Bachelorarbeit

Analyse der Druckberechnung mithilfe einer Zustandsgleichung im Vergleich zur Lösung eines Gleichungssystems in SPH-Flüssigkeitssimulationen

Pascal Hunkler

Gutachter: Prof. Dr.-Ing. Matthias Teschner

Albert-Ludwigs-Universität Freiburg
Technische Fakultät
Institut für Informatik
Professur für Graphische Datenverarbeitung

27. September 2022

Bearbeitungszeit
27. 06. 2022 – 27. 09. 2022

Gutachter
Prof. Dr.-Ing. Matthias Teschner

Erklärung

Hiermit erkläre ich, dass ich diese Abschlussarbeit selbständig verfasst habe, keine anderen als die angegebenen Quellen/Hilfsmittel verwendet habe und alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten Schriften entnommen wurden, als solche kenntlich gemacht habe. Darüber hinaus erkläre ich, dass diese Abschlussarbeit nicht, auch nicht auszugsweise, bereits für eine andere Prüfung angefertigt wurde.

Ort, Datum	Unterschrift

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

SPH Smoothed Particle Hydrodynamics

ISPH Incompressible SPH

IISPH Implicit Incompressible SPH

SSPH Standard SPH

1 Einleitung

Die Darstellung von möglichst realistischen virtuellen Welten spielt heutzutage eine immer größere Rolle. Sie werden in vielerlei Anwendungsbereichen benötigt, wie zum Beispiel für visuelle Effekte in Game Engines für Videospiele, in der Filmindustrie oder auch in Virtual Reality Anwendungen. Auch in der Entwicklung von Produkten, wie zum Beispiel Autos, kann im Rahmen von Tests auf eine virtuelle Umgebung zurückgegriffen werden, anstatt die Tests kostenintensiv in einer realen Umgebung durchzuführen.

Um diese virtuellen Welten möglichst detailgetreu umzusetzen, muss auch das Verhalten von Flüssigkeiten möglichst realistisch simuliert werden. Hier hat im Laufe der Zeit die Methode *Smoothed Particle Hydrodynamics (SPH)* zur Berechnung von Flüssigkeitssimulationen an Bedeutung gewonnen. In SPH-basierten Ansätzen wird die Flüssigkeit in viele verschiedene so genannte Partikel aufgeteilt. Dazu müssen Beschleunigungen zwischen diesen Partikeln berechnet werden, um die Positionen der Partikel zu verändern.

Teil davon ist auch die Berechnung des Drucks der verschiedenen Partikel. Im Laufe der Zeit wurden verschiedene Methoden entwickelt, um den Druck zu berechnen. Dazu gibt es mehrere mögliche Ansätze. Zum einen lässt sich der Druck mithilfe einer Zustandsgleichung berechnen, in der der Druck eines Partikels nur von der lokalen Umgebung des Partikels abhängt. Zum anderen kann der Druck global berechnet werden, in dem beispielsweise ein Gleichungssystem iterativ gelöst wird. Motivation dazu ist, dass damit erreicht werden soll, dass die Flüssigkeit in einem inkompressiblen Zustand bleibt, während lokale Methoden nicht direkt die Inkompressibilität zum Ziel haben. Beispiele für globale Druckberechnungsmethoden sind *Incompressible SPH (ISPH)* [SL03], oder aber auch *Implicit Incompressible SPH (IISPH)* [ICS+14], welches nach Ihmsen et al. [ICS+14] eine deutlich besser Performanz und Genauigkeit als ISPH aufweist, und auch heute noch eine Rolle spielt.

Im Rahmen dieser Arbeit wird der Frage nachgegangen, welcher Ansatz zur Druckberechnung sich in welchem Fall eher empfehlen lässt. Dazu wurde eine Software entwickelt, in welcher verschiedene Szenarien simuliert werden können. Die Druckberechnung kann lokal durch eine Zustandsgleichung erfolgen oder aber auch global mit der IISPH-Methode.

Zu Beginn dieser Arbeit wird auf verschiedene Konzepte eingegangen. Dazu wird die Navier-Stokes-Gleichung vorgestellt, welche die grundlegende Gleichung darstellt, die im Rahmen verschiedener Ansätze approximativ gelöst werden muss, um

Flüssigkeitssimulationen zu realisieren. Es wird auf das SPH-Konzept eingegangen, durch welches in partikelbasierten Ansätzen verschiedene Größen an den Partikeln diskretisiert werden können. Zudem wird gezeigt, wie sich durch SPH eine einfache, grundlegende Flüssigkeitssimulation umsetzen lassen könnte.

Im Anschluss werden die beiden Druckmethoden vorgestellt, die im Rahmen dieser Arbeit auch implementiert werden. Zum einen die Druckberechnung durch eine einfache Zustandsgleichung und zum anderen die Druckberechnung mit der IISPH-Methode nach Ihmsen et al. [ICS+14]. Hier wird ein lineares Gleichungssystem aufgestellt, welches zur Druckberechnung gelöst werden muss. Dazu wird gezeigt, wie dieses Gleichungssystem iterativ mit dem relaxierten Jacobi-Verfahren gelöst werden kann.

Danach wird gezeigt, wie die Konzepte aus den vorigen beiden Kapiteln in der entwickelten Software, welche in der Programmiersprache C++ programmiert wurde, implementiert wurden. Dazu werden die Konzepte an die Bedingungen, die in der Implementierung vorherrschen, angepasst. Neben den Flüssigkeitspartikeln müssen in der Implementierung auch Grenzpartikel berücksichtigt werden, welche die Flüssigkeit begrenzen und somit verhindern, dass diese einfach auseinanderfließt. Zudem können einzelne Konponenten der den Konzepten zugrunde liegenden Gleichungen vereinfacht werden. Es werden außerdem die Algorithmen in Pseudocode gezeigt, die in der tatsächlichen Implementierung verwendet wurden.

Um schließlich die Frage zu beantworten, welche der beiden Druckmethoden sich in welchem Fall eher eignen, werden verschiedene Szenarien simuliert und ausgewertet. Für die beiden Druckberechnungsmethoden werden in der Genauigkeit möglichst vergleichbare Simulationen durchgeführt. Anhand der daraus gewonnenen Daten wird die Performanz der beiden Methoden miteinander verglichen.

Zum Schluss wird ein Fazit aus diesen Ergebnissen und Erfahrungen gezogen und ein Ausblick auf zukünftige Entwicklungen gegeben.

2 Grundlagen

2.1 Navier-Stokes-Gleichung und Flüssigkeitssimulationen

Dieser Abschnitt behandelt die Navier-Stokes-Gleichung, welche die Grundlage dafür bildet, Flüssigkeitssimulationen zu realisieren. Sie beschreibt die Änderungsrate der Geschwindigkeit eines kleinen volumetrischen Flüssigkeitelements. Für die Navier-Stokes-Gleichung gibt es verschiedene Formen, die auf den eulerschen Ansatz oder den lagrangeschen Ansatz beruhen. Diese Arbeit beschäftigt sich im Wesentlichen auf lagrangesche, partikelbasierte Simulationen, der Vollständigkeit wegen wird jedoch auch noch kurz auf eulersche, gitterbasierte Ansätze eingegangen. Die Inhalte dieses Abschnitts basieren auf der Arbeit von Ihmsen et al. [IOS+14].

2.1.1 Partikelbasierte Simulation

In einer lagrangeschen, partikelbasierten Flüssigkeitssimulation wird die Flüssigkeit in so genannte Partikel unterteilt, die sich mit der Flüssigkeit mitbewegen. Jedes Partikel nimmt ein Teilvolumen der Flüssigkeit ein und besitzt eine Masse. In einem partikelbasierten Ansatz kann die Navier-Stokes-Gleichung wie folgt beschrieben werden:

$$\frac{d\mathbf{v}_i}{dt} = -\frac{1}{\rho_i}\nabla p_i + \nu\nabla^2\mathbf{v}_i + \frac{\mathbf{F}_i^{other}}{m_i} \tag{2.1}$$

Sie gibt die Änderungsrate der Geschwindigkeit $\frac{d\mathbf{v}_i}{dt}$ eines beweglichen Partikels mit Positionen $\mathbf{x}_i$ nach Zeit an, welche durch verschiedene Beschleunigungen beeinflusst wird: Die Druckbeschleunigung $-\frac{1}{\rho_i}\nabla p_i$, die Viskositätsbeschleunigung $\nu\nabla^2\mathbf{v}_i$, die durch die Viskosität ν beeinflusst wird, und der Einfluss anderer Kräfte $\frac{\mathbf{F}_i^{other}}{m_i}$, wie beispielsweise die Gravitation. Ziel einer partikelbasierten Simulation ist es, die verschiedenen Beschleunigungen zu berechnen, um die Geschwindigkeiten und Positionen der Partikel stets anzupassen. Eine Methode, um die Beschleunigungen an den Partikeln zu berechnen, ist Smoothed Particle Hydrodynamics (SPH), welches im nächsten Abschnitt vorgestellt wird. In partikelbasierten Simulationen muss regelmäßig die Nachbarschaft eines Partikels bestimmt werden, da diese frei beweglich sind und sich somit die Nachbarschaft ändert. Die Nachbarschaft eines Partikels wird benötigt, um die Beschleunigungen an einem Partikel mit SPH zu bestimmen. Im

Gegensatz zu gitterbasierten Simulationen, welche im nächsten Unterabschnitt vorgestellt werden, muss hier jedoch ein Term aus der Navier-Stokes-Gleichung weniger berechnet werden.

2.1.2 Gitterbasierte Simulation

In eulerschen, gitterbasierten Simulationen wird im Gegensatz zu partikelbasierten Simulationen mit Gittern statt Partikeln gearbeitet. Hier wird die Flüssigkeit nicht in frei bewegliche Partikel unterteilt, sondern in Teilvolumina, die fest an einem Ort sind und meist ein Gitter bilden. Die Navier-Stokes-Gleichung hat durch diesen anderen Ansatz folgende Form:

$$\frac{\partial \mathbf{v}_i}{\partial t} = -\frac{1}{\rho_i}\nabla p_i + \nu \nabla^2 \mathbf{v}_i + \frac{\mathbf{F}_i^{other}}{m_i} - \mathbf{v}_i \cdot \nabla \mathbf{v}_i \tag{2.2}$$

Betrachtet wird hier die Änderungsrate der Geschwindigkeit $\frac{\partial \mathbf{v}_i}{\partial t}$ an einer festen Position $\mathbf{x}_i$. Dadurch, dass die Positionen fest sind, kommt in der Navier-Stokes-Gleichung neben den Beschleunigungen noch der Term $-\mathbf{v}_i \cdot \nabla \mathbf{v}_i$ vor, welcher die Bewegung der Flüssigkeit ausgleicht. Im Gegensatz zu partikelbasierten Simulationen muss hier jedoch direkt erst einmal keine Nachbarschaftssuche stattfinden, da die Punkte nicht frei beweglich sind. Generell kann jedoch die Performanz und Genauigkeit zwischen partikelbasierten und gitterbasierten Ansätzen nicht allgemein verglichen werden.

2.2 SPH

Das Konzept Smoothed Particle Hydrodynamics (SPH), ursprünglich formuliert von Lucy [Luc77] und unabhängig davon von Monaghan und Gingold [GM77], entstand ursprünglich aus dem Bereich der Astrophysik, wird aber heute auch in diversen anderen Bereichen, unter anderem auch der Computergrafik angewandt. Mithilfe von SPH kann durch Diskretisierung und Berechnung von Größen wie der Druckbeschleunigung oder der Viskositätsbeschleunigung die Navier-Stokes-Gleichung gelöst werden. Es eignet sich daher gut für partikelbasierte Simulationen.

2.2.1 Diskretisierung mit SPH

Die Inhalte dieses Abschnittes basieren auf den Arbeiten von Monaghan [Mon05], von Price [Pri12] und von Koshier et al. [KBST20]. Für eine beliebige skalare Variable A gilt die Identität

$$A(\mathbf{x}) = \int A(\mathbf{x}')\delta(\mathbf{x} - \mathbf{x}')d\mathbf{x}' \tag{2.3}$$

δ ist hierbei die Dirac'sche Deltafunktion, die definiert ist als

$$\delta(\mathbf{x}) = \begin{cases} \infty, & \text{falls } \mathbf{x} = 0 \\ 0, & \text{sonst} \end{cases} \tag{2.4}$$

Die Dirac'sche Deltafunktion in Gleichung 2.3 kann mithilfe einer glättenden Kernelfunktion W mit endlicher Breite h approximiert werden.

$$A(\mathbf{x}) = \int A(\mathbf{x}')W(\mathbf{x} - \mathbf{x}', h)d\mathbf{x}' + O(h^2) \tag{2.5}$$

Damit die Approximation aus Gleichung 2.5 gültig ist, muss W folgende Eigenschaften besitzen:

$$\int_{\mathbb{R}^d} W(\mathbf{x}', h)dv' = 1 \qquad \text{(Normalisierung)} \tag{2.6}$$

$$\lim_{h\to 0} W(\mathbf{x}, h) = \delta(\mathbf{x}) \qquad \text{(Dirac-}\delta\text{)} \tag{2.7}$$

Weitere wünschenswerte Eigenschaften der Kernelfunktion W sind:

$$W(\mathbf{x}, h) \geq 0 \qquad \text{(Positivität)} \tag{2.8}$$

$$W(\mathbf{x}, h) = W(-\mathbf{x}, h) \qquad \text{(Symmetrie)} \tag{2.9}$$

$$W(\mathbf{x}, h) = 0 \text{ für } \|\mathbf{x}\| \geq \hbar \qquad \text{(Kompakter Support)} \tag{2.10}$$

$\forall \mathbf{x} \in \mathbb{R}^d, h \in \mathbb{R}^+$, während $\hbar$ der Kernelsupport ist. Eine beliebte Kernelfunktion ist der Cubic Spline Kernel [Mon92]. Er ist definiert als

$$W(\mathbf{x}, h) = \sigma_d \begin{cases} (2-q)^3 - 4(1-q)^3, & \text{für } 0 \leq q \leq 1 \\ (2-q)^3, & \text{für } 1 \leq q \leq 2 \\ 0, & \text{sonst} \end{cases} \tag{2.11}$$

mit $q = \frac{1}{h}\|\mathbf{x}\|$. Der Kernelnormalisierungsfaktor σ_d ist abhängig von der Dimension d und beträgt für $d = 1, 2, 3$ $\sigma_1 = \frac{1}{6h}, \sigma_2 = \frac{5}{14\pi h^2}, \sigma_1 = \frac{1}{4\pi h^3}$. In der Literatur gibt es verschiedene Formulierungen für den Cubic Spline Kernel, die sich im Wesentlichen in der Parametrisierung unterscheiden. Der Vorteil dieser Kernelfunktion ist, dass die Eigenschaften zu Positivität, Symmetrie, und kompakten Support, erfüllt werden. Zudem erzielt Cubic Spline trotz seiner Einfachkeit gute Ergebnisse.

Um die Interpolation aus Gleichung 2.5 bei einer Flüssigkeit zu diskretisieren, wird die Flüssigkeit in mehrere Partikel unterteilt. Jedes Partikel f besitzt eine Masse m_f, Dichte ρ_f und Position $\mathbf{x}_f$. Der Wert von A an einem Partikel f wird notiert als A_f, die Approximation davon als $\langle A_f \rangle$. Der Integral aus Gleichung 2.5 kann nun durch eine Summe, und die Masse ρdV durch die Partikelmasse m_f ersetzt werden.

$$A(\mathbf{x}) = \int \frac{A(\mathbf{x}')}{\rho(\mathbf{x}')} W(\mathbf{x} - \mathbf{x}', h)\rho(\mathbf{x}')d\mathbf{x}' + O(h^2) \tag{2.12}$$

$$\approx \sum_{f_f} m_{f_f} \frac{A_{f_f}}{\rho_{f_f}} W(\mathbf{x}_f - \mathbf{x}_{f_f}, h) =: \langle A_f \rangle \tag{2.13}$$

f_f sind hierbei alle Partikel der Flüssigkeit. $W(\mathbf{x}_f - \mathbf{x}_{f_f}, h)$ wird ab nun durch W_{ff_f} abgekürzt. Da beispielsweise der Cubic Spline Kernel einen Kernelsupport von $2h$ besitzt, muss hier nur über Partikel f_f summiert werden, die in direkter Nachbarschaft sind, da die Kernelfunktion für Partikel, die weiter als $2h$ von dem Partikel f entfernt sind, null ist. Daher ist die Eigenschaft, dass der Support der Kernelfunktion kompakt ist, wünschenswert.

Häufig ist es auch nötig, die Differentialoperatoren zu diskretisieren. Der Gradient einer Variable A kann durch folgende Approximation bestimmt werden:

$$\nabla A_f \approx \sum_{f_f} A_{f_f} \frac{m_{f_f}}{\rho_{f_f}} \nabla W_{ff_f} \tag{2.14}$$

Diese Approximation hat eine Genauigkeit 0-ter Ordnung, wenn folgende Bedingung erfüllt wird:

$$\sum_{f_f} \frac{m_{f_f}}{\rho_{f_f}} \nabla W_{ff_f} = \mathbf{0} \tag{2.15}$$

Wird zusätzlich die Bedingung

$$\sum_{f_f} \frac{m_{f_f}}{\rho_{f_f}} (\mathbf{x}_{f_f} - \mathbf{x}_f) \otimes \nabla W_{ff_f} = \not\Vdash \tag{2.16}$$

erfüllt, hat sie eine Genauigkeit erster Ordnung. Ist $\mathbf{A}$ höherdimensional, so können komplexere Differentialoperatoren wie folgt diskretisiert werden:

$$\nabla \mathbf{A}_f \approx \sum_{f_f} \frac{m_{f_f}}{\rho_{f_f}} \mathbf{A}_{f_f} \otimes \nabla W_{ff_f} \tag{2.17}$$

$$\nabla \cdot \mathbf{A}_f \approx \sum_{f_f} \frac{m_{f_f}}{\rho_{f_f}} \mathbf{A}_{f_f} \cdot \nabla W_{ff_f} \tag{2.18}$$

$$\nabla \times \mathbf{A}_f \approx -\sum_{f_f} \frac{m_{f_f}}{\rho_{f_f}} \mathbf{A}_{f_f} \times \nabla W_{ff_f} \tag{2.19}$$

wobei $a \otimes b = ab^T$ das dyadische Produkt ist. Leider führen diese „direkten“ Ableitungen zu einer schlechten Approximationsqualität und instabilen Simulationen. Daher gibt es mittlerweile viele alternative Formulierungen für diese Differentialoperatoren. Zwei davon werden im nächsten Unterabschnitt gezeigt.

2.2.2 SPH in partikelbasierten Simulationen

Dieser Unterabschnitt basiert auf die Arbeit von Koshier et al. [KBST20]. Ziel einer partikelbasierten Simulation ist es, die Beschleunigungen zu bestimmen, die in

der Navier-Stokes-Gleichung angegeben sind. Die Navier-Stokes-Gleichung für einen partikelbasierten Ansatz hat folgende Form:

$$\frac{d\mathbf{v}_f}{dt} = -\frac{1}{\rho_f}\nabla p_f + \nu\nabla^2\mathbf{v}_f + \frac{\mathbf{F}_f^{other}}{m_f} \tag{2.20}$$

Es muss also in erster Linie die Druckbeschleunigung $-\frac{1}{\rho_f}\nabla p_f$ und die Viskositätsbeschleunigung $\nu\nabla^2\mathbf{v}_f$ bestimmt werden. Die SPH Diskretisierung einer Variable A sieht wie folgt aus:

$$A_f \approx \sum_{f_f} m_{f_f}\frac{A_{f_f}}{\rho_{f_f}}W_{ff_f} \tag{2.21}$$

Hieraus ist ersichtlich, dass zur Berechnung von A die Dichte ρ_f aller Partikel benötigt wird. Diese kann jedoch leicht ermittelt werden, indem in Gleichung 2.21 A durch ρ ersetzt wird. Der Faktor $\frac{\rho_{f_f}}{\rho_{f_f}}$ kürzt sich weg und man erhält:

$$\rho_f \approx \sum_{f_f} m_{f_f}\frac{\rho_{f_f}}{\rho_{f_f}}W_{ff_f} = \sum_{f_f} m_{f_f}W_{ff_f} \tag{2.22}$$

Für die Berechnung der Druckbeschleunigung $-\frac{1}{\rho_f}\nabla p_f$ wird die Diskretisierung des Druckgradienten benötigt. Wie der Druck berechnet werden kann, wird im nächsten Kapitel vorgestellt. Ist der Druck berechnet, ist es naheliegend, die normale SPH Diskretisierung für den Gradienten zu verwenden:

$$-\frac{1}{\rho_f}\nabla p_f \approx -\frac{1}{\rho_f}\sum_{f_f} A_{f_f}\frac{m_{f_f}}{\rho_{f_f}}\nabla W_{ff_f} \tag{2.23}$$

Das Problem bei dieser Näherung ist jedoch, dass die daraus resultierende Beschleunigung nicht den Impuls und Drehimpuls der Flüssigkeit erhält, was zu einer instabilen Simulation führt. Für die Druckberechnung wird häufig stattdessen eine alternative Approximation verwendet, die auf folgender Diskretisierung des Gradienten einer Variable A beruht:

$$\nabla A_f \approx \rho_f \sum_{f_f} m_{f_f}\left(\frac{A_f}{\rho_f^2} + \frac{A_{f_f}}{\rho_{f_f}^2}\right)\nabla W_{ff_f} \tag{2.24}$$

Der Vorteil an dieser Approximation ist, dass das eben genannte Problem, dass daraus resultierende Kräfte oder Beschleunigungen den Impuls und Drehimpuls der Flüssigkeit nicht erhalten, behoben wird. Dies ist essentiell für robuste, stabile Simulationen. Der Nachteil, der in Kauf genommen wird, ist, dass nun konstante oder lineare Gradienten nicht mehr exakt berechnet werden können. Berechnet man die Druckbeschleunigung anhand dieser Diskretisierung erhält man:

$$-\frac{1}{\rho_f}\nabla p_f \approx -\sum_{f_f} m_{f_f}\left(\frac{p_f}{\rho_f^2} + \frac{p_{f_f}}{\rho_{f_f}^2}\right)\nabla W_{ff_f} \tag{2.25}$$

Nun muss noch die Viskositätsbeschleunigung $\nu\nabla^2\mathbf{v}_f$ berechnet werden. Die normale SPH Diskretisierung für den Laplace Operator ist bestimmt durch

$$\nabla^2\mathbf{v}_f = \sum_{f_f} \frac{m_{f_f}}{\rho_{f_f}} \mathbf{v}_{f_f} \nabla^2 W_{ff_f} \tag{2.26}$$

Es hat sich jedoch gezeigt, dass die Verwendung der zweiten Ableitung der Kernelfunktion sensibel ist gegenüber Störungen der Partikelanordnung und zu schlechteren Ergebnissen führt als eine alternative Formulierung, die ohne die zweite Ableitung auskommt:

$$\nabla^2\mathbf{v}_f = 2(d+2) \sum_{f_f} \frac{m_{f_f}}{\rho_{f_f}} \frac{\mathbf{v}_{ff_f} \cdot \mathbf{x}_{ff_f}}{\|\mathbf{x}_{ff_f}\| + 0.01h^2} \nabla W_{ff_f} \tag{2.27}$$

$\mathbf{x}_{ff_f} = \mathbf{x}_f - \mathbf{x}_{f_f}$, $\mathbf{x}_{ff_f} = \mathbf{x}_f - \mathbf{x}_{f_f}$ und d ist die Anzahl der räumlichen Dimensionen. Schließlich erhält man zur Berechnung der Viskositätsbeschleunigung [KBST20]:

$$\nu\nabla^2\mathbf{v}_f = 2(d+2)\nu \sum_{f_f} \frac{m_{f_f}}{\rho_{f_f}} \frac{\mathbf{v}_{ff_f} \cdot \mathbf{x}_{ff_f}}{\|\mathbf{x}_{ff_f}\| + 0.01h^2} \nabla W_{ff_f} \tag{2.28}$$

Diese Formulierung erfüllt einige wünschenswerte Eigenschaften: Sie ist galilei-invariant, was bedeutet, dass beispielsweise in einer sich gleichmäßig bewegenden Flüssigkeit die Viskositätsbeschleunigung nicht sich anders verhält als in einer äquivalenten ruhenden Flüssigkeit. Es entsteht unter anderem auch keine Reibung, wenn alle Partikel gleichmäßig rotieren. Zudem wird der Impuls und Drehimpuls erhalten.

2.3 Zeitintegration

Sind in einer partikelbasierten Simulation die Beschleunigungen für den Druck, die Viskosität und die restlichen Beschleunigungen, wie z.B. die Gravitation berechnet, muss eine numerische Zeitintegration durchgeführt werden, um die Geschwindigkeiten und Positionen der Partikel zu aktualisieren. Häufig wird dazu das Euler-Cromer Verfahren genutzt. [KBST20]

Die Position $\mathbf{x}_f^{t+\Delta t}$ und Geschwindigkeit $\mathbf{v}_f^{t+\Delta t}$ eines Partikels f zum Zeitpunkt $t + \Delta t$ wird hier aus der berechneten Beschleunigung $\mathbf{a}_f^t$ zum Zeitpunkt t wie folgt aktualisiert:

$$\mathbf{v}_f^{t+\Delta t} = \mathbf{v}_f^t + \Delta t \mathbf{a}_f^t \tag{2.29}$$

$$\mathbf{x}_f^{t+\Delta t} = \mathbf{x}_f^t + \Delta t \mathbf{v}_f^{t+\Delta t} \tag{2.30}$$

Δt ist hierbei die vergangene Zeit zwischen zwei Simulationsschritten, und wird auch als Zeitschritt bezeichnet.

Die Wahl des Zeitschritts Δt ist entscheidend für die Performanz und Stabilität der Simulation. Generell ist es gewollt, möglichst wenige Simulationsschritte berechnen

zu müssen, damit die Simulation insgesamt schneller berechnet wird. Daher wird ein möglichst großer Zeitschritt Δt gewählt. Ist der Zeitschritt jedoch zu groß gewählt, wird die numerische Integration ungenauer, was zu instabilen Simulationen führt, die zusammenbrechen. Um einen möglichst großen Zeitschritt zu wählen, der noch zu einer stabilen Simulation führt, kann sich an der CFL-Bedingung orientiert werden. Die CFL-Bedingung begrenzt den Zeitschritt Δt abhängig eines Parameters λ, der Partikelgröße $\hbar$, und der Geschwindigkeit des schnellsten Partikels $\mathbf{v}^{max}$:

$$\Delta t \leq \lambda \frac{\hbar}{\|\mathbf{v}^{max}\|} \tag{2.31}$$

Die Idee dahinter ist, dass für z.B. $\lambda = 1$ sich die Partikel nicht weiter als eine Partikelgröße bewegen. [KBST20]

2.4 Grundlegender Aufbau einer partikelbasierten Flüssigkeitssimulation

Dieser Abschnitt beschreibt, wie sich die Konzepte aus den vorigen Abschnitten und der Druckberechnung, die im nächsten Kapitel erklärt wird, kombinieren lassen, um daraus eine einfache Flüssigkeitssimulation umzusetzen. Zuerst wird für jedes Partikel f die Dichte berechnet mithilfe folgender Gleichung:

$$\rho_f = \sum_{f_f} m_{f_f} W_{ff_f} \tag{2.32}$$

Dann wird die Viskositätsbeschleunigung berechnet:

$$\mathbf{a}_f^v = 2(d+2)\nu \sum_{f_f} \frac{m_{f_f}}{\rho_{f_f}} \frac{\mathbf{v}_{ff_f} \cdot \mathbf{x}_{ff_f}}{\|\mathbf{x}_{ff_f}\| + 0.01h^2} \nabla W_{ff_f} \tag{2.33}$$

Es wird dann eine vorläufige Geschwindigkeit $\mathbf{v}_f^*$ der Partikel bestimmt, die unter dem Einfluss der Viskositätsbeschleunigung und anderen Beschleunigungen wie z.B. der Gravitation aus der ursprünglichen Geschwindigkeit $\mathbf{v}_f^t$ durch eine Zeitintegration berechnet wird:

$$\mathbf{v}_f^* = \mathbf{v}_f^t + \Delta t \left(\mathbf{a}_f^v + \mathbf{a}_f^{ext}\right) \tag{2.34}$$

Nun muss noch der Druck der Partikel p_f berechnet werden. Dies wird im nächsten Kapitel beschrieben. Wenn der Druck berechnet ist, wird die Druckbeschleunigung an jedem Partikel berechnet:

$$\mathbf{a}_f^p = -\sum_{f_f} m_{f_f} \left(\frac{p_f}{\rho_f^2} + \frac{p_{f_f}}{\rho_{f_f}^2}\right) \nabla W_{ff_f} \tag{2.35}$$

Anschließend werden die Geschwindigkeit und Position jedes Partikels aktualisiert:

$$\mathbf{v}_f^{t+\Delta t} = \mathbf{v}_f^* + \Delta t \mathbf{a}_f^p \tag{2.36}$$

$$\mathbf{x}_f^{t+\Delta t} = \mathbf{x}_f^t + \Delta t \mathbf{v}_f^{t+\Delta t} \tag{2.37}$$

3 Druckberechnung

Dieses Kapitel behandelt die Druckberechnung in partikelbasierten Flüssigkeitssimulationen. Die Berechnung des Drucks ist Voraussetzung für die Bestimmung der Druckbeschleunigung, welche Teil der Navier-Stokes-Gleichung ist. Es werden zwei Methoden vorgestellt, um den Druck zu bestimmen: Die Druckberechnung mithilfe einer Zustandsgleichung und das globale Berechnen des Drucks, welches von einer inkompressiblen Flüssigkeit ausgeht, mithilfe von Implicit Incompressible SPH (IISPH) [ICS+14], welches auf Incompressible SPH (ISPH) [SL03] aufbaut.

3.1 Druckberechnung mit einer Zustandsgleichung

Eine Methode, um den Druck zu berechnen, ist das Verwenden einer Zustandsgleichung. Hierbei wird der Druck direkt aus der Abweichung der berechneten Dichte ρ_f eines Partikels von der Ruhedichte ρ^0 bestimmt. Ein Beispiel für eine solche Gleichung ist folgende:

$$p_f = \max\left(k\left(\frac{\rho_f}{\rho^0} - 1\right), 0\right) \tag{3.1}$$

Der Druck ist immer positiv, da es sonst zu Problemen kommt, wenn $\rho_f < \rho^0$, was vor allem an der Oberfläche der Flüssigkeit auftritt, da die Partikel an der Oberfläche weniger Nachbarn haben. Die Konstante k ist eine Steifigkeitskonstante. In der Arbeit von Koschier et al. [KBST20] ist beschrieben, dass die Wahl von k einen Einfluss auf die Dichteabweichung hat. Größere Werte für k führen zu einer geringeren Dichteabweichung, jedoch muss ein kleinerer Zeitschritt gewählt werden, da die Simulation sonst instabil wird. Kleinere Werte für k hingegen führen zu größeren Dichteabweichungen und damit auch zu weniger realistischen Simulationen. Anschaulich erklären lässt sich dies anhand eines Beispiels mit einer ruhenden Flüssigkeit, die unter dem Einfluss von Gravitation steht. Da die Flüssigkeit ruht, gilt

$$\mathbf{g} - \frac{1}{\rho_f}\nabla p_f = \mathbf{0} \tag{3.2}$$

Durch Diskretisierung des Druckgradienten erhält man

$$\mathbf{g} = \sum_{f_f} m_{f_f}\left(\frac{p_f}{\rho_f^2} + \frac{p_{f_f}}{\rho_{f_f}^2}\right)\nabla W_{ff_f} \tag{3.3}$$

Verwendet man beispielsweise die Zustandsgleichung $p_f = k(\rho_f - \rho^0)$ erhält man daraus

$$\mathbf{g} = k \sum_{f_f} m_{f_f} \left(\frac{\rho_f - \rho^0}{\rho_f^2} + \frac{\rho_{f_f} - \rho^0}{\rho_{f_f}^2} \right) \nabla W_{ff_f} \tag{3.4}$$

Die Gravitation ist konstant, also kann man aus dieser Gleichung schließen, dass sich insgesamt aus einer großen Steifigkeitskonstante k geringere Dichteabweichungen ergeben.

3.2 Druckberechnung mit IISPH

Dieser Abschnitt behandelt die Druckberechnung mit der IISPH-Methode. Implicit Incompressible SPH (IISPH) wurde erstmals von Ihmsen et al. entwickelt [ICS+14], auf deren Arbeit dieser Abschnitt auch basiert. Im Gegensatz zu Standard SPH (SSPH), also der Druckberechnung durch eine Zustandsgleichung, wird hier ein lineares Gleichungssystem gelöst, um den Druck der Partikel zu berechnen.

3.2.1 Herleitung des Gleichungssystems

Um das lineare Gleichungssystem herzuleiten, wird erst einmal die Kontinuitätsgleichung betrachet:

$$\frac{D\rho}{Dt} = -\rho \nabla \cdot \mathbf{v} \tag{3.5}$$

Die Kontinuitätsgleichung stellt einen Zusammenhang her zwischen der Änderungsrate der Dichte und der Divergenz des Geschwindigkeitsfeldes. Diese Gleichung wird dann diskretisiert, indem die Änderungsrate der Dichte durch den Vorwärtsdifferenzquotienten $\frac{\rho_f(t+\Delta t) - \rho_f(t)}{\Delta t}$ ersetzt und die Divergenz des Geschwindigkeitsfeldes durch das SPH-Konzept der Divergenz $\nabla \cdot \mathbf{v}_f = -\frac{1}{\rho_f} \sum_{f_f} m_{f_f} \mathbf{v}_{ff_f} \nabla W_{ff_f}$. Dadurch erhält man:

$$\frac{\rho_f(t + \Delta t) - \rho_f(t)}{\Delta t} = \sum_{f_f} m_{f_f} \mathbf{v}_{ff_f}(t + \Delta t) \nabla W_{ff_f}(t) \tag{3.6}$$

$\mathbf{v}_{ff_f}(t + \Delta t)$ ist hierbei $\mathbf{v}_f(t + \Delta t)$ - $\mathbf{v}_{f_f}(t + \Delta t)$. Diese Geschwindigkeiten basieren auf Druckbeschleunigungen, welche linear in den Werten für den Druck sind, da

$$\mathbf{a}_f^p = -\sum_{f_f} m_{f_f} \left(\frac{p_f}{\rho_f^2} + \frac{p_{f_f}}{\rho_{f_f}^2} \right) \nabla W_{ff_f} \tag{3.7}$$

Verwendet man das Euler-Cromer Verfahren, so kann die Geschwindigkeit eines Partikels zum Zeitpunkt $t + \Delta t$ wie folgt geschrieben werden:

$$\mathbf{v}_f(t + \Delta t) = \mathbf{v}_f(t) + \Delta t \left(\mathbf{a}_f^{nonp} + \mathbf{a}_f^p \right) \tag{3.8}$$

$\mathbf{a}_f^{nonp}$ ist hierbei die Beschleunigung, die durch die Viskosität und andere Kräfte wie die Gravitation, entsteht. Wir betrachten ebenfalls die geschätzte Zwischen-Geschwindigkeit $\mathbf{v}_f^*$:

$$\mathbf{v}_f^* = \mathbf{v}_f(t) + \Delta t \mathbf{a}_f^{nonp} \tag{3.9}$$

Durch diese Zwischen-Geschwindigkeit entsteht auch eine Zwischen-Dichte

$$\rho_f^* = \rho_f(t) + \Delta t \sum_{f_f} m_{f_f} \mathbf{v}_{ff_f}^* \nabla W_{ff_f}(t) \tag{3.10}$$

Zudem gilt:

$$\mathbf{v}_f(t + \Delta t) = \mathbf{v}_f^* + \Delta t \mathbf{a}_f^p \tag{3.11}$$

Setzt man (3.11) in (3.6) ein, kann man Umformen:

$$\frac{\rho_f(t + \Delta t) - \rho_f(t)}{\Delta t} = \sum_{f_f} m_{f_f} \mathbf{v}_{ff_f}(t + \Delta t) \nabla W_{ff_f}(t) \tag{3.12}$$

$$= \sum_{f_f} m_{f_f} \mathbf{v}_{ff_f}^* \nabla W_{ff_f}(t) + \Delta t \sum_{f_f} m_{f_f} \mathbf{a}_{ff_f}^p \nabla W_{ff_f}(t) \tag{3.13}$$

$$= \frac{\rho_f^* - \rho_f(t)}{\Delta t} + \Delta t \sum_{f_f} m_{f_f} \mathbf{a}_{ff_f}^p \nabla W_{ff_f}(t) \tag{3.14}$$

Da wir erreichen wollen, dass die Dichte $\rho_f(t + \Delta t)$ die Ruhedichte ist, damit die Partikel in einem unkomprimierten Zustand sind, wird $\rho_f(t+\Delta t) = \rho^0$ angenommen. Mit Umformen erhält man dann:

$$\Delta t^2 \sum_{f_f} m_{f_f} \left(\mathbf{a}_f^p - \mathbf{a}_{ff_f}^p \right) \nabla W_{ff_f}(t) = \rho^0 - \rho_f^* \tag{3.15}$$

Setzt man nun (3.7) in (3.15) ein, ist der Druck linear und man erhält ein lineares Gleichungssystem

$$\mathbf{A}(t)\mathbf{p}(t) = \mathbf{b}(t) \tag{3.16}$$

Für jedes Partikel haben wir eine Gleichung der Form

$$\sum_{f_f} A_{ff_f} p_{f_f} = b_f = \rho_0 - \rho_f^* \tag{3.17}$$

3.2.2 Lösung des Gleichungssystems

Das Gleichungssystem, das gelöst werden muss, um den Druck zu berechnen, kann mit verschiedenen Methoden gelöst werden. Empfohlen von Ihmsen et al. [ICS+14] ist es, das relaxierte Jacobi-Verfahren anzuwenden. Die Herangehensweise dazu ist

in [ICS+14] ebenfalls beschrieben, worauf dieser Unterabschnitt auch basiert. Hierzu werden die Werte für den Druck p_f wie folgt iterativ gelöst:

$$p_f^{l+1} = (1-\omega)p_f^l + \omega \frac{\rho^0 - \rho_f^* - \sum_{f_f \neq f} A_{ff_f} p_{f_f}^l}{A_{ff}} \tag{3.18}$$

l ist hierbei der Iterationsindex und ω der Relaxierungsfaktor. Um dies zu berechnen, werden die Diagonalwerte A_{ff} der Matrix $\mathbf{A}$ benötigt und $\sum_{f_f \neq f} A_{ff_f} p_{f_f}^l$. Diese können jedoch effektiv berechnet werden. Um die Koeffezienten zu extrahieren, betrachten wir den Term $\Delta t^2 \mathbf{a}_f^p$:

$$\Delta t^2 \mathbf{a}_f^p = -\Delta t^2 \sum_{f_f} m_{f_f} \left(\frac{p_f}{\rho_f^2} + \frac{p_{f_f}}{\rho_{f_f}^2} \right) \nabla W_{ff_f} \tag{3.19}$$

$$= \underbrace{\left(-\Delta t^2 \sum_{f_f} \frac{m_{f_f}}{\rho_f^2} \nabla W_{ff_f} \right)}_{\mathbf{d}_{ff}} p_f + \sum_{f_f} \underbrace{-\Delta t^2 \frac{m_{f_f}}{\rho_{f_f}^2} \nabla W_{ff_f}}_{\mathbf{d}_{ff_f}} p_{f_f} \tag{3.20}$$

Dies kann nun in (3.15) eingesetzt werden, daraus erhält man:

$$\rho^0 - \rho_f^* = \sum_{f_f} m_{f_f} \left(\mathbf{d}_{ff} p_f + \sum_{f_f} \mathbf{d}_{ff_f} p_{f_f} - \mathbf{d}_{f_f f} p_{f_f} - \sum_{f_{f_f}} \mathbf{d}_{f_f f_{f_f}} p_{f_{f_f}} \right) \nabla W_{ff_f} \tag{3.21}$$

Beachtet werden muss, dass in $\sum_{f_{f_f}} \mathbf{d}_{f_f f_{f_f}} p_{f_{f_f}}$ auch Druckwerte p_f vorkommen, da f und f_f Nachbarn sind. Um diese Werte zu separieren, können wir auch schreiben

$$\sum_{f_{f_f}} \mathbf{d}_{f_f f_{f_f}} p_{f_{f_f}} = \sum_{f_{f_f} \neq f} \mathbf{d}_{f_f f_{f_f}} p_{f_{f_f}} + \mathbf{d}_{f_f f} p_f \tag{3.22}$$

Nun können wir die Teile mit Werten p_f von den Teilen mit Werten p_{f_f} und $p_{f_{f_f}}$ in (3.21) trennen. Daraus erhält man:

$$\rho^0 - \rho_f^* = p_f \sum_{f_f} m_{f_f} \left(\mathbf{d}_{ff} - \mathbf{d}_{f_f f} \right) \nabla W_{ff_f} + \tag{3.23}$$

$$\sum_{f_f} m_{f_f} \left(\sum_{f_f} \mathbf{d}_{ff_f} p_{f_f} - \mathbf{d}_{f_f f} p_f - \sum_{f_{f_f} \neq f} \mathbf{d}_{f_f f_{f_f}} p_{f_{f_f}} \right) \nabla W_{ff_f} \tag{3.24}$$

Jetzt können die Koeffizienten A_{ff} einfach berechnet werden durch

$$A_{ff} = \sum_{f_f} m_{f_f} \left(\mathbf{d}_{ff} - \mathbf{d}_{f_f f} \right) \nabla W_{ff_f} \tag{3.25}$$

Der iterative Schritt in (3.18) kann nun berechnet werden durch

$$p_f^{l+1} = (1-\omega)p_f^l + \omega\frac{1}{A_{ff}}\Bigg(\rho^0 - \rho_f^* - \tag{3.26}$$

$$\sum_{f_f} m_{f_f}\Bigg(\sum_{f_f}\mathbf{d}_{ff_ff_f}p_{f_f} - \mathbf{d}_{f_ff_ff_f}p_{f_f} - \sum_{f_{f_f}\neq f}\mathbf{d}_{f_ff_{f_f}f_{f_f}}p_{f_{f_f}}\Bigg)\nabla W_{ff_f}\Bigg) \tag{3.27}$$

4 Implementierung

Dieses Kapitel beschäftigt sich damit, wie die Konzepte aus den vorherigen Kapiteln implementiert wurden.

4.1 Programmierumgebung

Die Software zur Flüssigkeitssimulation wurde überwiegend in der Programmiersprache C++ geschrieben. Daneben sind die Shader für die Visualisierung in OpenGL Shading Language geschrieben. Als Entwicklungsumgebung wurde Microsoft Visual Studio Community 2022 verwendet. Verwendete Bibliotheken sind GLFW und GLAD für die Visualisierung mit OpenGL, GoogleTest für UnitTests und OpenGL Mathematics (GLM) für mathematische Strukturen und Berechnungen.

4.2 Grenzbehandlung

Neben den Flüssigkeitspartikeln gibt es in der Implementierung auch Grenzpartikel. In der Notation werden die Grenzpartiel mit dem Index b und somit auch die Nachbarn eines Flüssigkeitspartikels f, welche zur Grenze gehören, mit dem Index f_b identifiziert. Für die Grenzbehandlung wird der Ansatz aus [KBST20] verwendet, wo es mehrere Schichten von Grenzpartikel mit identischer Größe und Masse gibt.

Die Grenzpartikel sind fest an einem Ort und verhindern, dass die Flüssigkeit auseinanderfließt. Die Grenze besteht aus drei Schichten von Grenzpartikel. Die mehreren Schichten sorgen dafür, dass wenn ein Partikel nahe an der Grenze ist, noch stets genügend Nachbarn hat.

Für die Grenzpartikel werden Größen wie die Dichte ρ_b und der Druck p_b nicht berechnet. Da diese jedoch für SPH-Berechnungen benötigt werden, müssen dennoch für diese Größen Werte gewählt werden. Für die Dichte ρ_b wird die Ruhedichte ρ^0 gewählt. Wird der Druck p_{f_b} in einer SPH-Berechnung benötigt, wird stattdessen der Wert des Flüssigkeitspartikels gespiegelt und damit der Wert p_f verwendet.

4.3 Kernelfunktion, Kernelgradient

In der Implementierung wird der Cubic Spline Kernel verwendet. Da die Simulation zweidimensional ist, hat die Kernelfunktion damit folgende Form:

$$W(\mathbf{x}_f - \mathbf{x}_{f_f}, h) = \frac{5}{14\pi h^2} \begin{cases} (2-q)^3 - 4(1-q)^3, & \text{für } 0 \leq q \leq 1 \\ (2-q)^3, & \text{für } 1 \leq q \leq 2 \\ 0, & \text{sonst} \end{cases} \tag{4.1}$$

mit $q = \frac{1}{h}\|\mathbf{x}_f - \mathbf{x}_{f_f}\|$ und h die Partikelgröße.

Diese Funktion lässt sich recht einfach mit Algorithmus 1 implementieren.

Algorithm 1 Cubic Spline Kernelfunktion

```
1: function W(x_f − x_{f_f})
2:     q ← ‖x_f − x_{f_f}‖ / h
3:     t_1 ← max(1 − q, 0)
4:     t_2 ← max(2 − q, 0)
5:     σ ← 5 / (14πh^2)
6:     return σ · (t_2^3 − 4t_1^3)
7: end function
```

Nun, da die Kernelfunktion in Form des Cubic Spline Kernels implementiert ist, muss noch der dazugehörige Kernelgradient implementiert werden. Der Kernelgradient des Cubic Spline Kernels hat folgende Form:

$$\nabla W(\mathbf{x}_f - \mathbf{x}_{f_f}, h) = \frac{5}{14\pi h^2} \cdot \frac{\mathbf{x}_f - \mathbf{x}_{f_f}}{\|\mathbf{x}_f - \mathbf{x}_{f_f}\| h} \begin{cases} -3(2-q)^2 + 12(1-q)^2, & \text{für } 0 \leq q \leq 1 \\ -3(2-q)^2, & \text{für } 1 \leq q \leq 2 \\ 0, & \text{sonst} \end{cases} \tag{4.2}$$

ebenfalls mit $q = \frac{1}{h}\|\mathbf{x}_f - \mathbf{x}_{f_f}\|$ und h die Partikelgröße.

Für den Kernelgradient wird Algorithmus 2 angewandt.

Algorithm 2 Cubic Spline Kernelgradient

function $\nabla W(\mathbf{x}_f - \mathbf{x}_{f_f})$
  $q \leftarrow \frac{\|\mathbf{x}_f - \mathbf{x}_{f_f}\|}{h}$
  if $q == 0$ **then**
    return $\begin{pmatrix} 0 & 0 \end{pmatrix}^\mathsf{T}$
  end if
  $t_1 \leftarrow \max(1 - q, 0)$
  $t_2 \leftarrow \max(2 - q, 0)$
  $\sigma \leftarrow \frac{5}{14\pi h^2}$
  return $\sigma \frac{\mathbf{x}_f - \mathbf{x}_{f_f}}{\|\mathbf{x}_f - \mathbf{x}_{f_f}\| h} \cdot (-3t_2^2 + 12t_1^2)$
end function

4.4 Nachbarschaftssuche

Da für die SPH-Berechnungen die Nachbarn eines Partikels benötigt werden, muss eine Nachbarschaftssuche durchgeführt werden. Diese muss in jedem Simulationsschritt erfolgen, da sich die Nachbarn eines Partikels in jedem Simulationsschritt ändern können. Daher ist es sehr wichtig, dass die Nachbarschaftssuche effizient abläuft und nicht zu viel Zeit dafür aufgewandt wird.

Der naive Ansatz ist es, wenn für jedes Partikel die Distanz zu den restlichen Partikeln gemessen wird und überprüft wird, ob die Distanz geringer ist als der Kernelsupport. Pro Simulationsschritt gibt es damit einen Zeitaufwand in $O(n^2)$ für n Partikel in der Simulation. Dies ist nicht gerade optimal, denn schon bei etwas größeren Simulationen wird sehr viel Zeit für die Nachbarschaftssuche angewandt.

4.4.1 Uniformes Gitter

Um das Problem zu lösen, dass die Nachbarschaftssuche in quadratischer Zeit nach der Partikelanzahl abläuft, kann stattdessen ein Uniformes Gitter aufgebaut werden, was zur Folge hat, dass der Zeitaufwand für die Nachbarschaftssuche nun in $O(n)$ ist. Das Uniforme Gitter unterteilt den Raum, in dem sich die Partikel bewegen, in mehrere gleich große quadratische Zellen, deren Größe dem Kernelsupport entspricht. Sinn dahinter ist, dass als Nachbarn eines Partikels nur die Partikel in der gleichen Zelle oder einer Nachbarzelle in Frage kommen. Da die Anzahl der Nachbarzellen und die Anzahl der Partikel in einer Zelle konstant ist, kommen für jedes Partikel nur konstant viele Nachbarpartikel in Frage. Daher ist die Nachbarschaftssuche durch das Uniforme Gitter in $O(n)$ für jeden Simulationsschritt.

In der Implementierung wird ein Uniformes Gitter verwendet, das dem kompakten Gitter entspricht, was in der Arbeit von Lagae und Dutré [LD08] beschrieben wird.

4.4.2 Uniformes Gitter Aufbau

Kern der Implementierung des Uniformen Gitters sind zwei Arrays C und L. Der Array L enthält die Partikelindizes, welche sortiert sind nach den Zellen, in denen diese sich befinden. Der Array C wird zum Aufbau von L benötigt und enthält für jeden Zellenindex den Offset, an dem in L ein Partikelindex eingefügt wird. Nach dem Aufbau von L gibt C jeweils anhand eines Zellenindex den Index für das erste Partikel in dieser Zelle in L an. Beispielsweise sind alle Partikel von $L[C[j]]$ bis $L[C[j+1]-1]$ in der Zelle mit Zellenindex j. Der Zellenindex eines Partikels wird anhand dessen Position bestimmt.

Algorithm 3 Aufbau des Uniformen Gitters

```
1: for j ← 0, cells do
2:     C[j] ← 0
3: end for
4: for all particle i do
5:     j ← getCellIndex(x_i)
6:     C[j] ← C[j] + 1
7: end for
8: for i ← 1, cells do
9:     C[j] ← C[j] + C[j − 1]
10: end for
11: for all particle i do
12:     j ← getCellIndex(x_i)
13:     C[j] ← C[j] − 1
14:     L[C[j]] ← i
15: end for
```

4.4.3 Bestimmung der Nachbarn mithilfe des Uniformen Gitters

Ist das Uniforme Gitter aufgebaut, indem die Arrays L und C initialisiert wurden, so kann dieses für die Nachbarschaftssuche genutzt werden. Um die Nachbarn eines Partikels zu ermitteln, werden zuerst die Zelle des Partikels und dessen Nachbarzellen (bis zu 8 Nachbarzellen, je nachdem ob die Zelle am Rand ist oder nicht) überprüft. Alle Partikel in diesen Zellen kommen als Nachbar in Frage.

In einem zweiten Schritt wird dann überprüft, ob der Abstand zwischen diesen Partikeln und des Urprungspartikels geringer als der Kernelsupport $\hbar$ ist. Die Partikel, deren Abstand zum Ursprungspartikel geringer als der Kernelsupport ist, sind die Nachbarn des Ursprungspartikels.

Algorithmus 4 zeigt, wie die Ermittlung der Nachbarn eines Partikels f implementiert werden kann.

Algorithm 4 Nachbarschaftssuche mithilfe des Uniformen Gitters

```
possibleNeighbors ← empty list
neighbors ← empty list
for all neighbor cell index j do
    for i ← C[j], C[j + 1] − 1 do
        possibleNeighbors.ADD(L[i])
    end for
end for
for all i in possibleNeighbors do
    if ‖x_i − x_f‖ < ħ then
        neighbors.ADD(i)
    end if
end for
```

4.5 Berechnung der Dichte

Zur Implementierung der Dichteberechnung wird die normale SPH Approximation der Dichte benutzt:

$$\rho_f = \sum_{f_i} m_{fi} W_{ff_i} \tag{4.3}$$

In der Implementierung ist die Masse aller Partikel identisch, daher kann die Masse der Nachbarpartikel m_{fi} durch m_f ersetzt werden und stattdessen die Dichte wie folgt berechnet werden:

$$\rho_f = m_f \sum_{f_i} W_{ff_i} \tag{4.4}$$

Algorithm 5 Dichteberechnung

```
for all fluid particle f do
    ρ_f ← m_f Σ_{f_i} W_{ff_i}
end for
```

4.6 Berechnung des Drucks

Beim Start der Simulation kann ausgewählt werden, ob der Druck durch eine Zustandsgleichung oder durch IISPH berechnet wird. Daher gibt es für die Druckberechnung zwei verschiedene Implementierungen.

Für die Druckberechnung durh eine Zustandsgleichung wird folgende Zustandsgleichung verwendet:

$$p_f = \max\left(k\left(\frac{\rho_f}{\rho^0} - 1\right), 0\right) \tag{4.5}$$

Die Implementierung durch einen Algorithmus ist trivial.

Alternativ kann der Druck durch IISPH [ICS+14] berechnet werden. Dazu muss ein Gleichungssystem der Form $\mathbf{Ap} = \mathbf{s}$ gelöst werden. Bevor das Gleichungssystem gelöst wird, werden die Diagonalwerte A_{ff} der Matrix A und die Komponenten s_f des Lösungsvektors $\mathbf{s}$ berechnet. Im Kapitel „Druckberechnung“ wurde zur Bestimmung der Diagonalwerte folgende Gleichung aus [ICS+14] verwendet, die aus Notationsgründen modifiziert ist:

$$A_{ff} = \sum_{f_f} m_{f_f} \left(\mathbf{d}_{ff} - \mathbf{d}_{f_f f} \right) \nabla W_{ff_f} \tag{4.6}$$

$\mathbf{d}_{ff}$ ist definiert als:

$$\mathbf{d}_{ff} = -\Delta t^2 \sum_{f_f} \frac{m_f}{\rho_f^2} \nabla W_{ff_f} \tag{4.7}$$

$\mathbf{d}_{f_f f}$ ist definiert als:

$$\mathbf{d}_{f_f f} = -\Delta t^2 \frac{m_f}{\rho_f^2} \nabla W_{f_f f} \tag{4.8}$$

Da in der Implementierung die Massen der Partikel identisch sind, vereinfacht sich A_{ff} daher zu:

$$A_{ff} = \sum_{f_f} m_{f_f} \left(\mathbf{d}_{ff} - \mathbf{d}_{f_f f} \right) \nabla W_{ff_f} \tag{4.9}$$

$$= \sum_{f_f} m_f \left(\mathbf{d}_{ff} - \mathbf{d}_{f_f f} \right) \nabla W_{ff_f} \tag{4.10}$$

$$= \sum_{f_f} m_f \left(\left(-\Delta t^2 \sum_{f_f} \frac{m_f}{\rho_f^2} \nabla W_{ff_f} \right) - \Delta t^2 \frac{m_f}{\rho_f^2} \nabla W_{ff_f} \right) \nabla W_{ff_f} \tag{4.11}$$

$$= -\Delta t^2 \frac{m_f^2}{\rho_f^2} \sum_{f_f} \left(\left(\sum_{f_f} \nabla W_{ff_f} \right) + \nabla W_{ff_f} \right) \nabla W_{ff_f} \tag{4.12}$$

Bezieht man die Partikel der Grenze mit ein, erhält man:

$$A_{ff} = -\Delta t^2 \frac{m_f^2}{\rho_f^2} \Bigg(\sum_{f_f} \Big(\Big(\sum_{f_f} \nabla W_{ff_f} \Big) + \nabla W_{ff_f} \Big) \nabla W_{ff_f} + \tag{4.13}$$

$$\sum_{f_b} \Big(\Big(\sum_{f_b} \nabla W_{ff_b} \Big) + \nabla W_{ff_b} \Big) \nabla W_{ff_b} \Bigg) \tag{4.14}$$

s_f hat folgende Form:

$$s_f = \rho_f^0 - \rho_f^* \tag{4.15}$$

Die Zwischen-Dichte ρ_f^* wird aus den Zwischen-Geschwindigkeiten wie folgt berechnet unter Berücksichtigung der Grenzpartikel und den identischen Massen:

$$\rho_f^* = \rho_f + m_f \Delta t \left(\sum_{f_f} (\mathbf{v}_f^* - \mathbf{v}_{f_f}^*) \nabla W_{ff_f} + \sum_{f_b} \mathbf{v}_f^* \nabla W_{ff_b} \right) \tag{4.16}$$

Die Komponenten des Lösungsvektors werden also durch folgende Gleichung implementiert:

$$s_f = \rho_f^0 - \rho_f - m_f \Delta t \left(\sum_{f_f} (\mathbf{v}_f^* - \mathbf{v}_{f_f}^*) \nabla W_{ff_f} + \sum_{f_b} \mathbf{v}_f^* \nabla W_{ff_b} \right) \tag{4.17}$$

Ebenfalls wird der Druck $\mathbf{p}^0$ auf $0.5\mathbf{p}_f(t - \Delta t)$ initialisiert. Dieser Ausgangswert für den Druck soll nach Ihmsen et al. [ICS+14] optimal sein.

Nun kann iterativ der Druck berechnet werden. Der Index l bezeichnet die Anzahl der Iterationen. Dazu wird zuerst jeweils die Druckbeschleunigung $\mathbf{a}_f^p$ durch Algorithmus 7 berechnet. Danach wird $(\mathbf{Ap})_f^l$ berechnet, was definiert ist durch:

$$(\mathbf{Ap})_f^l = \Delta t^2 \sum_{f_f} m_{f_f} (\mathbf{a}_f^p - \mathbf{a}_{f_f}^p) \nabla W_{ff_f} \tag{4.18}$$

Unter Einbeziehung der Grenzpartikel und den identischen Massen, wird $(\mathbf{Ap})_f^l$ wie folgt berechnet:

$$(\mathbf{Ap})_f^l = m_f \Delta t^2 \left(\sum_{f_f} (\mathbf{a}_f^p - \mathbf{a}_{f_f}^p) \nabla W_{ff_f} + \sum_{f_b} \mathbf{a}_f^p \nabla W_{ff_b} \right) \tag{4.19}$$

Jetzt kann schließlich der Druck aktualisiert werden durch:

$$p_f^{l+1} = \max(p_f^l + \omega \frac{s_f - (\mathbf{Ap})_f}{\mathbf{A}_{ff}}, 0) \tag{4.20}$$

Der Relaxierungsfaktor ω ist in der Implementierung auf den Wert 0.5 festgelegt, da dieser Wert nach Ihmsen et al. [ICS+14] nahezu optimale Ergebnisse liefern soll. Der Druck wird auf einen positiven Wert gehalten, da die Partikel sich sonst gegenseitig anziehen würden, was zu unerwünschten Effekten führt, wie in [ICS+14] beschrieben ist.

Es wird ebenfalls der durchschnittliche Dichtefehler $\frac{1}{n} \sum_f \frac{(\mathbf{Ap})_f - s_f}{\rho_f^0}$ berechnet. In diese Summe werden nur die Partikel einbezogen, die aus der vorigen Berechnung einen positiven Druck besitzen, und nicht den Wert 0 zugewiesen bekommen haben. Ist dieser Fehler unter dem Wert von 0.001, so wird der letzte berechnete Wert des Druckes genommen, andernfalls wird weiter iteriert und ein neuer Druck berechnet.

Ihmsen et al. [ICS+14] empfiehlt, als Konvergenzkriterium für den durchschnittlichen Dichtefehler Werte von 0.1 % oder geringer zu verwenden. Zudem sollten mindestens zwei Iterationen durchlaufen werden.

Algorithmus 6 zeigt die Implementierung der Druckberechnung mit IISPH.

Algorithm 6 Berechnung des Drucks der Partikel mit IISPH

for all fluid particle f **do**
  $A_{ff} \leftarrow -\Delta t^2 \frac{m_f^2}{\rho_f^2} \left(\sum_{f_j} \left(\left(\sum_{f_j} \nabla W_{ff_j} \right) + \nabla W_{ff_j} \right) \nabla W_{ff_j} \right)$
  $s_f \leftarrow \rho_f^0 - \rho_f - m_f \Delta t \left(\sum_{f_f} (\mathbf{v}_f^* - \mathbf{v}_{f_f}^*) \nabla W_{ff_f} + \sum_{f_b} \mathbf{v}_f^* \nabla W_{ff_b} \right)$
  $p_f \leftarrow \frac{p_f}{2}$
end for
$e \leftarrow \infty$
$iterations \leftarrow 0$
while $e \geq 0.001$ **or** $iterations < 2$ **do**
  $e \leftarrow 0$
  $n \leftarrow 0$
  Compute pressure accelerations $\mathbf{a}_f^p$ using algorithm 7
  for all fluid particle f **do**
    $(\mathbf{Ap})_f \leftarrow m_f \Delta t^2 \left(\sum_{f_f} (\mathbf{a}_f^p - \mathbf{a}_{f_f}^p) \nabla W_{ff_f} + \sum_{f_b} \mathbf{a}_f^p \nabla W_{ff_b} \right)$
    if $(\mathbf{Ap})_f \neq 0$ **then**
      $p_f \leftarrow p_f + \omega \frac{s_f - (\mathbf{Ap})_f}{\mathbf{A}_{ff}}$
      if $p_f < 0$ **then**
        $p_f \leftarrow 0$
      else
        $e \leftarrow e + \left| \frac{(\mathbf{Ap})_f - s_f}{\rho_f^0} \right|$
        $n \leftarrow n + 1$
      end if
    end if
  end for
  $e \leftarrow \frac{e}{n}$
  $iterations \leftarrow iterations + 1$
end while

4.7 Berechnung der Druckbeschleunigung

Zur Berechnung der Druckbeschleunigung wird die alternative Form der SPH-Approximation verwendet, die den Impuls und Drehimpuls der Flüssigkeit erhält:

$$\mathbf{a}_f^p = -\frac{1}{\rho_f}\nabla p_f = -\sum_{f_f} m_{f_f}\left(\frac{p_f}{\rho_f^2} + \frac{p_{f_f}}{\rho_{f_f}^2}\right)\nabla W_{ff_f} \tag{4.21}$$

Auch hier ändert sich die Form durch die identische Masse der Partikel und unter Einbeziehung der Grenzpartikel zu:

$$\mathbf{a}_f^p = -\frac{1}{\rho_f}\nabla p_f = -m_f\left(\sum_{f_f}\left(\frac{p_f}{\rho_f^2} + \frac{p_{f_f}}{\rho_{f_f}^2}\right)\nabla W_{ff_f} + \right. \tag{4.22}$$

$$\left.\sum_{f_b}\left(\frac{p_f}{\rho_f^2} + \frac{p_f}{(\rho^0)^2}\right)\right) \tag{4.23}$$

Die Implementierung dazu findet sich in Algorithmus 7.

Algorithm 7 Berechnung der Druckbeschleunigungen

```
for all particle i do
    if particle i belongs to the boundary then
        a_i^p ← (0  0)^T
        continue
    end if
    acc_p ← (0  0)^T

    // Druckbeschleunigung an Partikel i
    for all neighbor j of particle i do
        if particle j belongs to the boundary then
            acc_p ← acc_p − ( p_i/ρ_i^2 + p_i/(ρ_f^0)^2 ) · ∇W_ij
        else
            acc_p ← acc_p − ( p_i/ρ_i^2 + p_j/ρ_j^2 ) · ∇W_ij
        end if
    end for
    acc_p ← acc_p · m_f

    a_i^p ← acc_p
end for
```

4.8 Berechnung der restlichen Beschleunigungen

Neben den Druckbeschleunigungen müssen auch andere Beschleunigungen berechnet werden. In der Implementierung wird die Viskositätsbeschleunigung berechnet und die Gravitationsbeschleunigung, welche konstant ist. Die Viskositätsbeschleunigung $\nu\nabla^2\mathbf{v}_f$ wird ebenfalls nicht durch die normale SPH-Approximation berechnet, sondern durch folgende alternative SPH-Approximation:

$$\nu\nabla^2\mathbf{v}_f = 2(d+2)\nu\sum_{f_f}\frac{m_{f_f}}{\rho_{f_f}}\frac{\mathbf{v}_{ff_f}\cdot\mathbf{x}_{ff_f}}{\|\mathbf{x}_{ff_f}\|+0.01h^2}\nabla W_{ff_f} \tag{4.24}$$

Da der Parameter ν frei gewählt ist, kann der Vorfaktor $2(d+2)$ weggelassen werden. Unter Berücksichtigung der identischen Masse und Einbeziehung der Grenzpartikel wird in Algorithmus 8 die Viskositätsbeschleunigung dann durch folgende Gleichung berechnet:

$$\nu\nabla^2\mathbf{v}_f = \nu m_f\Bigg(\sum_{f_f}\frac{1}{\rho_{f_f}}\frac{\mathbf{v}_{ff_f}\cdot\mathbf{x}_{ff_f}}{\|\mathbf{x}_{ff_f}\|+0.01h^2}\nabla W_{ff_f} \tag{4.25}$$

$$+\sum_{f_b}\frac{1}{\rho_f}\frac{\mathbf{v}_f\cdot\mathbf{x}_{ff_b}}{\|\mathbf{x}_{ff_b}\|+0.01h^2}\nabla W_{ff_b}\Bigg) \tag{4.26}$$

Algorithm 8 Berechnung der restlichen Beschleunigungen

for all particle i **do**
  if particle i belongs to the boundary **then**
    $\mathbf{a}_i^n \leftarrow \begin{pmatrix} 0 & 0 \end{pmatrix}^\intercal$
    continue
  end if
  $\mathbf{acc}_g \leftarrow \begin{pmatrix} 0 & -9.81 \end{pmatrix}^\intercal$
  $\mathbf{acc}_v \leftarrow \begin{pmatrix} 0 & 0 \end{pmatrix}^\intercal$

  // Viskositätsbeschleunigung an Partikel i
  for all neighbor j of particle i **do**
    if particle j belongs to the boundary **then**
      $\mathbf{acc}_v \leftarrow \mathbf{acc}_v + \frac{1}{\rho_i} \frac{(\mathbf{v}_i - \mathbf{v}_j)(\mathbf{x}_i - \mathbf{x}_j)}{(\mathbf{x}_i - \mathbf{x}_j)(\mathbf{x}_i - \mathbf{x}_j) + 0.01h^2} \cdot \nabla W_{ij}$
    else
      $\mathbf{acc}_v \leftarrow \mathbf{acc}_v + \frac{1}{\rho_j} \frac{(\mathbf{v}_i - \mathbf{v}_j)(\mathbf{x}_i - \mathbf{x}_j)}{(\mathbf{x}_i - \mathbf{x}_j)(\mathbf{x}_i - \mathbf{x}_j) + 0.01h^2} \cdot \nabla W_{ij}$
    end if
  end for
  $\mathbf{acc}_v \leftarrow 2\nu m_f \cdot \mathbf{acc}_v$

  $\mathbf{a}_i^n \leftarrow \mathbf{acc}_g + \mathbf{acc}_v$
end for

4.9 Simulationsschritt

Der Simulationsschritt ist der Kern der Simulation. In jedem Simulationsschritt werden die Beschleunigungen berechnet, die auf die Partikel wirken, und anschließend anhand diesen Beschleunigungen die Geschwindigkeiten und Positionen der Partikel aktualisiert.

Zu Beginn des Simulationsschrittes findet die Nachbarschaftssuche statt, damit die SPH-Approximationen berechnet werden können. Die Nachbarschaftssuche geschieht mithilfe des Uniformen Gitters, welches zuvor beschrieben wurde.

Nachdem die Nachbarn bestimmt sind, wird mithilfe von Algorithmus 5 die Dichte an jedem Flüssigkeitspartikel berechnet.

Dann werden die restlichen Beschleunigungen mit Algorithmus 8 berechnet, die nicht durch den Druck ausgelöst sind.

Anhand diesen Beschleunigungen wird eine Zwischen-Geschwindigkeit $\mathbf{v}_f^*$ an jedem Flüssigkeitspartikel f berechnet.

Nun wird der Druck berechnet. Je nachdem, welche Druckberechnung gewünscht ist, wird der Druck durch eine Zustandsgleichung oder durch IISPH mit Algorithmus 6 berechnet.

Anschließend werden die Druckbeschleunigungen berechnet mit Algorithmus 7.

Zum Schluss werden anhand des Euler-Cromer Verfahrens die Geschwindigkeiten $\mathbf{v}_f$ und Positionen $\mathbf{x}_f$ der Flüssigkeitspartikel aktualisiert.

Der Simulationsschritt ist in Algorithmus 9 dargestellt.

Algorithm 9 Simulationsschritt

```
Determine neighbors of each particle
Compute density ρ_f of each fluid particle using algorithm 5
Compute non-pressure accelerations a_f^n using algorithm 8
for all fluid particle f do
    v_f^* ← v_f + Δt a_f^n
end for
Compute pressure p_f of each fluid particle using algorithm 6 or state equation
Compute pressure accelerations a_f^p using algorithm 7
for all fluid particle f do
    v_f ← v_f^* + Δt a_f^p
end for
for all fluid particle f do
    x_f ← x_f + Δt v_f
end for
```

4.10 Visualisierung

Visualisiert wird die Simulation durch die Programmierschnittstelle *OpenGL*.

Jedes Partikel wird durch einen Kreis dargestellt. Beim Starten der Simulation kann die Partikelgröße h gewählt werden. Die Partikelgröße entspricht gleichzeitig dem Durchmesser des zugehörigen Kreises in Pixel. Wird also beispielsweise eine Partikelgröße $h = 10$ gewählt, so ist in der Visualisierung jeder Kreis auch vom Durchmesser 10 Pixel.

Die globalen Koordinaten $\mathbf{x}_f$ eines Partikels f, die in der Simulation verwendet werden, sind zu dessen Fensterkoordinaten $\mathbf{x}'_f$ gespiegelt an einer Parallelen der x-Achse, die sich in der Mitte des Fensters mit y-Koordinaten $y = \text{height}/2$ befindet.

Seien $\mathbf{x}'_f$ also die Fensterkoordinaten des Partikels f mit globalen Koordinaten $\mathbf{x}_f = \begin{pmatrix} x & y \end{pmatrix}^\mathsf{T}$. Dann gilt

$$\mathbf{x}'_f = \begin{pmatrix} x & \text{height} - y \end{pmatrix}^\mathsf{T} \tag{4.27}$$

Die Grenzpartikel werden alle in grauer Farbe dargestellt. Die Farbe eines Flüssigkeitspartikels f hängt von dem Betrag dessen Geschwindigkeit $\|\mathbf{v}_f\|$ ab. Dabei erscheinen ruhende Partikel komplett in blau, schnellere Partikel in grün und schließlich ab $\|\mathbf{v}_f\| > \frac{h}{\Delta t}$ komplett in rot. Das heißt, aus der Visualisierung lässt sich erkennen, ob die CFL-Bedingung

$$\Delta t \leq \lambda \frac{\hbar}{\|\mathbf{v}^{max}\|} \tag{4.28}$$

für $\lambda = 1$ erfüllt wird. Diese Bedingung sollte auf jeden Fall erfüllt werden, daher sollten im Laufe der Simulation die Partikel auf keinen Fall komplett in roter Farbe erscheinen. Die Übergänge zwischen den Farben sind fließend.

Sei $\mathbf{c}_f$ der Wert der Farbe des Partikels f im RGB-Farbraum. Dann wird $\mathbf{c}_f$ wie folgt berechnet:

$$\mathbf{c}_f = \begin{cases} \begin{pmatrix} 0 \\ \sin^2(\frac{\pi\|\mathbf{v}_f\|}{\frac{h}{\Delta t}}) \\ \cos^2(\frac{\pi\|\mathbf{v}_f\|}{\frac{h}{\Delta t}}) \end{pmatrix} & \text{falls } \|\mathbf{v}_f\| < \frac{h}{2\Delta t} \\ \begin{pmatrix} \cos^2(\frac{\pi\|\mathbf{v}_f\|}{\frac{h}{\Delta t}}) \\ \sin^2(\frac{\pi\|\mathbf{v}_f\|}{\frac{h}{\Delta t}}) \\ 0 \end{pmatrix} & \text{falls } \frac{h}{2\Delta t} \leq \|\mathbf{v}_f\| < \frac{h}{\Delta t} \\ \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} & \text{sonst} \end{cases} \tag{4.29}$$

5 Analyse

5.1 Szenarien

In der Simulationssoftware können fünf verschiedene Szenarien simuliert werden. Die verschiedenen Szenarien können beim Start der Software gewählt werden. Die Szenarien unterscheiden sich im Wesentlichen in den Anfangspositionen der Partikel. In diesem Abschnitt werden diese Szenarien vorgestellt.

Das erste Szenario ist eine klassische Dammbruchsimulation, die in vierlei Anwendungen von Flüssigkeitssimulationen vorkommt. Der Aufbau ist einfach und die Eigenschaften der Flüssigkeit kann gut beobachtet werden, so dass sich dieses Szenario gut zum Testen einer Simulation eignet. Sie stellt eine Situation dar, in der ein Damm, der die Flüssigkeit aufhält, plötzlich verschwunden ist. Der Aufbau des Dammbruchszenarios ist in Abbildung 5.1 dargestellt.

Das zweite Szenario ist sehr ähnlich zum Dammbruchszenario. Hier ist der Damm noch vorhanden. Dieser besitzt jedoch ein Loch, wodurch die Flüssigkeit durchfließt. In diesem Szenario kann beobachtet werden, wie die Flüssigkeit durch das Loch durchströmt und sich die Pegelstände vor und hinter dem Damm angleichen. Der Aufbau dieses Szenarios ist in Abbildung 5.2 dargestellt.

Das dritte Szenario zeigt zu Beginn die Flüssigkeit auf einer Erhöhung. Hier kann beobachtet werden, wie die Flüssigkeit auf beiden Seiten herunterfällt und sich zwei voneinander getrennte Flüssigkeiten bilden. Der Aufbau dieses Szenarios ist in Abbildung 5.3 dargestellt.

Im vierten Szenario kann beobachtet werden, wie die Flüssigkeit von einer schiefen Ebene in Richtung einer horizontale Ebene herunterfließt. Dargestellt ist der Aufbau dieses Szenarios in Abbildung 5.4.

Das fünfte Szenario zeigt eine ruhende Flüssigkeit. Beobachtet werden kann hier wie sich die Partikel anordnen. Dieses Szenario eignet sich auch gut, um den Verlauf der Durchschnittsdichte der Flüssigkeitspartikel zu beobachten. Dargestellt ist der Aufbau dieses Szenarios in Abbildung 5.5.

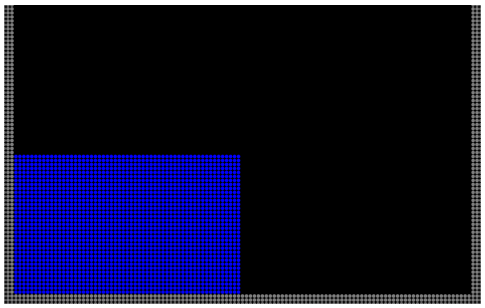

Abbildung 5.1: Brechender Damm

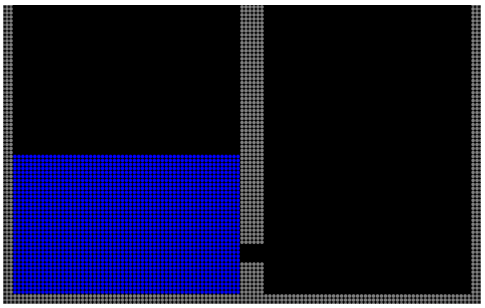

Abbildung 5.2: Damm mit Leck

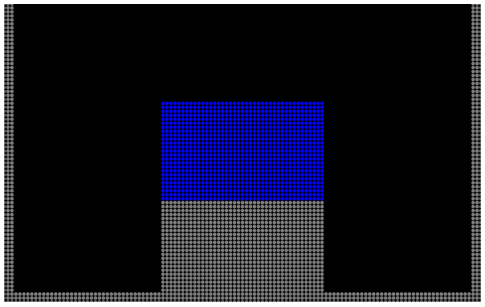

Abbildung 5.3: Von einer Plattform herunterfallende Flüssigkeit

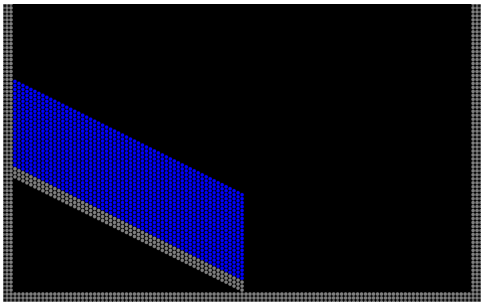

Abbildung 5.4: Von einer schiefen Ebene herunterfließende Flüssigkeit

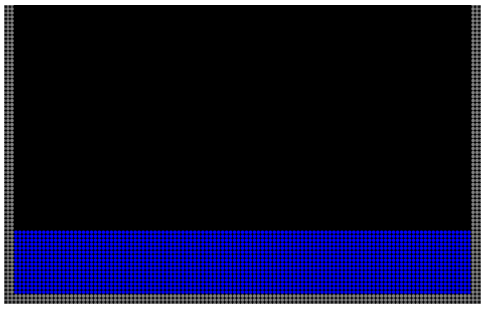

Abbildung 5.5: Ruhende Flüssigkeit

5.2 Ergebnisse

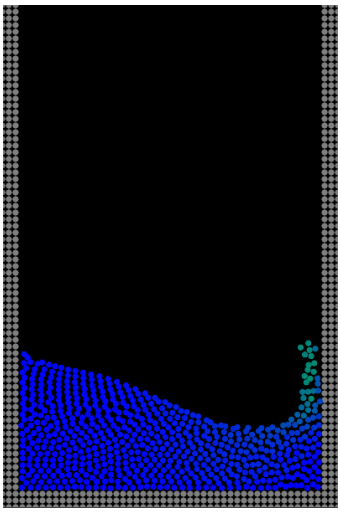

Abbildung 5.6: Dammbruch mit 1251 Partikel

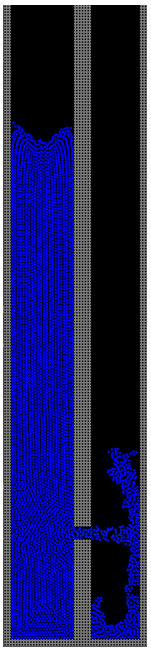

Abbildung 5.7: Dammleck mit 7199 Partikel

Um die Performanz zwischen SSPH und IISPH zu vergleichen, wurden Daten aus verschiedenen Dammbruchsimulationen gesammelt. Eine Szene davon ist in Abbildung 5.6 dargestellt. Die Flüssigkeitstiefe zu Beginn beträgt 30 Partikel, die Partikelgröße ist auf 8 festgelegt, insgesamt gibt es 1251 Partikel. Die Viskosität beträgt 200. Es wurde jeweils eine Zeitspanne von 50 Sekunden simuliert. Abhängig des Zeitschritts Δt ergab sich damit eine unterschiedliche Anzahl an berechneten Simulationsschritten.

Für die IISPH-Simulationen wurde ebenfalls die durchschnittliche Anzahl an Iterationen erfasst, die während der Druckberechnung benötigt werden, um einen Dichtefehler unter 0.01% zu ereichen. Zudem wurde die Zeit gemessen, die für die Druckberechnung und für den Simulationsschritt jeweils benötigt wurde. Die Daten dazu sind in Tabelle 5.1 dargestellt.

Aus diesen Daten ist zu erkennen, dass mit größeren Zeitschritten auch die Anzahl an Iterationen für die Druckberechnung zunimmt. Der Anteil der Zeit, der für die Druckberechnung aufgewandt wird, an der Zeit für den Simulationsschritt, nimmt damit ebenfalls zu. Insgesamt erreicht die IISPH-Simulation mit Zeitschritt $\Delta t = 0.04$ die beste Performanz, da die absolute Berechnungszeit für den Simulationsschritt hier mit 62.38 Sekunden am geringsten ist.

Δt	durchschn. Iter.	Berechnungszeit (s)		
		Druck	Simulationsschr.	Anteil Druckber. (%)
0.01	2.00	48.24	143.49	33.6
0.02	2.38	27.37	73.91	37
0.03	4.87	32.28	65.07	49.6
0.04	8.45	38.55	62.38	61.8
0.05	12.87	44.48	63.41	70.1
0.06	18.04	50.92	66.5	76.6
0.08	30.01	62.52	74.21	84.2

Tabelle 5.1: Performanz von IISPH im Dammbruchszenario mit 1251 Partikel und Dichtefehler 0.1 %

Um die Performanz von IISPH mit SSPH vergleichen zu können, wurde auch eine vergleichbare SSPH-Simulation durchgeführt. Es wurde eine Steifigkeitskonstante von $k = 7 \cdot 10^5$ gewählt, um den durchschnittlichen Dichtefehler zuverlässig im Bereich unter 0.01% zu halten. Wie im Abschnitt *Druckberechnung mit einer Zustandsgleichung* beschrieben, besitzt die Steifigkeitskonstante k einen Einfluss auf die Dichteabweichung. Größere Werte für k ergeben geringere Dichtefehler und vice versa.

Die Wahl des Zeitschritts gestaltete sich schwierig, da es keine eindeutige Grenze zwischen stabilen und instabilen Simulationen gibt. Für Zeitschritte $\Delta t \geq 0.019$ ist deutlich zu sehen, dass die Simulation instabil ist. In diesem Fall bewegen sich die Partikel mit sehr hoher Geschwindigkeit in alle Richtungen auseinander. Ebenfalls kann auch noch für Zeitschritte von etwa $0.011 \leq t < 0.019$ beobachtet werden, dass einzelne Partikel sich nicht sinnvoll verhalten, auch wenn die Durchschnittsdichte unter 0.01% liegt. Daher wurde der Zeitschritt $\Delta t = 0.01$ als obere Grenze festgelegt, in der die Simulation noch stabil ist. Es ergab sich daraus insgesamt eine Berechnungszeit von 98.09 Sekunden für den Simulationsschritt. Die Druckberechnungszeit ist hier mit 0.241 Sekunden vernachlässigbar.

Vergleicht man die Zeit von 98.09 Sekunden für die SSPH-Simulation mit der Berechnungszeit der performantesten IISPH-Simulation mit 62.38 Sekunden, so ist zu erkennnen, dass sich die Simulation mit IISPH insgesamt etwas schneller berechnen lässt. Ein weiterer Vorteil von IISPH gegenüber SSPH ist, dass für einen vorgegebenen maximalen Dichtefehler von 0.01% lediglich der Zeitschritt Δt gewählt werden muss. Für $0.03 \leq \Delta t \leq 0.06$ ist die Performanz recht ähnlich. Bei SSPH muss zunächst anhand von vorherigen Simulationen der durchschnittliche Dichtefehler ermittelt werden, und abhängig davon wird die Steifigkeitskonstante k gewählt. Anschließend muss der Zeitschritt Δt gewählt werden, der wie oben beschrieben, auch nicht so einfach zu bestimmen ist, wie im Falle von IISPH. Daher erzielt IISPH in diesem Szenario eindeutig bessere Ergebnisse.

Um zu beobachten, wie sich die Druckberechnungsmethoden mit einem geringeren erlaubten Dichtefehler verhalten, wurde das exakt gleiche Simulationsszenario durchgeführt, mit dem Unterschied, dass der erlaubte Dichtefehler nun 0.01% statt 0.1% beträgt. Die zu IISPH gesammelten Daten sind in Tabelle 5.2 dargestellt.

Δt	durchschn. Iter.	Berechnungszeit (s)		
		Druck	Simulationsschr.	Anteil Druckber. (%)
0.005	2.14	101.6	291.29	34.9
0.0075	3.43	94.89	220.65	43
0.01	5.59	107.65	205.07	52.5
0.015	11.87	139.21	202.44	68.8
0.02	20.18	171.46	218.79	78.4
0.025	29.91	202	240.58	84
0.03	40.32	224.01	255.39	87.7

Tabelle 5.2: Performanz von IISPH im Dammbruchszenario mit 1251 Partikel und Dichtefehler 0.01 %

Hier ist zu sehen, dass im Vergleich zum höheren erlaubten Dichtefehler für gleiche Zeitschritte Δt etwa 8 Mal mehr Iterationen benötigt werden. Daher müssen auch niedrigere Zeitschritte gewählt werden. Der Zeitschritt $\Delta t = 0.015$ mit einer Berechnungszeit für den Simulationsschritt von 202.44 Sekunden ergibt die beste Performanz. Aber auch Zeitschritte $0.0075 \leq \Delta t \leq 0.02$ liefern weniger als 10% langsamere Ergebnisse. Im Vergleich zu den Simulationen mit höheren erlaubten Dichtefehler sind die optimalen Zeitschritte etwa um den Faktor 3 kleiner und dementsprechend sind die Berechnungszeiten für den Simulationsschritt ebenfalls um den Faktor 3 herum länger.

Es wurde ebenfalls eine möglichst vergleichbare SSPH-Simulation durchgeführt. Hier wurde eine Steifigkeitskonstante von $k = 7 \cdot 10^6$ gewählt, damit die durchschnittliche Dichte unter 0.01% bleibt. Im Vergleich zur Simulation mit erlaubten Dichtefehler von 0.1% ist k hier um den Faktor 10 größer. Die Steifigkeitskonstante und der Dichtefehler scheinen damit sich antiproportional zueinander zu verhalten. Aus dieser Steifigkeitskonstante ergab sich damit ein maximaler Zeitschritt von $\Delta t = 0.004$. Auch hier war die Wahl dieses maximalen Zeitschritts nicht so einfach. Die Berechnungszeit für den Simulationsschritt betrug damit 244.73 Sekunden und 0.699 Sekunden für die Druckberechnung.

Im Vergleich zur besten IISPH-Simulation ist die SSPH-Simulation somit etwas langsamer, wobei der Unterschied hier weniger stark ausfällt. Der Vergleich der Berechnungsgeschwindigkeit ist jedoch etwas unpräzise, weil die Wahl des Zeitschritts bei SSPH schwierig ist und gleichzeitig einen großen Einfluss auf die Berechnungszeit hat. Auch hier ist bei IISPH die Wahl des Zeitschritts einfacher, weil der Einfluss des Zeitschritts auf die Geschwindigkeit der Berechnung weniger stark ausfällt.

Um die beiden Druckberechnungsmethoden in einem komplexeren Szenario zu vergleichen, wurden ebenfalls Simulationen im zweiten Szenario *Damm mit Leck* durchgeführt. Eine Szene davon ist in Abbildung 5.7 dargestellt. Dieses Szenario wurde gewählt, weil im Gegensatz zur normalen Dammbruchsimulation länger beobachtet werden kann, wie sich die Simulation unter der ursrünglichen Flüssigkeitstiefe verhält. Um die Simulation komplexer zu gestalten, wurde die Flüssigkeitstiefe auf 200 Partikel erhöht. Die Partikelgröße beträgt nun $h = 4$. Um die Berechnungszeit zu begrenzen, wurde die Fensterbreite von 400 Pixel auf 200 Pixel verringert. Es ergibt sich eine Partikelanzahl von 7199 Partikeln. In den verschiedenen Simulationen wurde hier eine Zeit von 20 Sekunden simuliert. Der erlaubte Dichtefehler beträgt 0.1%. Die Daten dazu sind in Tabelle 5.3 dargestellt.

Δt	durchschn. Iter.	Berechnungszeit (s)		
		Druck	Simulationsschr.	Anteil Druckber. (%)
0.004	2	326.94	988.52	33.1
0.005	2.76	324.73	843.64	38.5
0.006	3.97	352.95	777.8	45.4
0.008	6.87	423.05	749.99	56.4
0.01	10.24	484.2	738.65	65.6
0.012	14.61	560.69	777.3	72.13
0.015	22.53	685.44	857.31	80
0.02	38.82	858.17	987.67	86.9

Tabelle 5.3: Performanz von IISPH im Dammleckszenario mit 7199 Partikel

Aus den Daten für die IISPH-Simulationen zeigt sich, dass der optimale Zeitschritt für IISPH um den Wert $\Delta t = 0.1$ herum liegt mit einer Berechnungszeit der Simulationsschritte von 738.65 Sekunden. Aber auch Zeitschritte von $0.006 \leq \Delta t \leq 0.012$ liefern ähnlich gute Ergebnisse.

Für die SSPH-Simulation wurde eine Steifigkeitskonstante $k = 3 \cdot 10^6$ festgelegt, um den durchschnittlichen Dichtefehler unter 0.1% zu halten. Daraus ergab sich ein maximaler Zeitschritt $\Delta t = 0.004$, in der die Simulation noch stabil erscheint. Es wurden insgesamt vernachlässigbare 1.49 Sekunden für die Druckberechnung und 649.78 Sekunden CPU-Zeit für den Simulationsschritt benötigt.

Im Vergleich zur SSPH-Simulation mit Steifigkeitskonstante $k = 3 \cdot 10^6$ und Zeitschritt $\Delta t = 0.004$ ist die Berechnungszeit bei IISPH etwa um den Faktor 1.14 länger. Daher erreicht die SSPH-Simulation hier ein besseres Ergebnis. Jedoch muss auch in diesem Fall für die SSPH-Simulation zuerst die gewünschte Steifigkeitskonstante ermittelt werden, die den gewünschten Dichtefehler unterschreitet und zusätzlich ist die Wahl des richtigen Zeitschritts schwieriger, weil dieser möglichst groß sein muss für eine schnelle Berechnungszeit und gleichzeitig klein genug sein muss, damit die Simulation stabil bleibt.

Die schnellere Berechnung für die SSPH-Simulation erscheint überraschend, weil dieses Szenario etwas komplexer gestaltet ist als die vorherigen. Nach Koshier et al. [KBST20] erzielen globale Druckberechnungsmethoden in komplexeren Szenarien, die sich vor allem durch eine höhere Säule an Partikel unter dem Einfluss von Gravitation auszeichnen, bessere Ergebnisse als Druckberechnungsmethoden durch Zustandsgleichungen. Dieses Verhalten konnte in den beschriebenen Simulationsszenarien nicht beobachtet werden. Eine Erklärung dafür könnte sein, dass

6 Fazit und Ausblick

Ziel dieser Bachelorarbeit war es, in SPH-Flüssigkeitssimulationen die Druckberechnung durch eine Zustandsgleichung mit der Druckberechnung durch das Lösen eines Gleichungssystems zu vergleichen. Zum einen wurde die Druckberechnung mit Zustandsgleichung

$$p_f = \max\left(k\left(\frac{\rho_f}{\rho^0} - 1\right), 0\right) \tag{6.1}$$

implementiert. Zum anderen wurde IISPH implementiert, in der zur Druckberechnung ein Gleichungssystem iterativ gelöst wird.

Es wurden mit diesen Druckberechnungsmethoden verschiedene Simulationen durchgeführt. Dabei stellte sich heraus, dass sich die Druckberechnung mit IISPH im Vergleich zur Druckberechnung mit der Zustandsgleichung im gezeigten Dammbruchszenario etwas schneller berechnen lässt. Wird der erlaubte Dichtefehler etwas niedriger gesetzt, so zeigt sich ein ähnliches Bild. Hier muss bei IISPH ein geringerer Zeitschritt gewählt werden, damit die Druckberechnung nicht zu lange dauert, da die Anzahl an Iterationen bei größeren Zeitschritten zunimmt. Ebenso muss bei SSPH eine größere Steifigkeitskonstante k gewählt werden, um den Dichtefehler zu verkleinern. Hier muss ebenso ein geringerer Zeitschritt gewählt werden, damit die Simulation bei größeren Werten für k stabil bleibt. Im Dammleckszenario mit größerer Flüssigkeitssäule und mehr Partikeln dagegen war die Druckberechnung mit der Zustandsgleichung etwas schneller.

Berücksichtigt werden muss auch, dass bei IISPH für einen gegebenen erlaubten maximalen Dichtefehler lediglich der Zeitschritt gewählt werden muss. Dieser Zeitschritt kann großzügig gewählt werden, da für größere Zeitschritte zwar mehr Iterationen berechnet werden, jedoch müssen für den gleichen simulierten Zeitraum gleichzeitig weniger Simulationsschritte berechnet werden. Daher erlauben unterschiedliche Zeitschritte eine nahezu ähnlich schnelle Berechnung.

Bei SSPH dagegen kann der maximale Dichtefehler nicht direkt vorgegeben werden. Der Dichtefehler, der während der Simulation entsteht, ist abhängig von der gewählten Steifigkeitskonstante k. Diese Konstante muss so gewählt werden, dass im Laufe der Simulation der Dichtefehler unter dem maximal erlaubten Wert bleibt. Zudem muss der Zeitschritt so gewählt werden, dass der Zeitschritt für eine schnelle Berechnung möglichst groß gewählt ist. Gleichzeitig muss der Zeitschritt aber klein genug sein, damit die Simulation noch stabil ist. Daher müssen zur Abstimmung

dieser Parameter zu Testzwecken vorher viele verschiedene Simulationen berechnet werden.

Aus diesem Grund kann gesagt werden, dass die Handhabung der Simulationen bei SSPH wesentlich komplizierter ist als bei IISPH. Andererseits ist IISPH komplexer und damit auch fehleranfälliger zu implementieren. Es ergab sich bei den vorgestellten Szenarien bei ähnlicher Performanz dennoch ein klarer Vorteil für IISPH. In Folge dessen kann aus den Ergebnissen dieser Arbeit geschlussfolgert werden, dass sich bei der Berechnung von Flüssigkeitssimulationen überwiegend die IISPH-Methode zur Druckberechnung empfiehlt.

Literaturverzeichnis

[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. ISBN: 1365-2966 Publisher: Oxford University Press Oxford, UK.

[ICS+14] Markus Ihmsen, Jens Cornelis, Barbara Solenthaler, Christopher Horvath, and Matthias Teschner. Implicit Incompressible SPH. *IEEE Transactions on Visualization and Computer Graphics*, 20(3):426–435, March 2014.

[IOS+14] Markus Ihmsen, Jens Orthmann, Barbara Solenthaler, Andreas Kolb, and Matthias Teschner. SPH fluids in computer graphics. In *EUROGRAPHICS 2014/S. LEFEBVRE AND M. SPAGNUOLO*. Citeseer, 2014.

[KBST20] Dan Koschier, Jan Bender, Barbara Solenthaler, and Matthias Teschner. Smoothed particle hydrodynamics techniques for the physics based simulation of fluids and solids. *arXiv preprint arXiv:2009.06944*, 2020.

[LD08] Ares Lagae and Philip Dutré. Compact, fast and robust grids for ray tracing. In *Computer Graphics Forum*, volume 27, pages 1235–1244. Wiley Online Library, 2008. Issue: 4.

[Luc77] L. B. Lucy. A numerical approach to the testing of the fission hypothesis. *The Astronomical Journal*, 82:1013, December 1977.

[Mon92] Joe J. Monaghan. Smoothed particle hydrodynamics. *Annual review of astronomy and astrophysics*, 30:543–574, 1992. ISBN: 0066-4146.

[Mon05] Joe J. Monaghan. Smoothed particle hydrodynamics. *Reports on progress in physics*, 68(8):1703, 2005. ISBN: 0034-4885 Publisher: IOP Publishing.

[Pri12] Daniel J. Price. Smoothed particle hydrodynamics and magnetohydrodynamics. *Journal of Computational Physics*, 231(3):759–794, February 2012.

[SL03] Songdong Shao and Edmond Y.M. Lo. Incompressible SPH method for simulating Newtonian and non-Newtonian flows with a free surface. *Advances in Water Resources*, 26(7):787–800, July 2003.