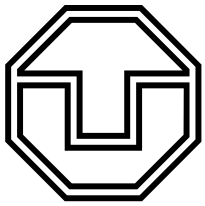

TECHNISCHE UNIVERSITÄT DRESDEN

Computergrafische Modellierung und Simulation von Rauch

Ein Hauptseminar an der
Technischen Universität Dresden,
Fakultät Informatik,
Institut für Software- und Multimediatechnik,
Professur für Computergrafik und -visualierung

eingereicht von
Stefan Koppitz
Dipl.cand Medien-Inf.
geboren am 02.10.1982,
Radebeul

und betreut von
Dipl.-Phys. Niels v. Festenberg
Prof. Dr. rer. nat. Stefan Gumhold

Inhaltsverzeichnis

Abbildungsverzeichnis

Kapitel 1: Einleitung

Diese Arbeit beschreibt eine Methode für die schnelle und stabile Rauchsimulation. Sie führt in die Dynamik von Fluiden ein und behandelt entsprechende mathematische Grundlagen. Es wird detailliert auf die Techniken und Algorithmen zur Simulation von Rauch, ausgehend von inkompressiblen Fluiden eingegangen.

1.1 Motivation

Fluide begegnen uns im Alltag an allen erdenklichen Stellen: Wasser in einer Pfütze, aufsteigender Rauch einer Zigarette, Dampf aus einem Teekessel. Die Modellierung solcher natürlicher Phänomene ist dagegen so schwer, wie oft die Phänomene in der Natur vorkommen. Das liegt vor allem an der sehr komplexen und turbulenten Bewegung, z. B. von Rauch. Das merkt man besonders bei der Betrachtung von Computerspielen. Bis vor kurzem gab es kaum Computerspiel in 3D, die Rauch oder verwandte Effekte enthielten, wobei Wasser- und Feuersimulationen schon länger zufriedenstellende Ergebnisse liefern.

Ein Fluidsimulator findet in diversen Bereichen seine Anwendung. In der Film- und Spieleindustrie herrscht ein großer Bedarf an überzeugender Nachahmung des Aussehens und Verhaltens von echten Fluiden wie Wasser, Rauch und Feuer. Daneben finden sich solche Simulatoren auch in wissenschaftlichen Disziplinen, in der Technik und im Ingenieurwesen wieder. Die meisten Aufgaben im Ingenieurwesen setzen voraus, dass die Simulation realistisch bezüglich der physikalischen Werte ist, um Fragen nach der Sicherheit und Performanz korrekt beantworten zu können. Die visuelle Qualität ist bei dieser Art von Anwendungen zweitrangig. In der Computergrafik sind dagegen Form und Verhalten der Fluide mehr von Bedeutung als die physikalische Korrektheit. Fluidsimulatoren sollten daher in der Computergrafik Realismus in Echtzeit erreichen. Diese Faktoren sind hier viel wichtiger als die physikalische Exaktheit, welche wesentlich mehr Rechenaufwand benötigen würde.

1.2 Fokus

Im Rahmen dieser Arbeit wird ein Modell vorgestellt, dass die Anforderungen der Computergrafik erfüllt. Realismus wird durch die numerische, implizite Lösung der inkompressiblen Navier-Stokes-Gleichungen erreicht. Um die Verwirbelungen realistisch darzustellen, wird das Vorticity Confinement eingesetzt. Desweiteren ist das Modell einfach zu implementieren, anzupassen und zu verwenden. Die Navier-Stokes-Gleichungen werden über eine grobes Voxelgitter gelöst, welches diverse vollständig eingetauchte und nicht vollständig eingetauchte Gegenstände bzw. Hindernisse beinhalten kann. Es können zudem physikalische Grenzen eingeführt werden, um die Bewegung des Fluids einzuschränken. Der eigentliche Rauch wird durch einen Hardware-basierten oder Photon-Map-Renderer dargestellt. Damit wird eine stabile, schnelle und robuste Rauchsimulation erreicht.

Die Techiken, die hier beschrieben werden, stammen aus dem Paper Fedkiw et al. „Visual Simulation of Smoke“ [FJ01] und bauen auf denen von [Sta99] und [FM97] auf.

1.3 Überblick

Im Kapitel 2 wird das Modell für die Beschreibung der Dynamik des Rauchs aufgestellt, hergeleitet und näher erläutert. Dabei wird auch auf den mathematischen Hintergrund eingegangen. Das Kapitel 3 befasst sich dann mit der Lösung des oben aufgestellten Modells. Das Kapitel 4 macht sich das finale Rendern des Rauchs mit Beleuchtung zum Gegenstand, um dann schließlich im 5. Kapitel einige Ergbnisse des hier vorgestellten Renderers auf Basis des vorgestellten Modells zu zeigen. Im 6. und letzten Kapitel wird das Modell der Rauchsimulation zusammengefasst und einen Ausblick auf die Zukunft des Themas in der Computergrafik gegeben.

Kapitel 2: Aufbau des Modells

Um die Dynamik, also die Bewegung des Rauchs über die Zeit hinweg, beschreiben zu können, muss ein Gleichungssystem aufgebaut werden. Das soll in diesem Kapitel geschehen. Dazu werden die mathematischen Grundlagen wie Vektorfelder, Divergenz, Rotation von Vektorfeldern und die beiden Operatoren Nabla und Laplace benötigt. Das Modell selbst besteht aus den Navier-Stokes-Gleichungen bzw. Euler-Gleichungen, den Temperatur- und Dichtetermen, dem thermischen Auftrieb und dem Vorticity Confinement.

2.1 Vorbetrachtungen

Um das Verhalten von Rauch zu simulieren, braucht man ein mathematische Darstellung des Zustandes zu jeder Zeit an jedem Ort. Hierbei ist die Geschwindigkeit des Rauchs die wichtigste Größe, die in dieser Simulation verwendet wird. Durch sie wird die Bewegung des Rauchs und der darin befindlichen Objekte direkt bestimmt. Diese Geschwindigkeit variiert in Zeit und Ort. Ein *Vektorfeld* kann zu jedem Ort zu jedem Zeitpunkt einen Zustand durch einen Geschwindigkeitsvektor darstellen. Unter einem Vektorfeld versteht man i. A. einen Raum, in dem jedem Punkt eine gerichtete Größe (Vektor) zugeordnet ist. In einem *Skalarfeld* wird dagegen eine ungerichtete Größe (Skalar) zugeordnet. Das Geschwindigkeitsfeld von Rauch wird hier so definiert, dass für jede Position $\vec{x} = (x, y, z)$ eine Geschwindigkeit zu einem Zeitpunkt t definiert ist:

$$\vec{u}(\vec{x}, t) = \begin{pmatrix} u(\vec{x}, t) \\ v(\vec{x}, t) \\ w(\vec{x}, t) \end{pmatrix} \tag{2.1}$$

Die Komponente $u(\vec{x}, t)$ stellt die Geschwindigkeit in x-Richtung, $v(\vec{x}, t)$ in y-Richtung und $w(\vec{x}, t)$ in z-Richtung dar.

Der Kern einer Fluidsimulation besteht aus der Bestimmung des korrekten Geschwindigkeitsfeldes nach jedem Zeitschritt. Das kann durch der Lösung eines Gleichungssystems erreicht werden, das die Entwicklung des Geschwindigkeitsfeldes über die Zeit und unter Einfluss verschiedener Kräfte beschreibt. Ist das Geschwindigkeitsfeld erst ermittelt,

kann man es benutzen, um Objekte, Dichten und andere Größen in ihm wandern zu lassen.

Um später die Gleichungssysteme des Modells lösen zu können, müssen weitere Eigenschaften des Vektorfeldes bzw. Skalarfeldes berechnet werden. Dazu benötigt man den *Nabla-Operator* ∇, ein formalen Vektor, dessen Komponenten aus den partiellen Differential-Operatoren bestehen. Der *Gradient* eines Skalarfeldes

$$\nabla p = \left(\frac{\partial p}{\partial x}, \frac{\partial p}{\partial y}, \frac{\partial p}{\partial z} \right) \tag{2.2}$$

nutzt diesen Operator und ist ein Vektor aus partiellen Ableitungen des Skalarfeldes. Später wird ein reguläres, diskretes Voxelgitter mit äquidistanten Gitterknoten eingeführt, auf dem man die Berechnung des Gradienten mit Hilfe der finiten Differenzen lösen kann:

$$\frac{p_{i+1,j,k} - p_{i-1,j,k}}{2h}, \frac{p_{i,j+1,k} - p_{i,j-1,k}}{2h}, \frac{p_{i,j,k+1} - p_{i,j,k-1}}{2h}, \tag{2.3}$$

wobei i, j, k Laufindizes in die x-, y- und z-Richtung des Gitters sind, und h dem Gitterknotenabstand entspricht. Die *Divergenz*

$$\nabla \cdot \vec{u} = \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} \tag{2.4}$$

wird für ein Vektorfeld berechnet und ist selbst ein Skalar. Numerisch wird sie mit

$$\frac{u_{i+1,j,k} - u_{i-1,j,k}}{2h} + \frac{v_{i,j+1,k} - v_{i,j-1,k}}{2h} + \frac{w_{i,j,k+1} - w_{i,j,k-1}}{2h} \tag{2.5}$$

berechnet. Wird der Divergenz-Operator auf das Ergebnis des Gradienten angewandt, erhält man den *Laplace-Operator* $\nabla \cdot \nabla = \nabla^2$. Angewendet auf ein Skalarfeld erhält man

$$\nabla^2 p = \frac{\partial^2 p}{\partial x^2} + \frac{\partial^2 p}{\partial y^2} + \frac{\partial^2 p}{\partial z^2}, \tag{2.6}$$

was man wieder numerisch mittels der finiten Differenzen lösen kann:

$$\frac{p_{i+1,j,k} - 2p_{i,j,k} + p_{i-1,j,k}}{h^2} + \frac{p_{i,j+1,k} - 2p_{i,j,k} + p_{i,j-1,k}}{h^2} + \frac{p_{i,j,k+1} - 2p_{i,j,k} + p_{i,j,k-1}}{h^2} \tag{2.7}$$

Die *Rotation*

$$\nabla \times \vec{u} = \begin{pmatrix} w_y - v_z \\ u_z - w_x \\ v_x - u_y \end{pmatrix} \tag{2.8}$$

kann ebenfalls nur für ein Vektorfeld berechnet werden und ist selbst wieder ein Vektor. Der Rotationsvektor gibt für jeden Ort an, wie stark und um welche Drehachse das Fluid im Vektorfeld rotiert. Hier sind keine finiten Differenzen nötig, da schon diskrete Werte vorliegen.

2.2 Veränderung von Geschwindigkeit und Druck

Das Verhalten von Rauch bzw. des Transportmediums, indem sich der Rauch bewegt, kann mittels eines mathematisch-physikalischen Modells beschrieben werden: den *Navier-Stokes-Gleichungen*. Diese Gleichungen beschreiben allgemein *inkompressible* und *homogene* Fluide. Die vereinfachenden Annahmen müssen wie so oft in der Physik getroffen werden, um komplexe Phänomene überhaupt beschreiben zu können. Ein Fluid ist inkompressibel, wenn das Volumen an jeder Stelle des Fluids über die Zeit konstant bleibt. Ein Fluid ist homogen, wenn seine Dichte im Raum konstant ist. Nimmt man beide Eigenschaften zur gleichen Zeit an, ist die Dichte des Fluids sowohl räumlich als auch zeitlich gesehen konstant. Diese Annahmen sind in der Fluid-Dynamik Standard und schränken keineswegs die Nutzbarkeit des Modells ein.

Zur Simulation der Dynamik von Rauch soll hier ein reguläres Voxelgitter mit räumlichen Koordinaten $\vec{x} = (x, y, z)$ und der veränderlichen Zeit t betrachtet werden. Das Transportmedium (typischerweise heißes, aufsteigendes Gas oder Luft) wird durch das Geschwindigkeitsfeld $\vec{u}(\vec{x}, t)$ und dem skalaren Druckfeld $p(\vec{x}, t)$ dargestellt. Beide Felder variieren in Zeit und Raum. Sind Geschwindigkeit und Druck zum initialen Zeitpunkt $t = 0$ bekannt, kann der Zustand des Fluids über die Zeit durch die Navier-Stokes-Gleichungen in inkompressibler Form beschrieben werden:

$$\frac{\partial \vec{u}}{\partial t} = -(\vec{u} \cdot \nabla)\vec{u} - \frac{1}{\rho}\nabla p + \nu \nabla^2 \vec{u} + \vec{F}, \tag{2.9}$$

$$\nabla \cdot \vec{u} = 0, \tag{2.10}$$

wobei ρ die (konstante) Dichte des Mediums, ν die kinematische Viskosität und $\vec{F} = (f_x, f_y, f_z)$ eine von außen auf das Medium einwirkende Kraft (z. B. Gravitation) sind. Dabei beschreibt Gleichung 2.9 die Impulserhaltung und Gleichung 2.10 die Massenerhaltung bzw. Divergenzfreiheit. Aus der Divergenzfreiheit folgt, dass genauso viel Masse in das System fließt, wie herausfließt. Die Masse bleibt somit im System konstant. Damit kann der Kraftterm in der Gleichung 2.9 nach dem 2. Newtonschen Axiom $F = m \cdot a$ als Beschleunigung verstanden werden. Die Richtungsvektoren $\vec{u}$ geben an einer beliebigen Position $\vec{x}$ an, mit welchem Betrag und in welche Richtung die Strömung fließt. Im dreidimensionalen Fall besteht ein Richtungsvektor aus drei Komponenten u, v und w. Pro Richtungskomponente gibt es somit eine partielle Differentialgleichung:

$$\frac{\partial u}{\partial t} = -(\vec{u} \cdot \nabla)u - \frac{1}{\rho}\nabla p + \nu \nabla^2 u + f_x, \tag{2.11}$$

$$\frac{\partial v}{\partial t} = -(\vec{u} \cdot \nabla)v - \frac{1}{\rho}\nabla p + \nu \nabla^2 v + f_y. \tag{2.12}$$

$$\frac{\partial w}{\partial t} = -(\vec{u} \cdot \nabla)w - \frac{1}{\rho}\nabla p + \nu \nabla^2 w + f_z. \tag{2.13}$$

Für ein dreidimensionalen Voxelgitter ist es nun möglich die vier Unbekannten (u, v, w und p) mit den vier Gleichungen zu bestimmen.

Die kinematische Viskosität aus der Gleichung 2.9 beschreibt, wie zäh ein Fluid ist. Werden aber eher weniger zähe Fluide, wie z. B. Gase und Rauch, simuliert, kann dieser Term vernachlässigt werden, besonders auf groben Gittern, wo numerische Instabilitäten auftreten, die wiederum als Viskosität betrachtet werden können. Die Navier-Stokes-Gleichung 2.9 vereinfacht sich so zu

$$\frac{\partial \vec{u}}{\partial t} = -(\vec{u} \cdot \nabla)\vec{u} - \frac{1}{\rho}\nabla p + \vec{F}, \tag{2.14}$$

und man erhält die sogenannte *Euler-Gleichungen*.

Um die Euler-Gleichung 2.14 besser zu verstehen, kann man diese in kleinere, einfachere Teile zerlegen:

Advektion. Die Geschwindigkeit eines Fluids bewirkt einen Transport von Objekten, Dichten und anderen Größen entlang der Strömung. Auch das Fluid selbst wird durch diesen Mechanismus transportiert, und diese Selbstadvektion wird durch den ersten Term repräsentiert.

Druck. Da sich die Moleküle in einem Fluid zu einem gewissen Grad frei bewegen können, breitet sich eine Kraftaufbringung nicht sofort aus. Es baut sich jedoch ein Druckunterschied auf, welcher dann zu einer Beschleunigung führt (siehe 2. Newtonsches Axiom: $F = m \cdot a$) - und zwar von Orten größeren Drucks zu Orten niedrigeren Drucks und somit entgegengesetzt zum Gradienten des Druckfeldes. Diese Beschleunigung wird durch den zweiten Term ausgedrückt und ist umgekehrt proportional zur Dichte des Mediums.

Externe Kräfte. Eine Beschleunigung des Fluids aufgrund von äußeren Einflüssen wird mit Hilfe des dritten Terms dargestellt. Diese Kräfte können lokale Kräfte sein, die nur auf einen bestimmten Bereich des Fluids wirken oder Kräfte, die gleichmäßig auf das gesamte Fluid wirken.

2.3 Veränderung von Temperatur und Dichte

Zur korrekten Simulation von Rauch benötigt man noch Gleichungen, die die zeitliche Entwicklung von Temperatur T_s und Dichte ρ_s des Rauch beschreiben. Hier nimmt man an, dass diese beiden Skalare einfach mit dem Transportmedium und so entlang des Geschwindigkeitsfeldes $\vec{u}$ bewegt werden (Advektion):

$$\frac{\partial T_s}{\partial t} = -(\vec{u} \cdot \nabla)T_s, \quad \frac{\partial \rho_s}{\partial t} = -(\vec{u} \cdot \nabla)\rho_s. \tag{2.15}$$

2.4 Thermischer Auftrieb

Schließlich muss noch der Effekt des thermischen Auftriebs beachtet werden, der vor allem bei heißen Gasen also Rauch auftritt. Durch den Auftrieb steigen Gase nach oben, wogegen schwere Gase durch Wirken der Schwerkraft nach unten fallen. Um diesen Effekt in das Modell einzubauen, wird eine zusätzliche externe Kraft definiert, die direkt proportional zur Dichte und Temperatur des Rauchs ist:

$$f_{buoy} = -\alpha\rho_s\vec{z} + \beta(T_s - T_{amb})\vec{z}, \tag{2.16}$$

wobei $\vec{z} = (0, 0, 1)$ direkt in die z-Richtung zeigt, T_{amb} die Umgebungstemperatur der Luft, α ein konstanter Massenskalierungsfaktor und β ebenfalls ein konstanter Skalierungsfaktor sind. Bei der obige Gleichung entsteht bei steigender Dichte des Rauchs und bei gleichbleibendem Volumen eine Kraft, die entgegen der Auftriebsrichtung wirkt. Grund ist die steigende Masse, die eine Kraft auf das Fluid ausübt und so den Rauch nach unten drückt. Der thermische Auftrieb führt auch nur dort Kraft hinzu, wo die lokale Temperatur höher ist als die Umgebungstemperatur. Die einfachste Art diese neue Kraft in die Simulation zu integrieren, ist das Hinzufügen eines Skalarfeldes für die Temperatur und Dichte des Rauchs.

2.5 Vorticity Confinement

Die Bewegung von Rauch, Luft und anderen nieder-viskosen Fluiden beinhaltet typischerweise rotierende und turbulente Strömungen in verschiedenen Skalierungsstufen. Wie Fedkiw et al. in [FJ01] feststellten, werden diese interessanten Details durch numerische Dämpfung, hervorgerufen durch die Verwendung von impliziten Näherungsverfahren auf groben Gittern, abgeschwächt. Eine Möglichkeit diese Turbulenzen wieder in die Simulation einzufügen, wäre die zufällige Generierung von Wirbeln im Fluid. Das hat aber den Nachteil, dass die Turbulenzen nicht an den Stellen hinzugefügt werden, wo sie verloren gegangen sind. Das führt dazu, dass der Rauch zu lebendig wirkt, da er sich nicht mehr physikalisch korrekt bewegt. Hier kommt das Vorticity Confinement zum Einsatz. Die Idee ist dabei die durch die numerische Dämpfung verloren gegangene Energie wieder an den Stellen einzufügen, wo sie verloren gegangen sind. Diese Stellen bzw. Verwirbelungen können durch das Rotationsfeld

$$\vec{\omega} = \nabla \times \vec{u} \tag{2.17}$$

gefunden werden. Dieses Vektorfeld, auch Vortizität oder Wirbelstärke genannt, beschreibt die Rotation des Geschwindigkeitsfeldes $\vec{u}$ an jedem beliebigen Ort zu jedem Zeitpunkt. Die Vektoren des Rotationsfeldes stehen dabei senkrecht auf dem Geschwindigkeitsfeld, und deren Länge ist ein Maß für die Stärke der Rotation an dieser Stelle (siehe Abbildung 2.1). Mit Hilfe des Rotationsfeldes berechnet man ein normiertes Vektorfeld:

$$\vec{N} = \frac{\eta}{|\eta|}, \tag{2.18}$$

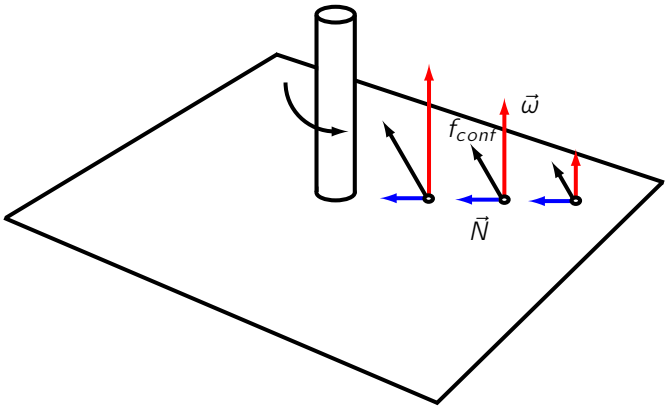

Abbildung 2.1: *Vorticity Confinement: Durch Dämpfung verloren gegangene Energie wird durch Einführen einer Kraft wiederhergestellt. Dazu wird zuerst ein Rotationsfeld (rot), dann ein Gradientenfeld (blau) bestimmt und anschließend die Kraft senkrecht zu den beiden Vektoren ausgeübt. Der Zylinder mit Pfeil soll die Rotationsrichtung im Geschwindigkeitsfeld darstellen.*

wobei $\eta = \nabla|\vec{\omega}|$ ist. Im Detail betrachtet, wird das Rotationsfeld in ein Skalarfeld transformiert, indem die Rotationsvektoren auf das Geschwindigkeitsfeld projiziert werden. Dabei gehen die Richtungsinformationen verloren, die Rotationsstärken sind jedoch als Skalare im Feld gespeichert. Wendet man nun den Nabla-Operator auf dieses Skalarfeld an, entsteht wieder ein Vektorfeld. Die Vektoren in diesem Gradienten-Vektorfeld zeigen von Orten niedrigerer Wirbelstärke zu Orten höherer Wirbelstärke. Schließlich wird eine Kraft berechnet, die die verloren gegangene Wirbelstärke annähernd wiederherstellt, indem diese senkrecht zum Gradientenfeld und Rotationsfeld hinzugefügt wird:

$$f_{conf} = \epsilon(\vec{N} \times \vec{\omega})h, \tag{2.19}$$

wobei ϵ positiv ist und einen nutzergesteuerten Skalierungsparameter darstellt, der den Betrag an wieder hinzuzufügender Energie steuert. h entspricht dem Abstand von zwei aufeinander folgenden Gittervoxel. Er sollt in alle drei Dimensionen gleich sein.

Kapitel 3: Lösung des Modells

Zur Lösung der im vorher benannten Modell aufgestellten Euler-Gleichungen nutzt man üblicherweise inkrementelle, numerische Verfahren. Denn für die benötigten Euler-Gleichungen gibt es keine bekannten analytischen Lösungen (exklusive einiger weniger, einfacher physikalischer Konfigurationen). Numerische, inkrementelle Verfahren liefern nach jedem Zeitpunkt ein darstellbares Ergebnis, stellen die Entwicklung der Strömung über die Zeit gut dar und reichen für diese Art Anwendung völlig aus.

Zunächst wird behandelt, wie mit völlig bzw. nicht völlig eingetauchten Objekten im Rauch verfahren wird, um dann schließlich den Algorithmus für die Lösung des aufgestellten Modells näher zu erläutern. Vorher soll noch die dem Algorithmus zu Grund liegende Raumstruktur erklärt werden.

3.1 Gestaffeltes Voxelgitter

Zur numerischen Lösung des Modells wird ein diskretisierter Raum benötigt, hier ein reguläres Voxelgitter mit äquidistanten Gitterknoten. Desweiteren wird die Finite-Element-Methode verwendet, die den Raum erst in die genannten, endlich aber gleich großen Voxel aufteilt. Zur Platzierung der Größen soll hier das sogenannte *gestaffelte Voxelgitter* Verwendung finden. Jenes definiert skalare Größen wie die Temperatur, die Rauchdichte und die externen Kräfte im Zentrum jeder Voxelzelle, wogegen die vektoriellen Größen wie die Geschwindigkeitskomponenten u, v und w an den Mittelpunkten der entsprechenden Voxelzellenseiten definiert sind (siehe Abbildung 3.1). Diese Methode ist numerisch stabiler und leidet unter weniger künstlicher Dämpfung. So tritt hier nicht das Problem einer nicht eindeutigen Nullstelle bei der Berechnung der Divergenz auf. Weiterhin lassen sich so die Randbedingungen für die Simulation besser modellieren.

Zu beachten ist noch die Indizierung der Vektorkomponenten. Da diese nicht im Zellmittelpunkt liegen, muss in entsprechender Richtung die Zellenseite durch Addition mit der Hälfte des Zellabstandes adressiert werden.

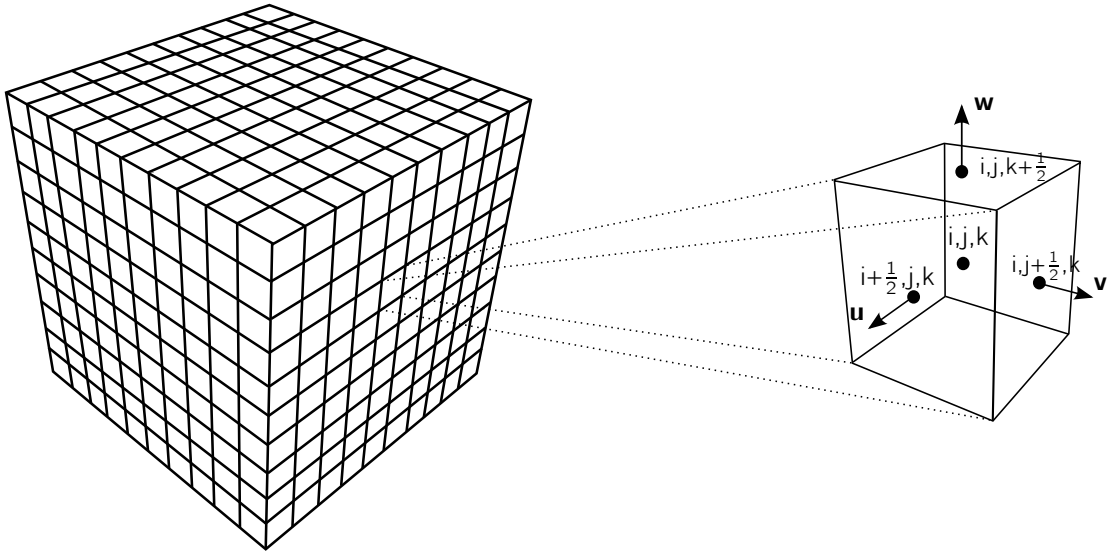

Abbildung 3.1: *Voxelgitter und -zelle: Durch Diskretisierung des Raums entsteht ein Voxelgitter bestehend aus gleichen, kubischen Voxelzellen (links). Die Geschwindigkeitskomponenten werden an den entsprechenden Seiten der Voxelzelle gespeichert, alle Skalare im Zentrum (rechts). Die Indizierung der Komponenten ist mit i, j und k angeben.*

3.2 Randbedingungen

Das vorgestellte Voxelgitter ermöglicht nicht nur die effiziente Lösung der Gleichungen, sondern macht es auch möglich das Verhalten der Interaktion von Rauch mit Objekten zu modellieren. Dabei werden alle Voxelzellen, die ein Objekt schneiden, markiert. Anschließend wird die Geschwindigkeit aller Seiten jener markierte Zellen auf die Geschwindigkeit des getroffenen Objekts gesetzt. Die Temperatur im Zentrum der markierten Zellen wird gleichermaßen an die des Objekts angepasst. Die Dichte innerhalb der markierten Zellen wird auf Null gesetzt. Um ein plötzlichen Abfall der Dichte am Rand von Objekten zu vermeiden, wird die Dichte an den Randvoxeln des Objekts an die Dichte der nächsten unmarkierten Voxelzelle angepasst (siehe Abbildung 3.2).

An den Begrenzungen des Simulationsraums wird die No-Slip-Bedingung angenommen. Das heißt, die Geschwindigkeit geht am Rand gegen Null. Zur korrekten Lösung der später vorgestellten Poisson-Gleichung benötigt man die reine Neumann-Randbedingung: $\partial p / \partial \vec{n} = 0$. Diese besagt, dass die Änderung des Drucks entlang der Normalen des Rands Null ist.

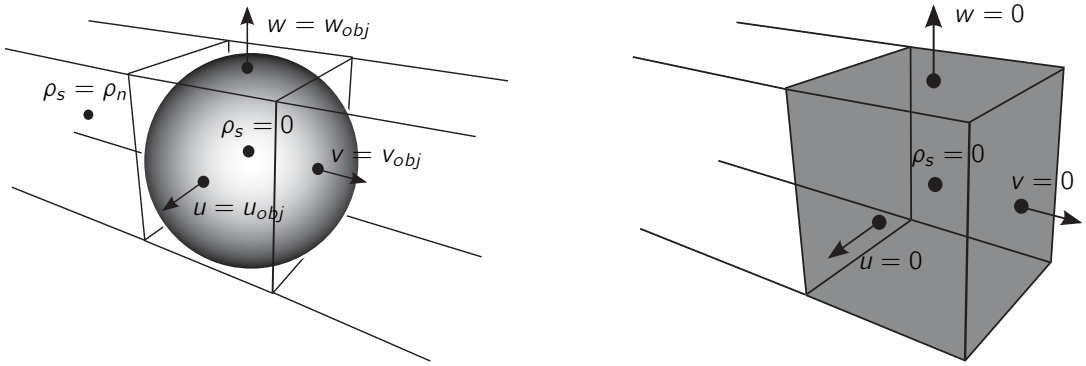

Abbildung 3.2: *Interaktion mit Hindernissen: Bei Interaktion mit einem im Rauch befindlichen Objekt, werden die geschnittenen Voxelzellen markiert und deren Werte wie in der Abb. gesetzt (links). An den Volumengrenzen wird die Geschwindigkeit und Dichte auf Null gesetzt, so dass kein Rauch durch diese gelangen (rechts).*

3.3 Lösung der Euler-Gleichungen

Im Folgendem werden zuerst die Euler-Gleichungen durch Dekomposition in einfache Schritte aufgeteilt. Diese Methode entspricht dem Verfahren von [Sta99]. Hier soll auch die mathematische Herleitung jedes einzelnen Schritts motiviert werden. Damit die Gleichungen numerisch lösbar sind, müssen sie noch umgeformt werden. Dazu sollten die vier bereits erklärten Gleichungen Gleichung 2.3, Gleichung 2.5, Gleichung 2.7 und Gleichung 2.8 genutzt werden.

3.3.1 Helmholtz-Hodge Dekomposition

Wie es möglich ist Vektoren in linear unabhängige Vektoren zu zerlegen, so ist es auch möglich Vektorfelder in eines Summe von Vektorfelder zu zerlegen. Nach der *Helmholtz-Hodge-Dekomposition* kann man ein Vektorfeld $\vec{w}$ im dreidimensionalem Raum D eindeutig in folgende Form zerlegen:

$$\vec{w} = \vec{u} + \nabla p, \tag{3.1}$$

wobei $\nabla \cdot \vec{u} = 0$ gilt. Das heißt, jedes Vektorfeld kann in eine Summe aus einem divergenzfreien Vektorfeld und einem Gradient eines Skalarfeldes zerlegt werden.

Zur Lösung der Euler-Gleichungen müssen zwei Berechnungen zum Update der Geschwindigkeit pro Zeitschritt gemacht werden: Advektion und Kraftausübung. Das Ergebnis ist ein neues Geschwindigkeitsfeld $\vec{w}$ mit einer Divergenz von ungleich Null. Aber Gleichung 2.10 setzt voraus, dass am Ende jedes Zeitschritts ein divergenzfreies Geschwindigkeitsfeld vorliegt. Hier hilft die Helmholtz-Hodge-Dekomposition, die besagt, dass die Geschwindigkeit in $\vec{w}$ durch Subtraktion des Gradienten des Druckfeldes korrigiert werden

kann:

$$\vec{u} = \vec{w} - \nabla p. \tag{3.2}$$

Um eine Gleichung zur Berechnung des Druckfeldes p zu jedem Zeitpunkt zu erhalten, wird auf beiden Seiten der Gleichung 3.1 der Nabla-Operator hinzumultipliziert:

$$\begin{aligned} \nabla \cdot \vec{w} &= \nabla \cdot (\vec{u} + \nabla p) && (3.3) \\ &= \nabla \cdot \vec{u} + \nabla^2 p. && (3.4) \end{aligned}$$

Durch die Annahme aus Gleichung 2.10 entsteht so

$$\nabla^2 p = \nabla \cdot \vec{w}. \tag{3.5}$$

In der letzten Zeile steht eine sogenannte *Poisson-Gleichung* mit Hilfe derer man in der Lage ist, das Druckfeld zu berechnen. Nachdem das divergente Geschwindigkeitsfeld $\vec{w}$ berechnet wurde, kann mit Hilfe dessen die Gleichung 3.5 gelöst werden. Hat man $\vec{w}$ und p, kann man anschließend Gleichung 3.2 berechnen und erhält so das divergenzfreie Geschwindigkeitsfeld $\vec{u}$.

3.3.2 Projektion

Zurzeit ist $\vec{w}$ noch sehr umständlich zu berechnen. Deshalb wird ein *Projektionsoperator* P eingeführt, der das Vektorfeld $\vec{w}$ unter Nutzung der Helmholtz-Hodge-Dekomposition auf das divergenzfreie Vektorfeld $\vec{u}$ abbildet. Wendet man diese Projektion auf Gleichung 3.1 an, entsteht

$$P\vec{w} = P\vec{u} + P(\nabla p). \tag{3.6}$$

Aus der Definition von P folgt $P\vec{w} = \vec{u} = P\vec{u}$. Das heißt der Term $P(\nabla p)$ ist Null. Wird nun diese Projektion auf Gleichung 2.14 angewendet

$$P\frac{\partial \vec{u}}{\partial t} = P\left(-(\vec{u} \cdot \nabla)\vec{u} - \frac{1}{\rho}\nabla p + \vec{F}\right), \tag{3.7}$$

entfällt der Druckterm und, da $\vec{u}$ divergenzfrei ist, auch der Projektionsoperator auf der linken Seite, und man erhält

$$\frac{\partial \vec{u}}{\partial t} = P\left(-(\vec{u} \cdot \nabla)\vec{u} + \vec{F}\right). \tag{3.8}$$

Nun hat man eine Gleichung in der der gesamte Algorithmus auf einem Blick ersichtlich ist. Zuerst wird der Term in der Klammer berechnet. Dazu müssen der Advektionsterm und Krafterm bestimmt werden. Das Ergbnis ist ein divergentes Geschwindigkeitsfeld $\vec{w}$, auf das der Projektionsoperator P angewendet wird, um das divergenzfreie Geschwindigkeitsfeld $\vec{u}$ zu erhalten. Die Projektion beinhaltet dabei die Berechnungen der Gleichung 3.5 und anschließend die Berechnung der Gleichung 3.2.

3.3.3 Advektionsterm

Da ein explizites Lösungsverfahren bei großen Zeitschritten zu Ungenauigkeiten bis zur Explosion des Ergebnisses führt, soll hier ein implizites, numerisches Verfahren verwendet werden: das Semi-Lagrange'sche Verfahren. Bei besagtem handelt es ich um ein partikelbasiertes Verfahren, wo Partikel durch ein Vektorfeld transportiert werden. Betrachtet man die Voxelmittelpunkte als Partikel, bewegen sich diese auf einer Bahn, die durch $\vec{u}(\vec{x}, t)$ beschrieben ist, durch das Fluid. Verfolgt man das Partikel in der Zeit vorwärts, muss dieses nicht unbedingt in einem Zellmittelpunkt landen, was jedoch nach Definition des gestaffelten Gitters gefordert ist. Deshalb müssen die Partikel gefunden werden, die nach einem Zeitschritt in einem Zellmittelpunkt enden. Das wird erreicht, indem man jedes Partikel rückwärts in der Zeit verfolgt, um so den neuen Wert für die mitbewegte Größe (z. B. Geschwindigkeit) zu erhalten. Der Wert an dieser Stelle wird dann an die entsprechenden Zellseiten des Startvoxels des Partikels - in der Zukunft - geschrieben:

$$q(\vec{x}, t + \Delta t) = q(\vec{x} - \vec{u}(\vec{x}, t)\Delta t, t). \tag{3.9}$$

Die Endposition des Partikel wird sich nicht unbedingt in einem Zellmittelpunkt befinden (siehe Abbildung 3.3). Der Wert an dieser Position $q(\vec{x} - \vec{u}(\vec{x}, t)\Delta t, t)$ kann jedoch durch trilineare Interpolation der Werte der vier nächsten Nachbarn berechnet werden.

Der Nachteil des Semi-Lagrange'schen Lösungsverfahrens ist die Dämpfung der Bewegung des Fluids. Dafür wurde aber schon wie besprochen das Vorticity Confinement eingeführt.

3.3.4 Kraftterm

Der Kraftterm besteht aus den verschiedenen extern auf den Rauch einwirkenden Kräfte:

$$\vec{F} = f_{buoy} + f_{conf} + f_{user} \tag{3.10}$$

Diese sind thermischer Auftrieb in Gleichung 2.16, Vorticity Confinement in Gleichung 2.19 und zusätzlich vom Nutzer eingebrachte Kräfte. Der Kraftterm wird durch Multiplikation mit dem Zeitschritt Δt und anschließender Addition mit der Geschwindigkeit in die Simulation eingebracht.

3.3.5 Poisson-Gleichung

Damit das Geschwindigkeitsfeld $\vec{w}$ masseerhaltend bzw. divergenzfrei wird, muss die Poisson-Gleichung 3.5 gelöst werden. Durch Diskretisierung mit Hilfe von Gleichung 2.7 kann Gleichung 3.5 dargestellt werden, und es entsteht ein dünn besetztes, lineares Gleichungssystem der Form $A\vec{x} = \vec{b}$. $\vec{x}$ ist der Vektor mit den gesuchten Variablen (hier p oder $\vec{u}$), $\vec{b}$ ein Vektor aus Konstanten und A eine Matrix (hier ∇^2). Dieses Gleichungssystem

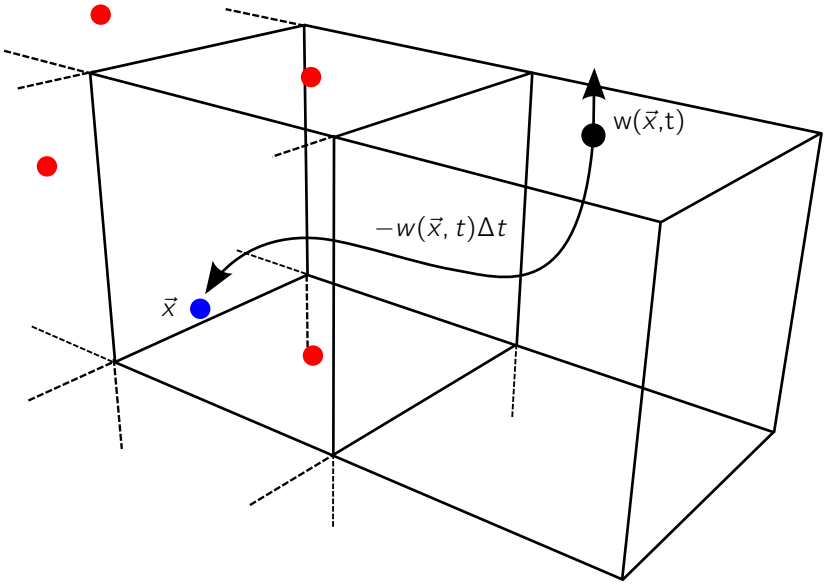

Abbildung 3.3: *Berechung des Advektionsterms: Ein impliziter Advektionsschritt durch Rückwärtsverfolgung der Geschwindigkeitskomponente w endet nicht in einem Voxelzellenzentrum. Durch trilineare Interpolation der vier nächsten Nachbarn (rot) wird der Wert bei $\vec{x}$ (blau) berechnet und an der Voxelzellenseite des Startvoxels (schwarz) gespeichert.*

kann mittels einfachen Relaxationsmethoden z. B. der Jacobi-Iteration berechnet werden. Besser ist es jedoch die Konjugierte-Gradienten-Methode zu verwenden, da diese auch leicht zu implementieren ist und besser konvergiert als o. g. Methoden. Um die Konvergenz weiterhin zu verbessern, haben die Autoren von [FJ01] einen Cholski-Preconditioner eingesetzt. Um ausreichend gute Ergebnisse zu erreichen, müssen pro Voxelzelle mehrere Iterationen durchgeführt werden.

3.4 Algorithmus

Für den Algorithmus werden zwei Voxelgitter benötigt, die die Geschwindigkeit und den Druck des Transportmediums und die Temperatur und Dichte des Rauchs nach der Defintion des gestaffelten Voxelgitters speichern. Das erste Voxelgitter dient als Quellgitter, das zweite als Zielgitter und nach einem Zeitschritt werden beide Gitter vertauscht. Das heißt, die im vorhergehenden Zeitschritt berechneten Werte im Zielgitter werden im nächsten Schritt als Quellwerte benutzt. Zum initialen Zeitpunkt sind diese Gitter bezüglich Geschwindigkeit und Druck leer: $\vec{u}_0 = \vec{u}(\vec{x}, 0)$ und $p = p(\vec{x}, 0)$. Dann wird zunächst das **Update des Geschwindigkeitsfeldes** durchgeführt. Dabei wird für jeden Zeitschritt Δt Gleichung 3.8 in drei Schritten ausgeführt. Nach jedem Schritt entsteht ein neues tempo-

räres Vektorfeld, was als Input für den nächsten Schritt dient: $\vec{w}_1$ nach der Kraftausübung auf $\vec{w}_0$ (das Vektorfeld zum Zeitpunkt t), $\vec{w}_2$ nach der Advektion und $\vec{w}_3$ nach der Projektion. Das Ergebnis nach der Zeit $t + \Delta t$ ist dann durch das Vektorfeld $\vec{w}_3$ gegeben. Die einzelnen Schritte zum Update des Geschwindigkeitfeldes werden im folgenden vorgestellt.

1. Krafterm. Es werden alle externen Kräfte in Gleichung 3.10 mit einem expliziten Eulerschritt dem Geschwindigkeitsfeld zugeführt:

$$\vec{w}_1(\vec{x}) = \vec{w}_0(\vec{x}) + \Delta t \cdot \vec{F}(\vec{x}, t). \tag{3.11}$$

   Wenn davon ausgegangen wird, dass die Kraft F gleichbedeutend mit einer Beschleunigung ist, ensteht durch Multiplikation mit einem Zeitschritt wieder ein Geschwindigkeitsfeld. Dabei sollte sich die Kraft während des Zeitschritt nicht ändern.

2. Advektionsterm. Es wird ein impliziter Advektionsschritt über die Zeit Δt wie in Gleichung 3.9 durchgeführt:

$$\vec{w}_2(\vec{x}) = \vec{w}_1(\vec{x} - \vec{u}(\vec{x}, t)\Delta t). \tag{3.12}$$

3. Projektion. Es wird der Projektionsoperator aus Gleichung 3.8 auf das Vektorfeld $\vec{w}_2$ ausgeführt, um jenes divergenzfrei zu bekommen:

$$\nabla^2 p = \nabla \cdot \vec{w}_2 \tag{3.13}$$

$$\vec{w}_3 = \vec{w}_2 - \nabla p. \tag{3.14}$$

Nach dem Update des Geschwindigkeitsfeldes folgen die **Updates des Temperatur- und Dichtefeldes**. Da dabei nur eine einfache Advektion stattfindet, wird wieder das Semi-Langrange'sche Verfahren verwendet. Diesmal aber mit den Werten im Voxelzellenzentrum. Es folgt das Vertauschen des Quellgitters mit dem Zielgitter, und der nächste Zeitschritt kann berechnet werden.

Kapitel 4: Rendering

Bis jetzt wurde nur das Fundament der Rauchsimulation besprochen, die Bewegung des Rauchs. Um nun aber auch die Auswirkungen dieser Bewegung zu sehen, muss der Rauch dargestellt werden. Die Rauchoberfläche sollte nicht durch geometrische Flächen dargestellt werden, da dies nicht der Realität entsprechen würde. Vielmehr sollte der Rauch als eine Partikelwolke gesehen werden. Das Aussehen von Rauch wird vor allem durch die Interaktion von Licht mit den Partikeln im Rauch verursacht, z. B. Emission, Absorption und Streuung. Fedkiw et al. [FJ01] stellen dazu zwei Renderer vor: einen schnellen Hardware-Renderer und eine visuell hoch-qualitativen Photon-Map-Renderer. Dabei ist der Hardware-basierte Renderer mehr als schneller Vorschau-Renderer zu verstehen, der rasches Feedback gibt. Um jedoch physikalisch-basierte Beleuchtung zu erreichen, ist der teurere und langsamere Photon-Map-Renderer vorzuziehen.

4.1 Vorbetrachtungen

Es soll für einen Punkt $\vec{x}$ die Strahlung berechnet werden, die diesen in Richtung $\vec{\omega}$ verlässt. Das Ziel ist also eine Gleichung, die sowohl die Strahlung berücksichtigt, die zu einem Strahl hinzukommt, als auch diejenige, die verloren geht.

Die Änderung der Strahlstärke L in Richtung $\vec{\omega}$ aufgrund von ausgehender Streuung ist

$$(\vec{\omega} \cdot \Delta)L(\vec{x}, \vec{\omega}) = -\sigma_s(\vec{x})L(\vec{x}, \vec{\omega}) \tag{4.1}$$

und aufgrund von Absorption ist:

$$(\vec{\omega} \cdot \Delta)L(\vec{x}, \vec{\omega}) = -\sigma_a(\vec{x})L(\vec{x}, \vec{\omega}). \tag{4.2}$$

σ_s ist dabei der Streuungskoeffizient und σ_a der Absorbtionskoeffizient. Durch beide o. g. Terme wird die Strahlstärke abgeschwächt und kann in einem Term zusammengefasst werden:

$$(\vec{\omega} \cdot \Delta)L(\vec{x}, \vec{\omega}) = -\sigma_t(\vec{x})L(\vec{x}, \vec{\omega}), \tag{4.3}$$

wobei $\sigma_t = \sigma_s + \sigma_a$ das Abschwächungsmaß ist. Dieses Maß ist direkt proportional zur Dichte des Rauchs und dem Abschwächungsquerschnitt C_{ext}:

$$\sigma_t = C_{ext}\rho. \tag{4.4}$$

Neben der Abschwächung gibt es noch einen Zuwachs an Energie durch einfallende Streuung:

$$(\vec{\omega} \cdot \Delta)L(\vec{x}, \vec{\omega}) = \sigma_s(\vec{x}) \int_{4\pi\Omega} p(\vec{x}, \vec{\omega}', \vec{\omega}) L_i(\vec{x}, \vec{\omega}') d\vec{\omega}', \tag{4.5}$$

wobei die einfallende Strahlstärke L_i über alle Richtungen auf der umgebenden Kugel $4\pi\Omega$ intergriert wird. Zusätzlich kann ein Partikel auch Licht emittieren, was auch einen Zuwachs an Energie bedeutet:

$$(\vec{\omega} \cdot \Delta)L(\vec{x}, \vec{\omega}) = \sigma_a(\vec{x}) L_e(\vec{x}, \vec{\omega}). \tag{4.6}$$

Kombiniert man alle o. g. Gleichungen, erhält man die gesamte Änderung der Strahlstärke:

$$\begin{aligned} (\vec{\omega} \cdot \Delta)L(\vec{x}, \vec{\omega}) &= \sigma_a(\vec{x}) L_e(\vec{x}, \vec{\omega}) - \sigma_t(\vec{x}) L(\vec{x}, \vec{\omega}) + \\ &\quad \sigma_s(\vec{x}) \int_{4\pi\Omega} p(\vec{x}, \vec{\omega}', \vec{\omega}) L_i(\vec{x}, \vec{\omega}') d\vec{\omega}' \end{aligned} \tag{4.7}$$

Nun wird mit dem Abschwächungsterm $e^{-\tau(\vec{x}, \vec{x}')}$ multipliziert. Dieser Term schwächt das Licht mit zunehmender Entfernung zur Lichtquelle ab. Durch anschließende Integration der Gleichung 4.7 mit dem Abschwächungsterm auf beiden Seiten nach der Segmentlänge s erhält man:

$$\begin{aligned} L(\vec{x}, \vec{\omega}) &= \int_0^s e^{-\tau(\vec{x}, \vec{x}')} \sigma_a(\vec{x}') L_e(\vec{x}') d\vec{x}' + \\ &\quad \int_0^s e^{-\tau(\vec{x}, \vec{x}')} \sigma_s(\vec{x}') \int_{4\pi\Omega} p(\vec{x}', \vec{\omega}', \vec{\omega}) L_i(\vec{x}', \vec{\omega}') d\vec{\omega}' d\vec{x}' + \\ &\quad e^{-\tau(\vec{x}, \vec{x} + s\vec{\omega})} L(\vec{x} + s\vec{\omega}, \vec{\omega}). \end{aligned} \tag{4.8}$$

Diese Gleichung ist allgemeinhin als Volumen-Rendering-Gleichung bekannt. Der erste Term beschreibt die durch Emission entlang des Strahls im Volumen dazukommende Leuchtdichte. Der zweite Term beschreibt dagegen die durch Einstreuung im Volumen dazukommende Leuchtdichte. Dazu wird wieder entlang des Strahls integriert und mit der Abschwächung skaliert. Da ein Photon beim Zusammentreffen mit einem Partikel in verschiedene Richtungen streut und mit weiteren Partikeln kollidiert (Mehrfachstreuung), wird im zweitem Integral auch über die umgebende Kugel $4\pi\Omega$ integriert. Die Phasenfunktion p wichtet dabei die Leuchtdichte so, dass Photonen, die mehr in Richtung des Auges gestreut werden, mehr Gewicht haben als andere. Der dritte Term des Volumen-Rendering-Integrals beschreibt den Anteil des Lichts von der Lichtquelle, der das Auge direkt erreicht - auch Einfachstreuung genannt (siehe Abbildung 4.1). Die optische Tiefe $\tau(\vec{x}, \vec{x}')$ ist hier gegeben durch

$$\tau(\vec{x}, \vec{x}') = \int_{\vec{x}}^{\vec{x}'} \sigma_t(t) dt. \tag{4.9}$$

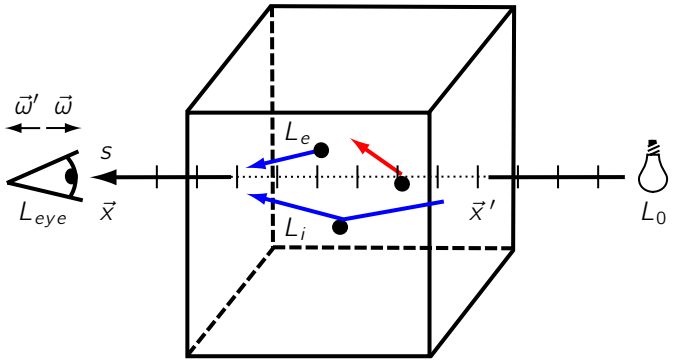

Abbildung 4.1: *Volumen-Rendering: Bei der Volumen-Rendering-Gleichung wird die Gesamtstrahlung berechnet, die durch ein Volumen beim Betrachter ankommt. Beim Durchwandern von Partikelwolken wird diese Strahlung beinflusst von Strahlung, die von Partikeln emmitiert, eingestreut und abgeschwächt wird.*

Die Phasenfunktion $p(\theta)$ beschreibt die Verteilung von gestreutem Licht in Partikelwolken und gibt an, wie wahrscheinlich ein Photon durch den Winkel θ abgelenkt wird. Die am meisten verwendete Phasenfunktion ist die empirische *Henyey-Greenstein-Phasenfunktion*:

$$p(\theta) = \frac{1 - g^2}{\sqrt{4\pi(1 + g^2 - 2g\cos\theta)^3}}. \tag{4.10}$$

g ist ein asymmetrischer Parameter im Bereich $[-1 \ldots 1]$. Ist $g > 0$, entsteht Vorwärtsstreuung, d. h. der Streuwinkel ist $0° \leq \theta < 90°$. Ist $g < 0$, ensteht Rückwärtsstreuung mit $90° \leq \theta \leq 180°$. $g = 0$ bedeutet dagegen von der Richtung unabhängige Streuung.

4.2 Photon-Map Renderer

Um Rauch möglichst realistisch zu rendern, muss die Interaktion des Lichts mit dem Rauch simuliert werden. Dazu ist es hilfreich den Rauch als Partikelwolke zu verstehen. Dabei handelt es sich um ein Volumen mit Partikeln, die den Lichttransport in Inneren des Volumens beeinflussen. Die Volumen-Rendering-Gleichung hilft dabei den Lichttransport im Rauch zu beschreiben und berücksichtigt alle möglichen Lichtinteratkionsmöglichkeiten. Für die Lösung dieser Gleichung wird das Photon-Mapping-Verfahren für Partikelwolken aus [JC98] verwendet.

4.2.1 Photon Mapping

Dieses Verfahren setzt sich aus zwei Schritten zusammen. Der erste besteht darin, Photonen (Lichtpakete) von den Lichtquellen aus in die Szene zu schießen und zu verfolgen. Der Schritt wird deshalb auch Photon Tracing genannt (siehe Abbildung 4.2). Die Photonen tragen eine bestimmte Energie, die wiederum von der Stärke der Lichtquelle abhängig ist. Treffen sie auf irgendein Objekt außer einer Partikelwolke, werden sie anhand der Materialeigenschaften des Objekts entweder absorbiert, gebrochen oder reflektiert. Jenes Ereignis des Treffens wird nun nicht an der Geometrie gebunden gespeichert, sondern in einer separaten Datenstruktur, der sogenannten *Photon Map*. Treffen die Photonen auf eine Partikelwolke wie Rauch kommt Mehrfachstreuung des Lichts vor, oder das Licht gelangt direkt zum Beobachter (Einfachstreuung). Im ersteren Fall, der Mehrfachstreuung, wird die sogenannte *Volume Photon Map* genutzt, um jene Photonen bei Kontakt zu speichern.

Im zweiten Schritt wird die Szene mit Hilfe der Informationen aus den beiden Photon Maps gerendert (siehe Abbildung 4.2). Dabei wird die globale Beleuchtung der Partikelwolke aufgrund der Mehrfachstreuung durch Integration entlang der Strahlen mittels Ray Marching berechnet. Für alle anderen Objekte wird die Beleuchtung mit Hilfe des einfachen Raytracing-Verfahrens berechnet, da hier keine Streuung sondern nur Reflektionen oder Brechungen vorkommen. Die Beleuchtung an einem Punkt wird in beiden Fällen mit Hilfe der Volume Photon Map bzw. der Photon Map geschätzt. Die direkte Beleuchtung eines Pixels wird ebenfalls durch Raytracing und ohne Photon Map berechnet, da hier ja kein Photonenkontakt mit der Umgebung stattgefunden hat.

4.2.2 Photon Tracing

Wird das Photon Mapping auf eine Partikelwolke wie Rauch angewendet, muss noch ermittelt werden, wann ein Photon mit dieser interagiert. Das Photon wird nicht sofort abgelenkt, wenn es auf das Volumen trifft. Sondern es wandert eine Strecke d bis es letztlich mit dem Volumen interagiert (Absorption oder Streuung). Bei einem homogenen Volumen ist

$$d = \frac{1}{\sigma_t}. \tag{4.11}$$

An diesem neuen Punkt muss nun entschieden werden, ob das Photon absorbiert oder gestreut wird. Dies wird durch das Rückstrahlvermögen

$$\Omega = \frac{\sigma_s}{\sigma_t}. \tag{4.12}$$

bestimmt. Anhand des Wertes Ω kann ein neues Photon generiert werden, dessen Energie mit Ω skaliert wird. Der Nachteil daran ist, dass dabei viele Photonen mit einer geringen Energie entstehen würden. Außerdem würde der Rechenaufwand exponentiell ansteigen. Deswegen wird das „Russische Roulette“ genutzt. Hierbei wird eine Zufallszahl $\xi \in [0, 1]$

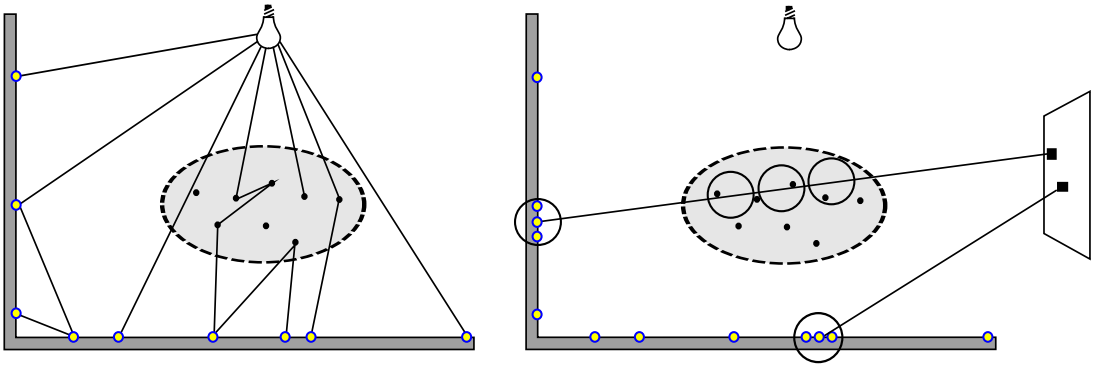

Abbildung 4.2: *Photon Tracing in eine Szene mit Rauch (links): Die Photonen, die in die Partikelwolke eintreten, können entweder gestreut, absorbiert (aktiv) oder hindurchgelassen (passiv) werden. Die aktiven Photonen werden am Interaktionspunkt in einer separaten Volume Photon Map gespeichert. Alle Interaktionen mit anderen Objekten werden in der normalen Photon Map verzeichnet. Rendering der Szene mittels Ray Marching/Tracing (rechts): die Beleuchtung innerhalb des Rauchs wird mittels Ray-Marching-Algorithmus und der Volume Photon Map berechnet. Die Beleuchtung außerhalb des Rauchs wird mittels Raytracing und der Photon Map berechnet.*

mit Ω verglichen. Falls $\xi \leq \Omega$ ist wird das Photon gestreut und wandert mit der gleichen Energie weiter, und ansonsten wird es absorbiert.

4.2.3 Volume Photon Map

Egal ob das Photon gestreut oder absorbiert wurde nachdem es in die Partikelwolke eingedrungen ist, das Ereignis wird in die Volume Photon Map gespeichert. Der Grund für die Lichtstreuung sind die rauhen Partikeloberflächen und Inhomogenitäten im Rauch. In den allermeisten Fällen kommt es sogar zu einer Mehrfachstreuung des Lichts. Das heißt, es wird an mehreren Partikeln hintereinander in verschiedene Richtungen gestreut. In der Volume Photon Map werden also nur die Photonen abgespeichert, die zur indirekten Beleuchtung der Partikelwolke beitragen, d. h. die zuvor mindestens einmal gestreut wurden. Der direkte Anteil, also die Einfachstreuung, wird wieder durch das Raytracing berechnet.

Nachdem alle Photonen von den Lichtquellen verschossen worden sind, ermittelt sich die Energie eines Photons aus dem Quotienten der Lichtquellenstärke durch die Anzahl aller verschossen Photonen dieser Lichtquelle.

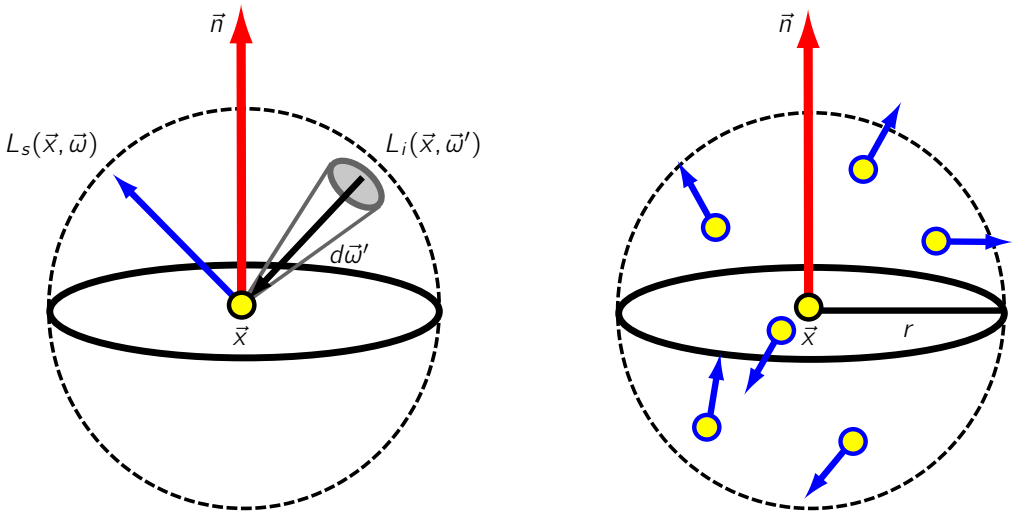

Abbildung 4.3: *Strahldichteschätzung: Bei der Berechnung des Einstreuungsterms wird über die durch $\vec{x}$ umgebende Kugel integriert. Dadurch erhält man die einfallende Strahlung pro Raumwinkel von $\vec{x}$ aus gesehen (links). Durch Bestimmung der nächsten Photonen in der Volume Photon Map, wird die lokale Photondichte gemessen. Dazu wird eine Kugel um den Punkt $\vec{x}$ ausgeweitet bis diese genug Photonen enthält(rechts).*

4.2.4 Strahldichteschätzung

Um die Volumen-Rendering-Gleichung 4.8 zu lösen, muss zuerst die Strahldichte L_s bestimmt werden, die an einem Punkt $\vec{x}$ in Richtung $\vec{\omega}$ gestreut wird. Dabei hat Licht, das direkt auf diesen Punkt trifft (Einfachstreuung), und Licht, das von mehreren Partikeln gestreut wurde, bevor es auf diesen Punkt trifft (Mehrfachstreuung), einen Einfluss auf die Berechnung. In der Volume Photon Map sind aber nur Photonen enthalten, die mindestens zweimal gestreut wurden. Das heißt, dass nur der Anteil der Mehrfachstreuung im Einstreuungsterm

$$(\vec{\omega}\cdot\nabla)L_s(\vec{x},\vec{\omega}) = \sigma_s(\vec{x})\int_{4\pi\Omega} p(\vec{x},\vec{\omega}',\vec{\omega})L_i(\vec{x},\vec{\omega}')d\vec{\omega}' \tag{4.13}$$

geschätzt werden muss (siehe Abbildung 4.3). Der Term $L_i(\vec{x},\vec{\omega}')$ beschreibt die Strahldichte, die aus einer Richtung $\vec{\omega}'$ aus dem kompletten umgebenden Raum $4\pi\Omega$ auf den Punkt $\vec{x}$ trifft. $p(\vec{x},\vec{\omega}',\vec{\omega})$ ist die Phasenfunktion, beschreibt die Verteilung des gestreuten Lichts und ist vergleichbar mit der BRDF für Oberflächen. Nimmt man nun den Zusammenhang zwischen Strahlungsfluss Φ und Strahlungsdichte L

$$L(\vec{x},\vec{\omega}) = \frac{d^2\Phi(\vec{x},\vec{\omega})}{\sigma_s(\vec{x})d\vec{\omega}dV} \tag{4.14}$$

und setzt diesen in Gleichung 4.13 ein, erhält man

$$(\vec{\omega} \cdot \nabla) L_s(\vec{x}, \vec{\omega}) = \int_{4\pi\Omega} p(\vec{x}, \vec{\omega}', \vec{\omega}) \frac{d^2\Phi(\vec{x}, \vec{\omega}')}{dV} \tag{4.15}$$

Die Volume Photon Map wird nun für die Schätzung des lokalen Strahlungsflusses eingesetzt. Dazu werden die n nächsten Nachbarn in einer Kugel mit dem Radius r um den Punkt $\vec{x}$ bestimmt (siehe Abbildung 4.3). Dessen Strahlungsflüsse werden aufsummiert und durch das Volumen der Kugel geteilt. So wird das Integral über den kompletten Raum geschätzt:

$$\begin{aligned} (\vec{\omega} \cdot \nabla) L_s(\vec{x}, \vec{\omega}) &= \int_{4\pi\Omega} p(\vec{x}, \vec{\omega}', \vec{\omega}) \frac{d^2\Phi(\vec{x}, \vec{\omega}')}{dV} \\ &\approx \sum_{k=1}^{n} p(\vec{x}, \vec{\omega}'_k, \vec{\omega}) \frac{\Phi_k(\vec{x}, \vec{\omega}'_k)}{\frac{4}{3}\pi r^3} \end{aligned} \tag{4.16}$$

Somit erhält man die durch Mehrfachstreuung entstandene Strahlung die beim Beobachter ankommt.

4.2.5 Forward Ray Marching

Diese Verfahren setzt voraus, dass wie beim Raycasting Strahlen vom Auge aus in die Szene geschossen und die Strahldichte entlang der Strahlen eingesammelt wird (siehe Abbildung 4.2). Dazu muss die Volumen-Rendering-Gleichung 4.8 für jeden Strahl gelöst werden. Wie auch bei den Navier-Stokes-Gleichungen ist es auch hier nicht möglich diese analytisch zu lösen. Ein Ausweg bietet wieder ein numerisches Verfahren, das Ray Marching. Dazu wird ein Strahl in gleichgroße Segmente n der Länge Δx_n aufgeteilt. Das einfallende Licht und die Eigenschaften des Mediums innerhalb eines solchen Segments sind konstant. Für jedes Segment wird ein Repräsentant bestimmt, der für alle Elemente im selben Segment die Eigenschaften bestimmt. Der hier vorgestellte Ray-Marching-Algorithmus bewegt sich vom Strahlursprung vorwärts zum Strahlende. Das ermöglicht einige Vereinfachungen, da verdeckter Rauch nicht unnötigerweise in die Berechnung der Strahldichte einfließt. Außerdem werden weniger Photonen benötigt, je tiefer man in den Rauch eintaucht.

Zur Bestimmung der Strahldichte innerhalb eines Segmentes n am Punkt $\vec{x}_n$ in Richtung $\vec{\omega}$ hat der Ray Marcher die Form:

$$L_n(\vec{x}_n, \vec{\omega}) = L_{n-1}(\vec{x}_{n-1}, \vec{\omega}) + e^{-\tau(\vec{x}_n)} \Delta\vec{x}_n (\vec{\omega} \cdot \nabla) L_s(\vec{x}'_n, \vec{\omega}), \tag{4.17}$$

wobei $\tau(\vec{x}_n)$ die optische Tiefe in Gleichung 4.9, L_s der in Richtung des Auges gestreute Anteil der durch Streuung dazukommenden Strahlstärke und $\vec{x}'_n$ ein zufällig gewählter

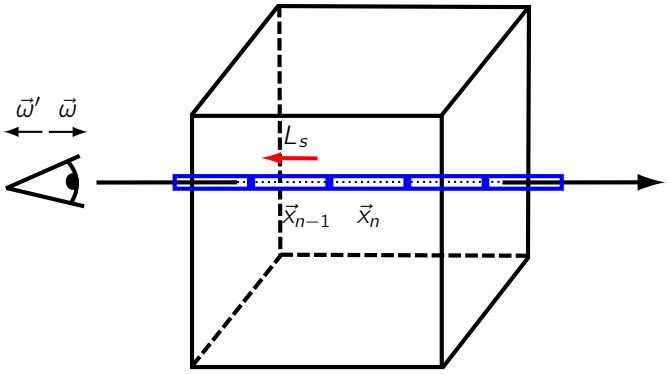

Abbildung 4.4: *Ray Marching: Der Ray-Marching-Algorithmus berechnet die Volumen-Rendering-Gleichung entlang eines Strahls durch Diskretisierung. Dabei wird der Strahl in kleinere Segmente aufgeteilt. Für jedes Segment ist das Medium homogen.*

Punkte im $n$ten Segment sind. Diese Gleichung wird rekursiv vom Strahlursprung zum Strahlende berechnet. Dabei wird im ersten Term auf die zuvor berechnete Strahlstärke des nächsten Segmentes $n-1$ zugegriffen. Dazu sollte die Leuchtdichte des 0ten Segmentes L_0 mit 0 als neutrales Element initialisiert werden. Im zweiten Term wird die Strahldichte ausgerechnet, die durch Streuung in die Richtung $\vec{\omega}$ dazukommt (siehe Gleichung 4.16). Diese Strahldichte, aufgrund der Diskretisierung skaliert mit der Segmentlänge Δx_n, wird noch durch den Absorptionsterm abgeschwächt (sieher Abbildung 4.4). Ist die Abschwächung wegen hoher Dichte bzw. großer Entfernung zum Auge sehr groß, ist der Einfluss auf die beim Auge ankommende Leuchtdichte eher gering und kann vernachlässigt werden. Dadurch kann man den Algorithmus eher beenden und spart so Zeit und Ressourcen.

Beim Ray Marching wird die Emission aus dem Volumen-Rendering-Integral vernachlässigt. Der direkte Anteil des Lichts von der Lichtquelle kommt nur beim Auge an, wenn die Dichte des Rauchs gering ist und somit die Berechnung nicht vorher abgebrochen wurde.

4.2.6 Raytracing

Um die Beleuchtung außerhalb des Rauchs zu berechnen, wird ein einfacher Raytracing-Algorithmus verwendet. Das heißt, es wird ein Strahl vom Auge durch die Bildschirmebene in die Szene geschossen. Trifft dieser auf ein Objekt, z. B. einer Begrenzung, wird am Schnittpunkt in die Photon Map geschaut, und es werden in einem bestimmten Radius um diesen Schnittpunkt die Energie aller Photonen aufsummiert. Es findet also keine Aufsummierung entlang des Strahls wie beim Ray Marching statt. Durch das Raytracing wird die indirekte Beleuchtung durch Reflektion und Brechung des Lichts bei der Berechnung der Beleuchtung der Objekte in der Szene mitbeachtet. Bei direkter Beleuchtung, d. h. wenn

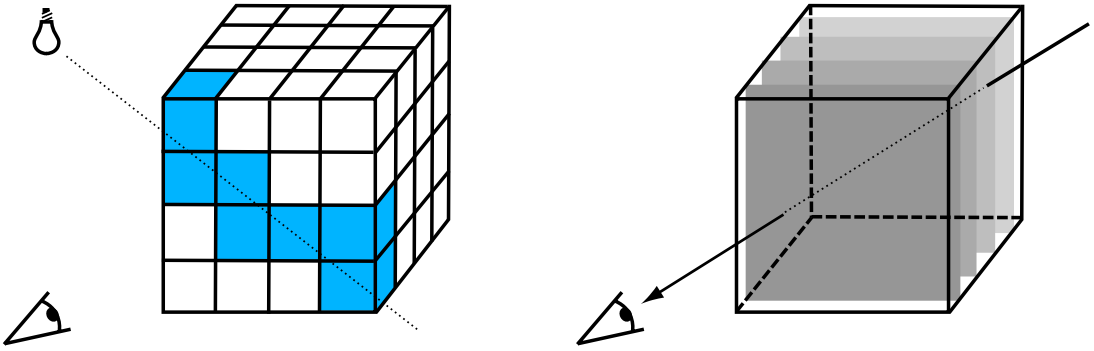

Abbildung 4.5: *Beim Hardware-basierten Renderer wird im ersten Schritt ein Voxel-Traversal von den Strahlen der Lichtquelle auf das Volumen angewandt und damit die Lichtmenge berechnet, die bei jedem Voxel ankommt (links). Im zweiten Schritt werden die transparenten Slices von hinten nach vorne mittels Blending gerendert (rechts).*

die Lichtquelle direkt vom Auge aus gesehen werden kann, nutzt man nur das Raytracing ohne Photon Map. Die Leuchtdichte am Pixel ist dann die Leuchtdichte der Lichtquelle.

4.3 Hardware-basierter Renderer

Der Renderer arbeitet in zwei Schritten. Im ersten Schritt wird für jedes Voxel die Lichtmenge gemessen, die bei diesem auf direktem Wege ankommt. Das Licht breitet sich entlang von Strahlen aus, welche das betrachtete Volumen durchqueren. Um die Voxel entlang eines solchen Strahls zu erkennen, wird ein Voxel-Traversal-Algorithmus nach [AW87] verwendet. Dabei werden von der Lichtquelle aus Strahlen in die Szene geschossen. Die Strahlen fungieren als eine Art Alpha-Puffer. Sie speichern einen Transparenzwert. Dieser Transparenzwert wird initial auf $T_{ray} = 1$ gesetzt. Jedes mal, wenn ein Voxel getroffen wird, wird der Transparenzwert des Strahls mit Hilfe der Voxeltransparenz

$$T_{vox} = e^{-C_{ext}h} \tag{4.18}$$

auf

$$T_{ray} = T_{ray}T_{vox} \tag{4.19}$$

angepasst. Der Strahl akkumuliert also alle Transparenzwerte der besuchten Zellen auf. h gibt dabei den Gitterknotenabstand an. Die Lichtmenge, die beim Voxel ankommt wird dann mittels der angepassten Strahltransparenz mit

$$L_{vox} = \Omega\, L_{light}(1 - T_{vox})T_{ray} \tag{4.20}$$

berechnet. Das heißt, die Lichtmenge, die beim Voxel ankommt, wird durch die Opazität des Voxels selbst (d. h. der Lichtanteil, die vom Voxel absorbiert wird) und durch

den Transparenzwert des Strahls (d. h. der Lichtanteil, der ungehindert durch die vorhergehenden Voxel gekommen ist) beeinflusst. Wenn der Strahl durch das Medium wandert, wird durch Gleichung 4.19 die Transparenz reduziert. Durch diesen Effekt wird die Selbstbeschattung des Rauchs simuliert.

Im zweiten Schritt wird wie in der direkten Volumen-Visualisierung das Voxelgitter von hinten nach vorn gerendert. Das Volumen wird dabei an den Voxelzentren in einzelne Scheiben, den sogenannten Slices, geschnitten. Das passiert entlang der Koordinatenachse, die mit der Beobachtungsrichtung am ähnlichsten ausgerichtet ist. Es ensteht dann eine Menge von 2D Slices, die durch transparente Quads gerendert werden. Die Farbe eines Pixels c_i entspricht der Ausstrahlung L_{vox} und die Transparenz $\alpha_i = T_{vox}$ des entsprechenden Voxels. Um die hinereinander liegenden, transparenten Quads korrekt miteinander zu mischen, wird das Blending der Grafikhardware genutzt:

$$c \leftarrow c_i + \alpha_i \cdot c,$$

mit c als Pixelfarbe im Farbpuffer. Hier braucht man keinen Alpha-Puffer wie beim ersten Schritt, da Back-to-Front die natürliche Richtung des Lichts ist, das beim Auge ankommt, und so keine Transparenzwerte mitgeführt werden müssen.

Kapitel 5: Ergebnisse

Folgende Seiten zeigen einige Beispiele für die Simulation von Rauch. Alle Bilder sind von [FJ01] entnommen und wurden nicht in Echtzeit gerendert. Für den aufsteigenden Rauch in Abbildung 5.1 wurden z. B. 30 Sekunden pro Frame gebraucht. Für den Rauch mit der Kugel wurden dagegen 75 Sekunden für ein Frame gerendert. Für die Animation in Abbildung 5.3 wurde nur eine Sekunde pro Frame gebraucht, da hier der Hardware-Renderer verwendet worden ist.

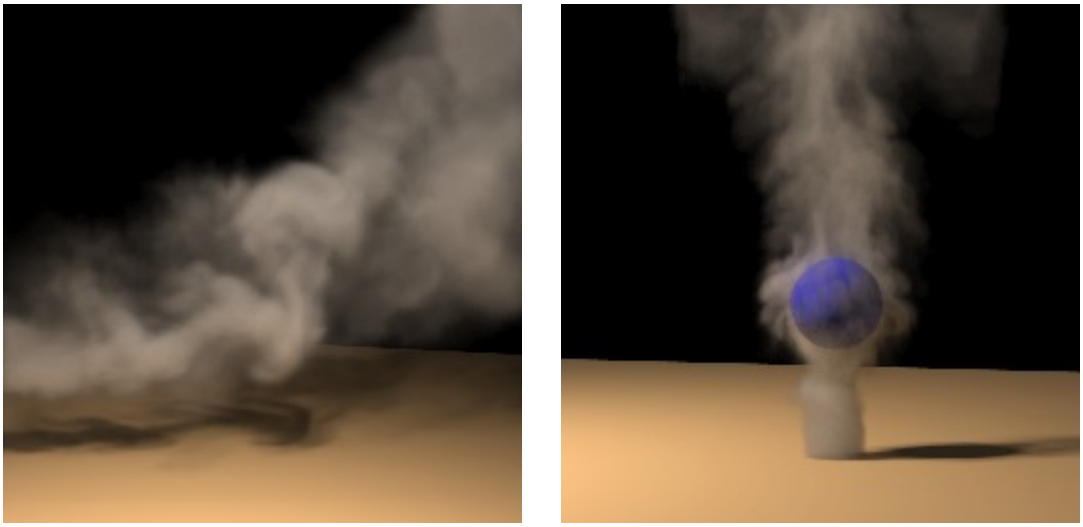

Abbildung 5.1: *Links ist ein Bild von einer Simulation von turbulentem, aufsteigendem Rauch mit Mehrfachstreuung des Lichts zu sehen. Rechts sieht man Rauch, der sich um einer Kugel herum ausbreitet. Bei beiden Simulationen sieht man den Erhalt der feinen Wirbel durch das Vorticity Confinement sehr gut.*

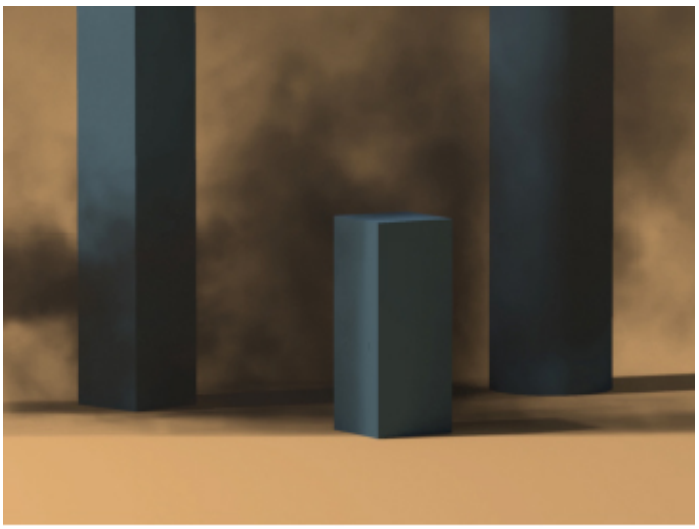

Abbildung 5.2: *Dunkler Rauch mit niedrigem Albedo bewegt sich durch eine Szene aus mehreren Objekten. Jedes dieser Objekte interagiert mit dem Rauch und verursacht daher Turbulenzen und Verwirbelungen.*

Abbildung 5.3: *Ein Folge von einzelnen Zeitpunkten bei einer Rauchsimulation. Hier ist die Veränderung der Rauchsäule über die Zeit zu beachten. Diese Simulation wurde mit dem interaktiven Hardware-basierten Renderer berechnet.*

Kapitel 6: Zusammenfassung

Im Rahmen dieser Arbeit wurde ein Modell zur Simulation von Gasen wie Rauch vorgestellt. Die Dynamik des Rauchs wurde mit Hilfe der Euler-Gleichungen beschrieben. Gelöst wurden diese, indem die Euler-Gleichungen in berechenbare Einzelteile zerlegt und nacheinander berechnet wurden. Der Advektionsterm wird durch eine Semi-Lagrange'sche Integration berechnet. Durch Einführung des sogenannten Vorticity Confinement wurden numerische Dämpfungen, ausgelöst durch den Semi-Lagrange'schen Ansatz, ausgeglichen, indem die verloren gegangene Energie wieder an den Stellen des Verlusts hinzugefügt wurden. Dabei bleibt das Modell stabil, solange die zugefügte Kraft unter einer bestimmten Grenze bleibt. Es ist, obwohl noch der Mehraufwand durch die Berechnung der zusätzlichen Kraft hinzukommt, ähnlich schnell wie das Modell von [Sta99]. Es ist auch wie bei [FM96] möglich, physikalische Grenzen und Hindernisse in das Modell einzufügen. Der Rauch wirbelt dann korrekt um diese herum bzw. vorbei.

Zum Rendern wurden zwei unterschiedliche Renderer vorgestellt. Zum einen war das der flexible und schnelle Hardware-basierte Renderer. Er stellt den Rauch in einem zweistufigen Verfahren ähnlich der direkten Volumenvisualisierung dar. Der zweite, physikalisch korrektere und visuell qualitativ bessere Renderer ist der Photon-Map Renderer. Er nutzt das Photon-Mapping-Verfahren für Partikelwolken aus [JC98].

Die Interaktion mit anderen Objekten und der Umgebung macht diese Verfahren zur Rauchsimulation ideal für computergrafische Belange, wo visuelle Qualität und Schnelligkeit im Vordergrund stehen.

Appendix A: Literaturverzeichnis

[AW87] J. Amanatides and A. Woo. A fast voxel traversal algorithm for ray tracing. *Eurographics Conference Proceedings 87*, pages 3–10, 1987.

[Fer04] R. Fernando. *GPU Gems: Programming Techniques, Tips, and Tricks for Real-Time Graphics*. Addison-Wesley, Pearson Education, Inc., 2004.

[FJ01] J. Fedkiw, R. Stam and H. W. Jensen. Visual simulation of smoke. *Proceedings of SIGGRAPH 2001*, 2001.

[FM96] N. Foster and D. Metaxas. Realistic animation of liquids. *Graphical Models and Image Processing, 58(5)*, pages 471–483, 1996.

[FM97] M. Foster and D. Metaxas. Modeling the motion of a hot, turbulent gas. *SIGGRAPH 97 Conference Proceedings, Anual Conference Series*, pages 181–188, 1997.

[JC98] H. W. Jensen and P. H. Christensen. Efficient simulation of light transport in scenes with participating media using photon maps. *SIGGRAPH 98 Conference Proceedings, Anual Conference Series*, pages 311–320, 1998.

[Jen01] H. W. Jensen. *Realistic Image Synthesis Using Photon Mapping*. A K Peters Natrick Massachusetts, 2001.

[Joh03] M. John. *Focus on Photon Mapping*. Premier Press, 2003.

[SRF05] A. Selle, N. Rasmussen, and R. Fedkiw. A vortex particle method for smoke, water and explosions. *SIGGRAPH 05 Conference Proceedings, Anual Conference Series*, pages 910–914, 2005.

[Sta99] J. Stam. Stable fluids. *Proceedings of SIGGRAPH 1999*, pages 121–128, 1999.