# Simulation der Verdunstung und Kondensation von SPH-basierten Flüssigkeiten

**Masterarbeit**

**im Fach Informatik**

**vorgelegt von**

Peter Marchel

Geboren am 16. Juli, 1986 in Swonarewka (Russland)

Angefertigt am

Lehrstuhl für Computergraphik und Multimediasysteme
Naturwissenschaftlich-Technische Fakultät
Universität Siegen

Betreuer:

Prof. Dr. A. Kolb, Lehrstuhl Computergraphik und Multimediasysteme, Universität Siegen

Dipl.-Inf. Hendrik Hochstetter, Lehrstuhl Computergraphik und Multimediasysteme, Universität Siegen

Beginn der Arbeit: 01. Juli 2015

Abgabe der Arbeit: 26. November 2015

**Eidesstattliche Erklärung**

Ich versichere, dass ich die Arbeit ohne fremde Hilfe und ohne Benutzung anderer als der angegebenen Quellen angefertigt habe und dass die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen hat und von dieser als Teil einer Prüfungsleistung angenommen wurde. Alle Ausführungen, die wörtlich oder sinngemäß übernommen wurden, sind als solche gekennzeichnet.

Siegen, den 26. November 2015

**Zusammenfassung**

Die Simulation von Flüssigkeiten wird in der Computergraphik oft diskutiert. Während dafür meistens SPH verwendet wird, wird dagegen die Simulation von Gasen eher auf Gittersystemen realisiert. Bei beiden Systemen gibt es Ansätze, Flüssigkeiten und Gase mit festen Objekten interagieren zu lassen. Die Verknüpfung von einem Gas und einer Flüssigkeit mittels Verdunstung und Kondensation wurde bisher vernachlässigt. Mit dieser Arbeit wird eine Möglichkeit geschaffen, diese Prozesse zu realisieren. Dabei laufen die Simulationen mit SPH und auf einem Gitter parallel ab. Bei einer Verdunstung oder Kondensation findet ein Massenaustausch zwischen den Systemen statt. Dabei ist darauf zu achten, dass bei dem Austausch Masse erhalten wird. Für eine stabile Simulation werden hierfür Partikel über die Zeit ausgeblendet oder eingeblendet anhand der Menge der Masse. So wird deren Einfluss auf die restlichen Partikel kontrolliert. Die Ergebnisse zeigen, dass dies eine geeignete Methode hierfür darstellt. Dabei muss jedoch auf das Verhältnis der Auflösungen beider Systeme geachtet werden.

**Abstract**

The simulation of fluids is often discussed in the computer graphics. While mostly SPH is used in these cases, however, the simulation of gases is rather implemented in grid systems. For both systems, there are approaches for interaction between solid objects and liquids or gases. The coupling of gases and liquids by evaporation and condensation has been neglected. With this work, a possibility is created to implement these processes. The simulations run in parallel with SPH and on a grid. During an evaporation or condensation there is a mass transfer between the two systems. The conservation of mass must be ensured. For the stability, the particles are blended in or out depending on the mass transfer. Thus, their influence on the other particles is controlled. The results show, that this is an appropriate method for this purpose. However, the ratio of the resolutions of both systems is important and must be taken care of.

# Inhaltsverzeichnis

# Verzeichnis der Bilder

# Verzeichnis der Tabellen

# Listings

# Abkürzungsverzeichnis

| | |
|---|---|
| COO | Coordinate (Matrix Format) |
| CPU | Central Processing Unit |
| CSR | Compressed Sparse Row (Matrix Format) |
| CUDA | Compute Unified Device Architecture |
| GPU | Graphics Processing Unit |
| MAC | Marker and Cell |
| PCISPH | Predictive-Corrective Inkompressible SPH |
| SM | Streaming Multiprocessor |
| SPH | Smoothed Particle Hydrodynamics |

# Kapitel 1

# Einleitung

In der Computergraphik haben sich zwei Methoden für die Simulation von Flüssigkeiten und Gasen etabliert. Bei einem Gittersystem werden feststehende Zellen verwendet, welche die Eigenschaften des Stoffes beinhalten. Die Smoothed Particle Hydrodynamics (SPH) beschreiben dagegen das Feld mittels Partikeln, die miteinander interagieren und frei beweglich sind. Zu beiden Systemen wurden Verfahren entwickelt, wie Flüssigkeiten mit festen Objekten interagieren können. Jedoch wurde bisher vernachlässigt zu untersuchen, wie Änderungen des Aggregatzustandes realisiert werden können.

Das Ziel dieser Arbeit ist es ein Verfahren zu entwickeln, mit dem Verdunstungs- und Kondensationseffekte simuliert werden können. Bei den beiden Vorgängen darf keine Masse verloren gehen. Um die Erhaltung der Masse zu gewährleisten, werden zwei Systeme parallel verwendet und miteinander verknüpft. Das Gitter ist für die Simulation der Gasphase zuständig und mit SPH wird die Flüssigkeit simuliert. Die bei einer Verdunstung freigegebene Masse soll von dem Gitter aufgenommen werden, wo die Ausbreitung durch die Zellen stattfindet. Kommt es durch eine Übersättigung zu einer Kondensation, so passiert genau das Gegenteil. Die Masse wird dem Gitter wieder entzogen und in das Partikelsystem übertragen. Dabei muss darauf geachtet werden, dass nur die Masse entzogen werden kann, die in den Zellen vorhanden ist. Der Wechsel des Aggregatzustandes ist ein längerer Prozess. Um die beiden Systeme dabei trotzdem stabil zu halten, sollen Partikel langsam über die Zeit eingeblendet bzw. ausgeblendet werden. Damit wird deren Einfluss auf andere Partikel langsam angepasst. Durch den langsamen Austausch soll auch verhindert werden, dass die Zellen des Gitters zu schnell mit Masse gefüllt werden.

Für die Realisierung der Kopplung der beiden System wird ein bestehendes Framework für SPH Simulationen mit einem Gitter erweitert. Die Berechnungen werden teilweise in C++ und teilweise in CUDA umgesetzt, um die hohe Parallelität der Grafikkarte auszunutzen.Dabei ist es interessant zu wissen in welchem Verhältnis sich die benötigten Zeiten für SPH und das Gittersystem gegenüberstehen.

## Gliederung

Nach der Einleitung in diesem Kapitel werden in Kapitel 2 die Grundlagen erläutert. Dabei wird zunächst ein Überblick über die Simulation von Flüssigkeiten gegeben und wie es mit einem Gitter und mit SPH realisiert wird. Auf die parallele Programmierung in CUDA wird auch kurz eingegangen. In Kapitel 3 werden einige bekannte Verfahren aufgelistet, welche sich mit SPH oder einem Gitter beschäftigen. Der neue Ansatz wird in Kapitel 4 vorgestellt. Dabei wird zunächst die generelle Idee erläutert und dann erklärt, worauf bei einem Massenaustausch zwischen den beiden System zu beachten ist. Daraus ergeben sich einige Anpassungen für das Gitter und SPH. Nach der Beschreibung des Verfahrens kommen in Kapitel 5 Details bei der Implementierung. Hier ist wieder der Massenaustausch wichtig und die Minimierung des Speicherplatzes. Die Ergebnisse dieser Arbeit werden dann in Kapitel 6 besprochen. Abschließen kommt in Kapitel 7 ein kleine Zusammenfassung und Vorschläge für mögliche Erweiterungen.

# Kapitel 2

# Theoretische Grundlagen

Bevor detaillierter auf die Umsetzung der Verdunstung und Kondensation mittels der Kopplung von SPH und Gitter eingegangen werden kann, müssen zunächst die Grundlagen beschrieben werden. Dafür wird zunächst auf die generelle Simulation von Flüssigkeiten und Gasen eingegangen. Danach erfolgt die Erläuterung des Gitters und von SPH. Beide Systeme setzen die Simulation auf verschiedene Weisen um. Das Gitter benutzt die eulersche Betrachtungsweise, bei der die Beobachterpunkte, also in diesem Fall die Zellen, fest an einem Platz sitzen und die Änderung über die Zeit an dieser Position beschreiben. In SPH dagegen wird die lagrangesche Betrachtungsweise verwendet. Dabei bewegen sich die Betrachtungspunkte mit der Strömung der Flüssigkeit mit.

## 2.1 Fluidsimulation

Um Fluide zu simulieren, benötigt man zunächst eine Beschreibung, wie sich Masse und Geschwindigkeiten über die Zeit hinweg verändern. Dazu kann es zum Beispiel kommen, wenn von außen eine Kraft darauf wirkt, sich in dem Simulationsgebiet Bereiche mit unterschiedlichen Drücken befinden oder durch Ausbreitung bzw. Durchmischung von Stoffen anhand von Zufallsbewegungen aufgrund der thermischen Energie. Und falls im Fluid Strömungen existieren, kommen noch Bewegungen entlang dieses Strömungsfeldes hinzu. Diese Vorgänge werden mit folgenden Begriffen beschrieben:

**Advektion** beschreibt den Transport innerhalb eines Strömungsfeldes. In so einem Feld lässt sich für jede Position eine bestimmte Geschwindigkeit feststellen. Befindet sich nun zum Beispiel Masse an einem Ort, an dem eine Strömung existiert, dann wird diese Masse entlang der Bewegungsrichtung transportiert. Betrachtet man ein Blatt welches auf einen Fluss fällt, wird diese durch die Strömung mitgerissen.

**Druck** ist in jedem System vorhanden und kann mit der allgemeinen Gasgleichung bestimmt werden. Durch unterschiedliche Prozesse ändert sich der Druck ständig und es kommt zu Druckunterschieden. Um diese wieder auszugleichen entstehen Kräfte, die ausgehend vom hohen Druck

in Richtung des niedrigen Drucks wirken. Öffnet man zum Beispiel eine Flasche Mineralwasser, in der ein höherer Druck herrscht, entsteht ein Strömungsfeld, welches das Gas entweichen lässt, um die Drücke anzupassen.

**Diffusion** beschreibt die Vermischung verschiedener Stoffe. Dabei werden die Konzentrationen der Stoffe über die Zeit hinweg ausgeglichen. Ein einfaches Beispiel hierfür ist ein Tropfen Tinte, der in ein Glas Wasser fällt. Zunächst ist die Farbe der Tinte noch klar zu erkennen. Aber über die Zeit vermischt es sich mit dem Wasser und löst sich darin auf, bis es nicht mehr zu erkennen ist.

Eine Diffusion gibt es auch in einem Strömungsfeld, welches hier jedoch als *Viskosität* bezeichnet wird. Hierbei passen sich die Geschwindigkeiten der näheren Umgebung an. Setzt sich ein Masseteilchen in Bewegung, dann versuchen anliegende Teilchen sich mit zu bewegen und bremsen je nach Zähigkeit andere Teilchen dadurch aus. Als bestes Beispiel hierfür dient der Honig, welcher als eine Flüssigkeit mit hoher Zähigkeit bekannt ist.

**Externe Kräfte** sind, Kräfte die von außen auf ein System einwirken. Dabei kann zwischen lokalen und globalen Kräften unterschieden werden. Lokale Kräfte wirken nur auf einen bestimmten Bereich des System, wie zum Beispiel ein Ventilator, welcher einen Luftstrom erzeugt. Globale Kräfte haben hingegen eine Auswirkung auf das ganze System. Das beste Beispiel hierfür ist die Schwerkraft, durch die jede Masse nach unten gedrückt wird.

### 2.1.1 Navier-Stokes

Alle beschriebenen Vorgänge können mit zwei Formeln zusammengefasst werden, die *inkompressiblen Navier-Stokes Gleichungen* [Bri08]. Sie bestehen aus mehreren partiellen Differentialgleichungen und dienen als Grundlage aller Fluidsimulationen. Ausgeschrieben lauten sie wie folgt:

$$\frac{\partial \mathbf{u}}{\partial t} = -\left(\mathbf{u} * \nabla\right)\mathbf{u} - \frac{1}{\rho}\nabla p + \nu\nabla^2\mathbf{u} + \frac{\mathbf{f}}{\rho} \tag{2.1}$$

$$\nabla\mathbf{u} = 0 \tag{2.2}$$

In ihr sind die Variablen $t$ für die Zeit, $\rho$ für die Dichte, $p$ für den Druck und $\nu$ für die kinematische Viskosität enthalten. Außerdem beinhaltet sie die Vektoren $\mathbf{u}$ und $\mathbf{f}$. Das letztere beschreibt die extern wirkenden Kräfte als Vektorfeld und wird als $\mathbf{f} = (f_x, f_y, f_z)$ ausgeschrieben. Der Vektor $\mathbf{u} = (u, v, w)$ beinhaltet die drei Komponenten der Geschwindigkeit in jede Dimension.

Betrachtet man die Formel etwas näher, so kann man die verschiedenen Teile der oben beschriebenen Vorgänge ablesen. Dabei beschreibt der erste Teil der Formel auf der rechten Seite mit $(\mathbf{u} * \nabla)\,\mathbf{u}$ die Advektion der Geschwindigkeit. Mit dem zweiten Term $\frac{1}{\rho}\nabla p$ kommen die Beschleunigungen hinzu, die von Druckdifferenzen stammen. Danach kommt der Term $\nu\nabla^2\mathbf{u}$, der die Geschwindigkeiten anhand der Viskosität anpasst. Und als letztes kommen die externen Kräfte, die lokale oder globale Beschleunigungen auslösen können. Die Formel 2.2 dient als Zusatz für inkompressible Fluide.

Dieser wird häufig bei Simulationen verwendet in denen die Dichte konstant ist oder sich nicht signifikant ändert. Bei gleichbleibender Dichte ändert sich das Volumen der Masse nicht. Um dies zu gewährleisten muss das Strömungsfeld Divergenzfrei sein.

### 2.1.2 Euler Gleichungen

Da die Luft in der Erdatmosphäre nur eine geringe Viskosität hat [Har03] kann dieser Term bei solchen Simulationen vernachlässigt werden. Somit verkürzen sich die inkompressiblen Navier-Stokes Gleichungen zu den *inkompressiblen Euler Gleichungen*:

$$\frac{\partial \mathbf{u}}{\partial t} = -\left(\mathbf{u} * \nabla\right)\mathbf{u} - \frac{1}{\rho}\nabla p + \frac{\mathbf{f}}{\rho} + \frac{\mathbf{f}_{buoy}}{\rho} \tag{2.3}$$

$$\nabla \mathbf{u} = 0 \tag{2.4}$$

Im Vergleich zu den Navier-Stokes Gleichungen ist hier nur der Term für die V entfallen. Die restlichen Teile bleiben dieselben. Außerdem wurde hier ein Teil der externen Kräfte raus gezogen und explizit angegeben. Es handelt sich um die Auftriebskraft $\mathbf{f}_{buoy}$.

### 2.1.3 Auftriebskraft

Von der Natur her ist bekannt, dass auf Stoffe unterschiedlicher Dichten und Temperaturen auch unterschiedlich starke Kräfte wirken. Während Gase mit höherer Temperatur eher tendieren zu steigen, wie es bei Heißluftballons zu sehen ist, tendieren Materialien mit höherer Dichte dazu eher zu sinken. Um diesen Effekt zu simulieren wird ein Term für den Auftrieb verwendet [Bri08], der als externe Kraft auf die Fluide wirkt.

$$\mathbf{f}_{buoy} = [\alpha s - \beta(T - T_{amb})]\mathbf{g} \tag{2.5}$$

Damit wird die Beschleunigung $\mathbf{g}$ durch die Gravitation mit einem Faktor angepasst, der abhängig von der Konzentration $s$ des Fluids und der Differenz der Temperatur $T$ des Fluids zur Lufttemperatur $T_{amb}$ ist. Außerdem sind in der Formel noch zwei nicht negative Koeffizienten $\alpha$ und $\beta$, die die Beschleunigung beeinflussen. Bridson [Bri08] führt in seinem Buch auf, dass $\alpha = (\rho_{fluid} - \rho_{air})/\rho_{air}$ und $\beta = 1/T_{amb}$ passende Werte für die Koeffizienten darstellen. Die Dichte $\rho_{air}$ kann mittels der idealen Gasgleichung

$$\rho_{air} = \frac{p}{RT} \tag{2.6}$$

berechnet werden. Dabei ist $p$ der mittlere Luftdruck der Atmosphäre und beträgt 10.013, 25 hPa, $R$ ist die spezifische Gaskonstante für Luft mit 287 $\frac{J}{kg \cdot K}$ und $T$ ist die absolute Temperatur der Luft. Die zweite Dichte $\rho_{fluid}$, welche in die Berechnung von $\alpha$ eingeht, ist die maximale Dichte des Fluides, die noch mit der Konzentration $s$ für die Gleichung 2.5 skaliert wird. An der Formel für den Auftrieb ist ersichtlich, dass bei nicht vorhandener Konzentration $s = 0$ und bei einer Fluidtemperatur gleich der Lufttemperatur $T - T_{amb} = 0$ keine Beschleunigung stattfindet.

### 2.1.4 Splitting

Um die Navier-Stokes bzw. die Euler Gleichungen zu lösen und dabei die Anforderung, dass das Vektorfeld Divergenzfrei bleibt, nicht zu verletzen, bietet es sich an, die Gleichung in Zwischenschritten zu berechnen. Dafür wird die Gleichung in kleine Gleichungen aufgeteilt, die separat voneinander gelöst werden können. Diese Methode wird dementsprechend auch als *Splitting* bezeichnet [Sta99] vorgestellt. Die Gleichung wird in die Schritte aufgeteilt, die in Abbildung 2.1 zu sehen sind. Da in jedem zu berechnenden Zeitschritt das Vektorfeld des vorherigen Zeitschrittes bekannt ist, ist dies der Anfangspunkt der Rechnung.

$$\mathbf{w}_0(\mathbf{x}) \overset{\text{add force}}{\longrightarrow} \mathbf{w}_1(\mathbf{x}) \overset{\text{advect}}{\longrightarrow} \mathbf{w}_2(\mathbf{x}) \overset{\text{diffuse}}{\longrightarrow} \mathbf{w}_3(\mathbf{x}) \overset{\text{project}}{\longrightarrow} \mathbf{w}_4(\mathbf{x})$$

Bild 2.1: Zwischenschritte zu den Navier-Stokes Gleichungen [Sta99]

Die Geschwindigkeit zum Zeitpunkt $t$ an der Position $\mathbf{x} = (x, y, z)$ ist gegeben durch $\mathbf{u}(\mathbf{x}, t)$. Dementsprechend wird zu Beginn der Rechnung $\mathbf{w}_0(\mathbf{x}) = \mathbf{u}(\mathbf{x}, t)$ gesetzt. Darauf wird nun zunächst die Änderung durch die Beschleunigung der externen Kräfte addiert. Dies führt zu dem ersten Zwischenergebnis $\mathbf{w}_1(\mathbf{x}) = \mathbf{w}_0(\mathbf{x}) + \Delta t \frac{\mathbf{f}(\mathbf{x},t)}{\rho}$. Danach wird die Geschwindigkeit noch mit der Advektion, der Diffusion und der Projektion, welches die divergenzfreiheit bewirkt, nacheinander angewandt, bis man mit $\mathbf{u}(\mathbf{x}, t + \Delta t) = \mathbf{w}_4(\mathbf{x})$ das Vektorfeld für den nächsten Zeitschritt berechnet hat.

### 2.1.5 Verdunstung/Kondensation

Verdunstung und Kondensation beschreiben Prozesse, bei denen sich der Zustand eines Stoffes ändert. Bei der Verdunstung geht ein Teil des flüssigen Stoffes in einen gasförmigen Zustand über. Bei der Kondensation hingegen wird ein Teil eines Gases in Flüssigkeit umgewandelt. Die Schnelligkeit mit der die Massen zwischen den beiden Zuständen ausgetauscht werden, hängt unter anderem mit der Sättigung der Luft ab. Ist die Luft noch ungesättigt, so kann sie noch viel Masse aufnehmen und die Verdunstung geschieht mit einer höheren Rate. Wenn sie schon gesättigt ist, kann nicht mehr viel Masse aufgenommen werden und die Verdunstungsrate nimmt ab. Ist die Luft jedoch schon übersättigt, so muss Masse verdrängt werden und es setzt die Kondensation ein.

Für die Bestimmung der Rate mit der ein Stoff verdunstet, wurden schon viele empirische Tests unter verschiedenen Bedingungen durchgeführt. Shah [Sha14] hat diese Untersuchungen alle miteinander verglichen. Bei den meisten Methoden kam eine Gleichung der Form

$$E = (a + bu)(p_w - p_a) \tag{2.7}$$

heraus. Dabei ist $E$ die Rate der Verdunstung in $\frac{kg}{m^2 \cdot h}$, $(p_w - p_a)$ ist der Druckunterschied zwischen Wasser und Luft und $u$ die Geschwindigkeit der Luft. Die restlichen Teile $a$ und $b$ sind Konstanten, die die Rate kontrollieren und sich von Untersuchung zu Untersuchung unterscheiden. In dieser Arbeit

wird die Formel von Smith und anderen [SLJ94] verwendet, die im Detail so aussieht:

$$\frac{\dot{m}}{A} = \frac{(30.6 + 32.1u_w)(p_w - p_a)}{\Delta H_v} \tag{2.8}$$

Dabei steht links mit $\frac{\dot{m}}{A}$ wieder die Verdunstungsrate in $\frac{kg}{m^2 \cdot h}$ und rechts mit $u_w$ die Luftgeschwindigkeit zusammen mit den Sättigungsdampfdrücken des Wassers $p_w$ und der Luft $p_a$. Dazu kommt noch die Latente Wärme $\Delta H_v$, die die aufgenommene oder abgegebene Energie bei gleichbleibender Temperatur bezeichnet.

Zu den Sättigungsdampfdrücken sei noch zu sagen, dass diese bei $p_w$ von der Temperatur des Wassers und bei $p_a$ von der Temperatur des Taupunktes abhängen. Dieser hängt von der Luftfeuchtigkeit ab und kann geschätzt werden durch [Law05]:

$$T_d \approx T - \left(\frac{100 - H_{rel}}{5}\right) \tag{2.9}$$

Der Taupunkt $T_d$ ergibt sich also aus der aktuellen Temperatur $T$ und der relativen Luftfeuchtigkeit $H_{rel}$ in Prozent. Mit diesem Wert lässt sich dann der Sättigungsdampfdruck $p_a$ berechnen, der in die Formel für die Verdunstung einfließt. Der Druck wiederum lässt sich mit der Magnus Formel [Son90] bestimmen.

$$p_w(T) = 6{,}112\text{hPa} \cdot \exp\left(\frac{17{,}62 \cdot T}{243{,}12°C + T}\right) \tag{2.10}$$

Der Sättigungsdampfdruck $p_w$ hängt dabei nur von der Temperatur $T$ ab.

Während es für die Verdunstung von Wasser mehrere Untersuchungen gegeben hat [Sha14], ist für die Rate der Kondensation nicht viel zu finden. Daher wurde für diese Arbeit entschieden für die Berechnung der Kondensation die gleiche Formel 2.8 wie für die Verdunstung zu verwenden. Ist nämlich die Luft übersättigt, dann ist die aktuelle Temperatur $T$ kleiner als die Temperatur $T_d$. Das wiederum bewirkt, dass der Sättigungsdampfdruck $p_a$ bei der Temperatur $T_d$ höher ist als der Sättigungsdampfdruck für die Lufttemperatur $T$. Dadurch ergibt sich nach der Formel (2.8) für die Verdunstungsrate ein negativer Wert. Da die Kondensation der entgegengesetzte Prozess zur Verdunstung ist, kann dieser Wert als die Rate für die Kondensation betrachtet werden.

## 2.2 Gittersystem

Ein Gittersystem zeichnet sich dadurch aus, dass ein bestimmter Raum in endlich viele Abschnitte (Zellen) zerlegt wird. Jede Zelle beschreibt dann die Eigenschaften zu dem Raumabschnitt, der sich innerhalb dieser Zelle befindet. Mögliche Eigenschaften hierbei sind zum Beispiel Masse, Temperatur oder Druck. Die wohl am häufigsten verwendete Variante ist das orthogonale Gitter, welches sich dadurch auszeichnet, dass alle Kanten bzw. Flächen rechtwinklig zueinander stehen. Aber es gibt auch unstrukturierte Gittermodelle, bei denen die Zellen die Form eines Dreiecks oder Tetraeders [Mav97] haben.

Mit Gittersystemen werden seit geraumer Zeit Fluide simuliert. Schon 1965 beschrieben Harlow und Welch [HW+65] wie man Bewegungen von Flüssigkeiten in einem vordefiniertem Bereich darstellen kann. Ihre Methode wurde als *Marker and Cell* (MAC) bezeichnet. Dabei verwenden sie einen Ansatz mit einem versetzten Gitter, dem so genannten *staggered grid*. Das besondere an diesem System sind die Positionen, an denen die Geschwindigkeiten gespeichert werden. Diese sind nämlich nicht in den Mittelpunkten der Zellen lokalisiert, sondern zwischen ihnen. In 2D Gittern befinden sich die Geschwindigkeiten an den Kanten und in 3D an den Flächen zwischen zwei nebeneinander liegenden Zellen. Eine Anschauung für den zweidimensionalen Fall bietet die Abbildung 2.2.

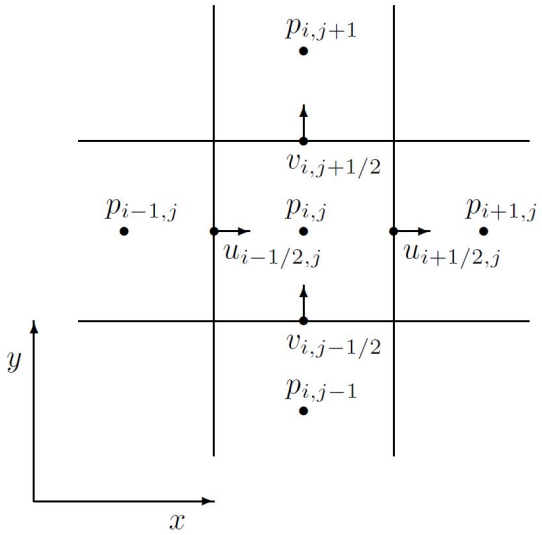



Bild 2.2: Skizze von einem staggered grid [Bri08]

Es ist zu sehen, dass der Druck $p_{i,j}$ sich im Mittelpunkt der Zelle befindet, während die verschiedenen Geschwindigkeitskomponenten an die Zelle angrenzen. An linker und rechter Seite sind die Geschwindigkeiten $u_{i-\frac{1}{2},j}$ und $u_{i+\frac{1}{2},j}$ in x-Richtung. Die Geschwindigkeiten $v_{i,j-\frac{1}{2}}$ und $v_{i,j+\frac{1}{2}}$ in y-Richtung dagegen befinden sich an der unteren und an der oberen Kante. Mit dieser besonderen Lage wird direkt ersichtlich, wie viel Masse in eine Zelle reinströmt und wie viel aus einer Zelle rausströmt.

Um zu verdeutlichen, wieso das versetzte Gittermodell für Fluidsimulationen besser geeignet ist als ein normales Gitter, braucht man sich nur ein Beispiel anzuschauen [Bri08]. Angenommen man hat ein eindimensionales Gitter mit den Werten $..., q_{i-1}, q_i, q_{i+1}, ...$ und will die Ableitung an der Position $q_i$ schätzen, wäre die Formel der zentralen Differenz eine gute Wahl.

$$\left(\frac{\partial q}{\partial x}\right) \approx \frac{q_{i+1} - q_{i-1}}{2\Delta x}$$

Das Problem bei der Formel ist jedoch, dass der mittlere Wert $q_i$, für den die Ableitung bestimmt wird, bei der Berechnung nicht mit einbezogen wird. Sind die Werte $q_{i+1}$ und $q_{i-1}$ beide gleich, dann ist die Ableitung immer Null, unabhängig davon welchen Wert $q_i$ hat. Es werden also Informationen

ausgelassen. Hier kommt das versetzte Gitter ins Spiel. Verschiebt man die Positionen an denen die Werte gespeichert werden um eine halbe Gitterbreite, sodass sie an den Grenzen der Zellen liegen, dann sind die Werte durch $..., q_{i-\frac{3}{2}}, q_{i-\frac{1}{2}}, q_{i+\frac{1}{2}}, q_{i+\frac{3}{2}}, ...$ gegeben. Um nun die Ableitung bei $q_i$ zu bestimmen, kann wieder die zentrale Differenzformel verwendet werden, allerdings ändert sie sich zu:

$$\left(\frac{\partial q}{\partial x}\right) \approx \frac{q_{i+\frac{1}{2}} - q_{i-\frac{1}{2}}}{\Delta x} \tag{2.11}$$

Mit dieser Form werden nun bei Ableitungen alle Werte benutzt, die dafür benötigt werden und dabei werden auch keine Werte übersprungen, die evtl. dafür berücksichtigt werden müssten. Damit ist es nun einfacher die Strömungsgleichungen 2.1 und 2.3, besonders was den Druckterm angeht, zu lösen.

Durch das versetzte Gitter können nun also zentrale Differenzen besser berechnet werden. Die Versetzung führt jedoch dazu, dass die Geschwindigkeiten immer interpoliert werden müssen. Denn unabhängig an welcher Position der Wert gebraucht wird, muss auf Werte der näheren Umgebung zurückgegriffen werden. Selbst dann wenn man einen Gitterpunkt oder versetzten Gitterpunkt betrachtet, ist eine bilineare oder trilineare Interpolation nötig. Da die Geschwindigkeiten versetzt sind, ergeben sich für jede Dimension unterschiedliche Gewichte für die Interpolation. Hier sind einige Beispiele, wie Interpolationen im zweidimensionalem aussehen können:

$$\begin{aligned}
\mathbf{u}_{i,j} &= \left(\frac{u_{i-1/2,j} + u_{i+1/2,j}}{2}, \frac{v_{i,j-1/2} + v_{i,j+1/2}}{2}\right) \\
\mathbf{u}_{i+1/2,j} &= \left(u_{i+1/2,j}, \frac{v_{i,j-1/2} + v_{i,j+1/2} + v_{i+1,j-1/2} + v_{i+1,j+1/2}}{4}\right) \\
\mathbf{u}_{i,j+1/2} &= \left(\frac{u_{i-1/2,j} + u_{i+1/2,j} + u_{i-1/2,j+1} + u_{i+1/2,j+1}}{4}, v_{i,j+1/2}\right)
\end{aligned}$$

Während man für einen Zellenmittelpunkt lediglich den Durchschnitt zweier Werte bilden muss, benötigt man auf den Zellgrenzen den Durchschnitt von vier Werten. Und für einen beliebigen Punkt im Raum wird jeweils eine Interpolation für jede Dimension mit den dazugehörigen Gewichten notwendig.

Das versetzte Gitter hat auch Auswirkungen auf die Speicherung der benötigten Werte. Da man in Arrays nur ganzzahlige Indizes verwenden kann, braucht man eine Vorschrift, wie man die Werte für die verschobenen Indizes $i + 1/2$ speichert. Man kann dabei den ganzzahligen Index nehmen, welcher jedoch auf einen verschobenen Wert verweist, der direkt vor oder nach diesem Index liegt. Ein Beispiel für den nächstliegenden Wert sieht so aus:

$$\begin{aligned}
\mathrm{p(i,j,k)} &= p_{i,j,k} \\
\mathrm{u(i,j,k)} &= u_{i+1/2,j,k} \\
\mathrm{v(i,j,k)} &= v_{i,j+1/2,k} \\
\mathrm{w(i,j,k)} &= w_{i,j,k+1/2}
\end{aligned}$$

Je nachdem, ob die Randgeschwindigkeiten mit gespeichert werden sollen, ergeben sich unterschiedliche Größen für die Arrays. Werden die Ränder benötigt, so braucht man bei einem Gitter der Größe $nx \times ny \times nz$ drei Arrays mit den Größen $(nx+1) \times ny \times nz$ für die u-Komponente, $nx \times (ny+1) \times nz$ für die v-Komponente und $nx \times ny \times (nz+1)$ für die w-Komponente. Lässt man die Ränder weg, so braucht man dementsprechend einen Wert weniger für die jeweiligen Komponenten.

### 2.2.1 Advektion

Wie schon in Kapitel 2.1 beschrieben handelt es sich bei der Advektion um einen Transport entlang des Strömungsfeldes. Die Basis Gleichung hierfür lautet:

$$Dq/Dt = 0$$

Der Wert von $q$ ist von der Position und von dem Zeitpunkt abhängig. Es ist also eine Funktion $q(x, y, z, t)$. Schreibt man nun die Formel für die Advektion mithilfe der Kettenregel aus, dann ergibt sich die Form:

$$\frac{\partial q}{\partial t} + u\frac{\partial q}{\partial x} + v\frac{\partial q}{\partial y} + w\frac{\partial q}{\partial z} = 0$$

Diese Formel lässt sich verkürzt schreiben als $\frac{\partial q}{\partial t} = -(\mathbf{u} * \nabla)\, q$ und ist so auch in den Navier-Stokes Gleichungen (2.1) enthalten. Die Lösung des Transportes in der Lagrange Betrachtungsweise ist trivial, da die Partikel bei der Bewegung mit dem Strömungsfeld alles mit transportieren. In der Euler Betrachtungsweise jedoch bleiben die Zellen immer an einer Stelle und die Eigenschaften ändern sich zeitlich abhängig von Strömungen.

Um dieses Problem zu lösen, hat Stam [Sta99] einen Ansatz vorgestellt, der beide Betrachtungsweisen vereint, nämlich die *Semi-Lagrange* Methode. Wie der Name schon vermuten lässt, wird dabei die Advektion in der Euler Betrachtungsweise mithilfe einer Lagrange Variante gelöst. Da die Zellpositionen immer fest sind, braucht man sich nur zu fragen, welcher Wert nach einem Zeitschritt an dieser Stelle landen würde. Der Zellmittelpunkt wird also wie ein Partikel betrachtet, welches sich zu einer bestimmten Zeit an dieser Position befindet. Da das Strömungsfeld bekannt ist, kann der Anfangspunkt des Partikels herausgefunden werden, indem man den Pfad entlang der Strömung zurück verfolgt.

Angenommen man will einen Wert $q_{Ziel}^{t+1}$ an der Position $\mathbf{x}_{Ziel}$ für den nächsten Zeitschritt $t+1$ bestimmen. Da die Geschwindigkeit an der Position bekannt ist, folgt man ihr für einen Zeitschritt $\Delta t$ zurück und schaut sich dort den Wert $q_{Start}^{t}$ an, den das imaginäre Partikel im vorherigen Zeitschritt hatte. Die einfachste Möglichkeit den Weg zurück zu verfolgen ist ein Euler Schritt. Damit ergibt sich die alte Position:

$$\mathbf{x}_{Start} = \mathbf{x}_{Ziel} - \Delta t \mathbf{u}(\mathbf{x}_{Ziel})$$

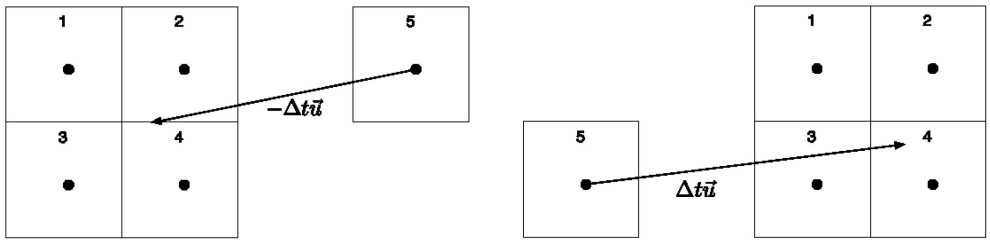
Man kann hier auch stabilere Techniken höherer Ordnung wie zum Beispiel das Runge-Kutta Verfahren verwenden. Unabhängig davon welches Verfahren benutzt wird, befindet sich die Anfangsposition

$\mathbf{x}_{Start}$ in den seltensten Fällen genau auf einem Zellmittelpunkt. Daher ist eine Interpolation an dem Punkt $\mathbf{x}_{Start}$ notwendig, um den Wert $q^t_{Start}$ zu erhalten. Dadurch erhält man Gewichte $\omega_{ij}$ die angeben, welcher Anteil von dem Wert $q_i$ der Quellzelle $i$ in die Zielzelle $j$ transportiert wird. In einer Formel ausgedrückt sieht das folgendermaßen aus:

$$q_j = \sum_i \omega_{ij} q_i$$

wobei $q_j$ die Zielzelle ist, $\omega_{ij}$ die Gewichte der Interpolation und $q_i$ die Zellen sind, zwischen denen interpoliert wird. Die Summe der Gewichte, die zur Zielzelle führen, ergeben immer $\sum_j \omega_{ij} = 1$. Führt man das nun für alle Zellen durch, lässt sich für jede Zelle $i$ bestimmen, welchen Beitrag sie für den Zeitschritt $t+1$ hat. Dies ist nämlich die Summe ihrer Gewichte $\sigma_i = \sum_i \omega_{ij}$. Ergibt sich eine Summe von 1, dann bedeutet dies, dass der ganze Anteil dieser Zelle im nächsten Zeitschritt wiederzufinden ist. Ist die Summe jedoch kleiner als 1, dann geht etwas verloren und ist die Summe größer als 1, dann kommt etwas in dem Zeitschritt hinzu.

Führt man die Advektion zum Beispiel für die Masse durch, so soll die Masse in jedem Zeitschritt erhalten bleiben. Hierfür muss $\sigma_i$ für alle Zellen 1 ergeben. Um dies zu gewährleisten, haben Lentine und andere[LGF11] eine Methode vorgestellt, die ohne Bedingungen stabil funktioniert. Ihre Idee beruht auf der Kombination von einem Semi-Lagrangen Rückwärtsschritt, wie er in diesem Kapitel vorgestellt wurde, und einem Semi-Lagrangen Vorwärtschritt, bei dem eine Bewegung von einer Position entlang der Strömung stattfindet. Beide Varianten sind im Bild 2.3 skizziert.




(a) (b)

Bild 2.3: Unconditional Stable Advection.

Um zu verhindert, dass Masse aus dem Nichts entsteht, soll zunächst $\sigma_i$ auf 1 begrenzt werden. Daher werden die Gewichte für alle Zellen $i$ mit $\sigma_i \geq 1$ angepasst. Dafür werden die Gewichte mit $\hat{\omega}_{ij} = \omega_{ij}/\sigma_i$ runterskaliert, sodass danach $\sum_i \hat{\omega}_{ij} = 1$ gilt. Der Fall $\sigma_i < 1$ wird etwas anders gelöst. Ist die Summe der Gewichte kleiner als 1, so bedeutet dies, dass nicht alles aus der Zelle $i$ in den nächsten Zeitschritt transportiert wird. Es fehlt also der Anteil $(1 - \sigma_i)$. Um diesen Verlust zu vermeiden kann für diesen Anteil noch ein Semi-Lagrange Vorwärtsschritt durchgeführt werden. Man erzeugt also einen Pfad entlang der Strömung und findet raus, wo ein Partikel landen würde,

wenn es aus dieser Zelle startet. Doch anders als beim Rückwärtsschritt, bei dem man den Wert durch Interpolation berechnet, wird in diesem Schritt der Wert auf die nächstgelegenen Zellen verteilt. Die hierbei entstandenen Gewichte werden als $f_{ij}$ bezeichnet. Damit am Ende die Summe der Gewichte 1 ergibt, müssen die neuen Gewichte runterskaliert werden und für die endgültigen Gewichte ergibt sich dann $\hat{\omega}_{ij} = \omega_{ij} + (1 - \sigma_i) f_{ij}$. Somit ist sichergestellt, dass für alle Zellen $i$ die Gleichung $\sum_i \hat{\omega}_{ij} = 1$ erfüllt ist. Somit geht nichts aus dem Zeitschritt $t$ verloren und es kommt nichts neues zum Zeitschritt $t + 1$ hinzu.

### 2.2.2 Herstellung der Inkompressibilität

Im Kapitel 2.1.2 wurden schon die Euler Gleichungen vorgestellt, jedoch wurde bisher nicht konkret darauf eingegangen, wie man die Inkompressibilität ($\nabla \mathbf{u} = 0$) herstellt. Weiterhin ist im Abschnitt 2.1.4 beschrieben, dass man die Gleichungen in Zwischenschritten berechnen kann. Wendet man das Splitting auf die Gleichung 2.3 an, bis nur noch der Druckterm übrig ist, sieht die Berechnung des Strömungsfeldes folgendermaßen aus:

$$\mathbf{u}^* = \mathbf{u}^t + \Delta t(-(\mathbf{u} * \nabla)\,\mathbf{u} + \frac{\mathbf{f}}{\rho} + \frac{\mathbf{f}_{buoy}}{\rho}) \tag{2.12}$$

$$\mathbf{u}^{t+1} = \mathbf{u}^* - \Delta t \frac{1}{\rho} \nabla p \tag{2.13}$$

An dieser Stelle kann nun der Druck dazu verwendet werden, um das Strömungsfeld inkompressibel zu machen. Zur Bestimmung des benötigten Druckes sollte man sich zunächst die Formel 2.13 ausgeschrieben für jede ihrer Komponenten vor Augen halten:

$$u^{t+1}_{i+1/2,j,k} = u_{i+1/2,j,k} - \Delta t \tfrac{1}{\rho} \tfrac{p_{i+1,j,k} - p_{i,j,k}}{\Delta x} \tag{2.14}$$

$$v^{t+1}_{i,j+1/2,k} = v_{i,j+1/2,k} - \Delta t \tfrac{1}{\rho} \tfrac{p_{i,j+1,k} - p_{i,j,k}}{\Delta x} \tag{2.15}$$

$$w^{t+1}_{i,j,k+1/2} = w_{i,j,k+1/2} - \Delta t \tfrac{1}{\rho} \tfrac{p_{i,j,k+1} - p_{i,j,k}}{\Delta x} \tag{2.16}$$

Um die Inkompressibilität zu erreichen, muss wie schon erwähnt, die Gleichung $\nabla \mathbf{u} = 0$ erfüllt sein. Schreibt man diese Gleichung auch vollständig aus, sieht es folgendermaßen aus:

$$(\nabla \mathbf{u})_{i,j,k} \approx \frac{u_{i+1/2,j,k} - u_{i-1/2,j,k}}{\Delta x} + \frac{v_{i,j+1/2,k} - v_{i,j-1/2,k}}{\Delta x} + \frac{w_{i,j,k+1/2} - w_{i,j,k-1/2}}{\Delta x} = 0 \tag{2.17}$$

Da diese Gleichung für die Inkompressibilität immer gelten muss, werden hier die Gleichungen 2.14 bis 2.16 eingesetzt. Somit kann man dann auf den Druck schließen. Nach dem Einsetzen und einer

Umformulierung kommt am Ende die Gleichung

$$\frac{\Delta t}{\rho}\left(\frac{6p_{i,j,k} - p_{i+1,j,k} - p_{i-1,j,k} - p_{i,j+1,k} - p_{i,j-1,k} - p_{i,j,k+1} - p_{i,j,k-1}}{\Delta x^2}\right) = -\left(\frac{u_{i+1/2,j,k} - u_{i-1/2,j,k}}{\Delta x} + \frac{v_{i,j+1/2,k} - v_{i,j-1/2,k}}{\Delta x} + \frac{w_{i,j,k+1/2} - w_{i,j,k-1/2}}{\Delta x}\right) \tag{2.18}$$

heraus. Sie hat die Gestalt einer *Poisson* Gleichung $-\Delta t/\rho \nabla \cdot \nabla p = -\nabla \cdot \mathbf{u}$. Auf der rechten Seite ist die Divergenz der Strömung, also die Angabe, wie viel in eine Zelle einströmt oder aus einer Zelle ausströmt. Und auf der linken Seite stehen die Drücke, die die Divergenzen ausgleichen sollen.

Die Gleichung 2.18 ist hier nur für eine allgemeine Zelle mit den Koordinaten $(i, j, k)$ gezeigt. Da das Gitter $nx \times ny \times nz$ Zellen hat, bekommt man dementsprechend genau so viele Gleichungen dabei raus. Alle Gleichungen zusammengenommen ergeben ein lineares Gleichungssystem. Es hat dann die Gestalt $Ax = b$.

Um auf die benötigten Drücke zu schließen, muss also zunächst die Divergenz für jede Zelle bestimmt werden. Danach bringt man die Matrix $A$ auf die rechte Seite, um den Druck für jede Zelle zu erhalten. Für diese Lösung gibt es unterschiedliche Möglichkeiten. Dazu kann man zum Beispiel das Gaußsche Eliminationsverfahren, die QR-Zerlegung oder die Cholesky-Zerlegung benutzen. Es ist auch möglich die Lösung mittels Iteration, wie zum Beispiel dem Gauß-Seidel oder dem Jacobi-Verfahren, zu schätzen. Je mehr Schritte man dabei durchführt werden, desto genauer wird sich der Lösung angenähert. Ist eine bestimme Fehlertoleranz erreicht, kann der Algorithmus abgebrochen werden.

## 2.3 Smoothed Particle Hydrodynamics

Durch SPH kann jede beliebige Funktion mithilfe einer Menge an bekannten Punkten beschrieben werden [Mon92]. Diese bekannten Punkte sind die Partikel, welche einige Werte mit sich tragen. Die restlichen Werte der Funktion können dann mithilfe einer Interpolation der bekannten Punkte bestimmt werden. Dadurch wird zwischen den Partikeln geglättet, was im Englischen *smoothed* heißt, wovon die Methode ihren Namen erhielt. Die Interpolation einer Funktion $Q$ an einer Position $\mathbf{x}$ wird mit einem Integral berechnet.

$$Q_I(\mathbf{x}) = \int Q(\mathbf{x}') W(\mathbf{x} - \mathbf{x}', h)\, \mathbf{dx}' \tag{2.19}$$

Die Funktion $W$ ist dabei der Glättungskernel und muss die zwei Eigenschaften

$$\int W(\mathbf{x} - \mathbf{x}', h)\, \mathbf{dx}' = 1 \tag{2.20}$$

$$\lim_{h \to 0} W(\mathbf{x} - \mathbf{x}', h) = \partial(\mathbf{x} - \mathbf{x}') \tag{2.21}$$

erfüllen. Da bei SPH nur endlich viele Partikel verwendet werden, kann nur eine Schätzung der Gleichung 2.19 vorgenommen werden. Für die Approximation der Interpolation wird die folgenden Summe verwendet:

$$Q_S(\mathbf{x}) = \sum_j m_j \frac{Q_j}{\rho_j} W(\mathbf{x} - \mathbf{x}_j, h) \tag{2.22}$$

In dieser Gleichung ist $Q$ ein skalarer Wert, der interpoliert werden soll und $\mathbf{x}$ ist die Position, wo der Wert bestimmt werden soll. Die Summe ist eine Iteration über alle Partikel $j$ und darin enthalten ist die Masse $m_j$ des Partikels $j$, seine Position $\mathbf{x}_j$, die Dichte $\rho_j$ und sein Wert $Q_j$. Auf den ersten Blick scheint es so, dass wirklich jedes Partikel in dieser Summe Beachtung findet. Tatsächlich haben jedoch nur wenige Partikel in der Nachbarschaft von der Position $\mathbf{x}$ eine Wirkung auf die Summe. Denn der Einflussbereich von den Partikeln wird mittels des Glättungskernels bestimmt. Dafür wird nämlich eine symmetrische Funktion genommen, die ab einem Radius $h$ so weit abflacht, dass der Wert gegen $0$ geht. Somit werden also immer nur die Partikel berücksichtigt, die im Einflussbereich $|\mathbf{x} - \mathbf{x}_j| \leq h$ liegen.

Die Masse und die Dichte tauchen in der Gleichung 2.22 auf, weil jedes Partikel $i$ einen gewissen Raum $V_i = m_i/\rho_i$ repräsentiert [MCG03]. Während die Masse $m_i$ für die Partikel bekannt ist und mit ihnen transportiert wird, ändert sich jedoch die Dichte $\rho_i$ abhängig von der Position eines Partikels. Durch Einsetzen der Dichte in die obere Gleichung (2.22) kann sie an der Position $\mathbf{x}$ bestimmt werden.

$$\rho_S(\mathbf{x}) = \sum_j m_j \frac{\rho_j}{\rho_j} W(\mathbf{x} - \mathbf{x}_j, h) = \sum_j m_j W(\mathbf{x} - \mathbf{x}_j, h) \tag{2.23}$$

Die eingesetzte Dichte kürzt sich mit der Dichte aus der Gleichung weg. Übrig bleiben nur noch die Massen der Partikel mit denen der Wert bestimmt wird. Da für die Animation von Flüssigkeiten auch Ableitungen von dem zu simulierendem Raum gebraucht, müssen dies für die Gleichung (2.22) bestimmt werden. Die Position kommt nur in dem Glättungskernel vor. Daher genügt es, wenn man nur diesen ableitet. Die ersten beiden Ableitungen werden dann mit den folgenden Formeln gebildet.

$$\nabla Q_S(\mathbf{x}) = \sum_j m_j \frac{Q_j}{\rho_j} \nabla W(\mathbf{x} - \mathbf{x}_j, h) \tag{2.24}$$

$$\nabla^2 Q_S(\mathbf{x}) = \sum_j m_j \frac{Q_j}{\rho_j} \nabla^2 W(\mathbf{x} - \mathbf{x}_j, h) \tag{2.25}$$

### 2.3.1 SPH Simulation

Bei der Animation mithilfe von SPH müssen wie bei einem Gittersystem die Massen- und die Impulserhaltung gelten. Im Kapitel 2.2.1 wurde beschrieben, wie die Masse in einem Gitter erhalten werden kann. Bei einem Partikelsystem ist die Massenerhaltung immer gegeben. Denn hierbei wird den Partikeln eine bestimme Masse zugeordnet. Bei einer Bewegung der Partikel bewegt sich deren Masse immer mit. Somit ändert sich immer nur die Position der Masse, jedoch nicht die Menge.

Es bleibt noch die Impulserhaltung zu betrachten. Bei SPH sind alle Eigenschaften des Feldes den Partikeln zugewiesen. Dazu gehört auch die Geschwindigkeit. Dies hat Auswirkungen auf die Advektion. Da jedes Partikel seine Geschwindigkeit kennt, braucht man bei der Advektion nicht mehr die Umgebung zu betrachten. Es genügt hierbei in jedem Zeitschritt die Partikel einfach entlang ihrer Geschwindigkeit zu bewegen. Dafür müssen die Geschwindigkeiten jedoch angepasst werden. Die Änderung erfolgt durch eine Beschleunigung [MCG03]

$$\mathbf{a}_i = \frac{d\mathbf{v}_i}{dt} = \frac{\mathbf{f}_i}{\rho_i} \tag{2.26}$$

Dabei ist $\mathbf{a}_i$ die Beschleunigung des Partikels $i$,was gleichbedeutend ist mit der Änderung seiner Geschwindigkeit $\mathbf{v}_i$ über die Zeit hinweg. Sie ist abhängig von den Kräften $\mathbf{f}_i$ und der Dichte $\rho_i$, welche sich an der Position des Partikels $i$ befinden.

### 2.3.2 Bestimmung der Kräfte

Die Kräfte, die auf die Partikel wirken, können unterteilt werden in externe Kräfte, Druckkräfte und Kräfte, die aus der Viskosität resultieren. Externe Kräfte wirken von außen direkt auf die einzelnen Partikel, wie zum Beispiel die Gravitation es tut. Bei der Viskosität und beim Druck muss jedoch die Umgebung der Partikel betrachtet werden. Die Viskosität kommt von den Unterschieden bei der Geschwindigkeit zu Stande. Durch schnelle Partikel werden langsame Partikel beschleunigt und langsame Partikel bremsen schnelle Partikel ab, sodass sich ihre Geschwindigkeiten angleichen. Müller und andere [MCG03] haben dafür die folgende Kraft für die Viskosität benutzt:

$$\mathbf{f}^{viscosity} = \mu \sum_j \frac{m_j}{\rho_j} \left(\mathbf{v}_j - \mathbf{v}_i\right) \nabla^2 W\left(\mathbf{x} - \mathbf{x}_j, h\right) \tag{2.27}$$

Dabei wird die Differenz der Geschwindigkeiten $(\mathbf{v}_j - \mathbf{v}_i)$ in der Umgebung des Partikels $i$ betrachtet. Diese wird noch mit der dynamischen Viskosität skaliert. Mit dieser Kraft werden die Partikel in die Richtung der Bewegung ihres Umfeldes beschleunigt. Sind alle Geschwindigkeiten gleich, dann hat diese Kraft keinen Einfluss auf die Beschleunigung.

Für die Kraft, die durch den Druck auf die Partikel wirkt, braucht man die erste Ableitung der Gleichung 2.22. Da dort durch Einsetzen des Druckes in die Gleichung immer die Werte in der Umgebung in Betracht gezogen werden, kommen unsymmetrische Kräfte dabei heraus. Um dies zu verhindern, muss bei der Berechnung der Druckkraft der eigene Wert mit einbezogen werden. In der Zusammenfassung von Monaghan [Mon05] ist deshalb die folgende Form angegeben:

$$\mathbf{f}^{pressure} = -m_i \sum_j m_j \left(\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2}\right) \nabla W\left(\mathbf{x} - \mathbf{x}_j, h\right) \tag{2.28}$$

Dabei wird sowohl der Druck des Partikels $i$ als auch aller Nachbarn $j$ berücksichtigt. Die so berechnete Kraft ist nun symmetrisch für jedes beliebige Partikelpaar. Der Druck, der in der Gleichung

vorkommt, kann mit der idealen Gasgleichung 2.6 bestimmt werden. Da es bei der Druckkraft nur um die Unterschiede der Drücke geht und nicht um den absoluten Wert, kann der Druck etwas anderes berechnet werden. Desbrun und Cani [DC96] haben in ihrer Arbeit bei der Berechnung des Drucks über die ideale Gasgleichung die Form $p = RT(\rho - \rho_0)$ verwendet. Dadurch soll die Dichte immer möglichst auf dem Wert $\rho_0$ gehalten werden, welches einem Inkompressiblen System entspricht.

Im Jahre 2009 haben Solenthaler und Pajarola[SP09] eine weitere Methode für die Bestimmung des Druckes entwickelt. Bei ihrer Variante werden zunächst die Positionen der Partikel für den nächsten Zeitschritt geschätzt und anhand der dort herrschenden Dichte werden die Drücke angepasst. Dieses Verfahren hat demnach den Namen *predictive-corrective incompressible* SPH (PCISPH) erhalten. Die einzelnen Schritte des Algorithmus können grob beschrieben werden mit:

```
1  while animating do
2    for all i do
3      find neighborhoods N_i(t)
4    for all i do
5      compute forces F^{v;g;ext}(t)
6      initialize pressure p(t) = 0.0
7      initialize pressure force F_p(t) = 0.0
8    while ( ρ*_err(t+1) > μ) || (iter < minIterations) do
9      for all i do
10       predict velocity v*_i(t+1)
11       predict position x*_i(t+1)
12     for all i do
13       predict density ρ*_i(t+1)
14       predict density variation ρ*_err(t+1)
15       update pressure p_i(t)+= f(ρ*_err(t+1))
16     for all i do
17       compute pressure force F_p(t)
18   for all i do
19     compute new velocity v*_i(t+1)
20     compute new position x*_i(t+1)
```

Listing 2.1: PCISPH Algorithmus

In jedem Zeitschritt werden zunächst in Zeile 2 die Nachbarschaften der einzelnen Partikel bestimmt. Dadurch lässt sich die Berechnung aller Werte mit der SPH Interpolation beschleunigen. Dann werden alle Kräfte in Zeile 5 ermittelt mit Ausnahme der Druckkraft. Sie wird nämlich zusammen mit dem Druck zunächst mit 0 (Zeile 6 und 7) initialisiert. Ab Zeile 8 beginnt das eigentliche Verfahren für die Bestimmung der Druckkraft. In dieser Schleife werden zunächst die Geschwindigkeiten und die Position der Partikel für den nächsten Zeitschritt vorhergesagt. Von der neuen Konstellation der Partikel ermittelt man die Dichten $\rho_i^*(t+1)$ und damit die Abweichung $\rho_{err}^*(t+1) = \rho_i^* - \rho_0$ von der idealen Dichte $\rho_0$. Mit diesem Fehlerwert wird in Zeile 15 der Druck $p_i(t)$ aktualisiert und mit diesem dann in Zeile 17 die Druckkräfte berechnet. Die Schleife für die Anpassung des Druckes läuft solange, bis die Abweichung der Dichten klein genug ist oder bis die maximale Anzahl an Iterationschritten

erreicht ist. Die hier beschriebenen Vorgänge sind nur grob beschrieben. Für den genaueren Hergang insbesondere für Korrektur des Druckes sei hier auf die Arbeit von Solenthaler und Pajarola [SP09] verwiesen.

### 2.3.3 Wärmetransport

Bisher wurde bei den Partikeln nur die Masse, die Geschwindigkeit und der Druck erwähnt. Da es in dieser Arbeit um Verdunstung und Kondensation geht, spielt auch die Temperatur eine große Rolle. Daher muss sie im Partikelsystem mit beachtet werdet. Wenn die Flüssigkeit erhitzt wird, dann soll die Temperatur der Partikel in der Nähe der Wärmequelle steigen und danach die Wärme über die Partikel weitergeleitet werden. Wie diese Wärmeleitung bei SPH funktioniert, wurde auch in Monaghans Zusammenfassung [Mon05] erwähnt. Die Gleichung dafür sieht dann so aus:

$$c_{p,i}\frac{dT_i}{dt} = \sum_j \frac{m_j}{\rho_i \rho_j}\left(\kappa_j + \kappa_i\right)\left(T_i - T_j\right)\frac{\nabla W\left(\mathbf{x} - \mathbf{x}_j, h\right)}{\left|\mathbf{x} - \mathbf{x}_j\right|} + \sum_k Q_k W\left(\mathbf{x} - \mathbf{X}_k, h\right) \tag{2.29}$$

Sie beschreibt die Änderung der Temperatur $T_i$ eines Partikels $i$ über die Zeit. Sie ist unter anderem abhängig von der Wärmekapazität $c_{p,i}$ bei konstantem Druck $p$, den Wärmeleitfähigkeiten $\kappa$ und den Temperaturen $T$. Diese Werte beziehen sich alle auf die Partikel. Befinden sich jedoch Partikel in der Nähe einer Wärmequelle oder Wärmesenke, dann kommen diese noch mit $Q_k$ gewichtet zu dem Abstand ihrer Position $\mathbf{X}_k$. Der Wert von $Q_k$ ist dabei Positiv, falls das Material erhitzt wird und negativ wenn es abgekühlt wird.

### 2.3.4 Glättungsfunktionen

Bei SPH spielt die Glättungsfunktion $W\left(\mathbf{x} - \mathbf{x}_j, h\right)$ eine große Rolle. Von ihrer Wahl hängt vieles ab, da sie bei SPH sehr oft verwendet wird. Änderungen an der Glättungsfunktion beeinflussen das ganze System. Es besteht auch die Möglichkeit verschiedene Funktionen für unterschiedliche Fälle zu verwenden. Müller und andere [MCG03] haben in ihrer Arbeit drei unterschiedliche Funktionen vorgestellt.

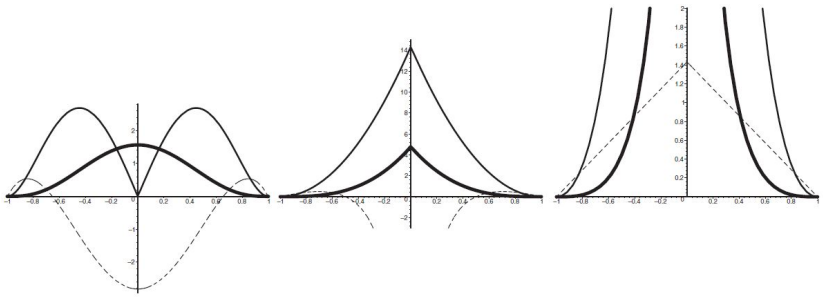


Bild 2.4: Die drei Glättungsfunktionen $W_{poly6}$, $W_{spiky}$ und $W_{viscosity}$. Dicke Linien zeigen die Funktion, dünne Linien die erste Ableitung und gestrichelte Linien die zweite Ableitung [MCG03]

Am häufigsten wird der Kernel $W_{poly6}$ verwendet. Dieser hat den Vorteil, dass dort der Abstand $x$ nur quadratisch vorkommt. Dadurch werden teure Berechnungen der Wurzel erspart.

$$W_{poly6}(\mathbf{x}, h) = \frac{315}{64\pi h^9} \begin{cases} \left(h^2 - x^2\right)^3, & 0 \leq x \leq h \\ 0, & \text{sonst} \end{cases} \tag{2.30}$$

Jedoch hat er Nachteile bei der Bestimmung der Druckkraft. Da die erste Ableitung von dem Kernel $W_{poly6}$ gegen 0 geht, wie im Bild 2.4 zu sehen ist, nimmt die Abstoßung bei sehr nahe beieinander liegenden Partikeln ab. Dadurch verkleben sie und überlagern sich teilweise, was zu höheren Dichten führt. Deshalb wird für die Druckkraft der Kernel $W_{spiky}$ verwenden, wie er schon von Desbrun und Cani[DC96] vorgestellt wurde.

$$W_{spiky}(\mathbf{x}, h) = \frac{15}{\pi h^6} \begin{cases} (h - x)^3, & 0 \leq x \leq h \\ 0, & \text{sonst} \end{cases} \tag{2.31}$$

Der Standard Kernel $W_{poly6}$ kann auch schlecht für die Viskosität verwendet werden. Die Viskosität wird durch eine Reibung verursacht. Je näher zwei Partikel zueinander stehen desto höher soll dieser Wert auch sein. Die zweite Ableitung des Kernels $W_{poly6}$ wird jedoch negativ bei kleinem Abstand (siehe Bild 2.4). Dadurch würde die Viskosität genau in die entgegengesetzte Richtung auf die Partikel wirken. Da die Viskosität für Geschwindigkeiten verwendet wird, bedeutet dies, dass die Partikel eher dazu tendieren sich gegen den Strom zu orientieren anstatt mit ihm mit zu schwimmen. Daher ist es besser dafür einen eigenen Kernel zu verwenden.

$$W_{viscosity}(\mathbf{x}, h) = \frac{15}{2\pi h^3} \begin{cases} -\frac{x^3}{2h^3} + \frac{x^2}{h^2} + \frac{h}{2x} - 1, & 0 \leq x \leq h \\ 0, & \text{sonst} \end{cases} \tag{2.32}$$

Dieser Kernel nimmt einen höheren Wert ein je kleiner der Abstand zweier Partikel ist. Dadurch werden Unterschiede bei der Geschwindigkeit schneller angeglichen anstatt ihn zu vergrößern.

### 2.3.5 Adaption

In manchen Fällen ist es von Vorteil, Partikel mit unterschiedlicher Größe zu simulieren. In Bereichen, wo das Feld relativ konstant ist, können große Partikel verwendet werden, um den Berechnungsvorgang zu beschleunigen. Andererseits sollten kleinere Partikel in Bereichen mit viel Dynamik verwendet werden, um das Feld genauer beschreiben zu können und schneller auf Veränderungen zu reagieren.

In der Vergangenheit wurden schon einige Beispiele für System mit unterschiedlichen Partikelauflösungen vorgestellt. Keiser [Kei06] hat ein Verfahren mit virtuellen Partikeln erarbeitet, indem nur Partikel gleicher Größe miteinander interagieren. Dazwischen gibt es eine Schicht bei der jeweils ein großes Partikel mit mehreren kleinen Partikel verbunden ist. Bei der Methode von Adams und

anderen [APKG07] kommt es nur zu einer Verschmelzung (merging) oder Spaltung (splitting), wenn für die neuen Partikel eine Position gefunden wird, bei der der Druck nicht zu hoch wird. Solenthaler und Gross [SG11] lassen zwei Auflösungen zur gleichen Zeit zu, die sich über künstliche Kräfte austauschen.

Um bei der Änderung der Patikelgröße hohe Drücke zu vermeiden, haben Orthmann und Kolb [OK12] eine Idee vorgestellt, bei der Partikel über die Zeit hinweg an Einfluss gewinnen bzw. verlieren. Dabei wird ein Gewicht $b \in [0, 1]$ eingeführt, welches den Einfluss eines Partikels über die Zeit kontrolliert. Der Einfluss wird in jedem Zeitschritt angepasst mit:

$$b(t + \Delta t) = b(t) + \begin{cases} \Delta b(t), & L \text{ splitted} \\ -\Delta b(t), & H \text{ merged} \end{cases} \tag{2.33}$$

Dabei wird das Gewicht vergrößert, wenn eine Zerteilung von der niedrigeren Auflösung $L$ stattfindet oder es verkleinert sich wenn Partikel von der höheren Auflösung $H$ verschmolzen werden. Der Betrag $\Delta b(t)$, um den das Gewicht angepasst wird, ist abhängig von einer Fehlerschätzung innerhalb des Feldes. Dies ist in der Arbeit von Orthmann und Kolb[OK12] genauer beschrieben.

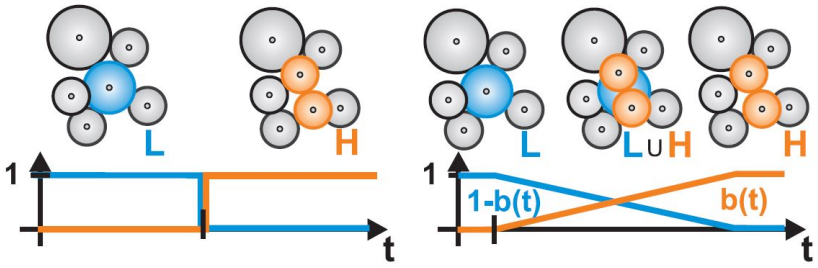



Bild 2.5: Beispiel für die Teilung eines großen Partikels in zwei kleine. Links werden die Teilung in einem Zeitschritt vollzogen. Rechts ist der Prozess auf mehrere Zeitschritte gestreckt mit Anpassung des Einflusses auf das Feld [OK12]

Die Änderungen für diesen Ansatz müssen auch bei der SPH Interpolation in der Formel 2.22 mitberücksichtigt werden. Die Summe muss aufgetrennt werden für Partikel, die sich momentan in einem Set $S$ befinden, in der eine Teilung oder eine Verschmelzung stattfindet, und für Partikel, die sich in keinem der Vorgänge befinden. Danach ergibt sich die Formel [OK12]:

$$\begin{aligned} Q(\mathbf{x}) = &\sum_{j \notin S} Q_j \frac{m_j}{\rho_j} W(\mathbf{x} - \mathbf{x}_j, h) \\ &+ \sum_S \left( b_S \sum_{j \in H_S} Q_j \frac{m_j}{\rho_j} W(\mathbf{x} - \mathbf{x}_j, h) + (1 - b_S) \sum_{j \in L_S} Q_j \frac{m_j}{\rho_j} W(\mathbf{x} - \mathbf{x}_j, h) \right) \end{aligned} \tag{2.34}$$

Dabei sind $S$ die Sets in denen die Auflösung angepasst wird. Darin enthalten sind jeweils die

Gewichte $b_S$ und jeweils Partikel in der höheren Auflösung $H_S$ und in der niedrigeren Auflösung $L_S$. Während man für die Berechnung eines Wertes $Q$ an einer beliebigen Position $\mathbf{x}$ für die Glättungsfunktion $W$ den Einflussradius $h$ der entsprechenden Partikel nehmen kann, bei einer Interaktion zwischen Partikeln unterschiedlicher Größe muss dieser jedoch angepasst werden. Eine gute Annahme ist dabei der Durchschnitt der beiden Radien.

$$W_{ij} = W\left(\mathbf{x} - \mathbf{x}_j, k\frac{h_i + h_j}{2}\right) \tag{2.35}$$

Der durchschnittliche Radius kann noch um $k$ skaliert werden. Orthmann und Kolb [OK12] benutzen dafür zum Beispiel einen Faktor $k = 1{,}25$. Die Anpassung des Einflussradius hat den Vorteil, dass Kräfte zwischen Partikeln unterschiedlicher Größen symmetrisch zueinander sind.

## 2.4 Paralelle Programmierung in CUDA

Die Leistungen von Prozessoren und Grafikkarten nehmen von Jahr zu Jahr zu (Bild 2.6). Während die Entwicklung bei CPUs (Central Processor Unit) eher linear verlief, stieg die Leistung der GPUs (Graphics Processing Unit) exponentiell an. Dies liegt zum Teil an den unterschiedlichen Architekturen. Eine CPU ist dafür ausgelegt einzelne Befehle sehr schnell hintereinander zu verarbeiten. Grafikkarten werden jedoch benötigt, um Bildpunkte (Pixel) auf den Bildschirm zu bringen. Da die Berechnungen dafür voneinander unabhängig sind, können diese parallel verarbeitet werden. Dadurch kann der Datendurchsatz erhöht werden.

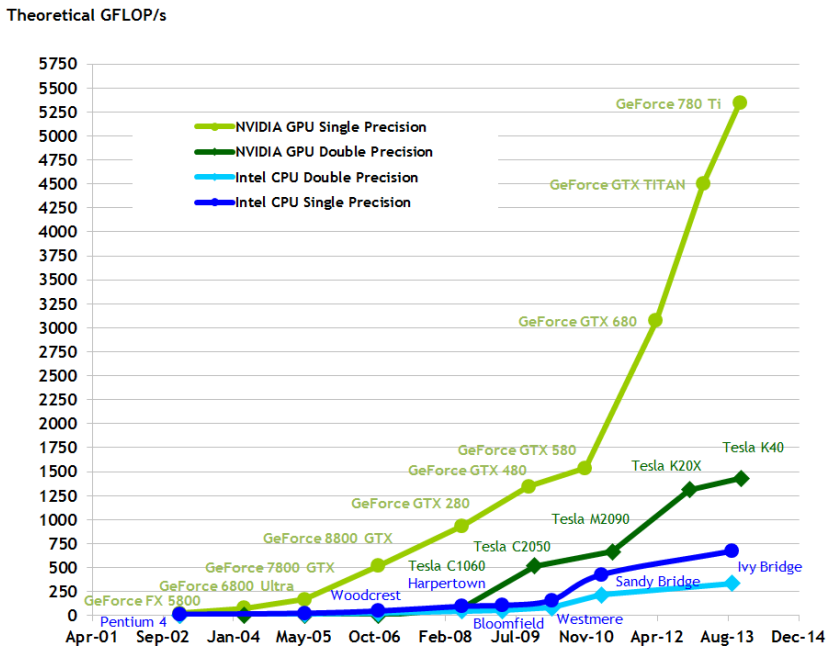



Bild 2.6: Vergleich der Leistungsentwicklung von GPU und CPU [Nvi]

Mit der Zeit haben sich die Grafikkarten soweit entwickelt, dass man damit nicht nur grafische Berechnungen machen kann. Mittlerweile ist die Rechenkraft der parallelen Arbeitsweise für beliebige Funktionen verfügbar. Eine Programmiersprache in der es möglich ist, Programme für eine GPU zu schreiben, nennt sich CUDA (Compute Unified Device Architecture [KWm12]).

Um in CUDA Programme für die GPU zu erstellen, benutzt man sogenannte *Kernel*. Das sind Funktionen die von *Host*, also der CPU, aufgerufen und auf dem *Device*, also der GPU, ausgeführt werden. Da die Grafikkarte nicht auf den Arbeitsspeicher zugreifen kann, müssen die Daten auf denen man arbeitet vor dem Aufruf eines Kernels auf den Speicher der Grafikkarte transferiert werden. Das Kopieren ist relativ langsam im Vergleich zur Berechnung, daher sollten die Daten immer an der Stelle gehalten werden, wo sie gebraucht werden.

Die Kernels werden parallel in *Threads* verarbeitet. Ein Thread ist die kleinste parallele Einheit. Mehrere Threads können in einem *Block* zusammengefasst werden. Und die Blöcke werden in einem *Grid* organisiert. So entsteht eine Struktur, wie sie in Bild 2.7 zu sehen ist.

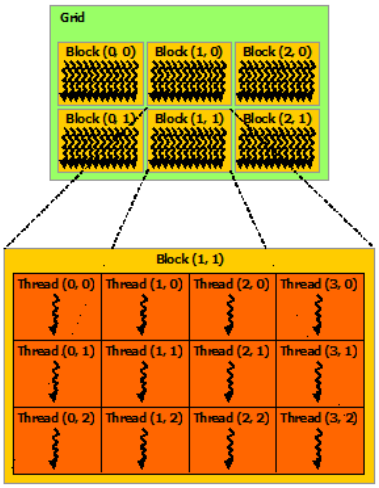



Bild 2.7: Hierarchie von Grids, Blocks und Threads [Nvi]

Eine Grid ist immer einem Kernel zugewiesen, sodass alle Blöcke und Threads darin diesen Kernel verarbeiten. Der Unterschied zwischen Blöcken und Threads liegt bei der Kommunikation. Die Blöcke haben keinerlei Informationen voneinander. Dagegen können Threads innerhalb eines Blockes Daten austauschen. Dafür gibt es den speziellen Speicher *Shared Mamory*. Um zu verhindern, dass Daten von Threads eines Block aus diesem Speicher gelesen werden, bevor etwas reingeschrieben wurde, besteht die Möglichkeiten der Synchronisation. Mit einem Befehl kann man alle Threads innerhalb eines Blockes dazu zwingen an einem bestimmten Punkt anzuhalten. Dann müssen alle Threads, die an dem Punkt sind bevor andere ihn erreichen, warten bis jeder Thread bei dieser Anweisung angekommen ist.

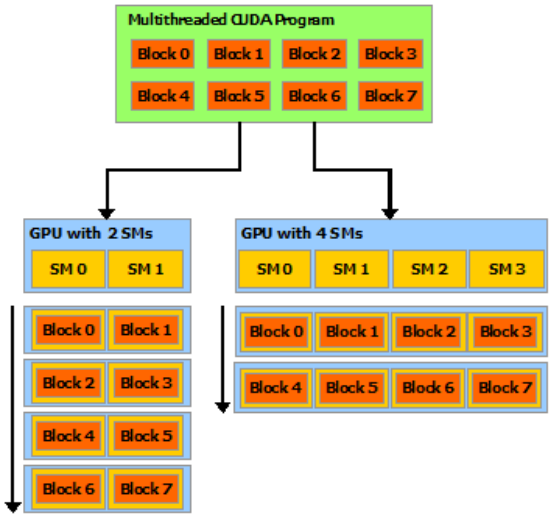



Bild 2.8: Skalierbarkeit durch SMs [Nvi]

Bei der Ausführung eines Kernels werden mehrere Blöcke einem *Streaming Multiprocessor* (SM) zugewiesen. Dieser kümmert sich um die Verteilung der Befehle, welche in Gruppen von 32 Threads ausgeführt werden. Eine solche Gruppe wird als *Warp* bezeichnet. Die Warps haben die Eigenschaft, dass alle Threads innerhalb des Warps immer dieselbe Anweisung ausführen. Sind in der Funktion Verzweigungen vorhanden, so sind nur die Threads des Warps aktiv, die in die Verzweigung rein kommen. Die restlichen Threads bleiben inaktiv bis eine Verzweigung vollständig abgearbeitet wurde. Falls die gesamte Anzahl an zu verarbeitenden Threads kein vielfaches von 32 ist, werden trotzdem immer 32 Thread pro Warp verwendet. Jedoch werden die Warps nicht vollständig ausgelastet. Das Ausführen von einer Anweisung in mehreren Threads zur gleichen Zeit wird als *Single Instruction Multiple Threads* (SIMT) Architektur bezeichnet. Dadurch müssen die Methoden seltener geholt werden, was zu einer höheren Performance führt.

Die SMs bilden eine gute Möglichkeit zur Skalierbarkeit (Bild 2.8). Sie arbeiten die Kernels blockweise ab. Je mehr SMs eine Grafikkarte hat, desto mehr Blöcke können parallel verarbeitet werden. Die Verteilung der Blöcke ist von den Ressourcen der GPU abhängig. Die Kernel müssen bei Verwendung einer anderen Grafikkarte nicht zwangsweise angepasst werden.

# Kapitel 3

# Bekannte Verfahren

Die Simulation von Flüssigkeiten und Gasen ist schon ein altbekanntes Thema, welches schon mehrfach auf Gitter und mit Partikeln realisiert wurde. Mit beiden Varianten wurden Möglichkeiten geschaffen, Stoffe verschiedener Konsistenzen miteinander interagieren zu lassen. Hier wird ein Überblick über bisherige Arbeiten gegeben.

Simulationen mit einem Gitter begannen schon sehr früh. Schon 1965 stellten Harlow und Welch [HW+65] die MAC-Methode vor, bei der Partikel sich mit der Strömung bewegen, welche mit einem Gitter System berechnet wird. Im Jahre 1999 entwickelte Stam [Sta99] eine Möglichkeit Rauch in einem Gitter zu simulieren. Diese wurde von Fedkiw und anderen [FSJ01] durch Verwirbelungen erweitert. Nguyen und andere [NFJ02] entwickelten eine Simulation von Feuer mit Rauch. Eine Wassersimulation mithilfe von Isoflächen wurde von Foster und Fedkiw [FF01] vorgestellt. Harris hat dann im Jahre 2003 in seiner Dissertation [Har03] gezeigt, wie Wolken zum Beispiel für Flugsimulationen erstellt werden können.

Eine Interaktion zwischen einem Fluid und einem Festkörper entwickelten Génevaux und andere [GHD03] mithilfe eines Austausches von Kräften. Carlson und andere [CMT04] haben eine Verbindung geschaffen, indem sie Festkörper als feste Flüssigkeiten betrachtet haben. Im Jahre 2005 stellten Guendelman und andere [GSLF05] eine Möglichkeit vor, wie Flüssigkeiten und Gase mit Stoffen interagieren können, die dünner sind als eine Gitterzelle. Batty und andere [BBB07] entwarfen eine Interaktion von Stoffen mit unregelmäßigen Rändern mit Flüssigkeiten, indem die Projektion des Druckes zur Inkompressibilität mit der Minimierung der kinetischen Energie angepasst wurde.

SPH wird seit 1977 verwendet als es unabhängig von Gingold und Monaghan[GM77] und von Lucy [Luc77] beschrieben wurde. 1992 gab es von Monaghan [Mon92] eine Zusammenfassung davon. Müller und andere [MCG03] haben im Jahre 2003 dieses System benutzt, um damit Flüssigkeiten wie zum Beispiel Wasser zu simulieren. Damit es bei der Simulation keine hohen Verdichtungen gibt, lassen Becker und Teschner [BT07] minimale Abweichungen zur Grunddichte zu auf Kosten von kleineren Zeitschritten. Solenthaler und Pajarola [SP09] hingegen versuchen den Druck des nächsten Zeitschritts zu schätzen und damit Abweichungen der Dichte zu korrigieren. Zwar geschieht die Anpassung mittels Iteration, jedoch können größere Zeitschritte verwendet werden, was zu einer

insgesamt höheren Performance führt.

Bei Simulationen von Flüssigkeiten mittels SPH hat die Größe der Partikel Einfluss auf das Ergebnis. Bei Benutzung von großen Partikel kann die Rechenzeit gesenkt werden und mit kleineren Partikeln können genauere Simulationen erzielt werden. Keiser [Kei06] verbindet die unterschiedliche Größen mithilfe von virtuellen Partikeln, um beide Vorteile zu vereinen. In der Arbeit von Adams und anderen[APKG07] werden die Größen der Partikel angepasst, wenn sich dadurch der Druck nicht zu stark verändert. Solenthaler und Gross [SG11] hingegen lassen die Simulation mit zwei Auflösungen parallel zu und lassen diese mithilfe von Randkräften interagieren. Eine Methode die Partikel mit der Zeit ein- oder ausblenden zu lassen geht auf Orthmann und Kolb [OK12] zurück. Die Rate der Größenänderung wird immer angepasst, sodass keine großen Fehler entstehen.

Interaktionen zwischen verschiedenen Stoffen wurden auch bei SPH erarbeitet. Müller und andere [MSKG05] haben ein Verfahren für die Interaktion zwischen zwei Fluiden vorgestellt. Eine Interaktion mit Luft ist auch möglich. Dabei werden neue Partikel für die Luftphase erstellt, falls dies gebraucht wird. Schechter und Bridson [SB12] simulieren die Luft mit Geisterpartikeln, die die Flüssigkeit umgeben. Dadurch entsteht eine Kraft auf die Oberfläche. Eine Möglichkeit Festkörper mit Flüssigkeiten interagieren zu lassen wurde von Akinci und anderen[AIA+12] und von Shao [SZMTW15] vorgestellt. Dabei werden die Objekte abgetastet und als Partikel dargestellt, die dann Auswirkungen auf das Fluid haben.

Eine Simulation von Verdunstung und Kondensation war bisher nicht bekannt. Die Arbeit die am ehesten dazu passt ist von Ren und anderen[RLY+14]. Dabei werden verschiedene Flüssigkeiten simuliert, die bei Berührung eine chemische Reaktion auslösen, bei der Dampf entsteht. Sowohl die Flüssigkeiten als auch der Dampf und die Luft werden mit Partikeln simuliert. Dazu sind zum einen eine hohe Anzahl an Partikel notwendig und zum anderen werden große Dichteunterschiede verwendet. Dabei ist bekannt, dass durch diese Unterschiede Lücken [SP08] zwischen den Stoffen entstehen können.

# Kapitel 4

# Verdunstung und Kondensation mittels Gitter und SPH

In diesem Kapitel soll das neue Verfahren vorgestellt werden, mit dem es möglich ist Verdunstungen und Kondensation zu simulieren. Das ganze wird mittels eines Gitters und SPH realisiert. Diese beiden Systeme interagieren durch einen Massenaustausch miteinander. Dafür wird zunächst einmal die Idee für die Interaktion geschildert. Danach wird erklärt, wie man die Rate für den Massenaustausch bestimmen kann, bevor der genaue Austausch erklärt wird. In den letzten beiden Abschnitten wird auf die Einzelheiten der beiden Systeme eingegangen, die zu Umsetzung dieses Verfahrens verwendet wurden.

## 4.1 Idee

Verdunstung und Kondensation ist ein kontinuierlicher Prozess. Flüssigkeiten und Gase befinden sich in einem ständigen Austausch. Die Rate mit der dies geschieht ist abhängig vom Druck dieser beiden Zustände. Der Druck ist wiederum abhängig von der Temperatur und der Sättigung des Dampfes. Die Rate der Verdunstung wurde schon mehrfach empirisch untersucht. Die verschiedenen Modelle hat Shah [Sha14] in einer Arbeit analysiert.

Bei der Verdunstung entweicht Masse aus der Flüssigkeit und wird in Dampf umgewandelt. Bei der Kondensation geschieht das Gegenteil. In einem Partikelsystem werden die Massen durch die einzelnen Partikel transportiert. Um also Masse aus der Flüssigkeit zu entfernen, gibt es zwei Möglichkeiten. Als erstes können dafür einzelne Partikel entfernt werden. Dies kann jedoch dazu führen, dass ein zu hoher Massenaustausch stattfindet. Dadurch kann die Luft schnell übersättigt werden und bei der Kondensation können plötzlich hohe Druckunterschiede auftauchen. Besser ist es daher die Masse der Partikel langsam über die Zeit zu verändern. So würden Partikel bei einer Verdunstung über die Zeit hinweg Masse abgeben, bis sie verschwinden und bei einer Kondensation stetig Masse aufnehmen, bis sie ihre volle Masse erreicht haben.

Dieser Prozess, bei dem die Partikel langsam an Einfluss gewinnen oder an Einfluss verlieren,

wurde von Orthmann und Kolb [OK12] für eine andere Problemstellung verwendet. Dabei ging es um die Änderung der Partikelauflösung. Jedoch haben die gezeigt, dass es möglich ist Partikel mit unterschiedlichen Gewichten miteinander interagieren zu lassen und dabei die Stabilität des Partikelsystems trotzdem zu gewährleisten. Diese Methode kann dahingehend abgewandelt werden, dass Partikel langsam an Masse verlieren während sie sich in einem Verdunstungsprozess befinden bzw. an Masse gewinnen, wenn eine Kondensation stattfindet. Die Masse die bei einer Verdunstung entweicht soll in ein Gittersystem überführt werden, um die Gasphase darauf zu simulieren. Kommt es zu einer Kondensation, so gibt das Gitter wieder Masse an die Partikel weiter.

## 4.2 Bestimmung der Rate des Massenaustausches

Bei Simulationen im Gitter und in SPH werden die Massen auf unterschiedliche Weise gespeichert und transportiert. Um den Verlust an Masse vorzubeugen, muss der Austausch kontrolliert werden. Zuvor muss jedoch bestimmt werden, ob ein Austausch stattfinden kann. Wie schon in Kapitel 2.1.5 beschrieben, kann die Rate der Verdunstung oder der Kondensation mit der Formel 2.8 bestimmt werden. Dort wird die Rate als $[\frac{\dot{m}}{A}] = \frac{kg}{m^2 \cdot h}$ angegeben. Für die Simulation hingegen ist die Rate in $\frac{kg}{h}$ interessanter. Dafür muss die Fläche $A$ auf die rechte Seite der Formel 2.8 gebracht werden.

$$\dot{m} = \frac{(30.6 + 32.1u_w)(p_w - p_a)}{\Delta H_v} A \tag{4.1}$$

Für die Verdunstung kann für die Fläche $A$ die Oberfläche der Flüssigkeit angenommen werden. Da die Flüssigkeit mit SPH simuliert wird und die Rate sich je nach vorhandener Temperatur und Luftfeuchte sich lokal unterscheiden kann, ist es hier besser auf die Werte der Partikel zurückzugreifen. So kann dafür die Oberfläche eines Partikels bzw. sein Anteil an der Oberfläche angenommen werden. Wendet man die selbe Formel für die Kondensation an, so ist ebenfalls eine Fläche notwendig. Dafür könnte man zwar zum Beispiel die Oberfläche von festen Objekten, wie beispielsweise einer Wand oder einer Decke nehmen, jedoch würde dann keine Kondensation in freier Luft stattfinden können. Damit werden aber Effekte wie zum Beispiel ein Nebel verhindert. Es ist also besser Kondensationen auch dort zuzulassen, wo es keine Oberfläche in dem Sinne gibt. Da aber ohne eine Oberfläche die Rate dafür nicht bestimmt werden kann, wird in dieser Simulation für diesen Fall auch die Oberfläche eines Partikels verwendet.

Die Rate mit der die Masse ausgetauscht wird, wird den Partikeln zugeordnet und ist ab hier mit $\Phi$ gekennzeichnet. Um die Rate zu bestimmen, werden zwei verschiedene Methoden angewendet. Zur Berechnung der Verdunstung eignet es sich besser die einzelnen Partikel anzuschauen und zu entscheiden, wie schnell dies passiert. Für die Kondensation ist es vorteilhafter sich den Dampf in den einzelnen Gitterzellen zu untersuchen, um auf diesen Prozess zu schließen.

Um vorzubeugen, dass zu viele Partikel existieren, die nicht ihre volle Masse haben, weil eine Verdunstung oder eine Kondensation abgebrochen ist, müssen alle Partikel einen Prozess abschließen, sobald dieser angestoßen ist. Ist einmal festgelegt, dass ein Partikel verdunsten soll, wird es solange

an Masse verlieren bis es vollständig weg ist. Die Rate lässt sich in dieser Zeit noch anpassen, jedoch kann der Vorgang nicht gestoppt und nicht mehr umgekehrt werden. Das Gleiche gilt für den entgegengesetzten Fall. Ferner werden für die Rate minimale und maximale Werte festgelegt, um damit die Anzahl der Partikel zu kontrollieren und zu gewährleisten, dass die Vorgänge nicht zu kurz oder zu lange andauern. Wie die Berechnung der einzelnen Raten stattfindet, wird anhand der folgenden Pseudocodes erläutert.

```
1  for each particle i do
2    j = getCellOf(i);
3    p_water = calculateSaturationPressure(i.Temperature);
4    p_air = calculateSaturationPressure(j.Temperature);
5    Φ = getEvaporationRate(p_water, p_air);
6    if (Φ > minEvaporation)
7      i.EvaporationRate = Φ;
```

Listing 4.1: Pseudocode für die Ermittlung der Verdunstungsrate

In dem Pseudocode 4.1 sind die Schritte für die Berechnung der Rate für die Verdunstung angedeutet. wie schon erwähnt, ist es hierbei besser die Schleife über die Partikel $i$ laufen zu lassen. Bevor man die Rate bestimmt, muss zunächst geschaut werden in welcher Zelle $j$ sich das Partikel momentan befindet, um auf die Eigenschaften dieser Zelle zugreifen zu können. Dann kann der Sättigungsdampfdruck für das Wasser und die Luft bestimmt werden. Hierbei dient das Partikel für die Temperatur des Wassers und die Zelle für die Temperatur der Luft. Aus diesen beiden Drücken wird nach der Formel 2.8 dann die Rate $\Phi$ berechnet. Damit die Partikel sich nicht zu lange in einem Verdunstungsprozess befinden, muss die Rate einen vorher bestimmten Schwellwert übersteigen. Ist dies der Fall, dann wird die errechnete Rate dem Partikel zugewiesen.

```
1  for each Cell j do
2    T_dew = calculatDewPoint(j.Temperature, j.realtiveHumidity);
3    p_dew = calculateSaturationPressure(T_dew);
4    p_air = calculateSaturationPressure(j.Temperature);
5    Φ = getEvaporationRate(p_dew, p_air);
6    Count = 0;
7    for each particle i in Cell j do
8      if (i.evaporationRate < 0)
9        i.evaporationRate = clamp(minCondensation, Φ, maxEvaporation);
10       Count++;
11   if (Count > 0)
12     Φ = Φ - maxCondensation;
13   if (Φ > minCondensation)
14     i = createNewParticle();
15     i.evaporationRate = Φ;
16 }
```

Listing 4.2: Pseudocode für die Ermittlung der Kondensationsrate

Für die Kondensation unterscheidet sich die Methode ein wenig. Die Schritte hierfür sind in dem Pseudocode 4.1 angedeutet. Für diese Methode ist es vorteilhafter eine Schleife über die Gitterzellen $j$ zu durchlaufen. Denn erst wenn es zu einer Kondensation kommt, werden neue Partikel generiert. Würde man hingegen eine Schleife über Partikel machen, könnte man zum Beispiel keine Kondensation in einem Raum erkennen, der voller Dampf ist aber keine Tröpfchen bzw. Partikel enthält. Zu einer Gitterzelle $j$ ist die Temperatur bekannt und die relative Luftfeuchte lässt sich bestimmen. Mit diesen beiden Werten kann auf die Tautemperatur $T_{dew}$ geschlossen werden, ab der der Dampf anfängt zu kondensieren. Danach bestimmt man den Sättigungsdampfdruck für die Tautemperatur und für die aktuelle Temperatur innerhalb der Zelle $j$. Nun lässt sich die Rate für die Kondensation wieder nach der Formel 2.8 bestimmen. Dabei deutet ein negativer Werte auf diesen Vorgang hin. Der negative Wert kann nur entstehen, falls der Sättigungsdampfdruck $p_{dew}$ am Taupunkt niedriger ist als der aktuelle Sättigungsdampfdruck $p_{air}$. Die ist wiederum nur möglich falls der Taupunkt $T_{dew}$ unter der Temperatur der Zelle $j$ liegt, welches nur eintreten kann bei einer Luftfeuchtigkeit von über $100\%$.

Die nächsten Schritte unterscheiden sich nun einiges von der Verdunstung. Die Rate $\Phi$ für die Kondensation wird nur anhand einer Partikeloberfläche bestimmt. Damit mehrere Partikel in einer Zelle kondensieren können, kommt hier der maximale Wert einer Kondensationsrate ins Spiel. Für alle Partikel, die sich innerhalb der Zelle $j$ befinden und in der Kondensationsphase (Zeile 8) sind, wird die Rate $\Phi_i$ angepasst. Diese wird jedoch auf den minimalen bzw. maximalen Wert begrenzt und die Anzahl der Partikel gezählt. In Zeile 11 findet eine Prüfung statt, ob schon Partikel in dieser Zelle kondensieren. Ist dies der Fall wird die Rate $\Phi$ um den maximalen Wert reduziert. Überschreitet danach die Rate immer noch den minimalen Wert der Kondensation, kommen neue Partikel in das System, welche diese Rate erhalten. Es ist zu beachten, dass es nur zwei mögliche Fälle gibt bei denen neue Partikel generiert werden. Entweder es gibt noch keine kondensierenden Partikel innerhalb der Zelle und die minimale Rate wird zum ersten mal überschritten, dann wird das erste Partikel erzeugt. Oder es existieren schon kondensierende Partikel und die Rate übersteigt einen maximalen Wert für die Rate und nochmals den minimalen Wert. Wenn die maximale Rate erreicht wird, dann bekommen alle Partikel mit einer negativen Rate diesen Wert zugewiesen. Die Leistung der vorhandenen Partikel ist somit erschöpft. Wenn nach der Reduktion der minimale Wert noch überschritten ist, bedarf es neuer Partikel um den Dampf aus der Zelle schneller in Wasser zu verwandeln. Das heißt, dass erst neue Partikel generiert werden müssen, wenn festgestellt wird, dass mit den schon vorhandenen Partikeln die Kondensation nicht schnell genug vonstattengehen kann.

## 4.3 Massenaustausch zwischen Gitter und SPH

Ist die Rate für die lokalen Verdunstungen und Kondensationen bekannt, so kann der Massenaustausch zwischen SPH und dem Gitter stattfinden. Die Einheit von $\Phi$ ist dabei $\frac{kg}{h}$. Um auf die Änderung der Masse vom Zeitschritt $t$ zum nächsten Zeitschritt $t+1$ zu schließen, bedarf es einer Multiplikation mit der Zeitdifferenz. Daraus resultiert dann die Formel:

$$m_i^{t+1} = m_i^t - \hat{m}_i \tag{4.2}$$

$$\hat{m}_i = \Delta t \Phi_i \tag{4.3}$$

Die neue Masse $m_i^{t+1}$ des Partikels $i$ zum Zeitschritt $t+1$ setzt sich also aus der Masse $m_i^t$ des jetzigen Zeitschrittes $t$ und dem Anteil der Rate $\Phi_i$ über die verstrichene Zeit $\Delta t$ zusammen. Bei einer Verdunstung ist $\Phi_i > 0$, somit wird mit $\hat{m}_i$ Masse abgezogen. Bei der Kondensation ist $\Phi_i < 0$ und somit kommt Masse hinzu. In dieser Form wurde jedoch vernachlässigt, dass die Masse eines Partikels nicht negativ sein kann und das die Partikelgröße begrenzt ist. Jedes Partikel $i$ kann maximal eine Masse $m_{i,max}$ tragen. Diese beiden Werte dürfen nicht unter oder überschritten werden. Zieht man dies in Betracht, dann muss die Formel angepasst werden in:

$$m_i^{t+1} = min\left(max\left(0, m_i^t - \hat{m}_i\right), m_{i,max}\right) \tag{4.4}$$

Hier wird zunächst mit $max()$ verhindert, dass die Masse negativ wird und dann mit $min()$ auf den Wert $m_{i,max}$ begrenzt. Nun ist also schon mal die Änderung der Masse für die Partikel beschrieben. Da bei einer Verdunstung Masse von SPH zum Gitter transportiert werden soll und bei einer Kondensation eine Gitterzelle Masse an Partikel abgibt, muss diese Veränderung sich auch im Gitter widerspiegeln. Der ganze Raum ist in Gitterzellen unterteilt. Jedes beliebige Partikel $i$ kann somit einer Zelle $j$ zugeordnet werden. Für die Änderung der Masse innerhalb der Zell $j$ müssen also alle Partikel in dieser Zelle mit berücksichtigt werden. Daraus ergibt sich für die Gitterzellen der Austausch mit:

$$m_j^{t+1} = m_j^t + \sum_{i \in j} \left(m_i^t - m_i^{t+1}\right) \tag{4.5}$$

Die neue Masse $m_j^{t+1}$ der Gitterzelle $j$ zum Zeitschritt $t+1$ setzt sich aus der Summe der Massenänderungen $m_i^t - m_i^{t+1}$ aller Partikel $i$ zusammen, die sich zum Zeitpunkt $t$ in der Gitterzelle befinden. Die Änderung hängt wiederum mit der Rate der Verdunstung und der Kondensation $\Phi$ zusammen. Diese zwei Prozesse können auch gleichzeitig in einer Zelle vorherrschen. Dabei können sie sich teilweise gegenseitig auslöschen, was zu einer geringeren absoluten Massenänderung im Gitter führt. Von außen betrachtet sieht das dann wie ein Massenaustausch zwischen Partikeln aus.

Während bei Partikeln die maximale Masse begrenzt ist, wird beim Gitter keine Grenze gesetzt. Sobald es zu einer Übersättigung kommt, erhöhen sich die Raten für die Kondensation und die Masse wird automatisch verdrängt. Jedoch können auch die Zellen keine negativen Massen enthalten. Dafür soll es eine Begrenzung geben. Hier ist es nicht ausreichend wie bei den Partikeln nur die Masse der Zelle begrenzen, weil sonst die Änderung der Masse von Partikeln und des Gitters unterschiedlich ist. Tritt dagegen so ein Fall ein, dann müssen die beiden Funktionen 4.4 und 4.21 angepasst werden. Diese sind über die Änderung $\hat{m}_i$ der Masse gekoppelt. Daher macht es Sinn diesen Wert zu ändern,

bevor ein Austausch stattfindet. Mit der Anpassung sieht dieser dann folgendermaßen aus:

$$\hat{m}_i = \begin{cases} \Delta t \Phi_i \cdot \frac{-m_j^t}{\sum_{i \in j} \Delta t \Phi_i}, & \sum_{i \in j} \Delta t \Phi_i < -m_j^t \\ \Delta t \Phi_i, & sonst \end{cases} \tag{4.6}$$

Dabei wird also $\hat{m}_i$ nur angepasst, sobald die Gesamtmasse, die einer Zelle abgezogen werden soll, die Masse der Zelle überschreitet. Es darf nicht vergessen werden, dass bei einer Kondensation $\Phi_i < 0$ ist. Daher tritt der Fall erst ein, wenn $\sum_{i \in j} \Delta t \Phi_i < -m_j^t$ gilt. Um negative Massen im Gitter zu vermeiden, werden nun die Raten für alle Partikel $i$, die sich in der Zelle $j$ befinden, mit einem Faktor angepasst. Dieser verändert die Raten soweit, dass wenn man mit diesen weiter rechnet, dann am Schluss genau eine Masse von Null rauskommt.

$$\begin{aligned} m_j^{t+1} &= m_j^t + \sum_{i \in j} \hat{m}_i \\ &= m_j^t + \sum_{i \in j} \left( \Delta t \Phi_i \cdot \frac{-m_j^t}{\sum_{i \in j} \Delta t \Phi_i} \right) \\ &= m_j^t + \frac{-m_j^t \sum_{i \in j} \Delta t \Phi_i}{\sum_{i \in j} \Delta t \Phi_i} \\ &= m_j^t - m_j^t \\ &= 0 \end{aligned}$$

Die Anpassung des Massenaustausches hat Einfluss auf die Reihenfolge der Ausführung der beiden Funktionen. Dieser kann nämlich nicht vollzogen werden, solange nicht überprüft wurde, ob Austausch für das Gitter verändert werden muss. Daher wird die Änderung der Massen für die Partikel erst danach ausgeführt. Die Aktionen sehen dann wie folgt aus:

1. Berechnung der Rate für Verdunstung und Kondensation
2. Anpassung des Massenaustausches für das Gitter
3. Ausführung des Massenaustausch zwischen Gitter und SPH

Damit ist sichergestellt, dass die Massen ohne Verlust ausgetauscht werden können. Alles was aus dem Gitter entfernt wird, kommt in das SPH System und alles was die Partikel an Masse verlieren, wird den jeweiligen Zellen wieder gutgeschrieben.

## 4.4 Anpassungen am Gitter

Das Gitter dient als Speicherung für den Dampf, der bei Verdunstung entsteht. Hierbei reicht es aus, wenn die Zellen relativ groß sind. Es soll die Verwendung von SPH für den Dampf ersetzen. Würde man den Simulationsraum nur mit Partikeln füllen, wäre eine hohe Anzahl an Partikel notwendig. Benutzt man jedoch ein grobes Gitter, so reichen nur wenige Zellen aus, um denselben Raum zu

füllen. Je größer die Zellen im Vergleich zu der Partikelgröße für das Gitter angenommen werden, desto mehr Zeit kann bei der Berechnung eingespart werden. Jedoch sollte das Gitter auch nicht zu grob aufgelöst sein, da sonst weniger Effekte damit sichtbar werden.

Auch wenn das Gitter in erster Linie für die Speicherung des Dampfes verwendet wird, soll damit trotzdem eine Gasphase simuliert werden. Die Masse wird dabei, wie im vorherigem Kapitel beschrieben, den Zellen zugeführt bzw. entfernt. Die Simulation des Dampfes benutzt dabei die Euler-Gleichungen 2.3 und 2.3 aus Kapitel 2.1.2, da die Viskosität vernachlässigbar ist. Für die Advektion wird der veränderte Ansatz von Lentine und anderen [LGF11] verwendet, wie er in Kapitel 2.2.1 beschrieben ist. Die Temperatur soll auch mit auf dem Gitter simuliert werden. Ohne sie kann keine Verdunstung oder Kondensation stattfinden.

### 4.4.1 Freies Volumen in Gitterzellen

Die beiden Systeme SPH und Gitter werden parallel simuliert. Es ist jedoch klar, dass an einer Position nicht beide Systeme gleichzeitig aktiv sein können. Es kann an einer Postion entweder Flüssigkeit existieren oder Dampf bzw. Luft, aber nicht beides zur gleichen Zeit. Daher müssen die beiden Systeme aufeinander abgestimmt werden. Die Zellen des Gitters haben immer eine feste Position und bewegen sich nicht. Die Partikel dagegen können sich immer frei im Raum bewegen. Befinden sich Partikel in einer bestimmten Gitterzelle, so nehmen sie dort einen bestimmten Platz weg. Damit wird das Volumen, welches für das Gas verwendet werden kann, verkleinert. Dies muss bei der Simulation mit berücksichtigt werden.

Eine Änderung des freien Volumens in den Zellen hat Auswirkungen auf die Simulation. Wird das Volumen kleiner, so erhöht sich der Druck und die Dichte darin und das Gas muss aus der Zelle raus strömen. Erhöht sich dagegen das zur Verfügung stehende Volumen einer Zelle, so entsteht Platz, welcher wiederum mit Gas gefüllt werden kann. Eine Zelle $j$ kann maximal ein Volumen von $V_{j,max} = \Delta x \cdot \Delta y \cdot \Delta z$ haben. Dabei ist $\Delta x$ die Breite, $\Delta y$ die Tiefe und $\Delta z$ die Höhe der einzelnen Zellen. Falls sich Partikel innerhalb der Zelle befinden, müssen deren Volumen abgezogen werden. Der zur Verfügung stehende Raum einer Zelle ergibt sich dann zu:

$$V_j = max\left(V_{j,min}, V_{j,max} - \sum_{i \in j} V_i\right) \tag{4.7}$$

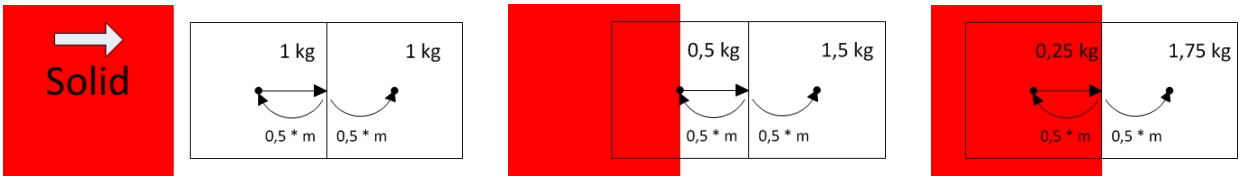
Das nutzbare Volumen $V_j$ einer Zelle $j$ ist das Volumen, welches nach Abzug der Volumen $V_i$ der Partikel $i$, die sich innerhalb der Zelle befinden, übrig bleibt. Da sich die Partikel leicht überschneiden können, besteht die Möglichkeit, dass die Summe aller einzelnen Volumen der Partikel einer Zelle das Volumen dieser Zelle überschreitet. Deshalb wird eine Begrenzung $V_{j,min}$ eingeführt, damit das restliche Volumen einer Zelle nicht negativ werden kann.

Die Anzahl der Zellen im Gitter bleibt die ganze Simulation über konstant. Selbst wenn einige Zellen voller Partikel sind, also nur die Flüssigkeit enthalten, werden sie nicht aus dem System genommen. Solchen Zellen wird ein minimales Volumen $V_{j,min}$ zugewiesen. Dies dient unter anderem

dazu, restliche Masse aus einer Zelle zu transportieren, die zu schnell mit Flüssigkeit gefüllt wurde. Genaueres hierzu wird im Verlauf dieses Kapitels erklärt.

### 4.4.2 Diffusion zur Verteilung der Masse

Der Transport der Masse innerhalb des Gitters wird normalerweise nur mit der Advektion realisiert. Dies genügt jedoch nicht immer, um Masse aus einer Zelle schnell genug raus zu transportieren, bevor der Raum der Zell gefüllt ist. Man kann dies an einem einfachen Beispiel sehen. Nehmen wir an, es gibt zwei Zellen mit jeweils einer Masse $m = 1kg$ und einer Breite $\Delta x = 1m$. Nun wird eine Zelle langsam von einem festen Objekt von einer Seite aus verdeckt. Angenommen das Objekt hat eine Geschwindigkeit $v = 0{,}5\frac{m}{s}$. Dann reichen zwei Sekunden aus, um die eine Zelle vollständig zu füllen. Nehmen wir ferner an, dass sich das Gas innerhalb der ersten Zelle mit der gleichen Geschwindigkeit transportiert wird. Hier wird einfacher halber eine Semi-Lagrange Forward Advektion angenommen. Wenn die Masse vom Mittelpunkt der Zelle transportiert wird, dann landet sie genau auf der Grenze zwischen den zwei Zellen. Durch Interpolation kommt heraus, dass eine Hälfte der Masse wieder in der gleichen Zelle landet und die andere Hälfte in der zweiten Zelle. Dies ist für die ersten zwei Zeitschritte auf folgender Skizze verdeutlicht:




(a) (b) (c)

Bild 4.1: Es wird nicht genügend Masse aus der Zelle verdrängt

Bei dem zweiten Zeitschritt passiert genau das Gleiche. Der Unterschied ist aber, dass diesmal weniger Masse transportiert wird. Von der Hälfte der Masse, die im ersten Zeitschritt übrig geblieben ist, bleibt wieder die Hälfte in der ersten Zelle übrig. Demnach enthält die verdeckte Zelle immer noch eine Masse $m = 0{,}25kg$, obwohl da kein Platz mehr übrig ist. Dieses Verhalten lässt sich mit einer allgemeinen Formel beschreiben:

$$m_{Rest}(t) = \left(1 - \frac{1}{t}\right)^t \tag{4.8}$$

Setzt man in diese Formel die Anzahl der Zeitschritte ein, die nötig sind, um eine Zelle gleichmäßig zu füllen, bekommt man den Anteil raus, der danach noch in der Zelle verbleibt. Für das obige Beispiel ergibt sich $m_{Rest}(2) = \left(1 - \frac{1}{2}\right)^2 = 0{,}25$, welches mit den Angaben übereinstimmt. Bei Simulationen hat die Größe des Zeitschrittes Einfluss auf die Stabilität. Bei kleineren Zeitschritten kommen genauere Ergebnisse raus. Demzufolge sollte weniger Masse übrig bleiben, je größer die

Anzahl der Schritte ist, die Zelle zu füllen. Nehmen wir als extremes Beispiel an, dass unendlich viel Schritte nötig sind. Dafür wird der Grenzwert für $t \to \infty$ gebildet.

$$\lim_{t\to\infty}\left(\left(1-\frac{1}{t}\right)^t\right) = e^{-1} \approx 0{,}3679 \tag{4.9}$$

Wie man sieht, verschlechtert sich der Wert noch weiter. Je höher die Anzahl der Zeitschritte ist, desto mehr Masse bleibt in der Zelle übrig. Eine höhere Auflösung der Zellen würde dem zwar entgegenwirken, jedoch wird dafür mehr Rechenzeit benötigt. Deshalb wird bei der Simulation des Gitters zusätzlich zu der Advektion auch eine Diffusion durchgeführt, um zu vermeiden, dass in den Zellen restliche Masse zurück bleibt. Die Diffusion dient dazu, die Massen aller Zellen so anzugleichen, sodass überall eine möglichst gleiche Dichte herrscht. Die allgemeine Formel dafür lautet:

$$\frac{\partial Q}{\partial t} = \alpha \frac{\partial^2 Q}{\partial x^2} \tag{4.10}$$

wobei $Q$ der Wert ist, der gleichmäßig verteilt werden soll und $\alpha$ ist eine Diffusionskonstante, die die Schnelligkeit der Verteilung reguliert. Eine bewährte Methode diese Gleichung zu lösen, stellt das Crank-Nicolson-Verfahren [CN47] dar. In kurzer Form sieht es so aus

$$\frac{Q_j^{t+1} - Q_j^t}{\Delta t} = \frac{\alpha}{2\left(\Delta x\right)^2}\left(\left(Q_{j+1}^{t+1} - 2Q_j^{t+1} + Q_{j-1}^{t+1}\right) + \left(Q_{j+1}^t - 2Q_j^t + Q_{j-1}^t\right)\right) \tag{4.11}$$

Auf der linken Seite steht die Änderung von $Q$ von dem Zeitschritt $t$ zum Zeitschritt $t+1$ bei einer vergangenen Zeit von $\Delta t$. Auf der rechten Seite sieht man deutlich die zwei 3-Punkte-Sterne für den eindimensionalen Fall. Im zweidimensional Fall entsteht dabei ein 5-Punkte-Stern und im dreidimensionalen ein 7-Punkt-Stern. Es wird deutlich, dass wenn die Werte $Q_{j-1}$, $Q_j$, und $Q_{j+1}$ alle gleich sind, es keine Änderung des Wertes $Q_j$ gibt. Um mit der Gleichung 4.11 die Änderung zu bestimmen, müssen jedoch schon die Werte $Q^{t+1}$ von nächsten Zeitschritt bekannt sein. Diese sollen meisten aber bestimmt werden. Deshalb ist es von Vorteil die Gleichung etwas umzustellen.

$$-\mu Q_{j+1}^{t+1} + \left(1+2\mu\right) Q_j^{t+1} - \mu Q_{j-1}^{t+1} = \mu Q_{j+1}^t + \left(1-2\mu\right) Q_j^t + \mu Q_{j-1}^t \tag{4.12}$$

Nun sind alle Werte $Q^{t+1}$, die man bestimmen muss auf der linken Seite und alle Werte $Q^t$, die zum Zeitpunkt $t$ bekannt sind auf der rechten Seite. Für eine verkürzte Schreibweise wurde hier $\mu = \frac{\alpha}{2(\Delta x)^2}$ definiert. Diese Gleichung ist nur ein Beispiel für die Berechnung des Wertes $Q_j^{t+1}$. Um die Werte für alle $j$ zu berechnen, kann die Form $L\vec{Q}^{t+1} = R\vec{Q}^t$ verwendet werden. Hierbei sind alle Daten in den Vektoren zusammengefasst. Die Matrizen $L$ und $R$ enthalten die Koeffizienten der Gleichung 4.12. In dieser Form können alle Werte bestimmt werden. Hierfür müssen nur die Matrizen $L$ und $R$ aufgestellt werden. Danach multipliziert man $R$ mit den alten Daten, um die rechte Seite komplett zu bestimmen. Um auf die neuen Werte zu schließen, muss dann nur noch die Matrix $L$ auf

die rechte Seite gebracht werden.

### 4.4.3 Blockierung des Massenaustausches

Bei der Diffusion, die bisher vorgestellt wurde, wird davon ausgegangen, dass zwischen allen Zellen ein normaler Austausch stattfinden kann. Bei einer Simulation, wie sie hier verwendet wird, entstehen jedoch oft Situationen in denen Zellen mit Partikeln befüllt sind und somit keine Masse aufnehmen können. Für eine Verhinderung eines Austausches zweier Zellen reicht es sogar aus ,wenn die Fläche zwischen den Zellen durch Partikel blockiert ist, wie es in Bild 4.2 skizziert ist. Um solche Situationen zu erkennen, müssen die Ränder jeder Zelle untersucht werden. Hier reicht es aus, nur die Partikel zu beobachten, die sich nah genug an einer Zellgrenze, also innerhalb einer Entfernung eines Toleranzabstandes davon befinden. Sind genügend Partikel in diesem Bereich zu finden, kann der Austausch blockiert werden.

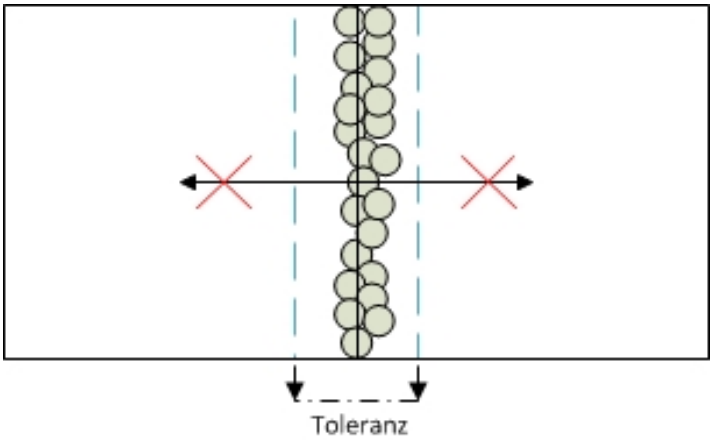



Bild 4.2: Zellkante ist mit Partikeln blockiert. Kein Austausch möglich.

Um eine genauere Vorhersage zu treffen, ob der Übergang verdeckt ist, kann auch eine andere Möglichkeit verwendet werden. Und zwar lässt sich für jedes Partikel bestimmen, ob es eine Überschneidung mit einer Grenzfläche zwischen zwei Zellen gibt. Ist so eine Situation gefunden, fehlt nicht mehr viel, um die genaue Schnittfläche zu bestimmen. Dafür bestimmt man zunächst den Schnittradius mit dem Satz des Pythagoras mit dem Radius des Partikels und des Abstandes des Partikels zu der Fläche.

$$intersectRadius^2 = particleRadius^2 - distance^2$$

Hat man den Schnittradius bestimmt, zieht man es ähnlich wie beim Volumen von der Gesamtfläche der Grenze zwischen zwei Zellen ab. Existieren so viel Partikel, dass ihre gesamte Schnittflächen die ganze Grenzfläche belegen, dann ist diese blockiert. Mit Symbolen sieht diese Beschreibung dann so aus:

$$A_{jk,blocked} = \begin{cases} True, & \left(A_{jk} - \sum_{i \cap A_{jk}} \pi \cdot R^2_{i \cap A_{ij}}\right) \leq 0 \\ False, & sonst \end{cases} \tag{4.13}$$

Dabei ist $A_{ij}$ die Fläche zwischen den Zellen $j$ und $k$. $A_{jk,blocked}$ gibt an, ob diese blockiert ist oder nicht. Das ist dann der Fall, wenn die Summe aller Schnittflächen mit Partikeln die Fläche $A_{jk}$ zwischen den beiden Zellen überschreitet. Dabei werden nur die Partikel $i \cap A_{ij}$ beachtet, die sich mit der Fläche schneiden. Mit dem Schnittradius $R_{i \cap A_{ij}}$ berechnet sich dann die Schnittfläche. Bei dieser Gleichung ist auch wieder zu beachten, dass die Partikel oft sehr nah beieinander liegen und sich teilweise überschneiden. Deshalb ist es möglich, dass die Summe aller Schnittflächen die Gesamtfläche übersteigt. Aus diesem Grund ist hier die Prüfung des Restes nicht genau auf Null, sondern auf $\leq 0$.

Wenn der Austausch einiger Zellen blockiert ist, dann muss auch die Diffusion entsprechend angepasst werden. Bei einer gesperrten Fläche kann keine Masse von einer Zelle zur anderen gelangen. Um dies zu verhindern, reicht es aus die Koeffizienten bei der Diffusion anzupassen. Diese hängen mit der Anzahl der Nachbarzellen zusammen, mit denen Masse ausgetauscht werden kann. Bei dem eindimensionalen Beispiel mit der Formel 4.11 hat jede Zelle zwei Nachbarn. Jeder davon erhält den Koeffizienten 1. Die Zelle für die die Diffusion bestimmt wird, bekommt die Anzahl an Nachbarn als Koeffizienten. In diesem Beispiel also eine 2. Im zweidimensionalem hat jede Zelle 4 Nachbarn und im dreidimensionalem 6 Nachbarn. Dementsprechend werden die Koeffizienten angepasst, wobei die Nachbarzellen immer den gleichen Koeffizienten haben.

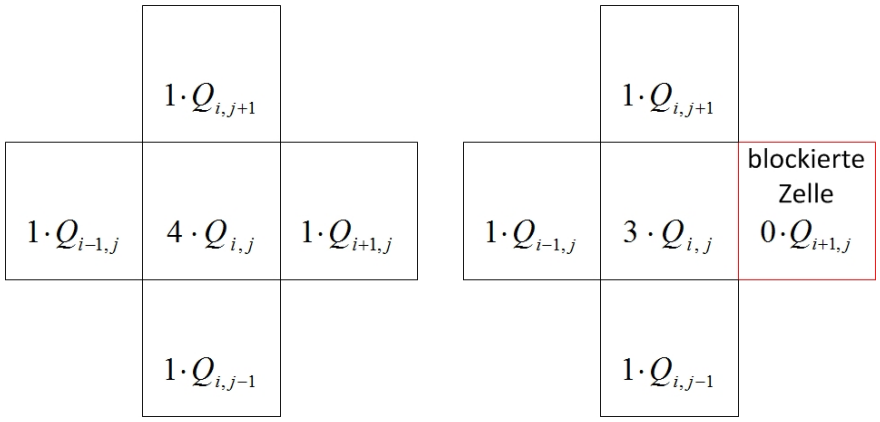



Bild 4.3: Bei einer blockierten Zelle wird aus einem 5-Punkte-Stern ein 4-Punkte-Stern

Ist eine Zelle blockiert, wie in Bild 4.3 skizziert ist, so werden die Koeffizienten an die Nachbarn angepasst. Normalerweise hat eine Zelle im zweidimensionalem 4 Nachbarn und dementsprechend denselben Koeffizienten. Besitzen zum Beispiel alle Zellen den gleichen Wert $Q$, so gibt es durch eine

Diffusion keine Änderung, denn durch den Laplace-Operator ergibt sich $4 \cdot Q_{i,j} - Q_{i-1,j} - Q_{i+1,j} - Q_{i,j-1} - Q_{i,j+1} = 0$.

Falls eine Nachbarzelle blockiert ist, bleiben nur noch 3 Zellen übrig, zwischen denen Masse ausgetauscht wird. Damit bei gleicher Dichte der Austausch wieder ausgeglichen wird, muss der Koeffizient der mittleren Zelle verringert werden. Denn bei 3 Nachbarn ergibt sich bei gleichen Werten $Q$ eine Änderung von $3 \cdot Q_{i,j} - Q_{i-1,j} - Q_{i+1,j} - Q_{i,j-1} - 0 \cdot Q_{i,j+1} = 0$. Es wird also wie erwartet keine Masse zwischen diesen Zellen transferiert,

Mit dieser Anpassung bei der Diffusion können also blockierte bzw. nicht erreichbare Zellen in diesem Vorgang mit berücksichtigt werden. Es kann jedoch erst ausgeführt werden, nachdem die Nachbarschadschaften der Gitterzellen bekannt sind. Die Reihenfolge sieht also so aus, dass zunächst die freien Volumen der einzelnen Zellen bestimmt wird. Dabei kann schon die Überprüfung stattfinden, ob irgendwelche Flächen versperrt werden. Danach können die Nachbarschaften für die Diffusion verwendet werden.

### 4.4.4 Angepasste Berechnung des Drucks

Das berechnete Volumen in den Gitterzellen und die blockierten Flächen können auch noch für eine andere Funktion benutzt werden. Im Kapitel 2.2.2 wurde schon über die Herstellung der Inkompressibilität geschrieben, womit das Strömungsfeld so ausgeglichen wird, dass es keine Quellen und Senken gibt. In der Formel 2.18 für die Berechnung des entsprechenden Drucks kommt die Dichte nur als eine Konstante vor. Da es bei Änderung des Volumens innerhalb der Zellen und beim Austausch von Masse zwischen SPH und dem Gitter zu Veränderungen der Dichten kommen kann, sollte eine etwas abgewandelte Form davon verwendet werden.

Batty und andere [BBB07] haben in ihrer Arbeit eine Methode vorgestellt, welche hierfür passt. Sie ist dafür gedacht feste Objekte mit unregelmäßiger Oberfläche mit Flüssigkeiten in einem Gittersystem zu verbinden. Dies machen sie durch die Minimierung der gesamten kinetischen Energie. Dabei werden die Massen an den Zellgrenzen in Betracht gezogen. Hier sei nur die daraus resultierende Formel gegeben.

$$\frac{\Delta t}{\Delta x^2}\begin{bmatrix}\left(\frac{V_{i+1/2,j,k}}{\rho_{i+1/2,j,k}} + \frac{V_{i-1/2,j,k}}{\rho_{i-1/2,j,k}} + \frac{V_{i,j+1/2,k}}{\rho_{i,j+1/2,k}} + \frac{V_{i,j-1/2,k}}{\rho_{i,j-1/2,k}} + \frac{V_{i,j,k+1/2}}{\rho_{i,j,k+1/2}} + \frac{V_{i,j,k-1/2}}{\rho_{i,j,k-1/2}}\right) p_{i,j,k} \\ -\frac{V_{i+1/2,j,k}}{\rho_{i+1/2,j,k}} p_{i+1,j,k} - \frac{V_{i-1/2,j,k}}{\rho_{i-1/2,j,k}} p_{i-1,j,k} \\ -\frac{V_{i,j+1/2,k}}{\rho_{i,j+1/2,k}} p_{i,j+1,k} - \frac{V_{i,j-1/2,k}}{\rho_{i,j-1/2,k}} p_{i,j-1,k} \\ -\frac{V_{i,j,k+1/2}}{\rho_{i,j,k+1/2}} p_{i,j,k+1} - \frac{V_{i,j,k-1/2}}{\rho_{i,j,k-1/2}} p_{i,j,k-1}\end{bmatrix} = -\frac{1}{\Delta x}\begin{bmatrix}V_{i+1/2,j,k} u_{i+1/2,j,k} - V_{i-1/2,j,k} u_{i-1/2,j,k} \\ +V_{i,j+1/2,k} v_{i,j+1/2,k} - V_{i,j-1/2,k} v_{i,j-1/2,k} \\ +V_{i,j,k+1/2} w_{i,j,k+1/2} - V_{i,j,k-1/2} w_{i,j,k-1/2}\end{bmatrix} \tag{4.14}$$

Die Gleichung hat sehr viel Ähnlichkeit mit der Gleichung 2.18 aus dem Kapitel 2.2.2. Auf der rechten Seite steht hier wieder die Divergenz der Zelle mit den Koordinaten $(i, j, k)$. Diesmal ist es jedoch zusätzlich skaliert mit dem Volumen an den Zellgrenzen. Hierfür lässt sich der Durchschnitt der beiden angrenzenden Zellen nehmen, also zum Beispiel $V_{i+1/2,j,k} = \frac{1}{2}(V_{i,j,k} + V_{i+1,j,k})$. Auf der linken Seite sieht man den 7-Punkte-Stern und die zu berechnenden Drücke. Hier sind wieder die Volumen der Zellen mit eingeschlossen und zusätzlich die Dichten an den Grenzen. Anders als in der Gleichung aus Kapitel 2.2.2 ist hier jedoch die Dichte mit in die Klammer reingezogen. Dafür wird nun auch kein konstanter Werte mehr genommen, sondern die jeweiligen Werte an den Zellgrenzen.

Mit der veränderten Variante lassen sich also die errechneten Volumen aus den Zellen für die Berechnung der Drücke verwendet. Falls es Zellen gibt, deren Raum mit Partikeln gefüllt ist und dementsprechend sich ihr Volumen verringert, wird dies in dieser Form berücksichtigt. Die dadurch resultierende höhere Dichte in der Zelle geht auch in die Berechnung der Drücke mit ein. Bei dieser Formel ist jedoch auch zu beachten, dass wie bei der Diffusion, einige Zellübergänge blockiert sein können. Hier kann dann die gleiche Anpassung vorgenommen, indem die Nachbarschaften mitberücksichtigt werden.

Für die Bestimmung der Drücke aller Zellen kann die Gleichung in eine Form der Art $A\vec{x} = \vec{b}$ gebracht werden. Die Einträge in der Matrix sind dann die Koeffizienten der Drücke auf der linken Seiten der Gleichung 4.14. Um auf den Druck zu schließen, muss die Matrix auf die linke Seite gebracht werden. Dafür wird in dieser Arbeit die Jacobi-Iteration verwendet. Mit diesem Verfahren nähert man sich mit jeder Iteration der richtigen Lösung an. Wenn ein vorher bestimmter Restfehler oder die maximal festgelegte Anzahl an Schritten erreicht ist, wird an der Stelle abgebrochen. Diese Methode hat den Vorteil, dass die maximale Rechenzeit begrenzt werden kann. Jedoch sollte man trotzdem die Schrittzahl nicht zu gering ansetzen, um noch eine gewisse Genauigkeit einhalten zu können.

### 4.4.5 Advektion der Temperatur

Zur abschließenden Beschreibung der Simulation mit dem Gittersystem fehlt noch die Einbeziehung der Temperatur. Diese ist notwendig für die Berechnung des Strömungsfeldes und für die Bestimmung der Rate der Verdunstung und der Kondensation. Die Temperaturen an den Grenzen des zu simulierenden Raumes können in die Diffusion mit dem Crank-Nicolson-Verfahren als Randbedingungen integriert werden. Die Advektion hingegen wird hier an die Eigenschaften der Temperatur angepasst.

Der Transport der Temperatur im Gitter kann nicht auf die gleiche Weise durchgeführt werden wie für die Masse, welche in Kapitel 2.2.1 beschrieben wurde. Die Bewegung des Dampfes hängt nur von dem Strömungsfeld ab. Dagegen ist die Temperatur auch von der Menge der Masse abhängig. Falls zum Beispiel eine Zelle nur eine sehr geringe Menge an Dampf mit hoher Temperatur beinhaltet und dieser in einem Zeitschritt aus der Zelle rausströmt, kann natürlich die Temperatur nicht plötzlich auf 0 Grad sinken. Deshalb ist es hier vorteilhafter die thermische Energie $E_{th}$, welche mit

$$E_{th} = mc \int_0^{T_1} dT = mc\frac{1}{2}T_1^2 \tag{4.15}$$

definiert ist, für die Advektion zu verwenden. Diese berechnet sich mittels der Masse $m$, der spezifische Wärmekapazität $c$ und der aktuellen Temperatur $T_1$ . Für den Transport kann dann die selbe Methode wie in Kapitel 2.2.1 benutzt werden. Da nicht nur der Dampf sondern auch die Luft selber eine bestimmte Temperatur besitzt, genügt es nicht hier nur die Masse des Dampfes zu verwenden. Die Luft selber sollte hier auch mit einbezogen sein. Dafür kann die Variante genutzt werden wie bei der Auftriebskraft in Kapitel 2.1.3. Laut Bridson [Bri08] lässt sich die kombinierte Dichte bestimmen mit:

$$\rho = \rho_{air}\left(1 + s\frac{\rho_{fluid} - \rho_{air}}{\rho_{air}}\right) \tag{4.16}$$

Darin ist $\rho_{air}$ die Dichte der Luft, $\rho_{fliud}$ die Dichte des Dampfes und $s$ die Konzentration des Dampfes, welches in diesem Fall der relativen Luftfeuchtigkeit entspricht. Zusammen mit dem vorhandenen Volumen nach Gleichung 4.7 kann damit die gesamte Masse der Luft und des Dampfes in der Zelle bestimmen. Diese wird dann für den Transport der thermischen Energie verwendet.

Für die Advektion der Temperatur muss also zunächst die kombinierte Masse der Zellen bestimmt werden, um damit die thermische Energie zu erhalten. Diese wird auf die gleiche Weise transportiert wie die Masse des Dampfes. Nach dem Verschieben der thermischen Energie lässt sich die Temperatur der Zelle raus finden, indem die Gleichung 4.15 nach der Temperatur umgeformt wird. Für die Masse $m$ darf jedoch nicht wieder der Dampf selber benutzt werden, sondern die Kombination aus Luft und Dampf. Diese lässt man am besten in gleichen Schritt mit der thermischen Energie transportieren, um dann auf die neue Masse und damit auf die neue Temperatur schließen zu können.

## 4.5 Anpassungen an SPH

Mit SPH wird die Flüssigkeit bei dieser Arbeit simuliert. Die Partikel bestimmen die Eigenschaften des flüssigen Stoffen. Dazu gehören unter anderem die Position, die Geschwindigkeit, die Masse und die Temperatur. Dieses System wird dabei nach der Art und Weise umgesetzt, wie es in Kapitel 2.3 beschrieben ist. Da jedoch durch die Verdunstung und Kondensation ein Massenaustausch zwischen SPH und einem Gitter stattfindet, gibt es kleine Anpassungen. Mit SPH werden normalerweise Flüssigkeiten simuliert bei denen die Masse konstant ist. Bei dieser Arbeit gibt es jedoch durch Verdunstung und Kondensation ständig Veränderung, auf die die Flüssigkeit reagieren muss.

In Kapitel 4.2 wurde schon darauf eingegangen, wie Masse zwischen den Systemen ausgetauscht wird. Da die Ideen dieser Arbeit möglichst in andere System integrieren zu können, sollten die absoluten Massen der Partikel nicht direkt angepasst werden. Es bietet sich eher an, ein Gewicht zu verwenden, ähnlich wie von Orthmann und Kolb [OK12] vorgestellt. Diese gibt den Anteil der Masse an, welches einem Partikel zugeordnet ist. Das Gewicht von einem Partikel $i$ sei hier als $\omega_i \in [0, 1]$

deklariert. Dadurch, dass es einen Anteil der Masse darstellt, lässt es sich dementsprechend folgendermaßen beschreiben:

$$\omega_i = \frac{m_i}{m_{i,max}} \tag{4.17}$$

Dabei ist $m_i$ die aktuelle Masse des Partikels $i$ und $m_{i,max}$ die maximale Masse, welche das Partikel haben kann. Damit der Anteil bei allen Berechnungen bei SPH berücksichtigt wird, müssen die Formeln ein wenig angepasst werden. Dafür genügt es, das Gewicht $\omega$ als skalaren Faktor in die Gleichungen mit rein zunehmen. Aus der grundlegenden Formel 2.22 für das Partikelsystem wird dann:

$$Q_S(\mathbf{r}) = \sum_j \omega_j \frac{m_{j,max}}{\rho_j} Q_j W(\mathbf{r} - \mathbf{r}_j, h) \tag{4.18}$$

Das heißt, dass fast die gleiche Gleichung für die Interpolation des Feldes verwendet werden kann. Es hat sich nur verändert, dass anstatt einer festen Masse $m_j$ für ein Partikel $j$ nun der Anteil $\omega_j m_{j,max}$ an seiner maximal möglichen Masse benutzt wird. Je höher dabei das Gewicht $\omega_j$ ist, desto mehr Einfluss hat das Partikel $j$ auf die Umgebung des Feldes. Ist dagegen der Anteil sehr gering, verkleinert sich der Einfluss dementsprechend. Dass Gewicht sagt auch etwas über die Größe des Partikels aus, denn es gilt:

$$V_i = \frac{m_i}{\rho_i} = \omega_i \frac{m_{i,max}}{\rho_i} = \omega_i V_{i,max} \tag{4.19}$$

Dadurch dass die Masse durch den Faktor angepasst wird, und das Volumen von der Masse und der Dichte abhängt, ist wiederum das Volumen von dem Gewicht $\omega$ abhängig. Die Änderung des Anteils muss natürlich bei einem Massenaustausch mit dem Gitter angepasst werden. Im Kapitel 4.2 wurde schon gezeigt, wie die Rate $\Phi$ für die Veränderung bei einer Verdunstung oder einer Kondensation bestimmt und wie die Massen in jedem Zeitschritt aktualisiert werden. Dort war von der Anpassung der absoluten Masse die Rede. Passt man nun statt den Massen die Gewichte an, so ergibt sich aus der Gleichung 4.4 dann

$$\omega_i^{t+1} = min\left(max\left(0, \omega_i^t - \frac{\hat{m}_i}{m_{i,max}}\right), 1\right) \tag{4.20}$$

Der Unterschied ist hier, dass nun das Gewicht $\omega_i$ eines Partikels aktualisiert wird. Dafür ist es nötig die Masse $\hat{m}_i$ mit der maximalen Masse $m_{i,max}$ zu teilen, um den Anteil zu erhalten. Da das Gewicht $\omega$ auf das Intervall $[0, 1]$ begrenzt ist, dürfen diese Grenzen nicht überschritten werden. Die Aktualisierung der Massen im Gitter muss dann auch etwas angepasst werden. Es ergibt sich nämlich:

$$m_j^{t+1} = m_j^t + \sum_{i \in j} \left(\left(\omega_i^t - \omega_i^{t+1}\right) m_{i,max}\right) \tag{4.21}$$

Der Austausch findet nicht mehr mit der Differenz der Massen der Partikel zwischen zwei Zeit-

schritten statt, sondern mit der Differenz der Gewichte $\omega$. Der Unterschied zwischen $\omega_i^t$ und $\omega_i^{t+1}$ spiegelt die Veränderung der Masse des Partikels $i$ wider. Dieser muss noch mit der maximal möglichen Masse skaliert werden, um den absoluten Wert des Massenaustausches zu kommen. Die beschriebenen Änderungen an dem SPH System reichen aus, um die Verdunstung und die Kondensation zu integrieren und dabei die Stabilität zu erhalten.

Bei der Simulation müssen zusätzlich noch einige Sachen beachtet werden. Bei dieser Arbeit wird nur die Verdunstung und die Kondensation vollzogen. Die Temperaturen werden nicht zu hoch angesetzt, damit kein Sieden stattfinden kann. Da die Verdunstung nur an der Oberfläche möglich ist, dürfen in diesem Prozess nur Partikel verwendet werden, die an der Oberfläche liegen. Eine Möglichkeit zum herausfinden, welche Partikel für diesen Vorgang verwendet können bietet der Ansatz, den Orthmann und andere [OHB+13] vorgestellt haben. Damit kann bestimmt werden, welche Partikel sich an der Oberfläche befinden und welchen Anteil sie an dieser haben. Somit braucht man nur diese Partikel für die Verdunstung zu betrachten. Zusätzlich bietet es sich an, den berechneten Anteil an der Oberfläche für den Übergang von der flüssigen zur gasförmigen Phase zu verwenden. Dieser Wert kann nämlich in die Berechnung der Rate der Verdunstung mit einfließen. Eine andere Möglichkeit eine Schätzung zu treffen, ob sich ein Partikel an der Oberfläche befinden, ist es sich seine Nachbarschaft zu betrachten. Partikel, die sich mitten in der Flüssigkeit aufhalten haben nämlich eine höhere Anzahl an Nachbarn als diejenigen, die am Rand sind. Somit könnte abhängig von dieser Anzahl aus eine Entscheidung getroffen werden, ob es zur Oberfläche gehört. Diese Methode ist jedoch sehr ungenau und lässt keine Angabe über den Anteil an der Oberfläche zu.

Bei der Kondensation muss drauf geachtet werden, wo neue Partikel platziert werden. Denn selbst wenn sie erst entstehen und nur einen sehr geringen Einfluss haben, können sie das System instabil machen. Werden neue Partikel sehr nah an der Position eines anderen Partikels erstellt, kann es zu hohen Kräften kommen. Dadurch entstehen hohe Beschleunigung, welche die Partikel dazu bringen auseinander zu fliegen. Bei dieser Arbeit wurde eine sehr konservative Methode gewählt. Demnach werden neue Partikel erst an einer Position erstellt, wenn es dort keine anderen Partikel in der nähe gibt. Dass heißt, dass der Umkreis des Einflussradius frei sein soll. Dadurch werden zunächst direkte Interaktionen vermieden. Erst wenn sich die Partikel durch äußere Kräfte einander annähern, fangen sie miteinander Kräfte auszutauschen und reagieren so aufeinander.

# Kapitel 5

# Implementation

Alles was in dieser Arbeit beschrieben ist, wurde in ein bereits existierendes Framework, welches C++ und CUDA verwendet, integriert. In dieser war schon eine fertige Simulation mit SPH vorhanden. Um Verdunstungen und Kondensation zu realisieren, wurde dieses um ein Gittersystem erweitert. Die Struktur des Programms ist so aufgebaut, dass alle Eigenschaften und Werte, welche den Partikeln und Zellen zugeordnet sind, als einzelne Arrays gespeichert sind, ähnlich wie es Nie und andere [NCX15] beschreiben. Dies ist besser als die Verwendung von einem Array, welches eine Struktur enthält, die alle Eigenschaften beinhaltet. Denn nicht alle Funktionen müssen auf alle Werte zugreifen können. Die meisten brauchen nur eine Teilmenge davon für die Berechnungen.

Da zwei unterschiedliche Systeme verwendet werden, haben die Array auch dementsprechend unterschiedliche Längen. Die Werte für das Gittersystem werden in Arrays der Größe $N_g = N_x \cdot N_y \cdot N_z$ gespeichert. Der Zugriff erfolgt über einen Index $ID_g(i, j, k) = i + jN_x + kN_xN_y$. Die Werte für die Partikel werden in Arrays der Größe $N_p$ gespeichert. Da durch Verdunstung und Kondensation sich die Anzahl der Partikel ändern kann, stellt dieser Wert die maximal mögliche Anzahl an Partikeln dar. Bei der Wahl dieser Größe muss darauf geachtet werden, dass dort immer genügend Platz für Partikel, die bei Kondensation entstehen, vorhanden ist.

Für die neu dazugekommenen Funktion für die Rahmenanwendung wurde eine zusätzliche Engine entwickelt, welche die Methoden umsetzt. Die Reihenfolge der Aufrufe ist nicht ganz frei wählbar, da einige Funktionen voneinander abhängig sind. Hierbei wurde folgende Abfolge verwendet:

```
1 while animate do
2   computeVolume();
3   velocityAdvection();
4   velocityProject();
5   vaporAdvektion();
6   performEvaporationCondensation();
7   diffusionVapor();
8   diffusionTemperature();
```

Listing 5.1: Reihenfolge der Funktionsaufrufe während eines Zeitschrittes

Als erstes wird das freie Volumen der Zellen bestimmt. Mit diesem kann dann die Dichte des Dampfes ermittelt werden und damit wiederum zusammen mit der Temperatur die relative Luftfeuchtigkeit. Bei der Berechnung des Volumens kann gleichzeitig die Prüfung stattfinden, welche Zellübergänge blockiert sind. Nach dem Volumen kommen dann die Schritte der Euler Gleichungen. Zuerst werden die Geschwindigkeiten in den Zellen entlang der Strömung verschoben. Im selben Schritt sind die Beschleunigungen durch äußere Kräfte berücksichtigt. Danach kommt die Projektion des Strömungsfeldes, welche das Feld divergenzfrei macht. Sobald Quellen und Senken bereinigt wurden, findet die Advektion des Dampfes statt. Der nächste Schritt ist der einzige der schreibend auf das Partikelsystem zugreift. Dort wird nämlich die Rate der Verdunstung und der Kondensation bestimmt. Anhand dieser wird die Masse ermittelt, welche zwischen SPH und Gitter ausgetauscht werden soll. Der Austausch selber findet auch im selben Schritt statt. Als letztes kommt dann die Diffusion für den Dampf und für die Temperatur dran.

## 5.1 Realisierung des Massenaustausches

Für die Beschreibung des Massenaustausches werden drei Arrays benutzt. Jedes davon hat die Länge $N_p$, um alle Partikel damit erfassen zu können. Zwei Arrays werden für die Identifizierung und ein Array für den Austausch gebraucht.

**massPtclIdx** dient zum speichern der IDs von den Partikeln

**massGridIdx** dient zum speichern der IDs der Zellen

**massTransfer** speichert die absolute Masse, die zwischen den Systemen ausgetauscht werden soll.

Diese drei Arrays sind so miteinander verknüpft, dass der Zugriff über denselben Index stattfindet. Die Positionen der Werte in diesen Arrays hängen immer zusammen. Dass heißt der $i$-te Wert in dem Array *massPtclIdx* gibt den Index an, welches Partikel Masse austauschen soll. An der $i$-ten Stelle des Arrays *massGridIdx* steht der Index der Zelle drin, in welcher sich das Partikel momentan befindet. Und schließlich gibt der $i$-te Wert des Arrays *massTransfer* an, wie viel Masse zwischen den beiden Objekten in dem jetzigen Zeitschritt ausgetauscht wird.

Die Methode für den Massenaustausch stellt die wichtigste Methode dieser Arbeit dar, weil mit ihr die beiden Systeme verknüpft werden. Daher soll sie hier detaillierter beschrieben werden. In der Auflistung 5.2 ist der Ablauf dargestellt, wie es auf der Grafikkarte ausgeführt wird. Dabei wurden die Ein- und Ausgaben des Kernels und die Kommentare ausgeblendet, um das Ganze etwas übersichtlicher zu gestalten. Die Funktionen der einzelnen Arrays werden im Text erklärt und auf die Programmaufrufe eingegangen.

Zuallererst werden bei der Methode alle Werte aus den Arrays geholt, welche für den Massenaustausch notwendig sind. Darunter sind zum einen die Indizes der Zelle *cell.idx* und des Partikels *ptcl.idx*, für die der Austausch stattfindet. Diese werden aus den Arrays, die wie oben beschrieben aufgebaut sind, geholt. Die Variable *index* gibt an, welcher Thread zur Zeit von der GPU verarbeitet

wird. Danach wird die maximale Masse *ptcl.mass* und der derzeitige Anteil *ptcl.wght* für das Partikel ausgelesen. Für die Zelle wird nur die aktuelle Masse *cell.mass* des darin enthaltenen Dampfes benötigt.

```
1  void kTransferMass()
2  {
3    cell.idx = massGridIdx[index];
4    ptcl.idx = massPtclIdx[index];
5    ptcl.mass = get_ptcl_mass(ptcl.lvl);
6    ptcl.wght = ptclWeightBufferIn[ptcl.idx];
7    cell.mass = gridVapor[cell.idx];
8
9    float diffMass = massTransfer[ptcl.idx];
10   float diffWeight = diffMass / ptcl.mass;
11   float newWght = clamp(ptcl.wght - diffWeight, 0.0, 1.0);
12
13   diffWeight = ptcl.wght - newWght;
14   diffMass = diffWeight * ptcl.mass;
15
16   ptclWeightBufferOut[ptcl.idx] = newWght;
17
18   atomicAdd( &gridVapor[cell.idx], diffMass);
19
20   if(newWght <= 0)
21   {
22     ptcl.pos = make_float4( INVALID_PTCL_RADIUS);
23     ptclPosBuffer[ptcl.idx] = ptcl.pos;
24   }
25   if(newWght >= 1)
26     ptclEvaporationRate[ptcl.idx] = 0.0;
27 }
```

Listing 5.2: Code für die Durchführung des Massenaustausches

In den Zeilen 8 bis 10 findet die Aktualisierung des Gewichtes für das Partikel statt. Dabei wird zunächst die Masse *diffMass*, die in diesem Zeitschritt transferiert wird, aus dem Array *massTransfer* gelesen. Da bei den Partikeln ein Gewicht für die Angabe der Masse verwendet wird, muss davon der Anteil *diffWeight* gebildet werden. Diesen zieht man nun von dem aktuellen Gewicht *ptcl.wght* ab und erhält nach Beachtung der Grenzen den neuen Anteil *newWght* für das Partikel. Damit ist die Bestimmung der neuen Masse für das Partikel fertig.

Um die neue Masse für die Zelle zu bestimmen, wird zunächst die Differenz *diffWeight* der tatsächlichen Änderung des Anteils der Masse an dem Partikel ermittelt. Und daraus resultiert die absolute Differenz der Masse $diffMass$. Hat man diese Werte bestimmt, können nun die Werte im Speicher aktualisiert werden. Für die Partikel reicht es aus, den neuen Wert des Gewichtes in ein Ar-

ray zu schreiben. Bei der Zelle muss jedoch beachtet werden, dass mehrere Partikel mit einer Zelle Masse austauschen können. Da die Threads auf der GPU parallel ausgeführt werden, kann es dazu kommen, dass mehrere Schreiboperation auf den selben Bereich durchgeführt werden. Das führt zur Verfälschung der Ergebnisse. Um dies zu vermeiden bietet CUDA spezielle Funktionen hierfür an, die *Atomic-Operations*. Dabei wird der Speicher für die Zeit des Vorgangs gesperrt, sodass kein weiterer Aufruf darauf zugreifen kann, und nach der Ausführung wieder freigegeben. Ist der Speicher schon gesperrt, wird solange gewartet, bis er freigegeben wurde. Dadurch wird sichergestellt, dass immer nur eine Schreiboperation zur gleichen Zeit durchgeführt werden kann. Diese Variante wird für die Aktualisierung der Masse für die Zelle in Zeile 17 verwendet.

Nach dem Transport der Masse fehlt noch ein Schritt für die Partikel. Falls die obere oder untere Grenze der Masse erreicht wird, muss drauf reagiert werden. Wenn das Gewicht *newWght* auf 0 sinkt, bedeutet dies, dass das Partikel keinen Einfluss mehr auf das System hat. Deshalb kann es entfernt werden. Dies geschieht hier durch die Setzung der Position auf einen ungültigen Wert. Damit wird signalisiert, dass das Partikel nicht mehr vorhanden ist und ist somit aus allen weiteren Berechnungen ausgeschlossen. Wenn das Gewicht jedoch den vollen Wert erreicht, genügt es einfach die Rate der Änderung auf 0 zu setzen. Die Richtung der Rate kann während einer Verdunstung oder einer Kondensation nicht geändert werden, bis der Vorgang vorbei ist. Durch das Setzen auf den Wert 0 ist die Kondensation vollständig abgeschlossen und es kann bei Bedarf für Verdunstungen eingesetzt werden.

## 5.2 Minimierung des Speicherbedarfs von Matrizen

In dieser Arbeit werden einige Gleichungen verwendet, die sich mittels einer Matrix beschreiben lassen. Dies trifft sowohl auf die Herstellung der Inkompressibilität (siehe Kapitel 2.2.2) des Strömungsfeldes und auf die Diffusion (siehe Kapitel 4.4.2) des Dampfes zu. Aber auch die Advektion (siehe Kapitel 2.2.1) der Masse und der Temperatur kann mittels einer Matrix realisiert werden.

Die Matrizen sind die dabei eingesetzt werden können, sind jedoch immer nur sehr dünn besetzt. In den Zeilen bzw. Spalten wird nur eine sehr geringe Menge für die Berechnungen verwendet. Der Rest der Einträge ist immer 0 und ist für die Berechnungen unbrauchbar. Wenn nun die Ganze Matrix inklusive aller 0er gespeichert wird, dann nimmt sie nur unnötig viel Platz ein. Angenommen es wird ein dreidimensionales Gitter mit insgesamt $32 \cdot 32 \cdot 32 = 32768$ Zellen simuliert. Um die Interaktion von jedem beliebigen paar von Zellen zu erlauben, wird eine Matrix mit 32768 Zeilen und 32768 Spalten benötigt. Wenn jedes der Werte innerhalb der Matrix als float gespeichert wird, welches 4 Byte pro Zahl einnimmt, so braucht die Matrix insgesamt einen Speicher von $32768^2 \cdot 4 = 4.294.967.296$ Bytes. Umgerechnet sind das genau 4 GB an Speicher, die nur für diese Matrix nötig wären. Versucht man ein dreidimensionales Gitter mit $128^3$ Zellen zu simulieren, werden schon 16.384 GB (siehe Tabelle 5.1) an Speicher benötigt. Dies übersteigt schon den gesamten Festplattenspeicher der meisten Systeme. Aber selbst die benötigten 4 GB für ein $32^3$ Gitter passen auf nur wenige Grafikkarten drauf.

Der benötigte Speicherplatz für sehr dünn besetzte Matrizen lässt sich mit einer anderen Schreibweise erheblich reduzieren. Beispiele hierfür sind das *Compressed Sparse Row* (CSR) Format und das *Coordinate* (COO) Format. Bei beiden werden nur die Werte gespeichert die ungleich 0 sind. Es werden jedoch zusätzliche Arrays benötigt, die angeben, wo diese Werte stehen. Da bei der Diffusion und bei dem Poisson immer nur die Nachbarn einer Zelle untersucht werden, benötigt man hierbei im dreidimensionalem nur 7 Werte, 6 Werte für die jeweiligen Nachbarzellen und einen Wert für die eigene Zelle. Für die Speicherung dieser Matrix bietet sich das CSR-Format an, welches aus den folgenden Arrays besteht:

**data** beinhaltet die Werte, die in der Matrix stehen
**col_idx** zeigt an, in welchen Spalten sich die Werte befinden
**row_ptr** gibt den Indizes an, an welchen Stellen der Arrays *data* und *col_idx* eine neue Zeile beginnt

Die Länge von den Arrays *data* und *col_idx* ist dabei gleich und ist genauso hoch wie die Anzahl der Werte, die ungleich 0 sind. Das Array *row_ptr* hat die Länge der Anzahl an Zeilen und zusätzlich einen Platz, der signalisiert, wo das Ende der Matrix ist. Wenn also nur 7 Werte pro Zeile gebraucht werden, dann hat *row_ptr* die Länge $N_g + 1$ und die anderen beiden die Länge $7 \cdot N_g$, wobei $N_g$ die Anzahl aller Gitterzellen ist. Für das obige Beispiel bedeutet dies, dass bei einem Gitter der Größe $32^3$ insgesamt nur $2 \cdot 7 \cdot 32768 + 32769 = 491.520$ Zahlen benötigt werden. Diese brauchen dann nur noch 1,875 MB an Speicher, statt den 4 GB für die volle Matrix, was eine erhebliche Reduzierung darstellt. Bei größeren Gittern wird der Unterschied noch gravierender, wie in Tabelle 5.1 gezeigt ist.

Die Multiplikation der Matrix mit einem Vektor wird dadurch auch beschleunigt. Da weniger Zahlen vorhanden sind, werden auch weniger Operationen benötigt. So müssen pro Zeile nur so viele Multiplikationen und Additionen durchgeführt werden, wie viele Werte tatsächlich enthalten sind. Ein Kernel für die Multiplikation einer Matrix im CSR-Format mit einem Vektor für die Grafikkarte könnte zum Beispiel so aussehen:

```
1 void kSpMV_CSR()
2 {
3   float dot = 0.0;
4   int row_start = row_ptr[row];
5   int row_end = row_ptr[row+1];
6   for (int elem = row_start; elem < row_end; elem++)
7     dot += data[elem] * inBuffer[col_idx[elem]];
8   outBuffer[row] = dot;
9 }
```

Listing 5.3: Beispielhafte Multiplikation von einer Matrix im CSR-Format mit einem Vektor

Teilergebnisse werden zunächst in der Variable $dot$ gespeichert. Diese wird mit dem Wert 0 initialisiert. Danach besorgt man sich die Indizes, an denen die Werte für die jetzige Zeile beginnen und wo sie enden. Dann genügt eine Schleife über alle Indizes, die zwischen den beiden Grenzen liegen, um die Multiplikation durchzuführen. Darin wird der Wert der Matrix aus dem Array $data$ geholt und

mit dem Wert des Vektors multipliziert, der zu der zugehörigen Spalte gehört. Diese Ergebnisse werden jeweils immer auf das Teilergebnis drauf addiert. Am Ende steht in $dot$ das Resultat von dieser Operation drin und kann in den Speicher geschrieben werden.

Für die Advektion der Masse und für die Temperatur bietet sich statt dem CSR-Format das COO-Format an. Mittels des Semi-Lagrange Backward und des Semi-Lagrange Forward Schritt muss dort nämlich auf unterschiedliche Weise auf die Matrix zugegriffen werden. Bei dem Backward Schritt stellt die Zelle das Ziel einer Bewegung dar. Dabei werden die Gewichte, die bei der Interpolation herauskommen in die entsprechende Zeile der Matrix geschrieben. Die Spalten, in die die Gewichte reinkommen, sind von den Zellen abhängig, die für die Interpolation notwendig sind. Bei dem Forward Schritt, der bei Bedarf ausgeführt wird, verhält es sich genau umgekehrt. Da werden die Gewichte in die Spalte der Startzelle geschrieben, und die Zeilen hängen von der Interpolation ab. Würden man also eine CSR-Matrix verwendet, so ist eine Änderung des Formates nötig, um die Werte in verschiedenen Weisen in die Matrix zu schreiben. Bei der Verwendung des COO-Formates ist keine Umformatierung notwendig.

Das COO-Format ist so aufgebaut, dass auf alle Stellen in der Matrix ohne große Umrechnungen zugegriffen werden kann. Dabei werden die Koordinaten der Position abgespeichert und der jeweilige Wert dazu. Dass heißt es werden insgesamt drei Arrays dafür verwendet.

**row_idx** speichert die Zeilen der Matrix, also die IDs der Zielzellen
**col_idx** speichert die Spalten der Matrix, also die IDs der Startzellen
**data** beinhaltet die Werte der Matrix, in diesem Fall also die Gewichte der Interpolation

Alle drei Arrays haben die selbe Länge, weil die einzelnen Einträge zusammen gehören. Für jeden Wert in der Matrix, wird jeweils ein Wert in die Arrays geschrieben, einer für die Zeile, einer für die Spalte und einer für den Wert. Somit ist die Länge der Arrays gleich der Anzahl der Werte der Matrix, die ungleich 0 sind.

| **Matrixformat** | $8^3$ | $16^3$ | $32^3$ | $64^3$ | $128^3$ |
|---|---|---|---|---|---|
| Normal | 1 MB | 64 MB | 4 GB | 256 GB | 16384 GB |
| CSR(7) | 30 KB | 240 KB | 1,875 MB | 15 MB | 120 MB |
| COO(16) | 96 KB | 768 KB | 6 MB | 48 MB | 384 MB |

Tabelle 5.1: Vergleich von dem benötigen Speicherplatz zwischen einer vollen Matrix, einer CSR Matrix mit 7 Werten pro Zeile und einer COO-Matrix mit 16 Werten pro Zeile

Bei der Advektion können für jede Zelle maximal zwei Interpolationen durchgeführt werden. Im dreidimensionalem gehen 8 Zellen in die Interpolation rein. Das heißt, dass für jede Zelle maximal 16 Einträge in der Matrix nötig sind. Bei drei Arrays für die Matrix müssen so maximal $16 \cdot 3 \cdot N_g$ Werte gespeichert werden. Um die Formate nochmals zu vergleichen, lässt sich auch hier eine Rechnung für den benötigten Speicherplatz aufstellen. Somit werden für ein Gitter der Größe $32^3$

maximal $16 \cdot 3 \cdot 32^3 = 1.572.864$ Zahlen benötigt. Diese verbrauchen etwa 6 MB an Speicher. In der Tabelle 5.1 können nochmals alle Daten für die unterschiedlichen Formate und für verschiedene Gitterauflösungen verglichen werden.

Die Advektion kann dann mit einer einfachen Matrix-Vektor Multiplikation durchgeführt werden. Um dies durch Parallelisierung zu beschleunigen, ist es besser einige Werte zusammen zu fassen. Am besten sortiert man das Array $row\_idx$, wo die Indizes der Zeilen, also damit auch der Zielzellen, stehen. Die Verschiebungen, die bei der Sortierung auftreten, müssen natürlich für alle drei Arrays des COO-Formats gemacht werden. Danach führt man zwei zusätzliche Arrays ein, die dann angeben, an welchen Stellen die Indizes für die jeweiligen Zielzellen anfangen und enden. Dann kann ein Kernel auf der GPU für jede Zeile parallel eine Multiplikation, ähnlich wie bei dem CSR-Format, durchführen. Durch die zusätzlichen Arrays können die Werte für die Zeilen Blockweise aus den Arrays des COO-Formats gelesen werden und sind damit unabhängig voneinander. Die gleiche Variante mit der Umsortierung kann man auch schon für den Forward Schritt verwenden. Dabei müssen die Gewichte der einzelnen Spalten aufsummiert werden, um eine Entscheidung treffen zu können, ob dieser Schritt notwendig. Dafür ist jedoch eine Sortierung nach den Spalten, also dem Array $col\_idx$ notwendig.

# Kapitel 6

# Ergebnisse

Die neue Methode, die in dieser Arbeit vorgestellt wurde, wurde auf einem System entwickelt und getestet, welches einen Intel i7-3770K Prozessor mit 3,50 GHz und einen 8 GB Arbeitsspeicher enthält. Für die parallele Programmierung ist eine GeForce GTX 660 Ti Grafikkarte eingebaut mit 2GB internem Speicher, 1344 CUDA Recheneinheiten, einem Kerntakt von 1020 MHz und einem Speichertakt von 1502 MHz.

Die neuen Funktionen wurden bereits in ein bestehendes Rahmenprogramm für die Simulation von SPH integriert. Dabei wird die Programmiersprache C++ verwendet und für die parallele Programmierung die Sprache CUDA. Alle hier beschriebenen Simulationen befinden sich in einem Würfel der Seitenlänge $100m$. Zu Start sind immer 679.424 Partikel vorhanden. Diese Zahl ändert sich im Laufe der Simulation durch Verdunstung und Kondensation. Die maximale Anzahl an Partikeln beträgt 1.048.576.

Zuallererst stellt sich die Frage in welchem Verhältnis die beiden Systeme stehen. Dabei ist es interessant zu wissen, wie viele Partikel durch das Gitter eingespart werden können, also wie viele Partikel in die Zellen passen, und wie viel Masse das Gitter aufnehmen kann von dem Dampf der verdunstet. Für den Partikelradius werden hier $0{,}5m$ eingesetzt. Für das Gitter wurden die Auflösungen $8^3$, $16^3$ und $32^3$ ausprobiert. Für die Breiten der Zellen ergeben sich also $12{,}5m$, $6{,}25m$, und $3{,}125m$. Das Volumen der Partikel beträgt etwa $0{,}5236m^3$. Das heißt bei einer Zellbreite von $3{,}125m$ passen theoretisch etwa 58 Partikel rein, bei einer Breite von $6{,}25m$ etwa 266 Partikel und 3730 Partikel würden in einer Zelle mit der Breite $12{,}5m$ reinpassen. Füllt man das Ganze $100^3m$ große Becken mit Partikeln, würden insgesamt 1.909.854 reinpassen. Bei der Simulation wird jedoch nur etwa ein Drittel davon verwendet.

Bei der Frage, wie viel Masse eine Zelle aufnehmen kann, muss bedacht werden, dass dies von der Temperatur abhängig ist. Kalte Luft kann nur sehr wenig Wasser aufnehmen, deshalb wird bei einer Abkühlung Masse durch Kondensation abgegeben. Heiße Luft dagegen kann mehr Wasser aufnehmen. Durch die erhöhte Aufnahmefähigkeit kann es durch ein Verdunsten der Flüssigkeit an Masse gewinnen. Welche Menge genau aufgenommen werden kann, kann durch den Sättigungsdampfdruck bestimmt werden, welcher sich mit der Formel 2.10 berechnen lässt. Mit dem ermitteltem Druck und

der idealen Gasgleichung (siehe Formel 2.6) kann man dann auf die maximale Masse schließen. Die folgende Tabelle zeigt einige Angaben zu verschiedenen Zellgrößen und Temperaturen, wie viel eine Zelle an Masse aufnehmen kann.

| **Zellgröße[m]/Temperatur[C]** | $20°$ | $30°$ | $40°$ | $50°$ | $60°$ |
|---|---|---|---|---|---|
| $12{,}5^3$ | 33,68 kg | 59,12 kg | 99,59 kg | 161,73 kg | 254,04 kg |
| $6{,}25^3$ | 4,21 kg | 7,39 kg | 12,45 kg | 20,21 kg | 31,75 kg |
| $3{,}125^3$ | 0,53 kg | 0,92 kg | 1,56 kg | 2,53 kg | 3,97 kg |

Tabelle 6.1: maximale Masse pro Zelle bei verschiedenen Zellgrößen und unterschiedlichen Temperaturen

Man kann deutlich den Unterschied zwischen kalter und warmer Luft erkennen. Während eine quadratische Zelle mit einer Breite von $12{,}5m$ bei $20°C$ nur $33{,}68kg$ Dampf aufnehmen kann, beträgt der Wert bei $60°C$ schon knapp das 8-fache. Bei den anderen Größen verhält es sich genauso. Ein Partikel mit einem Radius von $0{,}5m$ hat bei einer Wasserdichte von $1.000kg/m^3$ eine Masse von $523{,}6kg$. Man erkennt, dass selbst bei der niedrigsten Auflösung des Gitters ein Partikel zwei Zellen bei $60°C$ füllen kann. Bei kalter Luft können sogar fast 16 Zellen voll mit Dampf gefüllt werden. Es ist also klar zu sehen, dass für diese Simulationen ein relativ großes Verhältnis zwischen SPH und Gitter Sinn macht. Denn bei einer höheren Gitterauflösung müssen schon sehr viele Zellen mit Masse befüllt sein, bevor auch nur ein Partikel vollständig verdunsten kann. Laut der Tabelle können dies bis zu 1.000 Zellen sein.

Bei Simulationen im graphischen Bereich sind oft die Zeiten, die die Berechnungen in Anspruch nehmen interessant. Hierbei sei nicht nur die Beachtung auf die gesamten Zeiten gelenkt, sondern auch auf die Verhältnisse zwischen den beiden Systemen. Für die Messung der Performance wurden verschiedene Situationen durchgerechnet. Dabei wurden jedoch nur die Eigenschaften des Gitters verändert, während die Einstellungen für das SPH größtenteils gleich geblieben sind. Nur die Temperaturen wurden hierbei angepasst. Bei den Temperaturen wurde immer eine Seite festgelegt, die $340°K$ bzw. $66{,}85°C$ warm ist, und die gegenüberliegende Seite hat eine Temperatur von $290°K$ bzw. $16{,}85°C$ erhalten. Für den Dampf der beim Start in den Zellen vorhanden war, gab es auch zwei Fälle. Entweder enthielt die Luft überhaupt kein Wasser, war also vollkommen leer, oder die Zellen waren schon gefüllt. Wobei mit gefüllt gemeint ist, dass in jeder Zelle $10kg$ Dampf enthalten war. Durch die unterschiedlichen Temperaturen resultiert dies in verschiedene relative Luftfeuchtigkeiten.

Für die Angabe der gemessenen Zeiten wurden die Funktionen ausgewählt, die am längsten brauchen und in der Tabelle 6.2 zusammengestellt. Die ersten drei Spalten beziehen sich auf die Situation bei der der Boden des Beckens erhitzt wird und die Luft zu Anfang völlig trocken ist. Diese Messungen sind zum Vergleich für die Auswirkungen der unterschiedlichen Auflösungen. In den rechten beiden Spalten sind die Messungen bei denen die linke Seite des Beckens erhitzt und einmal die Luft trocken ist und einmal bereits mit Dampf gefüllt. Dies soll zeigen, ob die Berechnungen für unter-

schiedliche Massen sich unterschieden. Wie man jedoch erkennen kann, bleiben die Zeiten dort in etwa gleich, nur bei der Bestimmung des Drucks für das Gitter unterscheiden sie sich leicht.

| **Auflösung** | $8^3$ | $16^3$ | $32^3$ | $16^3$ | $16^3$ |
|---|---|---|---|---|---|
| **Rand mit hoher Temperatur** | unten | unten | unten | links | links |
| **Masse in Zellen** | leer | leer | leer | leer | voll |
| SPH Nachbarsuche | 44,8 | 45,2 | 47,7 | 45 | 45,3 |
| SPH Druckkraft | 82,5 | 71,6 | 73 | 71 | 74,1 |
| SPH Diffusion Temperatur | 12 | 12,1 | 12,2 | 12,1 | 12,1 |
| Gitter Volumen | 14 | 7,8 | 9 | 7,8 | 7,8 |
| Gitter Projektion/Druck | 17,1 | 206,4 | 376 | 86,7 | 70,3 |
| Gitter Massenaustausch | 19,5 | 9,5 | 14 | 9,6 | 9,7 |

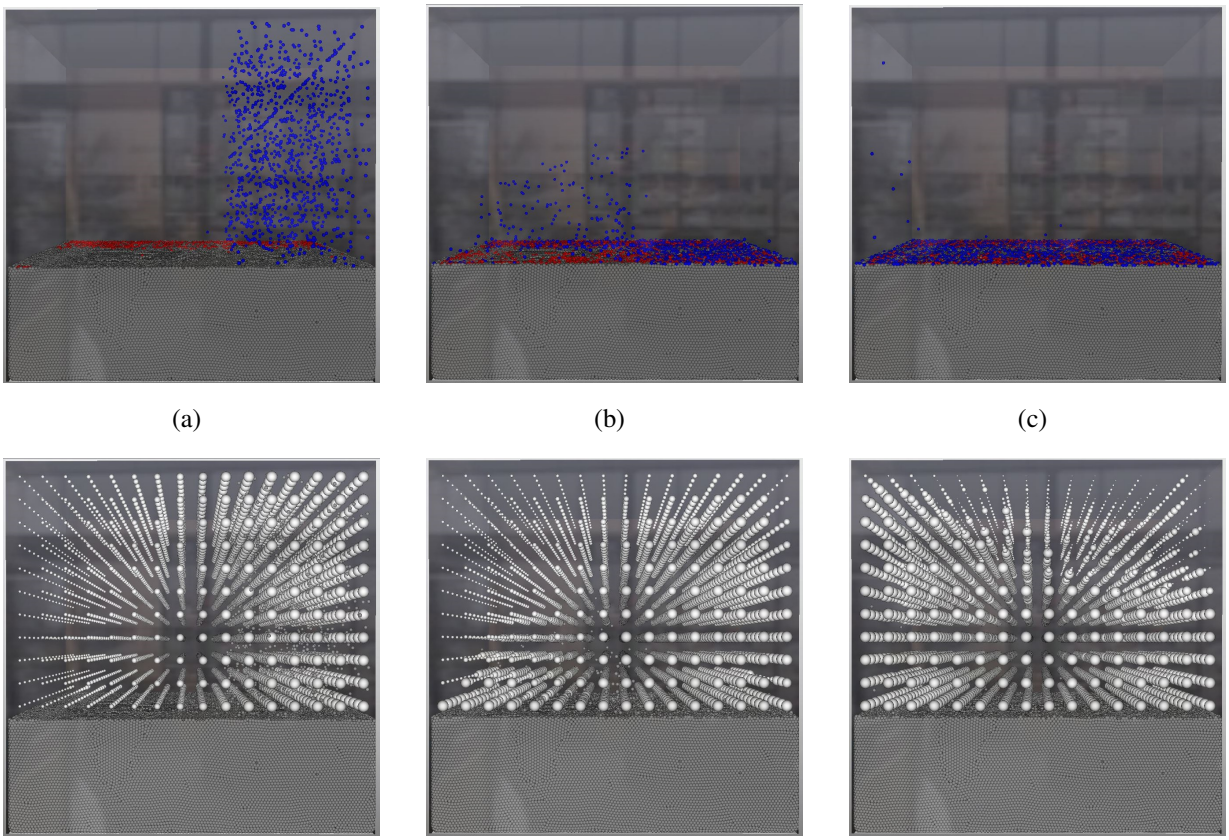
Tabelle 6.2: Durchschnittliche Zeiten für ausgewählte Funktionen in ms

Dadurch dass an dem Partikelsystem nur die Temperaturen verändert wurden, sieht man, dass die Zeiten für alle Situationen relativ nah bei einander liegen. Beim Gitter sind jedoch einige Unterschiede zu erkennen. Auffallend ist, dass die Herstellung der Inkompressibilität bei höheren Auflösungen die meiste Zeit verbraucht. Dies ist auch die Funktion, wo am meisten Berechnungen durchgeführt werden, da hier solange iteriert wird, bis eine bestimmte Genauigkeit erreicht ist. Vergleicht man diese Funktion mit de Berechnung der Druckkraft für SPH, so sieht man, dass bei den rechten beiden Beispielen die Zeiten ungefähr dieselben sind. Das Gitterzellen erstrecken sich jedoch über den ganzen Raum, wobei SPH nur etwa einen Drittel davon einnimmt. Bei einem Becken voller Partikel ist hier also ein dreifacher Wert zu erwarten.

Bei den drei linken Beispielen sieht man die Auswirkungen der unterschiedlichen Auflösungen des Gitters. Wie erwartet steigt die Berechnungszeit mit der Erhöhung der Auflösung an. Dies ist besonders bei der Projektion zu erkennen. Schon vom Wechsel von einem $8^3$ Gitter auf ein $16^3$ Gitter erhöht sich die Zeit auf über das zehnfache. Zwischen dem $16^3$ Gitter und dem $32^3$ Gitter beträgt die Zeit ungefähr doppelt so viel. Auffallend ist auch der Unterschied zwischen den Spalten 2 und 4. Obwohl dabei die selben Auflösungen benutzt wurden, unterscheiden sich die Zeiten für die Projektion deutlich. Dies kann damit zusammenhängen, dass bei einer Situation, bei der eine Seite warm und eine Seite kalt ist, einer Art Zirkulation entsteht und so die Divergenz leicht ausgeglichen werden kann. Ist dagegen nur der Boden heiß, so versucht der Dampf überall stetig nach oben zu steigen. Dadurch entsteht unten eine Quelle und oben eine Senke, die durch die Projektion ausgeglichen werden muss. Um diese Divergenzen auszugleichen ist eine durchschnittlich höhere Anzahl an Iterationsschritten notwendig. Bemerkenswert sind auch die Zeiten von dem Beispiel ganz links in der Tabelle. Dadurch, dass die Zellen sehr groß sind und dementsprechend sich viele Partikel dort aufhalten können, beanspruchen andere Funktionen mehr Zeit. So sind die Berechnungen für die Volumen und der

Massenaustausch ungefähr doppelt so hoch wie bei den restlichen Beispielen.

Nach der Bewertung der Performance und nach den Bestimmungen der maximal möglichen Massen innerhalb der Zellen, soll hier auf einige konkrete Beispiele eingegangen werden. Für die Messungen der Zeit wurden einige Situationen aufgezählt. Alle bis auf einen sollen nun anhand einiger Screenshots beschrieben werden. Während der Simulation wurden von mehreren Zeitschritten Bilder aufgenommen. Von jeder Simulation sind Aufnahmen zu drei verschiedenen Zeiten herausgesucht, die am aussagekräftigsten sind. Bei den oberen Reihen ist die Verdunstung und Kondensation gezeigt. Dabei sind Partikel die am verdunsten sind rot markiert und kondensierende Partikel blau. In den unteren Reihen ist der Dampf innerhalb der Zellen zu sehen. Die weißen Kugeln sind in der Mitte der Zellen platziert und geben den Dampf mittels relativer Luftfeuchte an. Je größer die Kugeln sind, desto höher ist die Luftfeuchte in den Zellen.



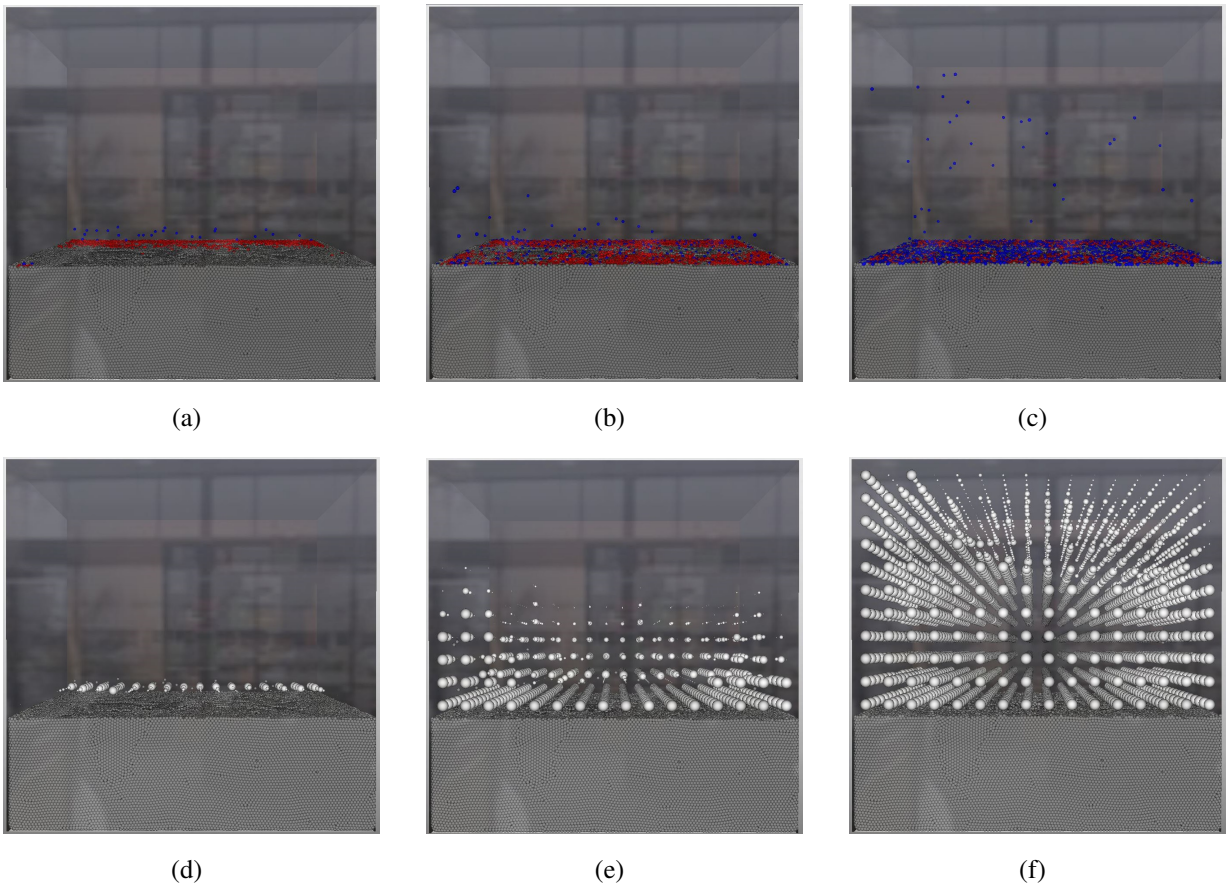
(a) (b) (c)

(d) (e) (f)

Bild 6.1: $16^3$ Gitter mit vollen Zellen und heißer Temperatur links bei $t = 0{,}5s; 5s; 40s$

In dem Beispiel von Bild 6.1 sind die Zellen zu Anfang mit $10kg$ Dampf gefüllt. Auf der linken Seite herrscht eine warme Temperatur und auf der rechten eine kalte. Die dadurch resultierenden unterschiedlichen relativen Luftfeuchtigkeiten sind in dem unteren linken Bild zu erkennen. In dem

Bild darüber ist schon nach $0{,}5s$ Simulationszeit deutlich die Kondensation in der rechten Hälfte des Raumes zu sehen. Nach $5s$ sind die Partikel durch die Gravitation nach unten gefallen und in der linken Hälfte ist mehr Dampf durch Verdunstung entstanden. Sogar so viel, dass selbst hier der Dampf anfängt zu kondensieren. Nach $40s$ hat sich ein Gleichgewicht eingestellt. Die Luft enthält zwar viel Masse, aber nicht so viel wie zum kondensieren notwendig mit kleineren Ausnahmen. Und die Masse, die über der Oberfläche verdunstet, geht direkt zu den kondensierenden Partikel über.



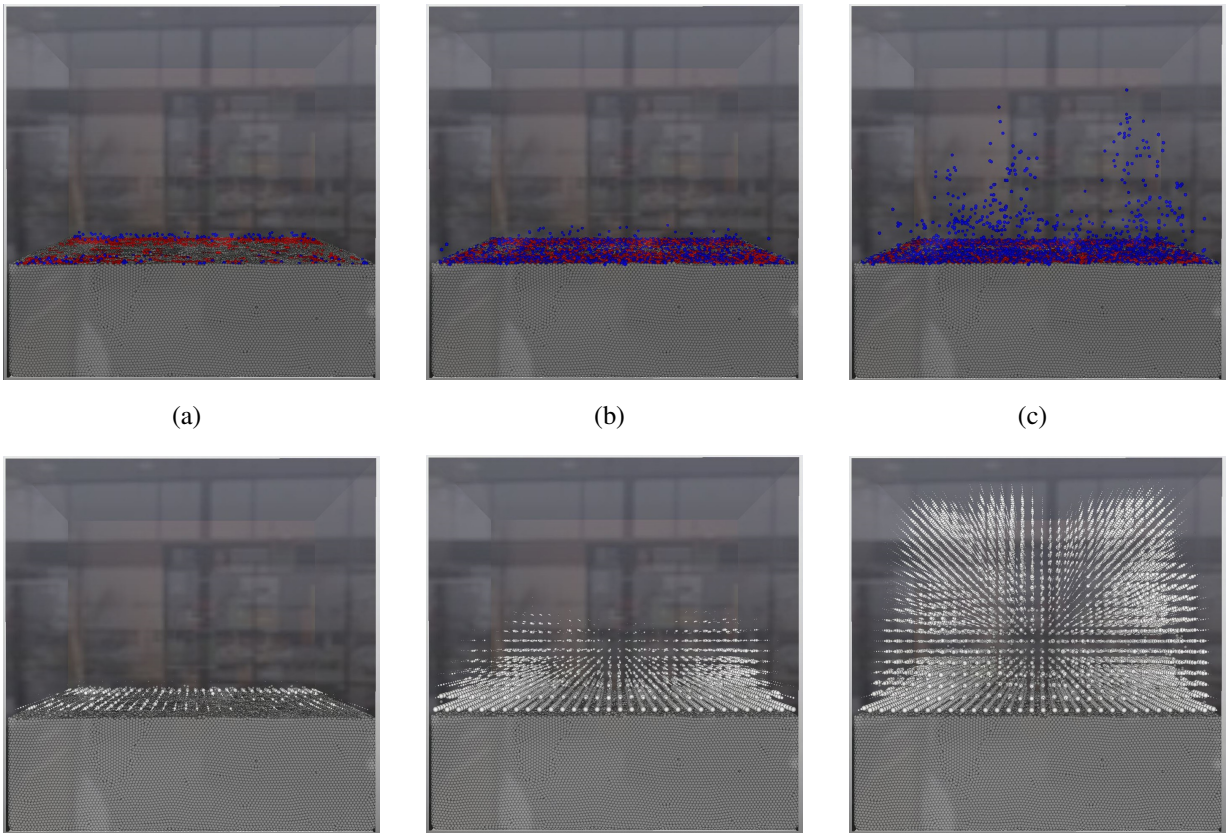
(a) (b) (c)

(d) (e) (f)

Bild 6.2: $16^3$ Gitter mit leeren Zellen und heißer Temperatur links bei $t = 0{,}5s; 10s; 60s$

Im zweiten Beispiel welches in Bild 6.2 zu sehen ist, wurde nun mit einer trockenen Luft angefangen. Bei $0{,}5s$ fängt schon das Wasser an zu verdunsten und die Luft füllt sich nah an der Oberfläche mit Dampf. Dabei wird sie so stark gesättigt, dass erste Kondensationen kurz über der Wasseroberfläche auftauchen. Nach $10s$ sieht man, wie der Dampf langsam nach oben steigt. Dabei ist die Luftfeuchtigkeit in der linken Hälfte etwas höher als in der rechten. Dies liegt daran, dass durch die höhere Temperatur auf der linken Seite die Rate der Verdunstung entsprechend höher ist. Auch die Kondensation auf dieser Seite ist etwas stärker, welche etwas höher über der Wasseroberfläche stattfindet. Nach $60s$ hat sich die Luft fast vollständig mit Dampf gefüllt, nur noch wenige Zellen mit niedriger

Luftfeuchtigkeit sind vorhanden. Im ganzen Raum entstehen vereinzelte Kondensationen, falls die Luft zu feucht wird. Direkt an der Wasseroberfläche sind Partikel sowohl am verdunsten als auch am kondensieren und stehen etwa im Gleichgewicht.



(a) (b) (c)

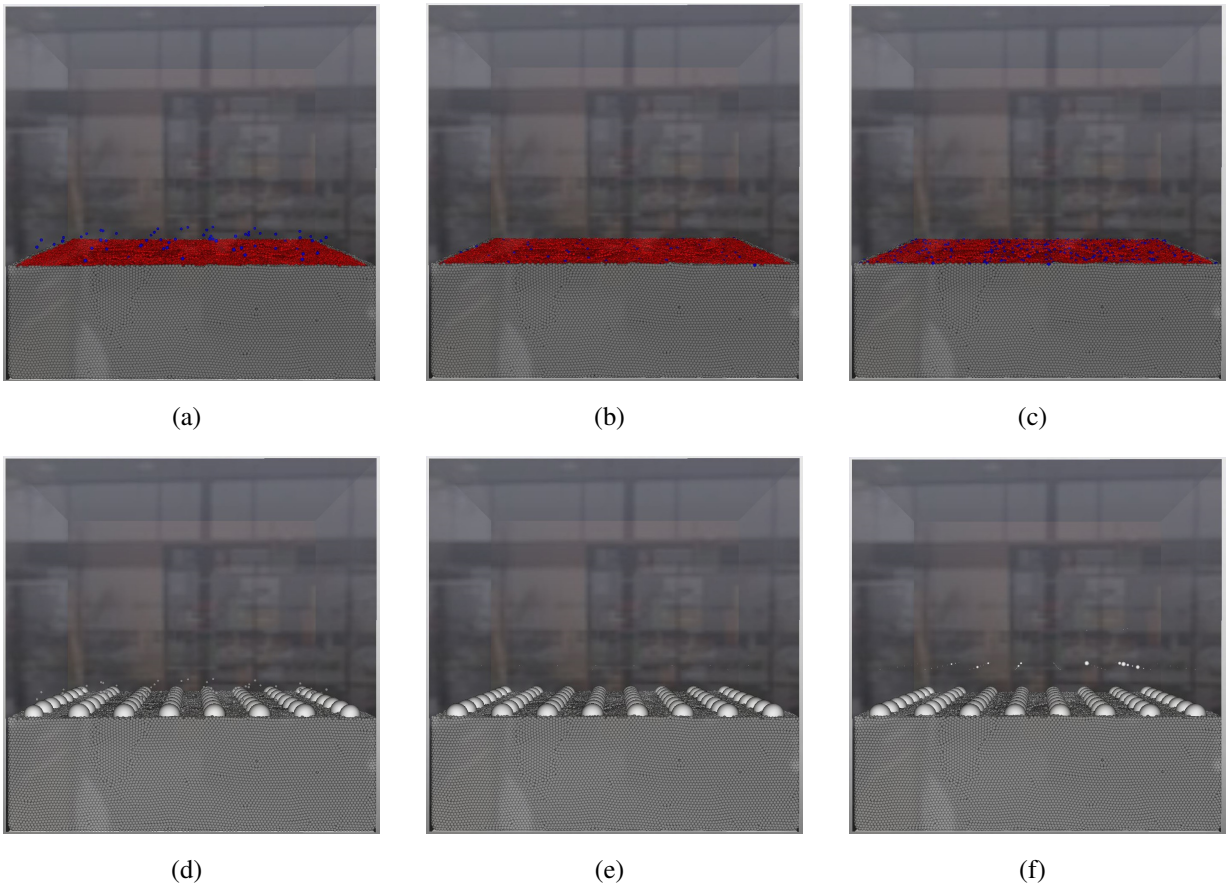
(d) (e) (f)

Bild 6.3: $32^3$ Gitter mit leeren Zellen und heißer Temperatur unten bei $t = 0{,}5s; 10s; 20s$

Im dritten Beispiel wurde nun ein Gitter mit einer höheren Auflösung gewählt. Die Zellen sind wieder leer und die Wärmequelle ist nun auf der unteren Ebene. In Bild 6.3 ist der Ablauf zu sehen. Nach nur $0{,}5s$ ist noch nicht sehr viel passiert. Die Oberfläche fängt an zu verdunsten und es bildet sich leichter Dampf an ihr. Vereinzelt kommt es zur Kondensation. Nach $10s$ füllt sich der Bereich über der Oberfläche mit Dampf. Die Luftfeuchtigkeit steigt stark an, sodass an der Wasseroberfläche eine höhere Kondensation entsteht. Durch den Auftrieb und die Diffusion verbreitet sich der Dampf nach oben. Nach $20s$ ist der Dampf soweit verbreitet, dass fast alle Zellen mit Dampf gefüllt sind. Die vorherrschende Luftfeuchte führt dazu, dass selbst hoch über der Wasseroberfläche durch Kondensation viele Tröpfchen entstehen, welche die Masse aus den Zellen ziehen.

Das letzte Beispiel, welches in Bild 6.4 zu sehen ist, ist ähnlich wie das vorherige, allerdings wird hier ein sehr grobes Gitter verwendet. Der große Platz in den Zellen führt dazu, dass fast an

der gesamten Oberfläche eine Verdunstung stattfinden. Schon nach $0{,}5s$ sind die Zellen, die an der Wasseroberfläche liegen, voll mit Dampf gefüllt. Dies führt zu einzelnen Kondensationen nah an der Oberfläche. Nach $10s$ sieht das Bild fast gleich aus. Die unteren Zellen sind immer noch voll und die neu entstandenen Partikel wurden durch die Gravitation nach unten gezogen. Selbst nach $20s$ ist keine Veränderung zu sehen. Während beim feinen Gitter der Dampf nach oben gestiegen ist, bleibt er hier nah an der Oberfläche.



(a) (b) (c)

(d) (e) (f)

Bild 6.4: $8^3$ Gitter mit leeren Zellen und heißer Temperatur unten bei $t = 0{,}5s; 10s; 20s$

Hier sieht man, dass ein zu grobes Gitter nicht für die Simulation geeignet ist. Durch die großen Abstände und großen Volumen findet fast keine Bewegung statt. Deshalb sollte nach Möglichkeit eine höhere Auflösung verwendet werden. Jedoch muss darauf geachtet werden, dass die Simulation in vertretbarer Zeit berechnet werden kann. Das Gitter darf auch nicht zu fein gewählt werden, das sonst eine Simulation mit reinem SPH evtl. schneller sein kann. Mit einer guten Balance zwischen den beiden Systemen lassen sich Verdunstungen und Kondensationen gut mit diesen Systemen simulieren.

# Kapitel 7

# Zusammenfassung und Ausblick

In dieser Arbeit wurde eine Möglichkeit vorgestellt, mit der sich die Prozesse der Verdunstung und Kondensation realisieren lassen. Hierfür wurde ein bestehendes Framework für SPH-basierte Flüssigkeiten um ein Gittersystem erweitert, welches für die Simulation der Gasphase dient. Beide Systeme können zwar jeweils für die Simulation von Flüssigkeiten und Gasen verwendet werden. Die Kopplung von SPH und Gitter lässt jedoch eine Interaktion zwischen zwei verschiedenen Aggregatzuständen zu. Die Verknüpfung erfolgt dabei durch einen verlustfreien Massenaustausch. Für die Verdunstung und Kondensation werden Partikel langsam über die Zeit hinweg aus- oder eingeblendet. So wird ihr Einfluss auf das System stetig angepasst und plötzlichen Änderungen im Feld entgegengewirkt. Während dem Einblenden oder Ausblenden wird die Masse in den Zellen des Gitters entsprechend angepasst.

Die Ergebnisse zeigen, dass die Verwendung des Gitters eine gute Möglichkeit darstellt, eine Gasphase zu simulieren, die durch Verdunstung und Kondensation beeinflusst wird. Die Simulationszeiten der beiden Systeme sind in etwa gleich. Und das obwohl das Gitter den ganzen Raum ausfüllt und SPH nur an Stellen verwendet wird, wo sich tatsächlich die Flüssigkeit befindet. Eine Simulation welche nur mit Partikeln realisiert wird, würde aufwendiger sein, als die vorgestellte Möglichkeit mittels der Verknüpfung zweier Systeme. Anhand der Ergebnisse wird auch klar, dass die gewählte Auflösung eine große Rolle spielt. Bei großen Zellen spart man zwar Zeit bei der Berechnung, jedoch können keine feinen Bewegungen realisiert werden. Mit kleinen Zellen können detailliertere Simulationen gemacht werden, die wiederum aufwändiger sind. Die Auflösung darf auch nicht zu hoch sein, denn dann wäre es von Vorteil alles mit SPH zu simulieren. Mit einer geeigneten Relation zwischen den Größen der Partikel und der Gitterzellen können beide Systeme nutzbringend verbunden werden.

## Ausblick

Es wurde gezeigt, dass eine Kopplung zwischen SPH und Gitter möglich ist. In dem hier dafür vorgestelltem Ansatz bestand die Verknüpfung nur über den Massenaustausch und die Volumenberechnung innerhalb der Zellen. Es wäre vorstellbar die beiden Systeme noch stärker miteinander zu verbinden. Dafür könnte man zum Beispiel einen Austausch der Temperaturen mit rein nehmen oder die gegenseitig wirkenden Kräfte an den Grenzen der beiden Systeme untersuchen. Durch die Auswirkungen der Kräfte können die Strömungsfelder mit einander gekoppelt werden. Als Ergebnis wäre erwartbar, dass dann Partikel, die durch Kondensation in freier Luft entstehen, dass dich die Auftriebskraft des Dampfes auch nach oben gezogen werden. Umgekehrt würde sich die Bewegung der Partikel auf das Geschwindigkeitsfeld im Gitter auswirken.

Als weiteres können auch dynamische Anpassungen der Auflösungen zur Laufzeit untersucht werden. Für SPH und Gitter gibt es schon Untersuchungen diesbezüglich, jedoch nur als einzelne Systeme. So würde die Verdunstung oder Kondensation nur mit den kleinsten Partikeln durchgeführt. Sobald sie ihre vollständige Masse erreicht haben, können sie mit anderen Partikeln zu einem größeren Partikel verschmelzen. Umgekehrt muss ein Partikel in kleinere unterteilt werden, sobald eine Verdunstung in Sicht ist. Das würde dazu führen, dass die Vorgänge schneller abgeschlossen werden. Gitterzellen können auch hierarchisch unterteilt werden, um die Genauigkeit zu erhöhen. Da muss jedoch ein Ausgleich zwischen der Größe und dem Aufwand gefunden werden.

# Literaturverzeichnis


[AIA+12] Nadir Akinci, Markus Ihmsen, Gizem Akinci, Barbara Solenthaler, and Matthias Teschner. Versatile rigid-fluid coupling for incompressible sph. *ACM Transactions on Graphics (TOG)*, 31(4):62, 2012.

[APKG07] Bart Adams, Mark Pauly, Richard Keiser, and Leonidas J Guibas. Adaptively sampled particle fluids. In *ACM Transactions on Graphics (TOG)*, volume 26, page 48. ACM, 2007.

[BBB07] Christopher Batty, Florence Bertails, and Robert Bridson. A fast variational framework for accurate solid-fluid coupling. In *ACM Transactions on Graphics (TOG)*, volume 26, page 100. ACM, 2007.

[Bri08] Robert Bridson. *Fluid simulation for computer graphics*. CRC Press, 2008.

[BT07] Markus Becker and Matthias Teschner. Weakly compressible sph for free surface flows. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 209–217. Eurographics Association, 2007.

[CMT04] Mark Carlson, Peter J Mucha, and Greg Turk. Rigid fluid: animating the interplay between rigid bodies and fluid. In *ACM Transactions on Graphics (TOG)*, volume 23, pages 377–384. ACM, 2004.

[CN47] John Crank and Phyllis Nicolson. A practical method for numerical evaluation of solutions of partial differential equations of the heat-conduction type. In *Mathematical Proceedings of the Cambridge Philosophical Society*, volume 43, pages 50–67. Cambridge Univ Press, 1947.

[DC96] Mathieu Desbrun and Marie-Paule Cani. *Smoothed Particles: A new paradigm for animating highly deformable bodies*. Springer-Verlag, Poitiers, France, August 1996. Published under the name Marie-Paule Gascuel.

[FF01] Nick Foster and Ronald Fedkiw. Practical animation of liquids. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 23–30. ACM, 2001.

[FSJ01] Ronald Fedkiw, Jos Stam, and Henrik Wann Jensen. Visual simulation of smoke. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 15–22. ACM, 2001.

[GHD03] Olivier Génevaux, Arash Habibi, and Jean-Michel Dischler. Simulating fluid-solid interaction. In *Graphics Interface*, volume 2003, pages 31–38. Citeseer, 2003.

[GM77] Robert A Gingold and Joseph J Monaghan. Smoothed particle hydrodynamics: theory and application to non-spherical stars. *Monthly notices of the royal astronomical society*, 181(3):375–389, 1977.

[GSLF05] Eran Guendelman, Andrew Selle, Frank Losasso, and Ronald Fedkiw. Coupling water and smoke to thin deformable and rigid shells. *ACM Transactions on Graphics (TOG)*, 24(3):973–981, 2005.

[Har03] Mark Jason Harris. *Real-time cloud simulation and rendering*. PhD thesis, University of North Carolina at Chapel Hill, 2003.

[HW+65] Francis H Harlow, J Eddie Welch, et al. Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface. *Physics of fluids*, 8(12):2182, 1965.

[Kei06] Richard Keiser. Multiresolution particle-based fluids. 2006.

[KWm12] David B Kirk and W Hwu Wen-mei. *Programming massively parallel processors: a hands-on approach*. Newnes, 2012.

[Law05] Mark G Lawrence. The relationship between relative humidity and the dewpoint temperature in moist air: A simple conversion and applications. *Bulletin of the American Meteorological Society*, 86(2):225–233, 2005.

[LGF11] Michael Lentine, Jón Tómas Grétarsson, and Ronald Fedkiw. An unconditionally stable fully conservative semi-lagrangian method. *Journal of computational physics*, 230(8):2857–2879, 2011.

[Luc77] Leon B Lucy. A numerical approach to the testing of the fission hypothesis. *The astronomical journal*, 82:1013–1024, 1977.

[Mav97] DJ Mavriplis. Unstructured grid techniques. *Annual Review of Fluid Mechanics*, 29(1):473–514, 1997.

[MCG03] Matthias Müller, David Charypar, and Markus Gross. Particle-based fluid simulation for interactive applications. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 154–159. Eurographics Association, 2003.

[Mon92] Joe J Monaghan. Smoothed particle hydrodynamics. *Annual review of astronomy and astrophysics*, 30:543–574, 1992.

[Mon05] Joe J Monaghan. Smoothed particle hydrodynamics. *Reports on progress in physics*, 68(8):1703, 2005.

[MSKG05] Matthias Müller, Barbara Solenthaler, Richard Keiser, and Markus Gross. Particle-based fluid-fluid interaction. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 237–244. ACM, 2005.

[NCX15] Xiao Nie, Leiting Chen, and Tao Xiang. Real-time incompressible fluid simulation on the gpu. *International Journal of Computer Games Technology*, 2015, 2015.

[NFJ02] Duc Quang Nguyen, Ronald Fedkiw, and Henrik Wann Jensen. Physically based modeling and animation of fire. *ACM Transactions on Graphics (TOG)*, 21(3):721–728, 2002.

[Nvi] Nvidia Corporation. Cuda c programming guide. `https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html`. letzteter Zugriff: 08.11.2015.

[OHB+13] Jens Orthmann, Hendrik Hochstetter, Julian Bader, Serkan Bayraktar, and Andreas Kolb. Consistent surface model for sph-based fluid transport. In *Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 95–103. ACM, 2013.

[OK12] Jens Orthmann and Andreas Kolb. Temporal blending for adaptive sph. In *Computer Graphics Forum*, volume 31, pages 2436–2449. Wiley Online Library, 2012.

[RLY+14] Bo Ren, Chenfeng Li, Xiao Yan, Ming C Lin, Javier Bonet, and Shi-Min Hu. Multiple-fluid sph simulation using a mixture model. *ACM Transactions on Graphics (TOG)*, 33(5):171, 2014.

[SB12] Hagit Schechter and Robert Bridson. Ghost sph for animating water. *ACM Transactions on Graphics (TOG)*, 31(4):61, 2012.

[SG11] Barbara Solenthaler and Markus Gross. Two-scale particle simulation. In *ACM Transactions on Graphics (TOG)*, volume 30, page 81. ACM, 2011.

[Sha14] Mirza Mohammed Shah. Methods for calculation of evaporation from swimming pools and other water surfaces. 2014.

[SLJ94] Charles C Smith, George Löf, and Randy Jones. Measurement and analysis of evaporation from an inactive outdoor swimming pool. *Solar Energy*, 53(1):3–7, 1994.

[Son90] D Sonntag. Important new values of the physical constants of 1986, vapour pressure formulations based on the its-90, and psychrometer formulae. *Zeitschrift fur Meteorologie*, 70:340–344, 1990.

[SP08] Barbara Solenthaler and Renato Pajarola. Density contrast sph interfaces. In *Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 211–218. Eurographics Association, 2008.

[SP09] Barbara Solenthaler and Renato Pajarola. Predictive-corrective incompressible sph. In *ACM transactions on graphics (TOG)*, volume 28, page 40. ACM, 2009.

[Sta99] Jos Stam. Stable fluids. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 121–128. ACM Press/Addison-Wesley Publishing Co., 1999.

[SZMTW15] X Shao, Z Zhou, N Magnenat-Thalmann, and W Wu. Stable and fast fluid–solid coupling for incompressible sph. In *Computer Graphics Forum*, volume 34, pages 191–204. Wiley Online Library, 2015.