

Modellierung von Gasen/ Volumetrische Wolken

VERFASST FÜR DAS
DIPLOMANTENSEMINAR „Computergrafik“ IM SS 07
VON SANDRA HOFFMANN

Übersicht

- Geschichte
 - Modellierung und Plazierung
- Wolkemodellierung
 - Bilder und Videos
- Allgemeines – Rauschen
- Rascharten
- Turbulenz
- Ansätze zur Modellierung von Gasen/Wolken
- Modellierung von Gas
 - Verwendung von Grafik-Hardware
 - Rendering System
 - Volumenrendering Algorithmus
 - Beleuchtung und Schatten
- Volumetrische Wolken
 - Wolkenfamilie
 - Wolkengattungen
 - Ansatz
 - Volumenvisualisierung
 - Rendering
 - Wolken-Renderer mit OpenGL
 - Bilder / Videos
 - Quellen

Modellierung von Gasen – Geschichte

- 1970 Modellierung von Gas in Computergrafik
- 1983 Voss / 1990 Musgrave
 - Fraktale für realistische Wolken- und Raucheffekte
- 1984 Kajiya und Von Herzen
 - Physikalsich basiertes Model für Wolkenmodellierung
- 1986 Max
 - Höhenfelder für die Simulation von Effekte mit diffusem Licht
- 1987 Nishita, Miyawaki und Nakamae
 - Konstantes Dichte Medium mit verschiedenen Ebenen von Dichte
 - Nur sehr wenige Geometrien für Gas konnten gemodelt werden
- 1985/90 Gardner
 - Realisitische Wolkenbilder mit Fourier Synthese
 - Kein realistisches 3D Model für Wolken

Modellierung von Gasen – Geschichte

- 1990 Ebert, Parent, Boyer und Roble
 - Volumen Dichte Funktion
 - Echtes 3D Model für die Geometrie von Gasen
- 1991/93/95 Stam und Fiume
 - 3D Geometriemodell für Gas mit „fuzzy blobbies“
 - „fuzzy blobbies“ entspricht volumetrischen Metaballs und Partikel Systemen
- 1999/2001 Stam, Fedkiw und Jensen
 - Erweiterung des Ansatzes von Stam und Fiume zu einem physikbasierten Navier-Stokes Lösung
 - Sehr realistische Effekte

Geschichte - Wolkenmodellierung

- in letzten 20 Jahren viel Forschung im Bereich Wolkensimulation und deren Darstellung
- 1985 Ken Perlin Erzeugung prozeduraler Texturen mittels Rausch-Funktion
 - viele weitere Verfahren aufbauend auf dieser Methode
- neue Arbeit von Y. Dobashi versuchte die prozedurale Eigenschaften von Wolken durch zellulare Automaten zu approximieren
- alle bisherigen Verfahren sind nicht echtzeitfähig
- Echtzeit Verfahren durch M. Harris mittels eines Billboard-Ansatzes interaktive Frameraten zu erreichen

Allgemeines – Rauschen

- Rauschen hat in der Natur einen zufälligen Charakter
- besonders an diesem Zufallsgenerator ist der funktionale Charakter
- bei Eingabe der gleiche (x,y,z) Werte wird immer der selbe Zufallswert erzeugt
- Wichtiges Kriterium zur Visualisierung von Gasen und Wolken
- Form und Oberflächenbeschaffenheit von Gasen und Wolken bestimmt
- Einfachste Rauschfunktion
 - Zufallsfunktion, die zu jedem Zeitpunkt einen anderen Wert ausgibt
- Ideale Noise-Funktion
 - Wiederholend und pseudo-zufällig,
 - Periode kann sehr lang gezählt werden, so dass sie nicht sofort ersichtlich ist
 - Wertebereich $[-1,1]$ oder $[0,1]$
 - Band-begrenzt, maximale Frequenz ist etwa 1
 - Behält auch nach einer Translation oder Rotation ihr typisches Aussehen

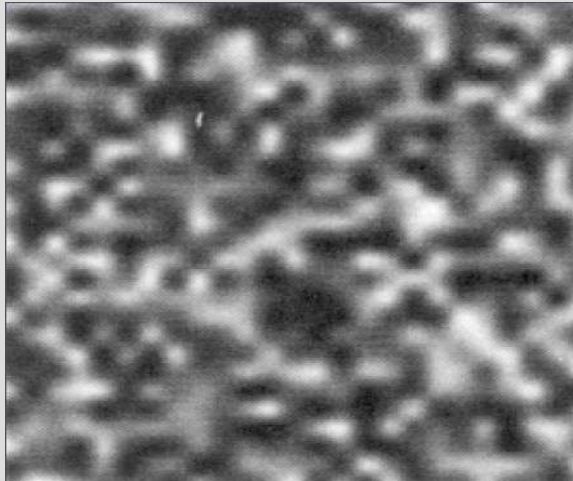
Rauscharten

- Lattice Noise
 - Rauschfunktionen aus Gruppe der Lattice Noise werden häufig verwendet
 - Rauschfunktion besteht aus 3 Teilen
 - Eindimensionales Array
 - Permutationstabelle
 - Funktion, mit der man an einer bestimmten Stelle in Array einen Wert ermitteln kann
 - Array wird zufällig gefüllt $[-1,1]$
 - Zwischen Punkten wird interpoliert
 - Qualität der Noise hängt von der Interpolationsmethode ab

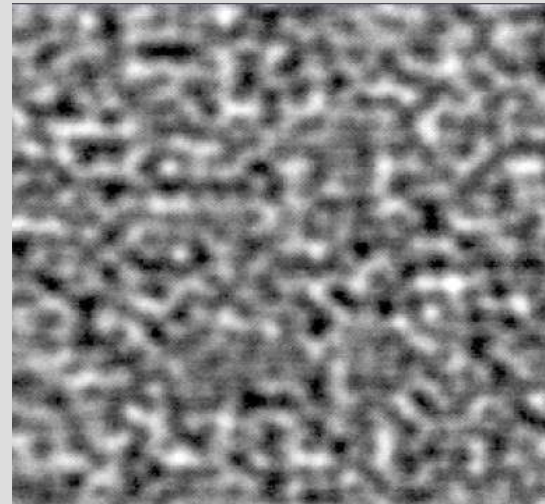
Rauscharten

- Value Noise
 - Erstellung einer zusätzlichen Tabelle mit zufälligen Werten zwischen $[-1,1]$
 - Lineare Interpolation -> nicht wirklich fein
 - Quadratische oder Kubische Spline Interpolation -> hat noch einige Artefakte vom Gitteraufbau
 - Kästchenmuster zu erkennen
- Gradient Noise
 - Kästchenmuster können durch Gradient Noise durchbrochen werden
 - Pro Gitterpunkt wird nicht nur eine Zahl sondern auch ein Vektor der Länge 1 gespeichert
 - Zur Ermittlung des Noise-Wertes wird das Skalarprodukt aus diesem Vektor und dem Nachkomma-Anteil des Ortsvektors des Punktes gebildet

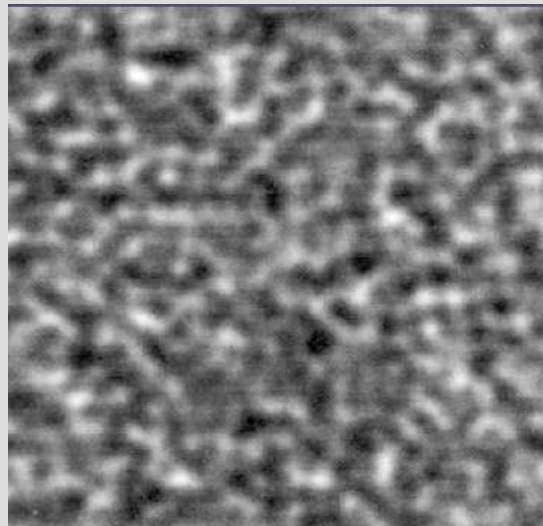
Rauscharten



Value Noise



Gradient Noise



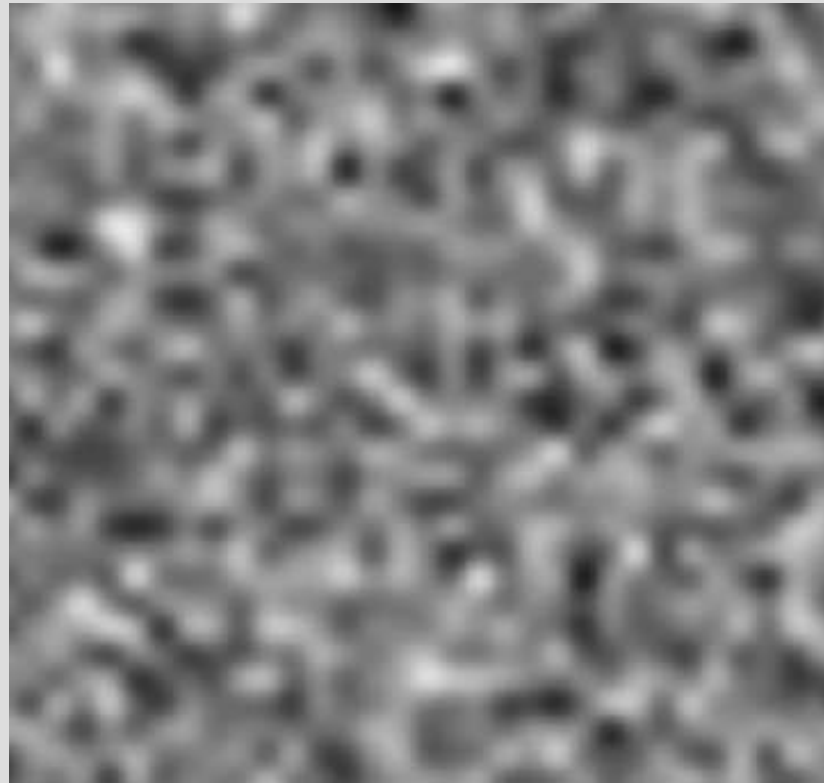
Value und Gradient
Noise

Rauscharten

- Wavelet Noise
 - „kleine Welle“
 - bezeichnet räumlich und zeitlich lokale Wellenfunktion
 - zur Berechnung der Noise-Funktion
 - verschiedene Wavelets an entsprechender Stelle ausgewertet
 - mit verschiedenen Wichtungen addiert
 - man kann unendlich weit hinein- und hinauszoomen
 - Heranzoomen
 - stärkere Ausblendung höhere Frequenzen und geringere Frequenzen werden sichtbar
 - Heranzoomen
 - Kleine Frequenzen werden schwächer und größere Frequenzen bekommen größeren Einfluss

Rauscharten

- Wavelet Noise



Turbulenz

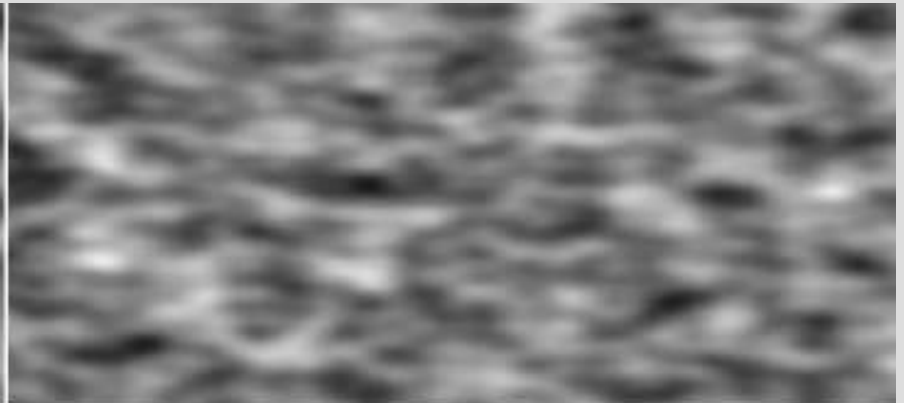
- Die einfache Noise-Funktion hat immer noch erkennbare Strukturen, da die Zufallswerte auf ganzzahligen Gitterpunkten gespeichert sind
- Um Unregelmäßigkeiten zu erzeugen, muss die Noise-Funktion erweitert werden
- Einfachste Möglichkeit
 - Mehrere Noises in verschiedenen Stärken übereinander legen
 - Passiert in der Turbulenz-Funktion
 - Noise werden gestapelt
 - jede Ebene ist nur halb so stark wie die vorhergehende, dafür aber doppelt so gross (skaliert)
 - Anzahl der Ebene = Oktaven

Turbulenz

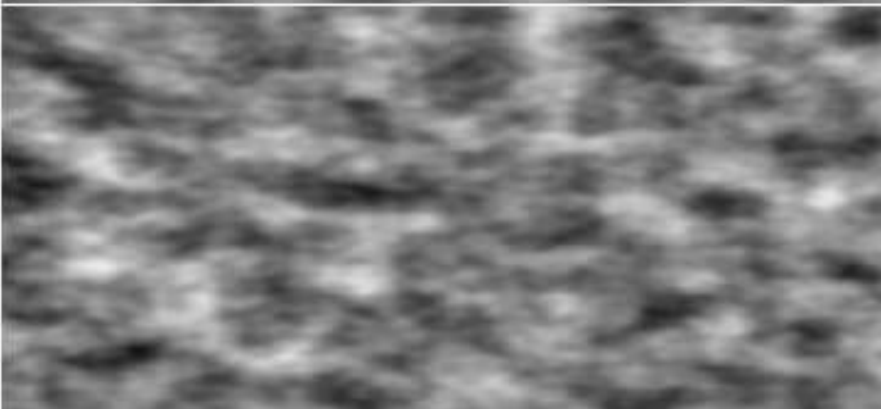
ein Oktave



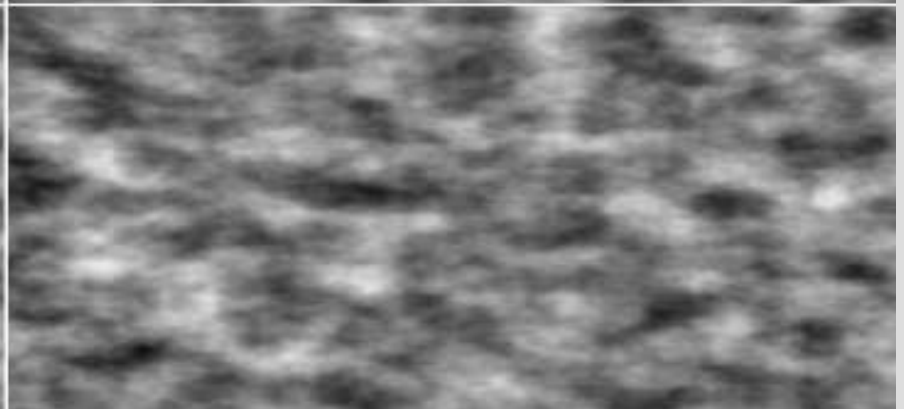
zwei Oktaven



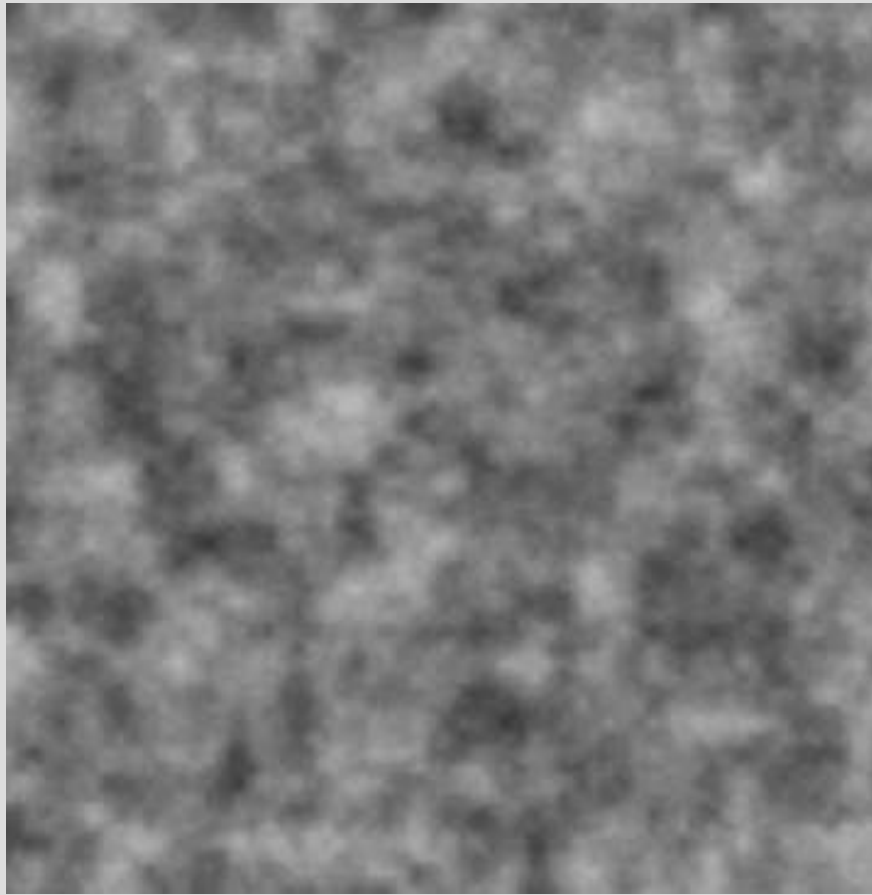
drei Oktave



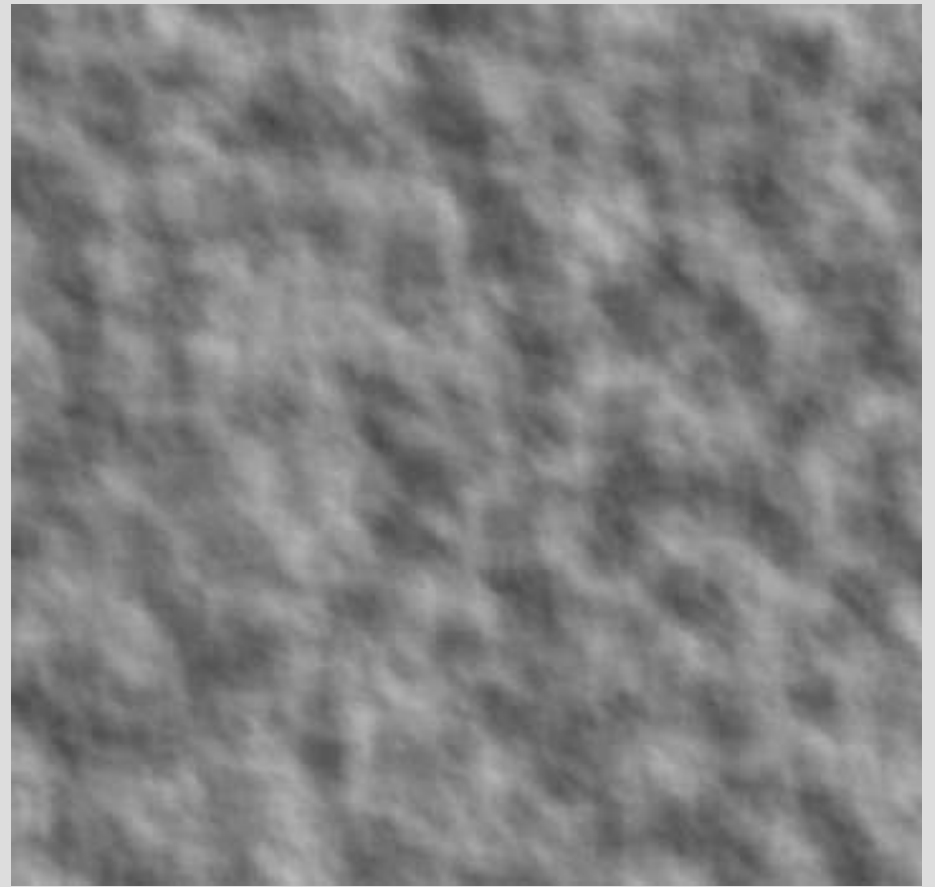
vier Oktaven



Turbulenz



Turbulenz-Rauschen mit Value Noise



Turbulenz-Rauschen mit Gradient Noise

Ansätze zur Modellierung von Gasen/Wolken

- Oberflächenbasierte Modelle von Objekten
- 3D Volumen Dichte Funktionen
- Partikelsystem
- Billboards
- 3D Hardware Texture Mapping
- Hardware beschleunigte Deckkraft/Schatten Mapping Technik
- Physikalisch basiert
- Smoothed Particle Hydrodynamics (SPH)
- Zellulare Automaten
 - Couple Map Lattice (CML)
 - Boltzmann Gitter Modell (BGM)

Ansätze zur Modellierung von Gasen/Wolken

- Claude Navier, George Stokes
- Strömungsverhalten von Fluiden
- Nichtlineare Partiellen Differential Gleichungen
 - Numerisches Lösungsverfahren
 - Kompromiss: Genauigkeit <-> Geschwindigkeit

$$\frac{\delta u}{\delta t} = \overbrace{-u \cdot \nabla u}^{\text{Advektionsterm}} - \overbrace{\frac{1}{\rho} \nabla p}^{\text{Druckterm}} + \overbrace{\mu \nabla^2 u}^{\text{Diffusionsterm}} + \overbrace{f}^{\text{Äußere Kräfte}}$$

$$\nabla \cdot u = 0$$

Modellierung von Gasen - Verwendung von Grafik-Hardware

- Verarbeitung von Bildern/Texturen
- Textur = Array von Werten
 - Repräsentation von Gittern
 - 3D Gitter als Reihe von 2D Texturen
- Nutzt schnelle Weiterentwicklung der Halbleitertechnologie
- Kompatible
- Problem: Genauigkeit

Modellierung von Gasen – Rendering System (Ebert/Parent)

- Echte 3D Bilder und Animationen von Gasen
- Gemischtes Render System
 - Scanline a-buffer Rendering Algorithmus für oberflächenbegrenzte Objekte
 - Per-pixel Volumen Raytracing Technik für Volumen modellierte Objekte
- Algorithmus
 - Ersellt a-Buffer (Liste) für Scanline, dieser beinhaltet alle Pixel die das aktuelle Pixel verdecken
 - Ausmaß des Volumenrenderings wird bestimmt
 - Volumenrendering wird ausgeführt -> Erstellung a-Buffer Fragment für den einzelnen Abschnitt des Volumens
 - Verfolgung stoppt wenn Objekt getroffen wird
 - Verfolgung läuft durch Objekt/Volumen und sammelt die Deckkraft und Farbe
 - Volumen a-Buffer Fragmente werden sortiert in die a-Buffer Fragment Liste basierend auf ihrer durchschnittlichen Z-Tiefe
 - a-Buffer Fragment Liste wird gerendert -> finale Farbe des Pixels

Modellierung von Gasen – Volumen Rendering Algorithmus

- Strahl vom Augpunkt durch das Pixel wird entlang der Geometrie verfolgt
- Aufruf der Funktion bei jedem Schritt durch das Volumen
- Farbe, Dichte, Deckkraft, Schattierung und Ausleuchtung für jede Auswahl bewertet
- Deckkraft = Dichte
 - Dichte -> Volumen Dichte Funktion ausgewertet wird multipliziert mit der Schrittgröße
 - Im Gasförmiges Model haben wir ein Integral angenähert um die Deckkraft entlang des Strahls zu berechnen
- Die Annäherung verwendet

$$Deckkraft = 1 - e^{-\tau \int_{t_{near}}^{t_{far}} p(x(t), y(t), z(t)) x \Delta t}$$

τ - optische Tiefe des Materials

$p()$ - Dichte des Materials

t_{near} - Startpunkt der Volumenverfolgung

t_{far} - Endpunkt

Modellierung von Gasen –

Beleuchtung

- Beleuchtungsmodell:

$$B = \sum_{t_{near}}^{t_{far}} e^{-tx} \sum_{t_{near}}^t p(x(u), y(u), z(u)) \times \Delta u \times I \times p(x(t), y(t), z(t)) \times \Delta t$$

- mit I $\sum_i I_i(x(t), y(t), z(t)) \times phase(\Theta)$

- die phase() Funktion, charakterisiert die gesamte Beleuchtung des Partikels
- $I_i(x(t), y(t), z(t))$ Summe des Lichtes vom Ausgangspunkt des Lichtes der vom Element zurückgeworfen wird
- Selbstschattierung kann in das I mit eingebracht werden, durch Abwägung der Helligkeit jeder Lichtquelle
- Abschätzung für ein „high-albedo“ Beleuchtungsmodell kann über hinzufügen eines ambienten Term, basiert auf dem albedo Wert des Materials hinzugefügt werden
- Albedo(weiß) = ist ein Maß für das Rückstrahlvermögen von diffus reflektierenden, also nicht selbst leuchtenden Oberflächen

Modellierung von Gasen – Volumetrischer Schatten

- volumetrischer Schatten ist wichtig für genaue Bilder
- Einfachste Möglichkeit
 - Verfolgung eines Strahles von jedem Volumen Element zu jedem Licht
 - Ermittlung der Deckkraft des Materials auf dem Strahl
- Schnellere Möglichkeit
 - Vorberechnete Tabelle nutzen die einmal pro Frame berechnet wird
 - Position der Schattenpunkte zu den Punkten in der Wolke
 - Punkt liegt in einer Parabellfläche (8 Tabelleneinträge)
- 2D Texture Mapping
 - Schatten von Semitransparenten Volumetrischen Objekten
 - Speichern einer Sichtbarkeitsfunktion in jedem Eintrag in der Tiefenschatten Map
 - Die Funktion speichert für jede Tiefe den Anteil des Lichtes das diese Tiefe erreicht
- Hardware beschleunigte Deckkraft Schatten Mapping Technik
 - Ähnlich zu dem 3D Hardware Textur basierten Volumen Rendering
 - nimmt eine Große Anzahl an Deckkraft Maps für die Berechnung Volumetrischen Schattens

Modellierung von Gasen – Solid Spaces

- Solid Spaces sind 3D Räume welche mit einem Objekt verbunden sind und die Attribute des Objektes verändern kann
- wird verwendet bei der Beschreibung von Objektattributen z.B. die Farbe
- Solid Farbraum wird für realistische Bilder von naturgemäßen Objekten verwendet

$$S(x, y, z) = F, F \in \mathbb{R}^n, n \in 1, 2, 3, \dots$$

- 3D Funktion kann auf N-Dimensionen ausgeweitet werden
 - z.B. auf die Zeit

Modellierung von Gasen – Gasmodellierung

Algorithmus

```
float gas (point P, float max_density, float exponent)
{
    float turb, density;
    turb = turbulence(pt);
    /* or turb = (1 + sin (turbulence(pt)*PI)/2) */
    density = pow(turb*max_density, exponent);
    return density;
}
```

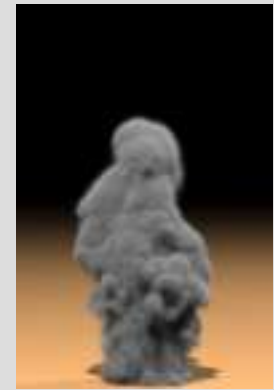
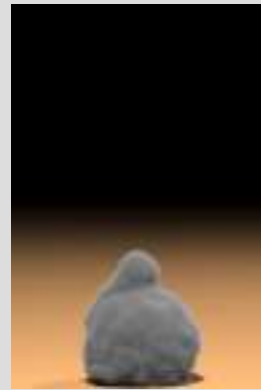
Modellierung von Gasen – Plazierung und Gasmodellierung



- Plazieren einiger primitiver Formen um Dichte Volumen zu erhalten
- Schwächer werdende Dichte um die Zerstreuung berücksichtigen zu können
- Dampfende Teetasse
 - schwächer werdende Dampf mit größer werdender Distanz vom Zentrum der Teeoberfläche
 - schwächer werdende Dampf zur Höhe über der Teeoberfläche

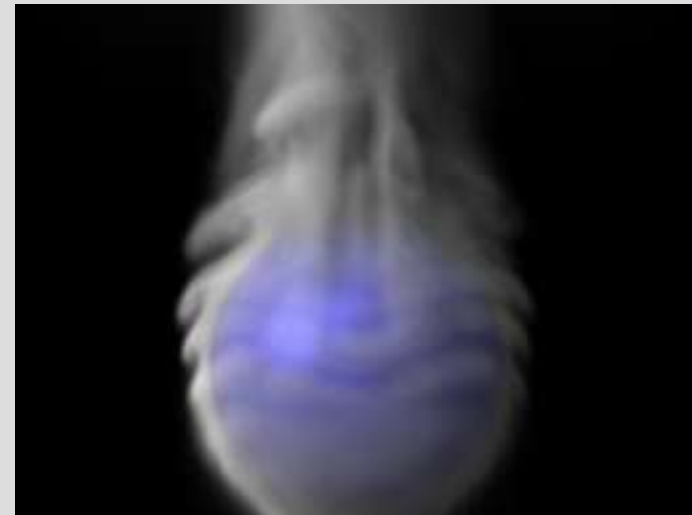
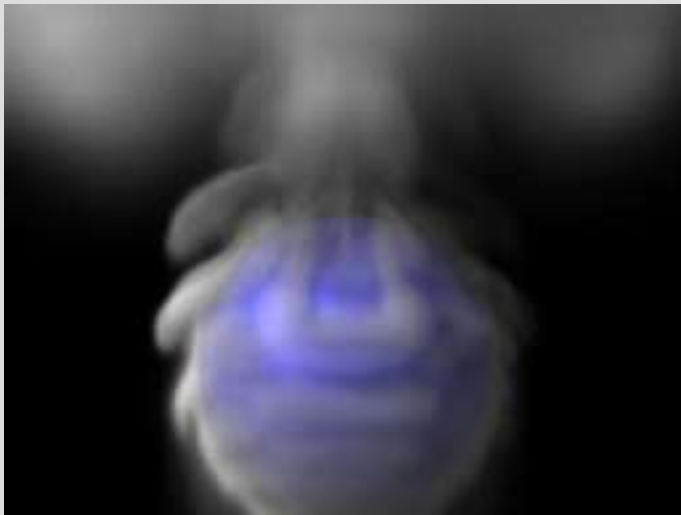
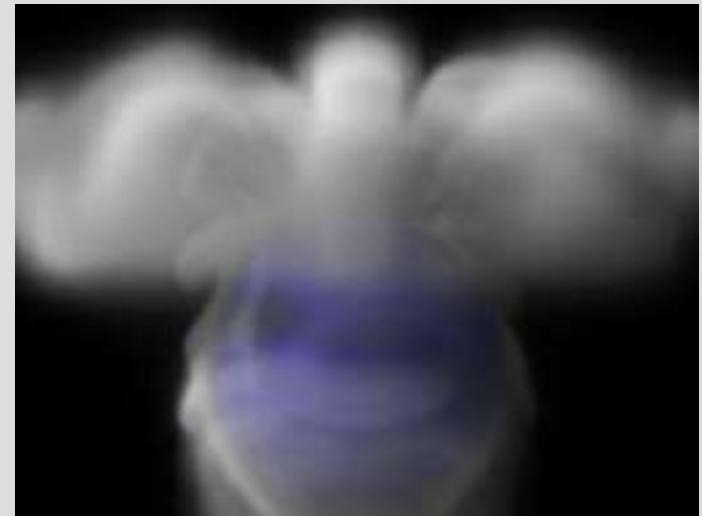
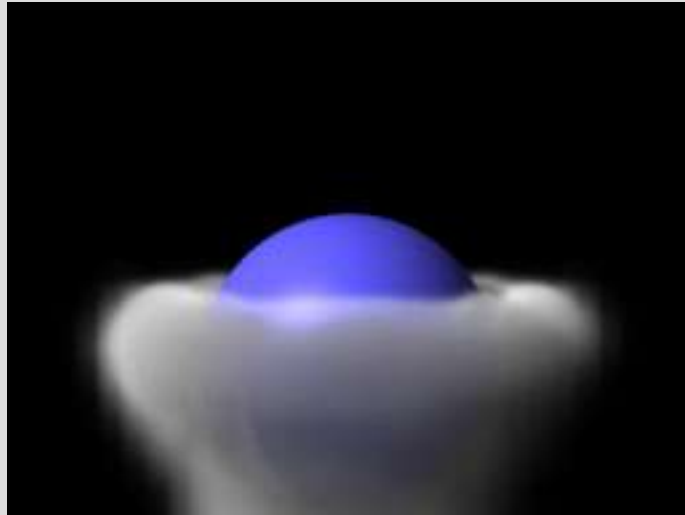
Modellierung von Gasen – Bilder

- Fedkiw



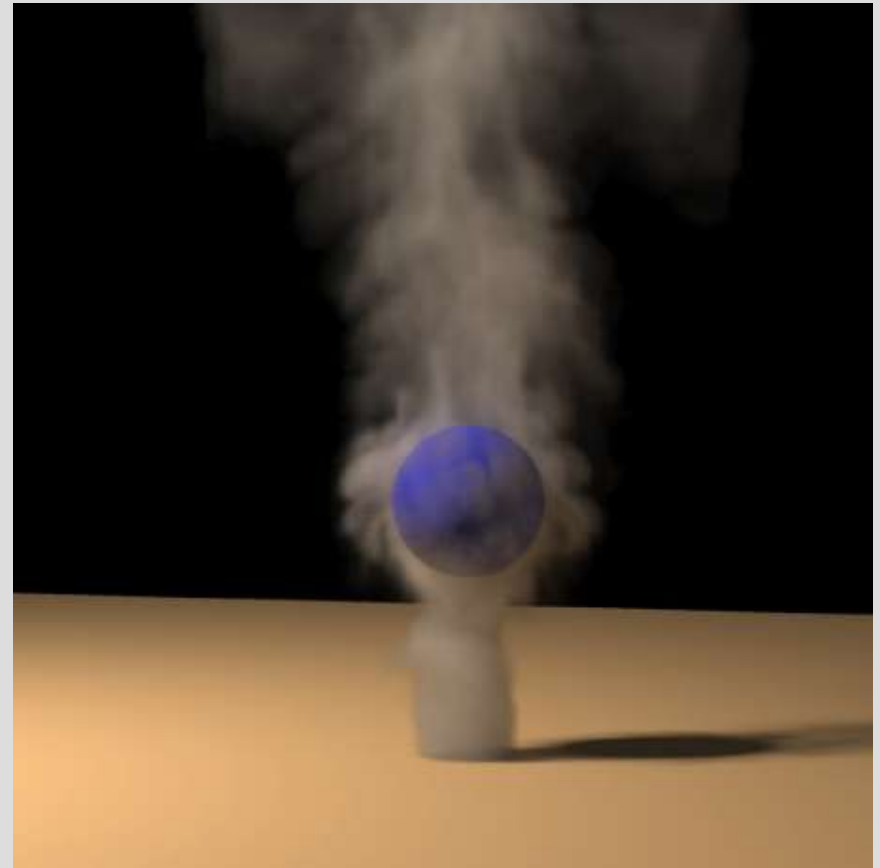
Modellierung von Gasen – Bilder

- Fedkiw



Modellierung von Gasen – Bilder

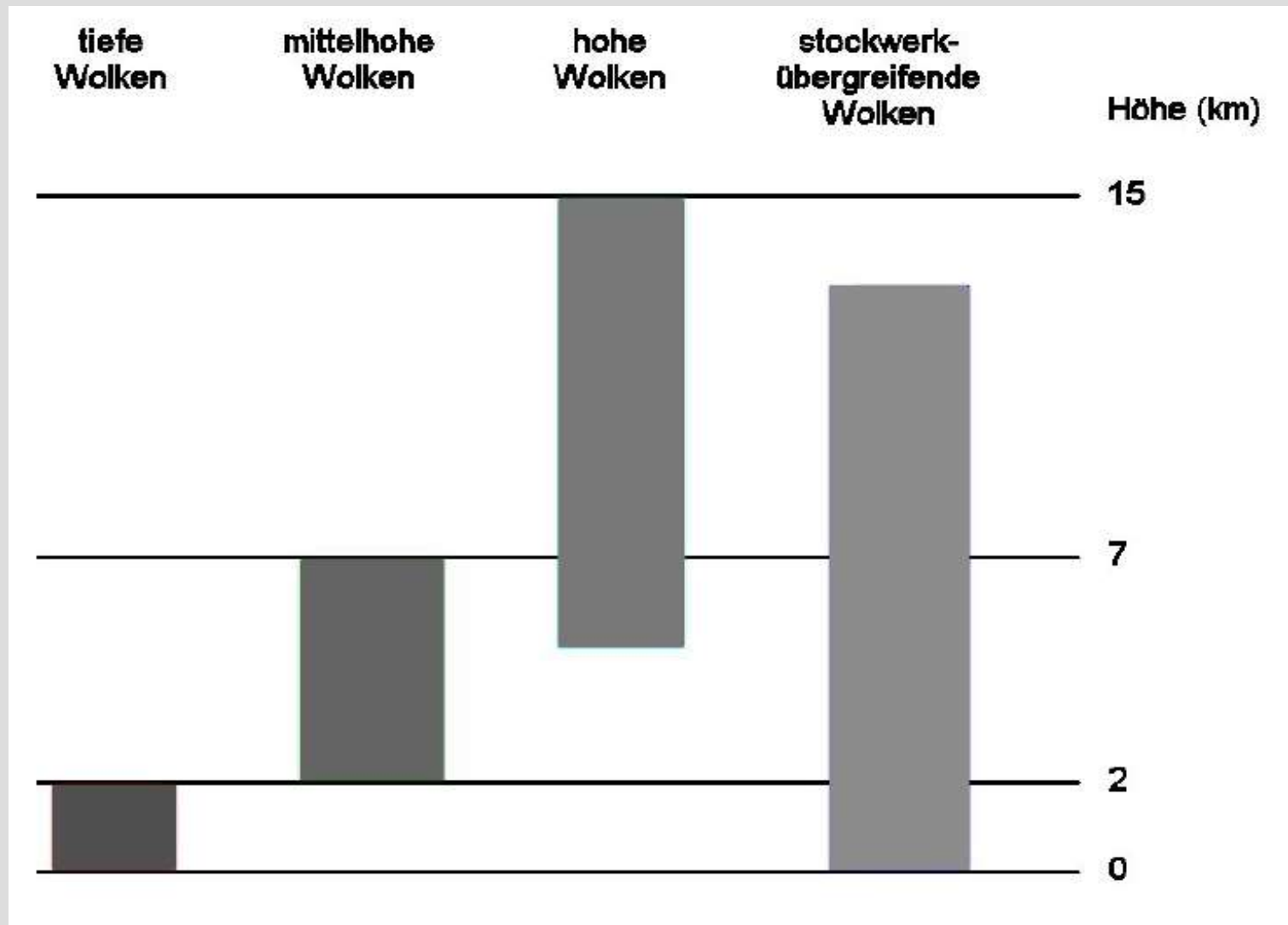
- Stam



Modellierung von Gasen – Videos

- Zwei Videos von Fedkiw

Volumetrische Wolken - Wolkenfamilie



Volumetrische Wolken - Wolkenfamilie

- Hohe Wolken
 - Untergrenze zwischen 5 und 15 Kilometer
 - Beginnen mit Cirr und sind Eiswolken
- Mittelhohe Wolken
 - Untergrenze 2 bis 7 Kilometer
 - Beginnen mit Alto und sind aus flüssigen und gefrorenem Wasser
- Tiefe Wolken
 - Höhe von 2 Kilometern
 - Nur flüssiges Wasser enthalten
- Stockwerkübergreifende Wolken
 - Vom Erdboden bis 13 Kilometer hoch reichen
 - In Namen Sileb nimb enthalten

Volumetrische Wolken - Wolkengattungen

- Haufenförmig
 - Enden auf cumulus
- Schichtförmig
 - Enden auf Silber stratus
- Schleierförmig
 - Heißen alle cirrus

Volumetrische Wolken - Ansatz

- zweidimensionale statische Bilder am Himmel plazieren -> Fotos oder am Rechner generierte Bilder
- Wolken ändern sich nicht in Abhängigkeit der Kamerabewegung
- Beleuchtung ändern sich nicht je nach Sonneneinfall auf die Wolke
- mit Hilfe der Rausch-Funktion zweidimensionale Wolken auf die virtuelle Himmelskugel abbilden
- gute Effekte: mehrere Ebenen übereinander stapeln, um Eindruck von 3D Wolken zu vermitteln

Volumetrische Wolken - Volumenvisualisierung

- bezeichnet die Repräsentation, Manipulation und Darstellung von Volumendaten
- umfasst alle Stufen der Visualisierungspipeline
 - Filterung der interessanten Daten aus den kompletten Rohdaten
 - Mapping -> Abbildung dieser Daten auf dargestellte Primitive
 - Rendering -> Bildgenerierung
- Volumendaten handelt es sich mathematisch gesehen um 3D-Skalarfelder
- Volumendaten liegen meist nur als skalare Werte an diskreten Gitterpunkten vor
 - Zwischenpunkte aus benachbarten Punkten werden durch Interpolation gewonnen
- Verfahren zur Visualisierung von Volumendaten lassen sich in zwei Klassen einteilen
 - indirekte Verfahren
 - direkte Verfahren

Volumetrische Wolken -

Indirekte Volumenvisualisierung

- bei indirekten Verfahren werden die Volumendaten einigen Vorverarbeitungsschritten unterzogen
 - intermediäre Repräsentation zu überführen
 - Transformation führt zu Informationsverlust im Vergleich zu den Rohdaten
 - erwünscht um Daten zu extrahieren
- Verfahren zur Merkmalsextraktion ist das Marching-Cubes-Verfahren
 - werden zu einem Schwellwert(Isowert), die Grenzfläche(Isofläche) extrahiert
 - generiert eine Geometrie, bestehend aus dem Eckpunkten von Dreiecken und deren Normalen
 - Gitterwürfels des Volumens einzeln betrachtet
 - für jeden Würfel werden die Eckpunkte danach klassifiziert, ob die Werte größer oder kleiner als der hervorgegebenen Isowert ist

Volumetrische Wolken -

Indirekte Volumenvisualisierung

- anhand dieser Klassifizierung läßt sich jedem Würfel ein Index zuweisen
- genaue Position für die Ecken der Dreiecke und deren Normalen lassen sich durch lineare Interpolation der Werte bzw. Normalen der Ecken bestimmen
- bei sehr komplexen Oberflächen kann die Zahl der durch das Marching-Cubes-Verfahren generierte Polygone sehr gross werden und die Optimierung wird schwer

Volumetrische Wolken -

Direkte Volumenvisualisierung

- Volumendaten direkt zur Darstellung verwendet
- zwei Bereiche unterteilen
 - Bildraumverfahren (image order)
 - Objektraumverfahren (object order)
- Bildraumverfahren
 - wird für jeden Bildpunkt ein Strahl vom Auge des Betrachters in das Volumen geschickt
 - um die Beträge des Volumens zu jedem Punkt zu ermitteln
- Verfahren die nur Primärstrahlen betrachten -> Ray-Casting-Verfahren
- Verfahren um Sekundärstrahlen zu generieren und zu verfolgen -> Ray-Tracing

Volumetrische Wolken - Föhnwolken (Altostratus lenticularis)



• http://www.positiv.ch/assets/images/db_images/db_foehnwolken_3001.jpg

- mittelhohen Wolken treten im Föhn auf
- Linsenförmige Scheiben
- Rausch-Funktion
 - Rand muss etwas unregelmäßig aussehen
 - Abhängig von zwei Eingangswerten (z- und y-Richtung) ->Unregelmäßigkeit
- Rand durch leicht abgeänderte Sinus-Funktion beschrieben -> nicht mehr symmetrisch
 - Addiert Bruchteil von Rausch Funktion -> Unregelmäßigkeit am Rand

Volumetrische Wolken - Haufenschichtwolken (*Stratocumulus stratiformis perlucidus*)



- <http://www.avvsilva.net/nuvensbaixas/nb360219.jpg>

- flache Haufenwolken mit einer großen horizontalen Ausdehnung
- gibt Wolkenlücken durch die Licht durchscheint
- kommt bei großen Wolken mit wenig Gitterpunkten aus
- Rausch-Funktion
 - Geringe Frequenz durch Skalierung der Rausch-Funktion
 - Große, zusammenhängende Flächen
 - Hoher Kontrast
- Turbulenz-Funktion verwendet um Eingangskordinaten zu verändern

Volumetrische Wolken - Cumulonimbus Capillatus Incus



http://upload.wikimedia.org/wikipedia/commons/thumb/9/98/Anvil_shaped_cumulus_panorama_edit_crop.jpg/750px-Anvil_shaped_cumulus_panorama_edit_crop.jpg

- Blumenkohlartige Oberflächenstruktur
- Rausch-Funktion
 - Große Flächen mit geringen Rauschen
 - durch Turbulenz-Funktion
 - Tiefe Einschnitte
 - Funktion an Mitte „umklappen“ und auf Turbulenzfunktion anwenden
 - Große zusammenhängende Flächen durch tiefe Furchen abgegrenzt

Volumetrische Wolken - Rendering

- Rendering besteht aus zwei Durchläufen
 - 1) Helligkeit an den Gitterpunkten berechnen
 - 2) Wolken werden gerendert
- Helligkeitsberechnung an den Gitterpunkten
 - Helligkeit einmal vor dem Rendern berechnet
 - Jeder Punkt des Wolkengitters bekommt eine Helligkeit zugewiesen
 - Verfahren:
 1. Berechne die entgegengesetzte Richtung des Lichtes in dem Koordinatensystem, das durch die Wolke beschrieben wird.
 2. Setze die Helligkeit der Gitterpunkte der drei beleuchteten Seiten auf 100%
 3. Ermittle den Richtungsvektor, der - ausgehend von einem Gitterpunkt - die nächste Ebene schneidet.
 4. Durchlaufe alle restlichen Gitterpunkte
 5. Berechne die Helligkeit anhand der benachbarten Punkte, die bereits bekannt sind
 6. Weise dem Helligkeitswert des Punktes die Transparenz zu

Volumetrische Wolken - Rendering

- Beim Rendern wird die Wolke Pixel für Pixel in das Bild gezeichnet
- Berechnung der Farbe und der Transparenz des Strahls vom Eintrittspunkt in die Wolke bis er sie wieder verläßt
- Intensität der Werte abhängig von der Dichte der Wolke
- Transparenz am Anfang der Berechnung gleich 1 nimmt in jedem Schritt in Abhängigkeit von der Dichte in der Mitte des Segments ab
- Farbe -> Dichte und Farbintensität muss beachtet werden

$$\text{Transparenz: } T_{neu} = T_{alt} - \left(\frac{1}{2}\right)^{p \cdot \text{falloff}^t}$$

$$\text{Intensität: } I_s = 1 - \left(\frac{1}{2}\right)^{p \cdot \text{falloff}^t}$$

$$\text{Absorption von Transparent: } I = I_s \cdot T_{alt} \\ \text{oder } I = T_{alt} - T_{neu}$$

Färbung des gesamten Strahles:

$$C_{neu} = C_{alt} + \text{Wolkenfarbe} \cdot \text{Sonnenfarbe} \cdot L \cdot I$$

p = Dichte der Wolke

falloff = Abnahme der Lichtintensität

Volumetrische Wolken - Wolken-Renderer mit OpenGL

- Grafikkarte kann keine volumetrischen Objekte darstellen
 - nur Polygone, Linien und Punkte
 - für Volumen rendern, müssen diese Objekte zu Hilfe genommen werden
 - „zerschneidet“ das Objekt in Scheiben, versieht jede mit einer Textur und rendert die Scheiben
- **Berechnung der Scheiben**
 - um Texturen auszunutzen, sollte Fläche die von Licht beschienen wird gross sein
 - als Normale der Fläche wird die Winkelhalbierende zwischen Lichtrichtung und Blickwinkel der Kamera verwendet
- **Berechnung der Helligkeit auf den Scheiben**
 - Berechnung einer Transformationsmatrix von Objektkoordinaten in Texturkoordinaten (geben Koordinaten in 3D-Textur an, welche die Dichteverteilung innerhalb der Wolke enthält)

Volumetrische Wolken - Wolken-Renderer mit OpenGL

- **Berechnung der Helligkeit auf den Scheiben**
 - Berechnung der Projektions-Matrix
 - Projektion je nach Lichtquelle verscheiden
 - Parallel-Projekt für Sonne
 - Perspektivische Projektion für Spot-Licht
 - Füllen der ersten Textur mit der Farbe der Lichtquelle
 - Rendern der weitem Texturen, mit aufsteigenden Abstand zur Lichtquelle
 - Vorhergehende Textur in aktuelle Textur kopiert
 - Abwächung des Lichtes der vorangegangenen Scheibe wirkt sich auf aktuelle Textur aus
 - Bei der trilinearen Interpolation (Grafikkarte) für die Abnahme des Lichtes und der Lichtfarbe, muss darauf geachtet werden, das die Scheibe mit dem größten Abstand zur Sonne später gerendet wird

Volumetrische Wolken - Wolken-Renderer mit OpenGL

- **Berechnung der Helligkeit auf den Scheiben**

- Stärke des Lichtes nach einer bestimmten Wegstrecke
- Weg abhängig von der Richtung der Normalen der Scheibe und der Richtung des Lichtes

$$S = \left(\frac{1}{2}\right)^{\frac{p \cdot s_L}{\text{falloff}}} \quad s_L = \frac{s_S}{\cos(\alpha)}$$

"Nach welcher Strecke nimmt die Helligkeit entlang des Strahles um die Hälfte ab bei voller Dichte"

p = Dichte der Wolke am Beginn des Lichtstrahles

s_L = Länge des zurückgelegten Weges

falloff = Abnahme der Lichtintensität

α = Winkel zwischen Normalen und Lichtrichtung

Volumetrische Wolken - Wolken-Renderer mit OpenGL

- Grafikkarte mischt das zu zeichnende Pixel mit dem dahinter liegenden Pixel

$$(r, g, b)_{neu} = a \cdot (r, g, b) + (1 - a) \cdot (r, g, b)_{alt}$$

- Licht wird gedunkelt, Farben verändern sich nicht

$$(r, g, b)_{neu} = (1 - a) \cdot (r, g, b)_{alt}$$

- Berechnung des Farb- und Alpha-Wertes des Pixels

$$(r, g, b, a) = \left(0, 0, 0, 1 - \left(\frac{1}{2} \right)^{P \frac{s_s}{(\vec{r}_L \cdot \vec{r}_N) \cdot falloff}} \right)$$

- $a = 0$
 - Hintergrundfarbe schein vollständig durch, kein Pixel wird gesetzt
 - Dichte an Stelle 0
- $a = 1$
 - Das neue Pixel ist schwarz, das Licht wird vollständig in Wolke absorbiert

Volumetrische Wolken - Wolken-Renderer mit OpenGL

- **Rendern der Wolke**
 - Scheiben absteigend nach ihrem Abstand zur Kamera sortiert und gezeichnet
 - Für jedes Pixel Farbe und Dichte der 3D-Wolkentextur ausgelesen
 - Wolkentextur mit Helligkeit an dem Punkt multipliziert
 - Scheiben unter Umständen zu sehen
 - 3D-Noise-Funktion als Textur in Grafikkarte speichern
 - Elementare Noise-Funktion als Textur speichern, Rest berechnen

Volumetrische Wolken –

Bilder

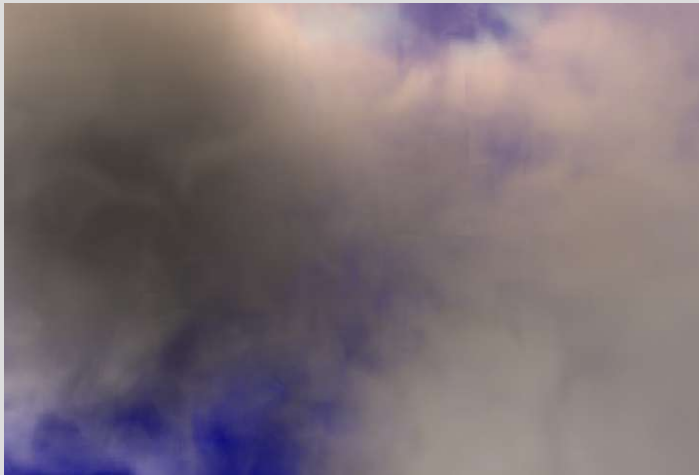
- Musgrave



Volumetrische Wolken –

Bilder

- Ebert



Volumetrische Wolken – Bilder

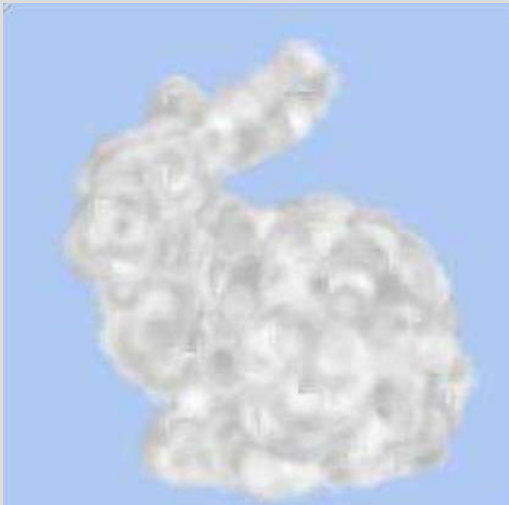
- T. Nishita und Y. Dobashi



Volumetrische Wolken –

Bilder

- N. Brickman, D.Olsen und G.Smith



Volumetrische Wolken –

Videos

- Videos von Musgave

Quellen

- Ebert, Musgrave, Peachey, Perlin, Worley Texturing und Modeling: „A Procedural Approach“
- <http://graphics.stanford.edu/~fedkiw/>
- <http://cobweb.ecn.purdue.edu/~ebertd/>
- n.Brickman, D. Olsen, G. Smith Shapes in the Clouds: Interactive, Artist-Guided Cloud Simulation
- T. Kunert Modellierung und Visualisierung von Wolken
- J. Krüger Echtzeitsimulation und -darstellung von Wolken
- T. Nishita und Y. Dobashi Modeling and Rendering of Various Natural Phenomena Consisting of Particles
- http://www.kenmusgrave.com/MojoWorld/How_To_Animate_Water.htm