auf die Erzeugung einer Animation ab, die die zeitliche Entwicklung einer Wolkenformation darstellt, wobei allerdings der Standort des Betrachters unverändert bleibt.
Eine interaktive Echtzeitdarstellung wäre beispielsweise auch mit den in [Gard85] beziehungsweise [EMPP98] beschriebenen Verfahren denkbar, da sie sich auf eine ontogenetische Darstellung der Wolkenformationen beschränken und daher die Darstellung mit einem sehr geringen Rechenaufwand möglich wird. Eine Möglichkeit hierzu wäre beispielsweise die Darstellung einer Stratus-Wolkenschicht durch eine prozedurale Textur.

6.2 Verbleibende Aufgaben

Es bestehen zahlreiche Möglichkeiten, die vorliegende Implementierung zu verbessern:

- Verbesserung der Modellierungsmöglichkeiten
- Berücksichtigung zusätzlicher Wolkentypen
- Verbesserung der Darstellungsqualität in der interaktiven Szenerie

- Erhöhung der Betrachtermobilität
- Dynamisch veränderliche Lichtverhältnisse

In der vorliegenden Implementierung ist die Modellierung der Wolken auf die Erstellung der Wolkenbeschreibungsdatei beschränkt. Eine sinnvolle Erweiterung hierzu wäre die Implementierung eines Wolkenmodelleditors, wie in [Taxe99] beschrieben. Bei der Verwendung eines solchen Editors würde sich der Aufwand für den Benutzer auf die Angabe einiger Wolkenparameter beschränken.

Zur Berücksichtigung zusätzlicher Wolkentypen ist zu sagen, daß die Erweiterung der Implementierung zur Darstellung von beispielsweise Stratuswolken mit sehr geringem Aufwand möglich wäre, da dazu nur die Verwendung einer oder mehrerer zum Boden paralleler Ebenen notwendig ist, auf die eine geeignete prozedurale Textur [EMPP98] eventuell mit Transparenz aufzubringen ist [Gard85]. Bei Verwendung einer solchen variablen Transparenz ist auch die Darstellung von Cirrus-Wolkenformationen problemlos möglich.
Diese Ansätze würden dann allerdings nicht mehr auf einer physikalischen Simulation beruhen. Diese ist ist für im Vergleich zu Cumuluswolken dünne Wolkenformationen schwieriger, da sich hier das Ray-Casting Verfahren nicht mehr so gut einsetzen läßt.

Die Verbesserung der Darstellungsqualität der interaktiven Szenerie ist auch ein Aspekt, der mit relativ wenig Programmieraufwand verbunden ist. Die Darstellungsqualität der Wolken ist in erster Linie von der Anzahl der Texturen abhängig. Die Zahl der Texturen, bei deren Bildern sich der Betrachter weit seitlich der Wolke befindet, ist im Programmcode als Konstante definiert und läßt sich so leicht verändern. Außerdem ist es noch notwendig, die entsprechenden Texturen zu erzeugen. Wieviele Texturen mit seitlichen Ansichten benötigt werden, hängt auch von der groben Form der Wolke ab: Falls die Länge und die Breite der Wolke stark differieren, ist es notwendig, mehr Texturen einzusetzen.
Wenn sich der Betrachter in der Szenerie direkt unter der Wolke befindet, wird eine Textur mit der Ansicht der Wolke von unten angezeigt. Hierbei wäre der Übergang von den seitlichen Texturen zu der Textur unterhalb der Wolke noch verbesserungswürdig, da hier durch die Überlagerung der Texturen visuelle Artefakte entstehen können. Zu diesem Zweck könnte man Bilder einsetzen, bei denen der Betrachter schräg unterhalb der Wolke steht.

Die Erweiterung der Betrachtermobilität dahingehend, das der Betrachter in seiner Navigation nicht mehr auf den Boden beschränkt ist, wäre mit einer weiteren Erhöhung der Texturanzahl zu erreichen. Die Vorgangsweise wäre symmetrisch

zum Fall der Wolkenbetrachtung von unten. Für diese Erweiterung wäre es notwendig, eine Darstellung der Wolke von oben zu erzeugen, um die Position der fiktiven Wolke im Raum vollständig mit Texturen zu umgeben, was zu einer kugelförmigen Struktur führen würde.
Diese Erweiterung löst aber noch nicht das Problem, daß ein direktes Hineinfliegen in die Wolke nicht möglich ist, da die Darstellung nur mit Hilfe texturierter Polygone erfolgt.

Die dynamische Veränderung der Lichtverhältnisse wäre eine etwas komplexere Erweiterung des Systems. Es gäbe hierzu folgende Möglichkeiten:

1. Dynamisches Austauschen vorabberechneter Texturen

2. Veränderung der Beleuchtung der texturierten Polygone

Zu Punkt eins ist zu sagen, daß dies nur eine Erweiterung des bisher verwendeten Ansatzes darstellt. Dabei ist allerdings zu bedenken, daß dies den Bedarf an Bildern und Texturen vervielfachen würde, was diesen Ansatz im Hinblick auf die Performanz nicht günstig erscheinen läßt.
Punkt zwei stellt eine interessante Möglichkeit dar, die in [WHON97] beziehungsweise [Wong98] beschrieben wird.

Kapitel 7

Zusammenfassung

Das Problem der realistischen Wolkendarstellung ist trotz zahlreicher Publikationen auf diesem Gebiet in den letztem Jahren aktuell geblieben, unter anderem deshalb, weil die Berechnungszeiten für eine Darstellung unter Berücksichtigung der physikalischen Vorgänge nach wie vor sehr hoch sind. Daraus ergibt sich auch, daß die bisher publizierten Verfahren sich auf die Erzeugung realistischer statischer Bilder oder Animationen beschränkten.
In dieser Arbeit wurde ein Verfahren vorgestellt, realistische Wolkendarstellungen in einer Echtzeitszenerie einzusetzen. Dabei wurde auf ein bereits publiziertes physikbasierendes Bilderzeugungsverfahren zurückgegriffen, und die erzeugten Darstellungen als Texturen eingesetzt, um die Darstellung in einer interaktiven dreidimensionalen Szenerie zu ermöglichen.

Anhang A

Verwendete Konstanten und Größen

A.1 Konstanten

Gravitationsbeschleunigung	$g = 9,81\, ms^{-2}$
Spezielle Gaskonstante für trockene Luft	$R_L = 287\, Jkg^{-1}K^{-1}$
Spezielle Gaskonstante für Wasserdampf	$R_W = 461,5\, Jkg^{-1}K^{-1}$
Universelle Gaskonstante	$R^* = 8,314\, Jmol^{-1}K^{-1}$

A.2 Größen

absolute Feuchtigkeit	ρ_w	[kg m^{-3}]
Dampfdruck	e	[mb]
(Luft)Dichte	$\rho_{(L)}$	[kg m^{-3}]
(Luft)Druck	$p_{(L)}$	[b]
Lichtintensität	I	[W]
Lichtwellenlänge	λ	[m]
Masse	m	[kg]
Massenmischungsverhältnis	μ_W	[Prozent]
relative Feuchtigkeit	f	[Prozent]
spezifische Feuchtigkeit	s	[kg m^{-3}]
Streuungskoeffizient	a	[Prozent]
Temperatur	T	[°K]
virtuelle Temperatur	T_v	[°K]
Volumen	V	[m^3]
Wasserdampfdruck	e	[b]
Zeit	T	[s]

Anhang B

Grammatiken

B.1 Wolkenbeschreibungsdatei

```
Cloud = "cloud" Minimal Maximal [Noise] {Ellipsoid}.
Minimal = "minimal coordinates" Tripple.
Maximal = "maximal coordinates" Tripple.
Ellipsoid = "ellipsoid" Center Semiaxis [CenterDensity] [BorderDensity]
Noise = "noise factor" integer.
Center = "center" Tripple.
Semiaxis = "semiaxis" Tripple.
CenterDensity = "center density" float.
BorderDensity = "border density" float.
Tripple = "<" integer "," integer "," integer ">".
```

B.2 Renderinginformationsdatei

```
RenderParams = "render parameters" Cloud ObserverPos ViewDir LightVecto
               PicPlaneDist PixelDist Resolution.
Cloud = "cloud" DataFile Position.
DataFile = name.
Position = Tripple.
ObserverPos = "observer" Tripple.
ViewDir = "view direction" Tripple.
LightVector = "light vector" Tripple.
PicPlaneDist = "picture plane distance" float.
PixelDist = "pixel distance" float.
Resolution = "resolution" integer "x" integer.
Tripple = "<" integer "," integer "," integer ">".
```

Literaturverzeichnis

[AfEM96] O. Aftreth, G. Emery, W. Morgan: *The Online POV-Ray Tutorial*
1996, http://library.thinkquest.org/3285/

[ArRP99] D. Aronov, M. Radomyselskiy, M. J. Po: *Fractals Unleashed*
1999, http://library.thinkquest.org/26242/full/index.html

[Bart97] H.-J. Bartsch: *Taschenbuch mathematischer Formeln*
Fachbuchverlag Leipzig 1997, ISBN 3-446-18717-0

[Bohr95] C. F. Bohren: *Atmospheric Optics*
Encyclopedia of Applied Physics, VCH Publishers 1995, Volume 12, ISBN 1-56081-071-8, pp. 405-434

[BuGr95] M. Buck, K. Grebner: *Werkzeuge zur Interaktion in virtuellen Welten*
MVD '95, infix 1995, ISBN 3-929037-98-X, pp. 57-68

[Caha94] R. F. Cahalan: *Bounded Cascade Clouds: Albedo and Effective Thickness*
Nonlinear Proc. Geophys. 1, 1994, pp. 156-167

[Clau97] U. Claussen: *Programmieren mit OpenGL*
Springer Verlag 1997, ISBN 3-540-57977-X

[CoWa93] M. F. Cohen, J. R. Wallace: *Radiosity and Realistic Image Synthesis*
AP Professional 1993, ISBN 0-12-178270-0

[Dalh98] M. K. Dalheimer: *GNU-Tools zur Programmierung*
O'Reilly 1998, ISBN 3-89721-209-9

[DiKL97] P. R. Dietmüller, U. Kreuzeder, B. Leisch: *Skriptum Programmierpraktikum 2 / C*
Forschungsinstitut für Mikroprozessortechnik, Universität Linz 1997

[DKYO00] Y. Dobashi, K. Kaneda, H. Yamashita, T. Okita, T. Nishita: *A Simple, Efficient Method for Realistic Animation of Clouds*
ACM SIGGRAPH 2000, pp. 18-28

[Eber94] D. S. Ebert: *Procedural Modeling and Animation of Gases and Fluids*
ACM SIGGRAPH 1994 Course Notes #8: Procedural Modeling, Texturing and Rendering Techniques

[Eber97] D. S. Ebert: *Volumetric Procedural Implicit Functions: A Cloud is Born*
ACM SIGGRAPH 1997 Technical Sketches Program

[EMPP98] D. S. Ebert, F. K. Musgrave, D. Peachey, K. Perlin, S. Worley: *Texturing & Modeling. A Procedural Approach*
Second Edition, AP Professional 1998, ISBN 0-12-228730-4

[Elia00] H. Elias: *Perlin Noise*
2000, http://freespace.virgin.net/hugo.elias/models/m_perlin.htm

[Elvi92] T. T. Elvins: *A Survey of Algorithms for Volume Visualization*
Computer Graphics 1992, Volume 26, Number 3, pp. 194-201

[Fell92] W. D. Fellner: *Computergrafik*
BI Wissenschaftsverlag 1992, ISBN 3-411-15122-6

[Gard85] G. Y. Gardner: *Visual Simulation of Clouds*
Computer Graphics 1985, Volume 19, Number 3, pp. 297-303

[Glas89] A. S. Glassner (editor): *An Introduction to Raytracing*
Academic Press 1989, ISBN 0-12-286160-4

[Hage84] P. J. W. ten Hagen: *Interactive Techniques*
Eurographics Tutorials '83, Springer Verlag 1984, ISBN 3-540-13644-4, pp. 73-95

[Hess61] W. Hesse: *Handbuch der Aerologie*
Leipzig 1961, Akademische Verlagsgesellschaft Geest & Portig K.-G.

[Hu96] Y. Hu: *The multiple scattering correction of the reflected radiance field for the perturbed scattering phase function*
1996, http://greenfield.fortunecity.com/healing/195/paper/iso/iso.html

[Irwi96] J. Irwin: *Full-Spectral Rendering of the Earth's Atmosphere Using a Physical Model of Rayleigh Scattering*
Proceedings of the 1996 Eurographics UK Conference, pp. 103-115 (Vol 1)

[Kauf91] A. Kaufman: *3D Volume Visualization*
Advances in Computer Graphics VI, Eurographic Seminars, Springer Verlag 1991, ISBN 3-540-53455-5, pp. 175-203

[Klass87] R. V. Klassen: *Modeling the Effect of Atmosphere on Light*
ACM Transactions on Graphics, Vol 6, No. 3, July 1987, pp. 215-237

[LeMS91] W. Leister, H. Müller, A. Stößer: *Fotorealistische Computeranimation*
Springer Verlag 1991, ISBN 3-540-53234-X

[Max94] N. Max: *Efficient Light Propagation for Multiple Anisotropic Volume Scattering*
Proceedings of the Fifth Eurographics Workshop on Rendering 1994 pp. 87-104

[Max95] N. Max: *Optical Models for Direct Volume Rendering*
IEEE Transactions on Visualization and Computer Graphics, Vol. 1, No. 2, June 1995

[Mayi94] F. Mayinger (editor): *Optical Measurements. Techniques and Applications*
Springer Verlag 1994, ISBN 3-540-56765-8

[Möll73a] F. Möller: *Einführung in die Meteorologie*
Band 1: Physik der Atmosphäre, BI Hochschultaschenbücher 1973, Band 276, ISBN 3-411-00276-X

[Möll73b] F. Möller: *Einführung in die Meteorologie*
Band 2: Physik der Atmosphäre, BI Hochschultaschenbücher 1973, Band 288, ISBN 3-411-00288-3

[Möss99] H. Mössenböck: *Übersetzer*
Informatik Handbuch, 2. Auflage, Hanser Verlag 1999, ISBN 3-446-19601-3, pp. 737-762

[NiDN96] T. Nishita, Y. Dobashi, E. Nakamae: *Display of Clouds Taking into Account Multiple Anisotropic Scattering and Sky Light.*
ACM SIGGRAPH 1996, pp. 379-386

[NiDo99] T. Nishita, Y. Dobashi: *Modeling and Rendering Methods of Clouds*
Proceedings of the 7th Pacific Conference, 1999-10, pp. 218-219

[NIDN97] T. Nishita, H. Iwasaki, Y. Dobashi, E. Nakamae: *A Modeling and Rendering Method for Snow by Using Metaballs*
Computer Graphics Forum, Vol. 16, No. 3, 1997

[OvWa82] R. D. Overheim, D. L. Wagner: *Light and Color*
John Wiley & Sons Inc. 1982, ISBN 0-471-08248-8

[Pov99] The POV-Team: *Persistence of Vision Ray-Tracer Version 3.1g. User's documentation*
1999, http://www.povray.org

[PrSS99] A. J. Preetham, P. Shirley, B. Smits: *A Practical Analytical Model of Daylight*
ACM SIGGRAPH 1999, pp. 91-100

[Schr94] M. Schroeder: *Fraktale, Chaos und Selbstähnlichkeit*
Spektrum akademischer Verlag 1994, ISBN 3-86025-092-2

[Shir94] P. Shirley: *Hybrid Radiosity / Monte Carlo Methods*
ACM SIGGRAPH 1994, Advanced Topics in Radiosity Course Notes, Chapter 11, pp. 1-24

[SpPr99] A. Spalt, M. Priglinger: *Vorlesungsunterlage Angewandte Computergrafik 1 (Visualisierung)*
Abteilung für Graphische und Parallele Datenverarbeitung, Institut für Technische Informatik und Telematik, Universität Linz 1999

[Taxe99] G. Taxen: *Cloud Modeling for Computer Graphics*
Master's thesis, Royal Institute of Technology, Stockholm, Sweden, 1999

[ToOb97] D. H. Tofsted, S. G. O'Brien: *Characterizing the effects of natural clouds on scene simulations*
SPIE Proceedings Vol. 3062, 1997, ISBN 0-8194-2477-3

[Volk98] J. Volkert: *Vorlesungsunterlage Computergrafik*
Abteilung für Graphische und Parallele Datenverarbeitung, Institut für Technische Informatik und Telematik, Universität Linz 1998

[Volk99] J. Volkert: *Vorlesungsunterlage Ausgewählte Kapitel aus Computergrafik: Virtual Reality*
Abteilung für Graphische und Parallele Datenverarbeitung, Institut für Technische Informatik und Telematik, Universität Linz 1999

[Watt89] A. Watt: *Fundamentals of Three-Dimensional Computer Grapics*
Addison Wesley 1989, ISBN 0-201-15442-0

[WHON97] T.-T. Wong, P.-A. Heng, S.-H. Or, W.-Y. Ng: *Image-based Rendering with Controllable Illumination*

Rendering Techniques 1997, Springer Verlag 1997, ISBN 3-211-83001-4, pp. 13-22

[Wong98] T.-T. Wong: *Time-Critical Modeling and Rendering: Geometry-based and Image-based Approaches*
PhD thesis, The Chinese University of Hong Kong, Hong Kong, China, 1998

[WNDS99] M. Woo, J. Neider, T. Davis, D. Shreiner: *OpenGL Programming Guide*
Third Edition, Addison Wesley 1999, ISBN 0-201-60458-2

Index

Eidesstattliche Erklärung

Ich erkläre an Eides statt, daß ich die vorliegende Diplomarbeit selbständig und ohne fremde Hilfe verfaßt habe. Ich habe dazu keine weiteren als die angeführten Hilfsmittel benutzt und die aus anderen Quellen entnommenen Stellen als solche gekennzeichnet.

Linz, April 2001 Paul Heinzlreiter