Modeling the Motion of a Hot, Turbulent Gas

Pascal Müller

Abhandlung über das gleichnamige Paper von
Nick Foster und Dimitris Metaxas

14. Juni 1998

Abstract

Das Paper [4] beschreibt eine neue Animationstechnik für die Bewegung von einem warmen Gas, das mit der Umgebung interagiert. Die physikalisch basierte Methode eignet sich vor allem für Szenen mit aufsteigendem Dampf, rollendem Rauch (z.B. nach einer Explosionen) und stürmischem Wind; oder auch für die komplexen verwirbelten Gasbewegungen die entstehen bei Hitzeübertragungen (z.B. einer heissen Teetasse) oder beim Ausströmen eines Gases aus einer Düse (z.B. ein Kamin). Die Methode kombiniert spezielle Formen von Gasbewegungsgleichungen mit dem Lösen von Volumen-Differential-Gleichungen bei grober Auflösung. Im Paper wird viel Wert auf Effizienz, Genauigkeit, Stabilität und Einfachheit des vorgestellten Algorithmus' gelegt.

1 Einleitung

Die realistische Darstellng von Wolken, Nebel, Dampf und Rauch ist in der Computergraphik eines der attraktivsten, aber leider auch schwierigsten Probleme. Das Interesse an der Visualisierung dieser Phänomene ist seit Anfang der 80er-Jahre vorhanden. Das so schwierig zu realisierende Verhalten von Gas entsteht, wenn Gase mit unterschiedlicher Temperatur und Geschwindigkeit sich mixen oder solide Objeke umfliessen.

Zur kompletten Simulation eines Gases benötigt man 3 Komponenten: Die Repräsentation des Gases, ein Modell für dessen zeitliches Verhalten im Raum und schlussendlich ein Beleuchtungsmodell.

Blinn [1] und Kajiya und von Herzen [7] gehörten zu den Ersten, die versucht haben, Gas zu visualisieren. Die Beleuchtungsmodelle, die sie entwickelten, waren schon sehr gut und gehören heute noch zum Standard, doch die Endresultate waren enttäuschend – das Problem waren die Gasmodellierungsalgorithmen.

1985 stellte Gardner [5] neue und äusserst einfache Modellierungs- und Renderingalgorithmen vor. Seine Wolken kreierte mit Hilfe von Fourier-Synthese, d.h. er benutzte eine Kombination von Sinus-Funktionen und veränderte dabei laufend Frequenz, Amplitude und Phase. Er führte also eine Inverse Fourier Transformation durch, wobei er ein $1/f$-Spektrum erzeugte (f ist die Frequenz). So einen Pseudeo-Zufallsgenerator zu entwickeln, war aber ständig ein Kampf gegen das Auftreten von regulären Mustern. Doch mit der Zeit gab es bestimmte 'Magical Numbers' die gute Resultate erzeugten. Er benutzte nicht wie Blinn und Kajiya Volume- Rendering zur Visualisation, sondern entwickelte eine frühe Form des soliden 3D-Texturing: Seine Fourier-Synthese lieferte in jedem Punkt im Raum einen Transparenz-Wert. Und nun benutzte Gardner die visuelle Eigenschaft der Wolken, dass sie die Form von Ellipsoiden haben. Er wertete einfach die Transparenz-Werte auf der Oberfläche eines Ellipsoiden aus und schwächte diese noch zweidimensional nach aussen ab. So waren natürlich keine 'Fly-Throughs' möglich, und 'Fly-Arounds' sahen auch nicht besonders gut aus. Doch die Wolken-Bilder, die Gardner erzeugte, waren absolut realistisch.

Im gleichen Jahr entwickelte Perlin [8] seine berühmten Noise- und Turbulenz-Funktionen. Diese wurden natürlich auch zum Modellieren von Gas benutzt. Besonders erfolgreich damit, war Ebert ab 1989 ([2] und [3]). Wie Gardner versuchte er beim Modellieren die visuellen Eigenschaften der Szenerie zu erfassen, und diese umzusetzten. Zum Beispiel: Je höher, der Dampf steigt, desto mehr verflüchtigt er sich – also hat Ebert die Turblenz-Funktion nach oben abgeschwächt. Besonders gute Resultate konnte er bei Animationen erzeugen, indem er mühsam, und mit viel 'Trial and Error' Wirbel von Hand einfügte und viele Paramter änderte. Letzteres ist besonders schwierig, da die Parameter meistens nicht physikalischer Herkunft sind. Ebert hat aber auch die Volume-Rendering-Algorithmen von Blinn und Kajiya perfektioniert.

Eine andere Methode entwickelte Sakas [10]: Er benutzte zur Erzeugung von Dichtefeldern auch spektrale Synthese. Nur berechnete er auch gleich noch die Entwicklung des Feldes über die Zeit, ein Art 4D-Turbulenz. Diese zeitliche Entwicklung ist aber nur auf einfache Translationen des Gases beschränkt, das bedeutet, dass die Methode nur in einfachen Szenerien überzeugen kann.

Die bisher realistischsten Animationen wurden aber von Stam und Fiume erzeugt ([11] und [12]). Sie modellierten nicht das Gas, sondern das Windfeld indem das Gas sich bewegt. Diese turbulenten Windfelder haben den Vorteil, dass Periodizitäten darin weniger auffallen. Sie waren auch die Ersten, die versuchten physikalische Ansätze umzusetzten. Doch besonders beeindruckend sind ihre Renderingmethoden.

Alle diese Ansätze haben gemeinsam, dass sie nur ein Medium simulieren und den Einfluss des Anderen abschätzen müssen. Das obwohl das Mischen von Gasen unterschiedlicher Temperatur und Geschwindigkeit der Hauptgrund für die interessanten Gas-Phänomene ist. Dieses Paper stellt nun eine total physikalisch basierte Animationsmethode vor, die allen Phänomenen gerecht wird. Dazu werden in Teil 2 die physikalischen Grundlagen erläutert. In Teil 3 werden die Erkenntnisse umgesetzt und es entsteht ein Algorithmus. Am Schluss werden noch die Resultate, Vor- und Nachteile diskutiert.

2 Ein Gas-Modell für Computergraphik

Die Bewegung eines Gases wird durch folgende Faktoren beeinflusst: Erstens, ist die Geschwindigkeit, mit der es in die umgebende Luft strömt, entscheidend. Mixt sich das schnellere Gas mit dem Langsameren, dann werden durch Reibung Kräfte übertragen und das langsame Gas wird mitgeschleppt. Auf diese Weise können neue Wirbel entstehen. Zweitens spielt die Temperatur eine grosse Rolle, denn heisseres Gas steigt schneller als solches, das sich bereits mit der kühleren Umgebung vermischt hat. Folge dieses thermischen Auftriebes sind weitere Turbulenzeffekte. Auch Objekte, um die das Gas herumfliessen soll, haben chaotischen Einfluss auf dessen Verhalten. Schlussendlich hält die Molekularbewegung das Gas selbst dann noch in Bewegung, wenn die Bedingungen ruhig sind

Zur Repräsentation eines Gases benötigt man ein skalares Temperaturfeld T, ein skalares Druckfeld p und ein vektorielles Geschwindigkeitsfeld $\mathbf{u}$. Mit Hilfe dieser Angaben kann nun in den beiden folgenden Abschnitten ein massgeschneidertes Modell entwickelt werden.

2.1 Navier-Stokes-Gleichung

Die Navier-Stokes-Gleichung ist die Grundgleichung für die Hydrodynamik viskoser Flüssigkeiten. Zusammen mit der Kontinuitätsgleichung (div $\mathbf{u} = 0$) kann sie alle Strömungen inkompressibler Flüssigkeiten, insbesondere auch turbulente Strömungen, beschreiben.

Beschränkt man das Modell auf die alltäglichen Bedingungen, d.h. keine von Explosionen herführenden Schockwellen, so lässt sich die Navier-Stokes-Gleichung folgendermassen vereinfachen: Solange die maximale Gasgeschwindigkeit unterschall ist, kann man annehmen, dass das Gas inkompressibel ist. Auch den Einfluss der molekularen Bewegung und der Schwerkraft kann vernachlässigt werden.

Die Navier-Stokes-Gleichung beschreibt die Änderung der Geschwindigkeit des Gases in einem finiten Element an einem fixen Ort im Raum – also nicht die Geschwindigkeitsänderung eines mitfliegenden Partikels:

$$\frac{\partial \mathbf{u}}{\partial t} = \eta \nabla \cdot (\nabla \mathbf{u}) - (\mathbf{u} \cdot \nabla)\mathbf{u} - \nabla p, \qquad (1)$$

wobei ∇ der Gradientenoperator ist, und $\cdot$ ist das innere Produkt. Der zweite Term rechts ($(\mathbf{u} \cdot \nabla)\mathbf{u}$) stellt die Geschwindigkeitsübertragung durch die Nachbarn dar. Ist der Druck in einer benachbarten Region tiefer rsp. die Geschwindigkeit höher, so entsteht eine Sogwirkung entlang dem Gradienten des Drucks – dies ist enthalten im dritten Term (∇p). Der erste Term rechts ($\eta \nabla \cdot (\nabla \mathbf{u})$) hängt von der Krümmung der Geschwindigkeitsverteilung ab und drückt die Reibungskraft aus. Dementsprechend bezeichnet η die dynamische Viskosität. Die Werte für η sind im Bereich von 0.01 (je kleiner, desto dickflüssiger das Gas).

2.2 Thermischer Auftrieb

Die Kräfte, die durch den thermischen Auftrieb entstehen, können folgendermassen modelliert werden:

$$\mathbf{F}_{bv} = -\beta \mathbf{g}_v (T_0 - T_k), \qquad (2)$$

wobei $\mathbf{g}_v$ ist die Gravität in vertikaler Richtung, β ist der Koeffizient der thermischen Expansion (im Bereich von 10^{-3}), T_0 ist eine initiale Referenztemperatur (*z.B.* Zimmertemperatur: $28°C$) und T_k ist die mittlere Temperatur an der Wand zwischen zwei Gaszellen.

Um (2) benutzen zu können, muss auch die Entwicklung der Temperatur über die Zeit berechnet werden. Die Differentialgleichung dazu lautet

$$\frac{\partial T}{\partial t} = \lambda \nabla \cdot (\nabla T) - \nabla \cdot T\mathbf{u}. \qquad (3)$$

Diese Gleichung beschreibt, wie sich die Temperatur in einem Punkt im Raum verändert. Zwei Faktoren spielen hier eine Rolle: Benachbarte Zellen übertragen die Wärme durch den Fluss, der zwischen ihnen stattfindet ($\nabla \cdot T\mathbf{u}$) und Temperaturänderungen, die durch molekulare Bewegungen entstehen – dies entspricht dem ersten Term rechts ($\lambda \nabla \cdot (\nabla T)$). λ repräsentiert dabei die turbulente und molekulare Diffusion, normalerweise Werte zwischen 0.1 und 10.

Hat man (3) in jedem Punkt des Volumen gelöst, kann man daraus mittels (2) die Auftriebskraft berechnen. Diese Kraft beeinflusst die (vertikale) Geschwindigkeit und kann zur Navier-Stokes-Gleichung (1) hinzuaddiert werden:

$$\frac{\partial \mathbf{u}}{\partial t} = \eta \nabla \cdot (\nabla \mathbf{u}) - (\mathbf{u} \cdot \nabla)\mathbf{u} - \nabla p + \mathbf{F}_{bv}. \qquad (4)$$

Die beiden Gleichungen (3) und (4) zusammen bilden nun den physikalischen Grundbaustein, auf dem diese Animationsmethode aufbaut.

3 Erstellen eines effizienten Animations-Systems

Basierend auf dem im 2.Teil vorgestellten Gas-Modell und dessen Gleichungen, wird jetzt ein Animations-System gebildet. Die Autoren haben die Szenerie und die Gleichungen dermassen approximiert, sodass Effizienz und Genauigkeit akzeptabel bleiben.

3.1 Approximation der Szenerie

Die gesamte Simulationsumgebung wird mittels kubischen Zellen approximiert (Abbildung 1), denn so kann die Komplexität der Szenerie stark reduziert werden. Auch, wie man im nächsten Abschnitt sehen wird, können so die Gleichungen (3) und (4) optimal und auf einfache Art und Weise gelöst werden.

Einen Voxel mit Seitenlänge $\Delta\tau$ kann man sich folgendermassen vorstellen (Abbildung 2): Er wird identifiziert durch seine Position i, j, k im Raum (bzw. im Volumendatensatz) relativ zum Ursprung in x, y, z-Richtung. Im Zentrum einer Zelle repräsentieren die Variablen $T_{i,j,k}$ und $p_{i,j,k}$ die Durchschnittstemperatur und den mittleren Druck innerhalb einer Zelle. In der Mitte jeder Seitenwand wird

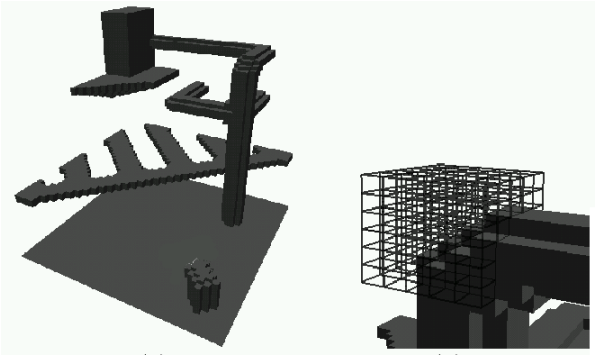

Abbildung 1: Voxel-Approximation der Szenerie.

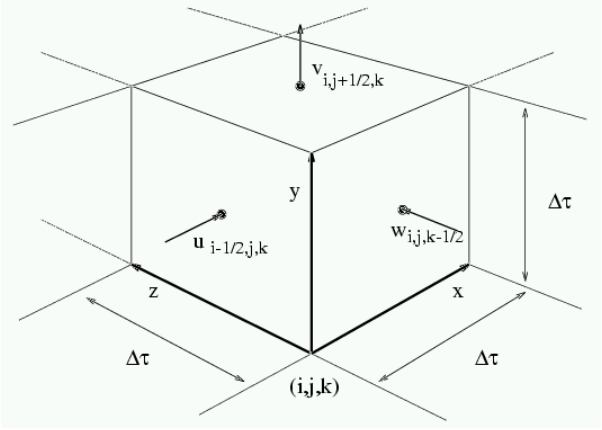

Abbildung 2: Ein Voxel mit Seitenlänge $\Delta\tau$.

die Geschwindigkeit des Flusses zwischen den beiden benachbarten Zellen angegeben: $\mathbf{u}_x, \mathbf{u}_y, \mathbf{u}_z$ bzw. u, v, w. Genau zwischen den beiden Zellen i, j, k und $i+1, j, k$ fliesst der Fluss mit Geschwindigkeit $u_{i+1/2,j,k}$. Hat man dieses Gitter aufgebaut, kann man mit Hilfe von (3) und (4) die Entwicklung von Temperatur, Druck und Geschwindigkeit in jedem Voxel über die Zeit berechnen. Dann kann mittels linearer Interpolation zu jeder Zeit die Temperatur, Druck und Geschwindigkeit in jedem beliebigen Punkt berechnet werden. Ein Beispiel: die Geschwindigkeit im Zentrum der Zelle i, j, k in x-Richtung ist $(u_{i-1/2,j,k} + u_{i+1/2,j,k})/2$.

3.2 Anwenden der Gleichungen auf das Gitter

Mit der Methode der Finiten Differenzen formt man (3) und (4) um, sodass die Gleichungen ideal auf das Gitter angewandt werden können. Anhand von (3)

wird im Folgenden gezeigt, was man unter "Finiten Differenzen" versteht. Ein Differential-Term wie

$$\frac{\partial T}{\partial y}$$

kann mit der Taylor-Entwicklung approximiert werden. Es entsteht dann

$$\frac{\partial T}{\partial y} = \frac{1}{2h}(T(y+h) - T(y-h)) + O(h^2), \qquad (5)$$

wobei h die finite Distanz, über die abgeleitet wurde, darstellt. Ähnlich kann die Ableitung zweiter Ordnung geschrieben werden als:

$$\begin{aligned}\frac{\partial^2 T}{\partial y^2} &= \frac{1}{h^2}(T(y+h) - 2T(y) + T(y-h)) \\ &+ O(h^2). \end{aligned} \qquad (6)$$

Wählt man $\Delta\tau$ für h, dann kann man (6) folgendermassen approximieren:

$$\frac{\partial^2 T}{\partial y^2} = \frac{1}{\Delta\tau^2}(T_{i,j+1,k} - 2T_{i,j,k} + T_{i,j-1,k}).$$

Mit dieser Technik ist es nun möglich (3) nur mit den Variablen im Gitter auszudrücken. Zuerst muss man aber (3) expandieren:

$$\begin{aligned}\frac{\partial T}{\partial t} &= \lambda\left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} + \frac{\partial^2 T}{\partial z^2}\right) \\ &- \left(\frac{\partial Tu}{\partial x} + \frac{\partial Tv}{\partial y} + \frac{\partial Tw}{\partial z}\right). \end{aligned} \qquad (7)$$

Jetzt kann die Methode der Finiten Differenzen angewandt werden. Für die Ableitungen zweiter Ordnung wird wie oben $h := \Delta\tau$ gesetzt. Für die Ableitungen erster Ordnung ist $h := \frac{1}{2}\Delta\tau$.

$$\begin{aligned}T^{n+1}_{i,j,k} &= T^n_{i,j,k} \\ &+ \Delta t\big\{(1/\Delta\tau)[(Tu)^n_{i-1/2,j,k} - (Tu)^n_{i+1/2,j,k} \\ &+ (Tv)^n_{i,j-1/2,k} - (Tv)^n_{i,j+1/2,k} \\ &+ (Tw)^n_{i,j,k-1/2} - (Tw)^n_{i,j,k+1/2}] \\ &+ (\lambda/\Delta\tau^2)[(T^n_{i+1,j,k} - 2T^n_{i,j,k} + T^n_{i-1,j,k}) \\ &+ (T^n_{i,j+1,k} - 2T^n_{i,j,k} + T^n_{i,j-1,k}) \\ &+ (T^n_{i,j,k+1} - 2T^n_{i,j,k} + T^n_{i,j,k-1})]\big\}, \end{aligned} \qquad (8)$$

wobei ein Term wie $(Tu)^n_{i+1/2,j,k}$ repräsentiert den Temperaturfluss zwischen den beiden Zellen (i,j,k) und $(i+1,j,k)$. Er lässt sich leicht berechnen:

$$(Tu)^n_{i+1/2,j,k} = \frac{u^n_{i+1/2,j,k}}{2}(T^n_{i,j,k} + T^n_{i+1,j,k}).$$

Zur Erläuterung von (8): Der Index von T zeigt dessen zeitliche Entwicklung an: T^n ist der Wert von T zu einem Zeitpunkt t, während T^{n+1} der Wert zum Zeitpunkt $t + \Delta t$ ist. Auf die selbe Weise, wie man aus (3) mittels Finiten Differenzen (8) entwickelt hat, kann man auch (4) umformen, sodass am Schluss $\mathbf{u}^{n+1}_{i,j,k}$ durch $\mathbf{u}^n_{i,j,k}$ ausgedrückt werden kann. Das Druckfeld wird im nächsten Abschnitt behandelt.

3.3 Sicherstellen der Genauigkeit

Die Voxel-Approximation ermöglicht eine effiziente Lösung der Gleichungen, hat aber den Nachteil, dass sich Berechnungsfehler der Ordnung $O(\Delta\tau^2)$ einschleichen (in Folge der Taylor-Approximation bei (5) und (6)). Bei der Temperatur T kann dieser Fehler vernachlässigt werden; nicht aber bei der Geschwindigkeit $\mathbf{u}$, denn sie repräsentiert auch die Masse, die durch die Zelle hindurch fliesst. Es sollte gleichviel Gas aus der Zelle hinausströmen, wie hereinströmt, denn ansonsten wird Masse erschaffen oder vernichtet – und das widerspricht dem Satz der Massenerhaltung. Das bedeutet, dass in keinem Punkt im Raum eine Quelle oder Senke vorhanden sein darf, d.h. die Divergenz muss überall gleich Null sein. Die Kontinuitätsgleichung drückt dies aus:

$$\nabla \cdot \mathbf{u} = 0. \qquad (9)$$

Diesen Term wieder mit Taylor approximiert, und anschliessend in Gitter-Variablen ausgedrückt, ergibt die Massen-Divergenz im Zentrum einer jeden Zelle:

$$\begin{aligned}(\nabla \cdot \mathbf{u})_{i,j,k} &= (1/\Delta\tau)[u_{i+1/2,j,k} - u_{i-1/2,j,k} \\ &+ v_{i,j+1/2,k} - v_{i,j-1/2,k} \\ &+ w_{i,j,k+1/2} - w_{i,j,k-1/2}]. \end{aligned} \qquad (10)$$

Dieses Skalarfeld muss nun in jeder Zelle gleich Null sein. Dazu müssen die entstandenen Fehler im Geschwindigkeitsfeld korrigiert werden. Dieses Problem entspricht dem Lösen der klassischen Poisson-Gleichung. Im Folgenden ein bekannter numerischer Lösungsansatz.

Man definiert ein Potentialfeld Ψ, das mit Nullen initialisiert wird, und bei jedem Frame, wird Ψ solange geupdatet, bis die Abbruchbedingung erfüllt ist.

$$\begin{aligned}\Psi^{h+1}_{i,j,k} &= \frac{2}{8/\Delta\tau^2}\{-(\nabla \cdot \mathbf{u})_{i,j,k} \\ &+ \frac{1}{\Delta\tau^2}[\Psi^h_{i+1,j,k} + \Psi^h_{i-1,j,k} \\ &+ \Psi^h_{i,j+1,k} + \Psi^h_{i,j-1,k} \\ &+ \Psi^h_{i,j,k+1} + \Psi^h_{i,j,k-1}]\} - \Psi^h_{i,j,k} \end{aligned} \qquad (11)$$

ist die Updating-Regel, die die $(\nabla \cdot \mathbf{u})_{i,j,k}$-Werte aus dem in (10) berechneten Skalarfeld benutzt. Gestoppt wird die Iteration, wenn für jede Zelle Folgendes gilt:

$$\left| \frac{|\Psi^{h+1}_{i,j,k}| - |\Psi^{h}_{i,j,k}|}{|\Psi^{h+1}_{i,j,k}| + |\Psi^{h}_{i,j,k}|} \right| < \varepsilon. \tag{12}$$

Konvergenz wird meistens nach 8-20 Iterationen pro Frame erreicht. Wobei ε im Bereich von 10^{-4} liegt.

Ist die Iteration konvergiert, muss mit Ψ nun das Geschwindigkeitsfeld $\mathbf{u}$ korrigiert werden. Dabei repräsentiert Ψ die relative Diskrepanz der Masse zwischen benachbarten Zellen. Das bedeutet, um (9) zu erfüllen, wird $\mathbf{u}$ mit den Gradienten von Ψ korrigiert:

$$\begin{aligned} u^{n+1}_{i+1/2,j,k} &= u^{n+1}_{i+1/2,j,k} - \frac{\Psi_{i+1,j,k} - \Psi_{i,j,k}}{\Delta\tau}, \\ v^{n+1}_{i,j+1/2,k} &= v^{n+1}_{i,j+1/2,k} - \frac{\Psi_{i,j+1,k} - \Psi_{i,j,k}}{\Delta\tau}, \\ w^{n+1}_{i,j,k+1/2} &= w^{n+1}_{i,j,k+1/2} - \frac{\Psi_{i,j,k+1} - \Psi_{i,j,k}}{\Delta\tau}. \end{aligned} \tag{13}$$

Auf diese Weise bleibt die Berechnung physikalisch korrekt. Der tolle Seiteneffekt: Es kann gezeigt werden, dass der Gradient des Druckfeldes $p_{i,j,k}$ dem Gradienten des Potentialfeldes $\Psi_{i,j,k}$ entspricht.

Stabilität

Genauigkeit eines Algorithmus' ist sehr wichtig, ist jedoch wertlos, wenn dieser nicht stabil läuft. Es müssen zwei Bedingungen erfüllt werden, sodass Stabilität garantiert werden kann. Erstens, kein Gas-Partikel darf innerhalb eines Zeitschrittes Δt durch einen ganzen Voxel hindurch:

$$\Delta t |\mathbf{u}| < \Delta\tau. \tag{14}$$

Wobei $|\mathbf{u}|$ die maximale Gasgeschwindigkeit ist und Δt ist ungefähr $\frac{1}{30} sec$. Die zweite Bedingung beeinträchtigt die dynamische Viskosität η. Lineare Analysis hat gezeigt, um die Navier-Stokes-Gleichung mit Finiten Differenzen stabil zu lösen, gilt für η:

$$\eta > (\Delta t/2) max[u^2, v^2, w^2]. \tag{15}$$

3.4 Randbedingungen

Das Voxel-Gitter ermöglicht nicht nur effiziente Lösungen der Gleichungen, sondern man kann mit ihm das Verhalten der Interaktion vom Gas mit Objekten modellieren. Die Oberflächeneigenschaften eines Objektes – die Randbedingungen – können beliebig eingestellt werden.

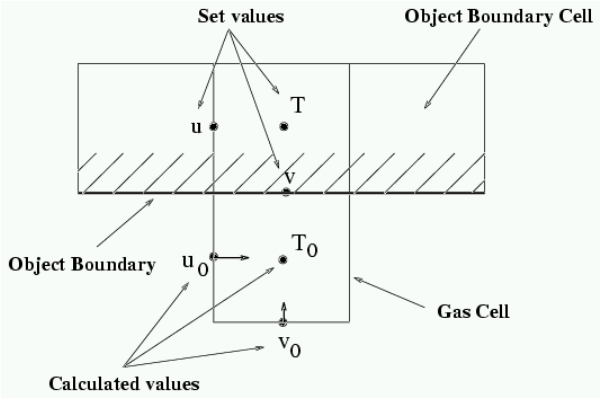

Abbildung 3: Temperatur und Geschwindigkeit an der Grenze zwischen Objekt und Gas

Objekt-Typ	u	v	T
Rauh, steinig (viel Turbulenz)	$-u_0$	0	T_0
Standard (wenig Turbulenz)	0	0	T_0
Glatt (keine Turbulenz)	u_0	0	T_0
Offenes Fenster	0	v_x	T_x
Heisse Gas-Düse	0	v_x	T_x
Warme Suppe	0	0	T_x

Tabelle 1: Beispiele verschiedener Randbedingungen. Index x bedeutet, dass der Animator den Wert wählen kann.

Eine solche "Randsituation" wird in Abbildung 3 zweidimensional skizziert. Die oberen 3 Zellen sind Teil des Objektes, während die untere Zelle eine Gaszelle mit Temperatur T_0 und Geschwindigkeiten u_0 und v_0 ist. Wie müssen nun T, u und v gewählt werden, wenn man zum Beispiel einen Heizkörper simulieren will? Da Gas nicht durch die Oberfläche hindurchdringen kann, ist die Geschwindigkeit genau an der Grenze gleich Null (d.h. $v := 0$). Entlang einer ideal glatten Oberfläche sollte das Gas ungestört fliessen können; um diesen Effekt zu erreichen, werden die zur Oberfläche tangentialen Geschwindigkeiten gleichgesetzt (d.h. $u := u_0$). T entspricht der konstanten Temperatur des Heizkörpers.

Nun sind dem Animator keine Grenzen gesetzt, denn er kann mit dieser Methode beliebige Objekte kreieren – auch offene Fenster (siehe Tabelle 1). Für ein Standardobjekt gilt: $u := 0$, $v := 0$ und T entspricht der Umgebungstemperatur T_0. Übrigens wird jetzt klar, weshalb die Autoren das Geschwindigkeitsfeld an der Seitenfläche der Voxels modelliert haben und nicht im Zentrum derer ($\mathbf{u}_{\underline{i+1/2},j,k}$ anstatt $\mathbf{u}_{\underline{i},j,k}$).

3.5 Der schlussendliche Algorithmus

Der komplette Algorithmus zur Animation von turbulentem Gas kann in zwei Phasen unterteilt werden. In der ersten Phase – der Initialisationsphase – muss der Animator viele Entscheidungen treffen:

1. Unterteilen der Umgebung in Voxel der Seitenlänge $\Delta\tau$.
2. Wählen der Randbedingungen für Temperatur und Geschwindigkeit wie in Tabelle 1.
3. Bestimmen der dynamischen Viskosität η, der thermischen Expansion β und der molekularen Diffusion λ.
4. Bestimmen von Δt (mindestens $\frac{1}{30} sec$), wobei man die Stabilitätsbedingungen (14) und (15) beachten muss.

Sind alle Parameter gewählt, kann die zweite Phase beginnen:

5. An den Oberflächen der Objekte die Randbedingungen anwenden, und dort die neuen Werte für T und $\mathbf{u}$ berechnen.
6. Mit den Finiten Differenz-Approximationen von (3) und (4) können Temperatur und Geschwindigkeit in jeder Zelle geupdatet werden. Der Druck-Gradient muss nicht berechnet werden, dazu kann man Ψ benützten.
7. Mit (10) das Divergenzfeld $(\nabla \cdot \mathbf{u})_{i,j,k}$ berechnen.
8. Solange die Abbruchbedingung (12) bezüglich des Potentialfeldes Ψ nicht in jeder Zelle erfüllt ist, muss Ψ gemäss (11) geupdatet werden.
9. Mit dem Gradienten des berechneten Potentialfeldes kann nun der Fehler im Geschwindigkeitsfeld korrigiert werden (13).
10. Zurück zu Schritt 5 und mit neuem Frame beginnen.

3.6 Rendering

Es gibt viele verschiedene Vorschläge, wie Gas gerendert werden kann. Für dieses Animationssystem im Paper eignen sich Partikel-Systeme (siehe [9]) hervorragend: Massenlose Partikel werden dem System in einer wichtigen Stelle zugefügt, zum Beispiel in der Öffnung des Kamins. Wie die Partikel da verteilt werden, sei dem Animator überlassen (z.B. kann er die Randbedingungen nützen, d.h. die Verteilung proportional zu T und $\mathbf{u}$ wählen, oder eine Noise-Funktion nützen). Sind die Partikel einmal eingeführt, kann mit dem Geschwindigkeitsfeld und der alten Position die Neue (im neuen Frame) bestimmt werden:

$$\mathbf{x}_k^{n+1} = \mathbf{x}_k^n + \Delta t \mathbf{u}_{\mathbf{x}}^n.$$

Hier bezeichnet $\mathbf{u}_{\mathbf{x}}$ die Geschwindigkeit in der Position $\mathbf{x}$. $\mathbf{u}_{\mathbf{x}}$ kann mittels trilinearer Interpolation aus den Gitterwerten berechnet werden. Die Partikel haben keinen Einfluss auf das Geschwindigkeitsfeld, sie dienen nur zur Visualisation des Vektorfeldes.

Basierend auf der momentanen Verteilung der Partikel kann bei jedem Frame ein Dichtefeld ρ generiert werden. Doch auf welche Art der Animator das macht, sei ihm überlassen – je nachdem, welchen Effekt er erzeugen will. Eine Möglichkeit ist, jedem Partikel 1/50 'Masse' zuzuweisen. Sind dann 50 Partikel in einer Zelle (i, j, k) ist die Dichte $\rho_{i,j,k} = \rho(i, j, k) = 1$. Und wieder kann mittels trilinearer Interpolation die Dichte an jedem Ort bestimmt werden. Bei diesem Dichtefeld kann übrigens eine beliebig feinere Auflösung gewählt werden, unabhängig von der Auflösung der Gaszellen (je feiner die Auflösung, desto weniger Voxelartefakte des Volume-Renderers entstehen).

Hat man ein Dichtefeld aufgebaut, können verschiedene Volume-Rendering-Techniken verwendet werden. Das einfachste Verfahren (aus [2]): Für jeden Pixel im Bild einen Strahl durch das Dichte-Volumen schicken und die Transparenz der Wolke in diesem Pixel berechnen. Nur hat man hier, nicht wie beim α-Blending, keine vorgegebene Attenuationsfunktion. Doch die Dichte entlang dem Strahlparameter s bildet die Attenuationsfunktion. Diese muss aufintegriert und exponential abgeschwächt werden. Numerisch ($\sum$ statt $\int$) geschrieben sieht das Ganze folgendermassen aus:

$$\text{Transparenz} = e^{-\tau \sum_{s_{near}}^{s_{far}} \rho(x(s),y(s),z(s))\Delta s}.$$

τ ist hier die optische Tiefe des Materials, s_{near} ist der Punkt indem der Strahl in den Volumendatensatz eintaucht und bei s_{far} tritt er aus.

Natürlich kann auch ein bessere Low-Albedo Beleuchtungsmodell eingesetzt werden. Das lohnt sich vor allem, wenn Lichtquellen in der Szenerie vorhanden sind. Die Gesamtintensität ergibt sich nun durch Integration des Dichtefeldes entlang dem Strahlparameter s, gewichtet mit der jeweils davorliegenden Attenuation:

$$\text{Helligkeit} = \sum_{s_{near}}^{s_{far}} e^{-\tau \sum_{s_{near}}^{s} \rho(x(u),y(u),z(u))\Delta u}$$

$$\cdot \; I \cdot \rho(x(s), y(s), z(s)) \cdot \Delta s.$$

I bezeichnet den Einfluss der Lichtquellen auf das jeweilige Element:

$$I = \sum_i I_i(x(s), y(s), z(s)) \cdot \text{phase}(\theta).$$

θ ist der Winkel zwischen Lichtquelle und Auge. Mit θ macht die Phasenfunktion den Einfluss der Lichtquelle vom Winkel abhängig. Als Phasenfunktionen eignen sich vor allem Henyey-Greenstein-Streufunktionen. $I_i(x(s), y(s), z(s))$ bezeichnet den direkten Einfluss der Lichtquelle i. Hier kann man nun sogenanntes 'Self-Shadowing' erzeugen: Die Gas-Partikel näher der Lichtquelle werfen einen Schatten auf diejenigen hinter ihnen. Dazu muss (für jede Lichtquelle) ein Strahl von der Lichtquelle weg zum Element hin betrachtet werden und wieder dessen Attenuation berechnet werden.

Es gibt noch viele weitere Verbesserungsmöglichkeiten um den Realismus der Szene zu erhöhen, z.B. könnte man mit ambienten Termen ein High-Albedo Belechtungsmodell approximieren, man könnte versuchen den Schatten des Gases auf andere Objekte zu visualisieren oder wie das Gas von einer spiegelnden Oberfläche reflektiert wird. Besonders herauszuheben sind die Methoden von Stam und Fiume, die unglaubliche Resultate erzeugten. Diese Animationmethode im Paper würde sich für deren 'warped Blobs' eignen [12].

4 Resultate

In den folgenden Abschnitten, werden nun drei Beispiele beschrieben. Alle wurden auf einer SGI Indigo2 mit 64MB RAM berechnet. Zum Rendering benutzten die Autoren das Tool BMRT, eine FreeWare-Implementation des RenderMan-Standards. Die Animationen sind auf dem Siggraph'97-Video. In Tabelle 2 sind die entsprechenden Berechnungszeiten angegeben.

Dampf

Bild 4 zeigt einen Heizungsraum; bei dieser Animation kann man sehr gut sehen wie das Gas um das Gitter und die anderen Objekte herumfliesst. Eine Auflösung von $40 \cdot 60 \cdot 40$ mit $\Delta\tau := 0.1$ wurde benutzt. Das Gas wird durch ein Druckventil abgelassen, das durch ein paar Voxels mit entsprechenden Randbedingungen simuliert wird. Die Eintrittsgeschwindigkeit beträgt $0.3m/s$ und der Dampf ist $80°C$ warm. Daraus resultiert eine Höchstgeschwindigkeit des Gases $|\mathbf{u}|_{max}$ von $0.35m/s$.

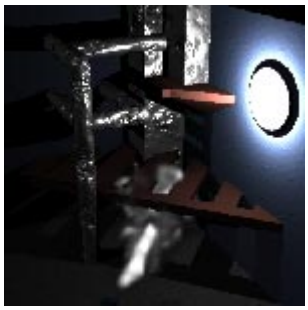

Abbildung 4: Dampf entweicht in ein Heizungsraum

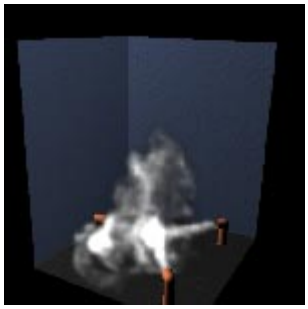

Abbildung 5: Drei Ventile erzeugen Turbulenzen

Unter fast denselben Umständen simuliert die zweite Animation (Bild 5) drei solcher Ventile. Bei beiden Beispielen wurden durchschnittlich 2000 massenlose Partikel pro Sekunde eingeführt.

Rauch

Das Frame aus der dritten Animation (Bild 6) zeigt, wie der Rauch eines Kamins aufsteigt und vom Wind mitgeschleppt wird. Der Wind hat eine Geschwindigkeit zwischen $2m/s$ und $3m/s$. Er wird simuliert, indem die Zellen links am Rand des Gitters folgende Eigenschaft haben:

$$\mathbf{u}_w^{n+1} = 0.98\mathbf{u}_w^n + 0.24\Delta t\phi(t).$$

Wobei $\phi(t)$ eine Zufallszahl zwischen -1 und 1 generiert. Die Konstanten haben keinen physikalischen Hintergrund, es sind nur 'Magical Numbers' die gute Resultate liefern. Der Rauch verlässt den Kamin mit einer Geschwindigkeit von $0.8m/s$ mit einer Temperatur von $46°C$. Die Voxel-Auflösung liegt bei $60 \cdot 60 \cdot 45$ mit $\Delta\tau := 1.0$ (und $|\mathbf{u}|_{max} = 3.4m/s$). Aus Tabelle 2 ist ersichtlich, dass wegen dem Windfeld fast doppelt soviele Ψ-Iterationen benötigt werden wie in den Animationen vorher.

Abbildung 6: Rauch tritt aus dem Kamin aus

Bild	Berechnungszeit $(s/frame)$	Ψ-Zyklen	Renderzeit $(m/frame)$
4	24	10	38
5	28	13	45
6	49	20	14

Tabelle 2: Die durchschnittlichen Berechnungs- und Renderingzeiten für alle drei Beispiele.

5 Stärken und Schwächen der Methode

Das Verfahren im Paper ermöglicht, auf effiziente Art und Weise, das Verhalten von Gas zu simulieren. Da das Ganze aber ein Kompromiss zwischen Effizienz und Genauigkeit ist, hat der Algorithmus auch ein paar Schwächen. Erstens, es können keine Wirbel erzeugt werden, die kleiner als die Gitterauflösung sind. Will man also genauere, kleinere Wirbel, muss man die Auflösung verfeinern. Doch da muss (14) beachtet werden: Halbiert man $\Delta\tau$ so muss auch Δt halbiert werden. Das hat zur Folge, dass man mehr Frames berechnen muss, als nötig wären (rendern muss man nur jedes Zweite). Aber das Gitter hat jetzt 2^3-mal mehr Zellen als vorhin, das bedeutet, dass der Algorithmus mindestens 8-mal länger benötigt ein Frame zu berechnen. Die Animation mit dem Kamin benötigt jetzt $49sec + 14min \approx 15min$ insgesamt für ein Frame alle $\frac{1}{30}sec$. Wird $\Delta\tau$ halbiert, benötigt man für ein Frame alle $\frac{1}{30}sec$ mindestens $2 \cdot 8 \cdot 49sec + 14min \approx 28min$. Wünschenswert wäre eine Art 'Multi-Resolution-Grid', mit dem man in den interessanten Regionen die Auflösung beliebig verfeinern könnte, ohne dabei im ganze Volumen mit der hohen Auflösung rechnen zu müssen.

Eine Schwäche der Finiten Differenzen ist, dass die Orientierung der Zellen die Resultate verfälschen kann. Gaspartikel, die diagonal durch eine Zelle strömen, weisen mehr Diffusion auf, als diejenigen, die Achsen-parallel strömen. Der Fehler kann im Allgemeinen vernachlässigt werden (siehe Animation mit den 3 Ventilen), aber es ist doch empfehlenswert, das Voxelgitter bestmöglich zu platzieren.

Auf bewegte Objekte wurde im Paper auch nicht eingegangen, obwohl damit viele interessante Effekte erzeugt werden könnten. Aber die Autoren haben sich in ihren früheren Paper über 'Fluid Dynamics' damit beschäftigt und schreiben, dass man die dort entwickelten Methoden auch in diesen Algorithmus implementieren könnte.

Eine weitere Schwäche der Methode ist, dass Vektorfelder berechnet werden und Partikelsysteme zur Visualisation gebraucht werden müssen. Partikelsysteme erfordern grosse Datenstrukturen. Natürlich sind die früher durch Turbulenz erzeugten Dichtefelder viel weniger aufwendig zu handhaben.

Die früheren Methoden verlangten grosses Geschick und viel Geduld vom Animator, um die visuellen Effekte von Gas nachzuahmen. Doch all diese 'Noise'-Methoden, vom besten Animator kreiert, können in keinster Weise mit der Methode im Paper mithalten, wenn man den erzielten Realismus betrachtet: Es können Wirbel erzeugt werden, Objekte werden vom Gas umflossen, kaltes Gas trifft auf heisses Gas, der thermische Auftrieb wird simuliert und Randbedingungen können auf einfache Art und Weise erstaunende Effekte erzeugen. Diese Methode im Paper ist eine der Ersten, die das Problem der Visualisation von Gasphänomenen mit physikalischen Ansätzen (Navier-Stokes) versucht zu lösen, und wie die Beispiele zeigen, mit grossem Erfolg.

Literatur

[1] J. F. Blinn. "Light Reflection Functions for Simulation of Clouds and Dusty Surfaces". *ACM Computer Graphics (SIGGRAPH '82)*, 16(3):21-29.

[2] D. S. Ebert und R. E. Parent. "Rendering and Animation of Gaseous Phenomena by Combining Fast Volume and Scanline A-buffer Techniques". *ACM Computer Graphics (SIGGRAPH '90)*, 24(4):357-366.

[3] D. S. Ebert et al. "Texturing and Modeling: A Procedural Approach". *Academic Press, London, 1994.*

[4] N. Foster und D. Metaxas. "Modeling the Motion of a Hot, Turbulent Gas". *ACM Computer Graphics (SIGGRAPH '97).*

[5] G. Y. Gardner. "Visual Simulation of Clouds". *ACM Computer Graphics (SIGGRAPH '85)*, 19(3):297-303.

[6] M. Gross. "Graphische Datenverarbeitung II (Skript zur Vorlesung)". *Informatik, ETH Zürich, 1997.*

[7] J. T. Kajiya und B. P. von Herzen. "Ray Tracing Volume Densities". *ACM Computer Graphics (SIGGRAPH '84)*, 18(3):165-174.

[8] K. Perlin. "An Image Synthesizer". *ACM Computer Graphics (SIGGRAPH '85)*, 19(3):287-296.

[9] W. T. Reeves. "Particle Systems. A Technique for Modeling a Class of Fuzzy Objects". *ACM Computer Graphics (SIGGRAPH '83)*, 17(3):359-376.

[10] G. Sakas. "Modeling and Animating Turbulent Gaseous Phenomena Using Spectral Synthesis". *The Visual Computer, 9, 1993*, 200-212.

[11] J. Stam und E. Fiume. "Turbulent Wind Fields for Gaseous Phenomena". *ACM Computer Graphics (SIGGRAPH '93)*, 369-376.

[12] J. Stam und E. Fiume. "Depicting Fire and Other Gaseous Phenomena Using Diffusion Processes ". *ACM Computer Graphics (SIGGRAPH '95)*, 129-136.

[13] H. H. Thomann. "Strömungslehre I + II (Skript zur Vorlesung)". *AMIV-Verlag, ETH Zürich, 1998.*