Physikalisch-basierte Simulation von Flüssigkeiten in interaktiven Umgebungen

Dissertation
zur
Erlangung des akademischen Grades
Doktor-Ingenieur (Dr.-Ing.)
der Fakultät für Informatik und Elektrotechnik
der Universität Rostock

vorgelegt von
Dipl.-Inf Dipl.-Phys. Hilko Cords
geboren am 04.04.1978 in Oldenburg
wohnhaft in Rostock

Rostock, 01.07.2009

urn:nbn:de:gbv:28-diss2009-0190-4

Gutachter:

1. Prof. Dr. Dietmar Jackèl, Universität Rostock
2. Prof. Dr. Oliver G. Staadt, Universität Rostock
3. Prof. Dr. Matthias Teschner, Universität Freiburg

Datum der Verteidigung: 06.10.2009

Classification (ACM CCS 1998): I.3.7 Computer Graphics - *Three-Dimensional Graphics and Realism*, I.6.8 Simulation and Modeling - *Types of Simulation*.

Keywords: Real-Time Liquid Simulation, Physically-based Liquid Simulation, Water Simulation, Surface Extraction from Point Clouds, Interactive Labeling.

URN: urn:nbn:de:gbv:28-diss2009-0190-4

Kurzfassung Die realistische, physikalisch-basierte Repräsentation von Flüssigkeiten in der Computergraphik umfasst die drei Schritte der *(Strömungs-) Simulation*, *Oberflächenextraktion* und *Darstellung*. Die Erzeugung realistischer Animationen mit Hilfe dieser Schritte ist rechentechnisch kostenintensiv und in der Regel nicht in Echtzeit durchführbar. Verfahren, die dennoch plausible Ergebnisse in Echtzeitumgebungen anstreben, müssen daher einen Kompromiss zwischen Qualität und Performanz eingehen.

Das Ziel dieser Dissertation ist die Beschleunigung der einzelnen Schritte der Repräsentation, um den Detailgrad virtueller Flüssigkeiten in Echtzeitumgebungen zu vergrößern. Hierzu wird ein allgemeines Schema eingeführt, welches die Verwendung verschiedener Simulationstechniken systematisiert, um problemangepasste und effiziente Repräsentationen zu ermöglichen. Der Schwerpunkt liegt in der Reduktion der Dimensionalität einer *Simulation* unter Berücksichtigung der zu repräsentierenden physikalischen Effekte. Zur *Oberflächenextraktion* dreidimensionaler Partikelwolken, wie sie zum Beispiel aus einer Flüssigkeitssimulation resultieren können, wird eine schnelle, Bildraum-basierte Methode eingeführt. Bezüglich des Aspekts der *Darstellung* werden effiziente Ansätze zur Approximation optischer Effekte und Visualisierung physikalischer Eigenschaften beschrieben.

Die Möglichkeiten und Grenzen der beschriebenen Konzepte wurden anhand prototypischer Implementierungen evaluiert. Dabei zeigte sich, dass die vorgestellten Methoden die Laufzeit beziehungsweise Qualität bestehender Techniken der interaktiven Flüssigkeitsrepräsentation verbessern.

Abstract Realistic physically-based visual representations of liquids require three main steps to be conducted: *simulation*, *surface extraction* and *rendering*. Since each of these steps in itself is a computationally complex task, realistic representations of liquids is a time-consuming endeavor. As a result, approaches that aim for real-time rendering and interactivity usually have to trade off quality for rendering speed.

The goal of this thesis is to accelerate the representations of realistic liquids to achieve real-time frame rates. This is accomplished by devising novel dedicated acceleration techniques for each of the above mentioned steps. Firstly, a general scheme for adaptive simulation is developed, which serves as the basis to appropriately choose from the set of available simulation techniques. The key goal of this scheme is to reduce simulation dimensionality and resolution while considering the requirements of the physical effects to be simulated. Secondly, efficient approaches for surface extraction from dynamic point clouds are described. Thirdly, rendering is improved by effective approximations of optical effects and visualization of physical properties.

Furthermore, the concepts described by this thesis were implemented as a prototype, based on which the concepts' benefits and drawbacks were analyzed. The results show that the presented concepts significantly improve the quality of realistic real-time representations of liquids while on the other hand allowing for real-time frame rates.

Danksagung Diese Arbeit ist im Rahmen des Graduiertenkollegs 466 *Verarbeitung, Verwaltung, Darstellung und Transfer multimedialer Daten - technische Grundlagen und gesellschaftliche Implikationen* durch die DFG gefördert worden. Ich möchte mich bei der DFG für die finanzielle Unterstützung bedanken, die neben dem Stipendium auch bereichernde Forschungsaufenthalte im Ausland mit eingeschlossen hat.

Vor allen Dingen möchte ich mich aber bei meinem Betreuer Herrn Prof. Dietmar Jackel bedanken, der sich schon vor Beginn meiner Promotion sehr für mich eingesetzt hat und es mir im Rahmen des Graduiertenkollegs ermöglichte, mich ausführlich mit einer spannenden Thematik auseinander zu setzen — auf meine Weise und mit großen Freiheiten. Des Weiteren möchte ich mich bei Herrn Prof. Oliver Staadt für die fruchtbare Zusammenarbeit nach dem Ausscheiden von Herrn Prof. Dietmar Jackel bedanken. Herrn Prof. Matthias Teschner danke ich für die freundliche, kurzfristige und unkomplizierte Bereitschaft diese Arbeit extern zu begutachten.

Außerdem möchte ich mich bei Martin Luboschik bedanken, ohne dessen Beistand diese Arbeit wohl kein erfolgreiches Ende gefunden hätte. Ausdrücklicher Dank gilt auch Falko Löffler, Angela Brennecke und Sebastian Schwanke, ohne die das Promovieren und die damit verbundenen Kaffeepausen definitiv wesentlich weniger Spaß gemacht hätten. Mein herzlicher Dank gilt auch Frau Prof. Heidrun Schumann und ihrer Arbeitsgruppe für ihre stete Unterstützung und Anregungen während meiner Promotion.

Hilko Cords,
Rostock, Oktober 2009.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1. Einleitung

Motivation Materie im flüssigen Aggregatszustand wird als *Flüssigkeit* bezeichnet. Das physikalische Verhalten von Flüssigkeiten wird im Rahmen der Strömungsmechanik beschrieben. Die numerische und möglichst akkurate Simulation des Strömungsverhaltens von Flüssigkeiten ist wichtiger Bestandteil der Forschung und Entwicklung. Aufgrund der komplexen Dynamik von Flüssigkeiten sind derartige physikalische Simulationen jedoch zu rechenaufwendig für interaktive Anwendungen, wie zum Beispiel Virtual Reality-Umgebungen oder Computerspiele.

Die Computergraphik adaptiert existierende physikalische Simulationsmethoden, um realistische Animationen virtueller Flüssigkeiten zu realisieren. Ziel dieser Bemühungen ist eine automatische, realistische und plausible Dynamik und Darstellung. Vergleichbaren Realismus mit klassischen Animationstechniken zu erzielen, ist mit großem handwerklichen Zeitaufwand verbunden. In Echzeitumgebungen, in denen der Benutzer mit der Umgebung interagiert, gibt es keine nennenswerte Alternative zu Simulationen, da die Interaktion unvorhersehbar ist. In der Regel werden approximierende, physikalische Simulationsmethoden mit klassischen Animationsmethoden kombiniert, so dass für die Simulationsmethoden in der Computergraphik der Begriff *physikalisch-basiert* verwendet wird. Trotz der Verwendung physikalisch-basierter Simulationstechniken zur Animation von Flüssigkeiten benötigt die Erzeugung photo-realistischer Ergebnisse jedoch in der Regel viele Minuten oder Stunden pro Bild (zum Beispiel [LTKF08, LAD08]). Denn neben der *Simulation* gilt es eine sich permanent ändernde *Oberfläche* der Flüssigkeit zu extrahieren und *optische Eigenschaften* realistisch darzustellen. Diese drei Schritte sind rechentechnisch aufwendig. In dieser Arbeit wird das Ergebnis dieser drei Schritte zusammenfassend mit *Flüssigkeitsrepräsentation* bezeichnet.

Die Leistungsverbesserungen von Standard-PC-Hardware hat die Ambitionen der Forschung hinsichtlich der Flüssigkeitsrepräsentation auf diesen Systemen auch in Richtung der interaktiven oder echtzeitfähigen Darstellung geschürt. Das Deutsche Institut für Normung (DIN) bezeichnet Echtzeitsysteme als Rechensysteme, deren *Verarbeitungsergebnisse innerhalb einer vorgegebenen Zeitspanne verfügbar sind.* Diese Definition kann den Anforderungen an Echtzeitsysteme in der Computergraphik nicht gerecht werden, da lediglich eine vorgegebene und keine beschränkte Zeitspanne gefordert ist. Dennoch gibt es keine eindeutige und allgemein gültige Definition von Echtzeit in der Computergraphik. Insofern bezieht sich diese Arbeit bei den Begriffen *Interaktivität* und *Echtzeit* in Anlehnung an [CLHM97, AMHH08] und das allgemeine Verständnis in der Computergraphik auf Bildwiederholraten, die eine Simulation und Darstellung bei einer Geschwindigkeit realisieren, die für

den Betrachter als interaktiv wahrgenommen werden kann und in keiner spürbaren Latenz (Zeitverzögerung < 15 ms) resultiert. Dementsprechend werden die beiden Begriffe in dieser Arbeit verwendet. Wenngleich auch keine eindeutige Definition existiert, so sei doch als Größenordnung erwähnt, dass in der Computergraphik oftmals Bildwiederholraten größer 10 Hz mit *interaktiv* und größer $30 - 50$ Hz mit *echtzeitfähig* bezeichnet werden. Ab Bildwiederholraten von 72 Hz und mehr ist eine wahrnehmbare Grenze erreicht und ein Unterschied ist für den Betrachter nicht mehr wahrzunehmen [AMHH08]. Die Problematik der interaktiven, physikalisch-basierten Flüssigkeitsrepräsentation besteht darin, innerhalb dieser sehr kleinen Zeitfenster Simulationsschritte auszuführen und gleichzeitig die Flüssigkeit darzustellen. Denn nur so kann eine direkte Interaktion mit der virtuellen Flüssigkeit erfolgen. Anwendungsfelder liegen in den Bereichen der Virtual Reality-, Simulations- und Virtual Surgery-Umgebungen, aber vornehmlich im Bereich der Computerspiele.

Diese Arbeit präsentiert neue Methoden für die Flüssigkeitsrepräsentation in interaktiven Umgebungen. Denn obwohl hierzu bereits Ergebnisse präsentiert wurden wurden (zum Beispiel [MCG03, YHK07]), werden neue Methoden benötigt, um mehr Details darzustellen oder die simulierte Flüssigkeitsmenge zu vergrößern. Mit dieser Blickrichtung präsentiert diese Arbeit zunächst eine abstrakte Sicht auf die Flüssigkeitsrepräsentation. Es wurden sechs Methoden entworfen, die neue Beiträge in allen drei Schritten der interaktiven Flüssigkeitsrepräsentation darstellen.

Im nächsten Abschnitt erfolgt eine Problembeschreibung, anschließend werden die konkreten Beiträge dieser Arbeit benannt. Den Abschluss dieses Kapitels stellt eine Übersicht über den Aufbau dieser Arbeit dar.

Problembeschreibung Reale Flüssigkeiten besitzen eine Vielzahl physikalischer Eigenschaften, die zum Beispiel zu komplexen und teilweise chaotischen Bewegungen und Verhaltensweisen führen können. Darüber hinaus unterliegt die Oberfläche in dynamischen Situationen einer permanenten Veränderung. Zudem besitzen Flüssigkeiten im Allgemeinen und transparente Flüssigkeiten im Besonderen komplexe optische Eigenschaften, die zum Beispiel zur Reflexion, Brechung und Absorption von Lichtstrahlen führen. Die realistische Repräsentation dieser Merkmale ist rechentechnisch aufwendig.

Entsprechend kann die Flüssigkeitsrepräsentation in der Computergraphik in drei unabhängig voneinander auszuführende Schritte unterteilt werden, die in dieser Arbeit in der *allgemeine Liquid-Pipeline* zusammengefasst werden — siehe Abbildung 1.1.

Die Grundproblematik der interaktiven Flüssigkeitsrepräsentation besteht in der effizienten Ausführung der allgemeinen Liquid-Pipeline. In *nicht-interaktiven* Umgebungen, also Umgebungen, in denen keine Beschränkungen der Berechnungszeit existieren, besteht das Vorgehen in der Regel darin, für den ersten Schritt eine hochaufgelöste, physikalische Strömungssimulation durchzuführen. Die sich ergebende freie Oberfläche wird bei hoher Auflösung extrahiert — zum Beispiel mit Hilfe eines *Marching Cubes*-Algorithmus [LC87]. Die resultierende Oberfläche wird dann unter Ver-

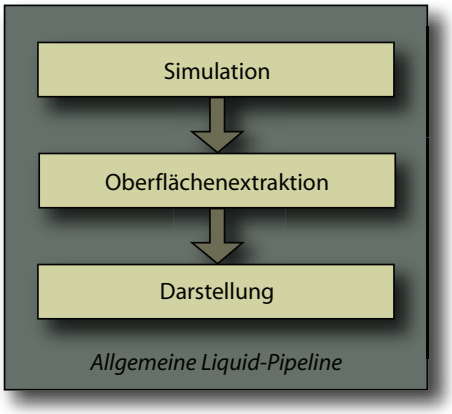


Abb. 1.1.: Die allgemeine Liquid-Pipeline.

wendung von Techniken wie *Ray Tracing* [Whi80] und *Photon Mapping* [JC95] realistisch dargestellt. Ein derartiges Vorgehen ist in interaktiven Umgebungen jedoch nicht möglich. Interaktivität kann lediglich durch Verwendung verhältnismäßig niedriger Simulations- und Oberflächenextraktionsauflösungen erzielt werden, so dass die resultierenden Oberflächen vergleichsweise wenig Details aufweisen. Die optischen Eigenschaften werden approximiert, um visuelle Effekte wie optische Brechungen mit hoher Geschwindigkeit darstellen zu können.

Das Ziel der Forschungsbemühungen im Bereich der interaktiven Repräsentation ist es, die drei Schritte der allgemeinen Liquid-Pipeline in Echtzeit auszuführen und dabei dennoch realistisch wirkende Ergebnisse zu erzielen. Obwohl bereits beachtliche Fortschritte im Bereich der interaktiven Repräsentation erzielt wurden, werden neue Techniken benötigt, um Flüssigkeiten

- detaillreicher und
- in größeren Mengen

in Echtzeitumgebungen repräsentieren zu können. Diese beiden Punkte stellen auch existierende dreidimensionale, interaktive Flüssigkeitssimulationen vor erhebliche Probleme. In gewisser Weise bedingen sich diese beiden Punkte gegenseitig. Der Aufwand wächst mit der simulierten Flüssigkeitsmenge, so dass die Details abnehmen und vice versa. Ziel ist es somit, einen möglichst ausgewogenen Kompromiss zwischen Qualität und Performanz zu finden.

An dieser Stelle sei explizit Wasser erwähnt, da es für die Computergraphik die wohl am häufigsten zu repräsentierende Flüssigkeit darstellt. Gerade die Repräsentation transparenten Wassers mit seiner sehr niedrigen Viskosität und der daraus resultierenden hohen Dynamik ist eine besondere Herausforderung.

Beiträge Die Beiträge dieser Arbeit schließen sich den im vorhergehenden Abschnitt beschriebenen Bemühungen zur Verbesserung der Qualität und Möglichkeiten existierender Methoden zur interaktiven Flüssigkeitsrepräsentation an. Im Fokus steht die Realisierung einer effizienten und flexiblen Repräsentation von Flüssigkeiten. Dabei präsentiert die vorliegende Arbeit Beiträge für jeden der drei Schritte der allgemeinen Liquid-Pipeline. Ein Schwerpunkt liegt auf dem Schritt der Simulation, da dieser das endgültige Verhalten der Flüssigkeit maßgeblich bestimmt. Beiträge für die schnelle Oberflächenextraktion und die -darstellung komplementieren die Verbesserungen im Bereich der Simulation.

Ausgangspunkt ist die Beobachtung, dass eine einzige Repräsentationsmethode nicht notwendigerweise den Anforderungen in interaktiven Umgebungen genügt — also zum Beispiel nicht zugleich große Flüssigkeitsmengen, wie eine Ozeanoberfläche, und dreidimensionale Flüssigkeitseffekte, wie einen Wasserfall, darstellen kann. Aus diesem Grund thematisiert diese Arbeit zunächst eine

- **abstrakte Sicht auf die Simulation von Flüssigkeitsphänomenen in interaktiven Umgebungen**. Diese wird in einem allgemeinen Schema zusammengefasst. Anhand des Schemas kann für konkrete Umgebungen entschieden werden, welche Simulationsmethoden geeignet bzw. erforderlich sind, um die gewünschten Eigenschaften zu repräsentieren. Dabei ermöglicht das Schema nicht nur einen allgemeinen Blick auf die Realisierung räumlich getrennter Simulationen. Unter Verwendung des Schemas wurden direkte Kopplungen von Simulationsmethoden realisiert, deren Ergebnisse, Geschwindigkeit oder Möglichkeiten über die existierender Verfahren hinausgehen. Zusätzlich können anhand des Schemas existierende Arbeiten klassifiziert werden, die verschiedene Simulationsmethoden verwenden.

Konkret präsentiert diese Arbeit neue, problemangepasste Methoden zur Simulation von Flüssigkeiten, die aus dem entwickelten Schema abgeleitet werden können und die durch Verwendung verschiedener Simulationsmethoden unterschiedlicher Dimensionen die Möglichkeiten erweitern und die Qualität verbessern.

- **Bewegte Oberflächensimulationsgitter [CS09b]** Ziel dieses Ansatzes ist es, eine adaptive und effiziente Simulation großer Füssigkeitsflächen (zum Beispiel Ozean) mit Objekt-Interaktion (zum Beispiel Boote) zu ermöglichen. Dazu werden Simulationen lediglich in notwendigen Bereichen durchgeführt, die bei Bedarf dynamisch sein können. Zudem werden effektive Methoden zur Erzeugung von Heckwellen und zur Animation von Schaum gegeben.

- **2D Strömungssimulation in Kombination mit 2D Oberflächensimulation [Cor08]** Diese Kombination ermöglicht die effiziente Beschreibung der Wellenausbreitung in zweidimensionalen Strömungssimulationen. So können Objekte von der Strömung beeinflusst werden und zugleich selbst Oberflächenwellen erzeugen. Zudem können Oberflächenwellen in strömenden Gewässern modelliert werden.

- **3D Strömungssimulation in Kombination mit 2D Oberflächensimulation** [**Cor07b**] In Echtzeitumgebungen können dreidimensionale Strömungssimulationen wegen der hohen Komplexität lediglich undetaillierte Flüssigkeiten repräsentieren. Eine effiziente Oberflächensimulation, die an die Strömungssimulation gekoppelt wird, ermöglicht zusätzlich die Darstellung filigraner Kapilarwellen. Diese könnten mit einer reinen Strömungssimulation lediglich bei sehr hohen Simulationsauflösungen realisiert werden und wären dann nicht echtzeitfähig.

- **Brechende Wellen** [**CS08**] Eine dreidimensionale brechende Welle kann aus Kombinationen zweidimensionaler Strömungssimulationen in Echtzeit erzeugt werden. Die Symmetrie einer brechenden Welle wird dabei ausgenutzt, um aus einer zweidimensionalen brechenden Welle eine dreidimensionale zu konstruieren. Dieser vorgestellte Ansatz ist der Erste, der eine physikalisch-basierte Simulation zur Repräsentation brechender Wellen in Echtzeitumgebungen verwendet.

Im Bereich der Oberflächenextraktion, dem zweiten Schritt der allgemeinen Liquid-Pipeline, stellt diese Arbeit das folgende Verfahren vor:

- **Effiziente Oberflächendarstellung Partikel-basierter Flüssigkeiten** [**CS09a**] Dabei handelt es sich um eine Bildraum-basierte Oberflächenextraktion aus dynamischen Partikelwolken. Die Methode wird vollständig auf der GPU ausgeführt und erzeugt kein polygonales Gitter. Deshalb ist der Ansatz in der Ausführung ausgesprochen kostengünstig und gut geeignet für die Darstellung dreidimensionaler Flüssigkeiten in interaktiven Umgebungen.

Im Bereich der Darstellung von Flüssigkeiten präsentiert diese Arbeit neben einigen Ideen zu Approximationen optischer Effekte folgende neue Beiträge:

- **Optische Brechung oberflächenschneidender Objekte** [**Cor07a**] Die Darstellung von Flüssigkeiten in interaktiven Umgebungen erfolgt unter Verwendung von Approximationen für Reflexionen und Brechungen. Die Verwendung existierender, approximativer Methoden führt jedoch bei polygonalen Objekten, die die Flüssigkeitsoberfläche schneiden, zu Artefakten bzw. unrealistischen optischen Unterbrechungen an der Grenzschicht. Die präsentierte Methode verringert dieses Problem für dynamische, Höhenfeld-basierte Flüssigkeitsoberflächen. Ein weiterer Fokus der Methode liegt auf der Darstellung der für optische Brechungen typischen Skalierung und Winkelverschiebung von Objekten in Abhängigkeit des Sichtwinkels.

- **Beschriftung Partikel-basierter Flüssigkeiten** [**LSC08, CLS09**] Die Visualisierung der Parameter einer Partikel-basierten Flüssigkeitssimulation kann mit der vorgestellten interaktiven, Partikel-basierten Beschriftungsmethode erfolgen. Diese kann während der Entwicklung Partikel-basierter Simulationen, aber auch der Exploration der generierten Datensätze, ein wichtiges

Hilfsmittel darstellen. So können physikalische Parameter der Partikel während der Simulation mit hoher Geschwindigkeit quantitativ dargestellt werden. Die Methode ist derart allgemein und flexibel, dass Anwendungsmöglichkeiten in vielen weiteren Bereichen gegeben sind, wie zum Beispiel der Informations-Visualisierung.

Es sei angemerkt, dass die Veröffentlichungen [CS08], [CS09a] und [CS09b] in Zusammenarbeit mit Oliver Staadt und die Veröffentlichungen [LSC08] und [CLS09] in Zusammenarbeit mit Martin Luboschik und Heidrun Schumann erstellt worden sind.

Struktur der Arbeit Diese Arbeit ist in 8 Kapitel unterteilt. Im folgenden Kapitel 2 werden die physikalischen Grundlagen für eine Flüssigkeitssimulation in der Computergraphik dargestellt. In Kapitel 3 wird der Stand der Technik der Flüssigkeitssimulation im Feld der Computergraphik aufgearbeitet und diskutiert. Anschließend wird in Kapitel 4 ein allgemeines Schema zur Flüssigkeitssimulation in Echtzeitumgebungen eingeführt. Darauf aufbauend werden neue, konkrete Methoden in den Kapiteln Simulation (Kapitel 5), Oberflächenextraktion (Kapitel 6) und Darstellung (Kapitel 7) präsentiert und diskutiert. Schließlich erfolgt eine Zusammenfassung und ein Ausblick in Kapitel 8. Da diese Arbeit Beträge auf allen Stufen der allgemeinen Liquid-Pipeline präsentiert, orientiert sich die Struktur der Kapitel 3,4 und 5–7 für eine systematische Präsentation an der Pipeline.

2. Physikalische Grundlagen

Dieses Kapitel stellt die für diese Arbeit fundamentalen physikalischen Modelle vor. Es liefert dabei keine vollständige Betrachtung der Modelle, sondern eine kurze Einführung in die wichtigsten physikalischen Grundlagen, die für die Repräsentation von Flüssigkeiten in der Computergraphik notwendig sind. Im weiteren Verlauf dieses Textes wird dann auf die entsprechenden Abschnitte des vorliegenden Kapitels verwiesen.

Begonnen wird im Folgenden mit den Navier-Stokes-Gleichungen, die die Grundlage der Strömungssimulation darstellen. Es folgt die Beschreibung der Wellengleichung und starren Körpern unter Berücksichtigung der Interaktion mit Flüssigkeiten. Schließlich erfolgt ein kurzer Überblick über wichtige optische Effekte, die bei transparenten Flüssigkeiten zu modellieren sind.

2.1. Navier-Stokes Gleichungen

Die allgemeine Bewegung von Fluiden (Gase und Newtonsche Flüssigkeiten) wird physikalisch durch die Navier-Stokes Gleichungen beschrieben. Die Gleichungen sind nach den Physikern Claude-Louis Navier (1785–1836) und George Gabriel Stokes (1819–1903) benannt, die unabhängig voneinander das zugehörige Gleichungssystem vorgestellt haben.

Die Navier-Stokes Gleichungen können in einer *kompressiblen* und einer *inkompressiblen* Form formuliert werden. Ändert ein Stoff unter externer Druckeinwirkung sein Volumen, so wird diese Eigenschaft als Kompressibilität bezeichnet. Umgekehrt bezeichnet Inkompressibilität die Eigenschaft, wenn bei externer Druckeinwirkung keinerlei Volumenänderung auftritt. Reale Materialien sind immer kompressibel. Im makroskopischen werden Gase jedoch als kompressibel und Flüssigkeiten und Festkörper in der Regel als inkompressibel betrachtet, da der Grad der Kompression derart gering ist, dass er oftmals vernachlässigt werden kann. Aus diesem Grund werden im Folgenden lediglich die inkompressiblen Navier-Stokes Gleichungen betrachtet. Es sei erwähnt, dass die inkompressible Betrachtung jedoch nicht die Darstellung von Effekten ermöglicht, die aus Dichteschwankungen resultieren. So ist zum Beispiel die Repräsentation von Schallausbreitungen oder extremen Temperaturgradienten mit der inkompressiblen Darstellung nicht möglich. Derartige Effekte sind jedoch in der Computergraphik von untergeordneter Bedeutung.

Die inkompressiblen Navier-Stokes Gleichungen beschreiben die Impulsänderungen einer Flüssigkeit in Abhängigkeit der dämpfenden Viskositätskräfte, der Änderung des Druckes, Gravitation und anderer externer Kräfte. Damit stellen

die Navier-Stokes Gleichungen im Prinzip das Pendant des zweiten Newtonschen Gesetzes für Fluide dar, welches in der Mechanik die Impulsänderung in Abhängigkeit der wirkenden Kräfte beschreibt. Die große Bedeutung der Gleichungen in Wissenschaft und Technik liegt in den zahlreichen Anwendungen begründet, denn die Strömung von Luft- und Wassermassen wird unter anderem mit den Navier-Stokes Gleichungen beschrieben. Unter Berücksichtigung von Grenzschichten können Strömungen beispielsweise innerhalb eines Rohres oder um Körper modelliert werden. Anwendungsbereiche liegen zum Beispiel im Feld der Meteorologie, Ozeanographie und Klimaforschung, des Flugzeug-, Schiffs- und allgemein Fahrzeugbaus.

Die inkompressiblen Navier-Stokes Gleichungen viskoser Strömungen stellen ein System aus nichtlinearen partiellen Differentialgleichungen zweiter Ordnung dar und bestehen zum einen aus dem Impulssatz

$$\rho\left(\frac{\partial \mathbf{v}}{\partial t} + (\mathbf{v}\cdot\nabla)\mathbf{v}\right) = -\nabla p + \mu\triangle\mathbf{v} + \rho\mathbf{F}_{\text{ext}} \tag{2.1}$$

und der Kontinuitätsgleichung, die die Massenerhaltung beschreibt:

$$\frac{\partial \rho}{\partial t} + \nabla\cdot(\rho\mathbf{v}) = 0. \tag{2.2}$$

Dabei ist $\mathbf{v}$ das Geschwindigkeitsfeld, ρ das Dichtefeld, p das Druckfeld, Nabla $\nabla = (\frac{\partial}{\partial x_1}, \frac{\partial}{\partial x_2}, \frac{\partial}{\partial x_3})^T$, Laplace-Operator $\triangle = \nabla^2$, μ die Viskositätskonstante und $\mathbf{F}_{\text{ext}}$ die wirkenden externen Kräfte (zum Beispiel Gravitation). Die linke Seite des Impulssatzes beschreibt die Beschleunigung der Flüssigkeit (zeitabhängige und zeitunabhängige Konvektion) und die rechte Seite beschreibt die Druckabhängigkeit, die Viskosität und externe wirkende Kräfte. Da Flüssigkeiten als inkompressibel betrachtet werden, kann die Dichte als konstant angenommen werden ($\frac{\partial \rho}{\partial t} = 0$). Damit ergibt sich aus der Kontinuitätsgleichung 2.2 die folgende Massenerhaltung:

$$\nabla\cdot\mathbf{v} = 0. \tag{2.3}$$

Folglich handelt es sich um divergenzfreie Strömungen, so dass bei konstanter Dichte in keinem Volumenelement Masse erzeugt oder vernichtet werden kann. Somit entspricht das in ein gegebenes Volumen einfließende Flüssigkeitsvolumen dem im gleichen Zeitraum ausfließenden Volumen.

Numerische Lösungen der Navier-Stokes Gleichungen Die numerischen Methoden zur Lösung der Navier-Stokes Gleichungen können klassifiziert werden in

- Euler Verfahren und
- Lagrange Verfahren.

Erstere bezeichnen Gitter-basierte Verfahren, die den gesamten Simulationsraum unter Verwendung von Gitterzellen diskretisieren. Letztere bezeichnen Partikel-basierte Verfahren, wobei die Flüssigkeit unter Verwendung von Partikeln diskretisiert wird. In der Computergraphik werden beide Verfahren verwendet. Die Gitter-basierten Ansätze verwenden in der Regel Finite Differenzen Methoden (FDM) oder Finite Volumen Methoden (FVM). Physikalische Eigenschaften werden anhand benachbarter Gitterzellen bestimmt. Grenzschichten müssen bei Euler-Verfahren explizit behandelt werden. Für Kollisionsobjekte erfolgt die Behandlung über Randbedingungen, die die Strömung in den zugehörigen Gitterzellen steuern, damit zum Beispiel keine Strömung in Objekte hineinfließen kann. Entsprechend werden auch die freien Oberflächen behandelt, da diese sich permanent ändern und in jedem Zeitschritt aktualisiert werden müssen. Es sind zahlreiche Verfahren entwickelt worden, um diese geeignet abzutasten (siehe dazu auch Kapitel 3). Für detaillierte Ergebnisse — auch in turbulenten Situationen — sind jedoch hohe Gitterauflösungen notwendig, so dass die Methode schnell aufwendig wird. Ein Nachteil der Gitter-basierten Verfahren ist die numerische Diffusion. Diese führt in computergraphischen Applikationen zu sichtbarem Verlust von Flüssigkeitsvolumen. Für weitere Details sei auf Kapitel 3 und zum Beispiel [GDN95] verwiesen.

Lagrange Verfahren diskretisieren das Volumen unter Verwendung von Partikeln. Das Verhalten der Flüssigkeit wird mit Hilfe der Partikelpositionen und -geschwindigkeiten simuliert. Die physikalischen Strömungseigenschaften können aus diesen Informationen bestimmt werden. Da die Partikel das Flüssigkeitsvolumen direkt diskretisieren, kann es nicht zu numerischer Diffusion kommen. Die Inklusion von Randbedingungen erfolgt über Partikelkollisionen und deren Behandlungen. Zudem ist Massenerhaltung direkt garantiert, da die Partikel selbst die Masse ihrer Umgebung repräsentieren und keine Partikel erzeugt oder vernichtet werden. Zusätzlich kann der konvektive Beschleunigungsterm $(\mathbf{v}\cdot\nabla)\mathbf{v}$ vernachlässigt werden, da die Partikel selbst die Strömungsinformationen tragen und damit die Konvektion direkt einschließen. Eine in der Computergraphik häufig verwendete Lagrange Methode ist die Smoothed Particle Hydrodynamics (SPH) Methode, da sie sehr effizient ausgeführt werden kann. Für weitere Details wird auf Kapitel 3, Anhang A und beispielsweise [Mon92] verwiesen.

2.2. Wellengleichung

Die allgemeine Wellengleichung beschreibt die zeitliche und räumliche Wellenausbreitung mit konstanter Ausbreitungsgeschwindigkeit. In der Computergraphik kann die zweidimensionale Wellengleichung verwendet werden, um die radiale Ausbreitung von Oberflächenwellen zu beschreiben. Mit der Zeit t und Position $\mathbf{x}$ kann sie mit folgender linearen partiellen Differentialgleichung zweiter Ordnung angegeben

werden:

$$\triangle f(\mathbf{x},t) - \frac{1}{c^2}\frac{\partial^2 f(\mathbf{x},t)}{\partial t^2} = 0. \tag{2.4}$$

Dabei bezeichnet $\triangle = \nabla^2 = \sum_{i=1}^{2} \frac{\partial^2}{\partial x_i^2}$ den zweidimensionalen Laplace Operator und c die Ausbreitungsgeschwindigkeit. In Hinblick auf die Simulation von Oberflächenwellen sei erwähnt, dass die Lösung der Wellengleichung als Höhenfeld interpretiert werden kann.

2.3. Starre Körper

Die Simulation starrer Körper ist notwendig, damit Objekte sich im Rahmen einer Flüssigkeitssimulation natürlich bewegen und zugleich mit dieser interagieren können. Ziel ist zum Beispiel die Repräsentation schwimmender Objekte. Die Physik starrer Körper stellt ein Teilgebiet der Mechanik dar. Die Bewegung eines starren Körpers wird in einen Translationsanteil $\mathbf{x}(t)$ und einen Rotationsanteil $\mathcal{R}(t)$ unterteilt. $\mathbf{x}(t)$ beschreibt die Position des Schwerpunktes mit einer Translationsgeschwindigkeit $\mathbf{v}(t)$ und $\mathcal{R}(t)$ die Rotation des Körpers um den Schwerpunkt mit einer Winkelgeschwindigkeit $\omega(t)$. $\mathcal{R}(t)$ stellt dabei eine 3×3 Rotationsmatrix dar. Die Orientierung von $\omega(t)$ beschreibt die aktuelle Rotationsachse der Drehbewegung und der Betrag $|\omega(t)|$ repräsentiert die Rotationsgeschwindigkeit.

Der Schwerpunkt $\mathbf{r}_0$ des starren Körpers ist definiert über die Dichte ρ und das Volumen V:

$$\mathbf{r}_0 = \frac{1}{m}\int_V \mathbf{r}\,\rho\,dV. \tag{2.5}$$

Die Translationsbewegung des starren Körpers wird entsprechend dem 2. Newtonschen Gesetz bestimmt:

$$\mathbf{F} = m\ddot{\mathbf{r}}_0. \tag{2.6}$$

Die Rotationsbewegung des starren Körpers wird über das Drehmoment $\mathbf{M}$ bestimmt. Für Massenpunkte, die den starren Körper diskretisieren, gilt:

$$\mathbf{M} = \mathbf{r} \times \mathbf{F}. \tag{2.7}$$

Das gesamte wirkende Drehmoment kann durch Addition aller wirkenden Momente bestimmt werden. Das Drehmoment $\mathbf{M}$ zeigt in Richtung der Rotationsachse $\mathbf{x}_{\text{rot}}$. Das Massenträgheitsmoment $J_{\mathbf{x}_{\text{rot}}}$ der Rotationsachse $\mathbf{x}_{\text{rot}}$ ist definiert über

$$J_{\mathbf{x}_{\text{rot}}} = \int_V r^2\,\rho\,dV. \tag{2.8}$$

Die Winkelbeschleunigung $\alpha_{\mathbf{x}_{\text{rot}}} = \frac{d\omega_{\mathbf{x}_{\text{rot}}}}{dt}$ der Drehung um Achse $\mathbf{x}_{\text{rot}}$ ergibt sich dann mit

$$M_{\mathbf{x}_{\text{rot}}} = J_{\mathbf{x}_{\text{rot}}} \cdot \alpha_{\mathbf{x}_{\text{rot}}}. \tag{2.9}$$

Die Rotationsmatrix $\mathcal{R}(t)$ wird schließlich entsprechend der Winkelgeschwindigkeit rotiert. Für weitere Details sei an dieser Stelle auf [Bar97] verwiesen.

Die Kopplung des starren Körpers mit einer Flüssigkeit erfordert zusätzlich die Behandlung des Druckes, welcher unter anderem in Form von Auftrieb das physikalische Verhalten starrer Körper in Flüssigkeiten maßgeblich beeinflusst. Die Auftriebskraft für schwimmende, unbewegte Objekte wird nach dem Archimedischen Prinzip bestimmt. Demnach entspricht die Auftriebskraft $\mathbf{F}_{\text{a}}$ der Gewichtskraft der durch das Objekt verdrängten Flüssigkeitsmenge:

$$\mathbf{F}_{\text{a}} = \rho_{fl}\, \mathbf{g}\, V_0. \tag{2.10}$$

Dabei bezeichnet ρ_{fl} die Durchschnittsdichte der Flüssigkeit, V_0 die verdrängte Flüssigkeitsmenge und die Fallbeschleunigung wird mit $\mathbf{g} = (0, 0, -9,81\, m/s^2)^T$ beschrieben. Für schwimmende oder schwebende Objekte im Gleichgewicht entspricht die Auftriebskraft der Gewichtskraft des Körpers: $\mathbf{F}_a = \mathbf{F}_g = m\mathbf{g}$. Wenn die Dichte der Flüssigkeit die durchschnittliche Dichte des Objektes übersteigt, schwimmt das Objekt, andernfalls sinkt es.

Bei Objekten, die sich mit Hilfe eines Antriebs durch eine Flüssigkeit bewegen (zum Beispiel Boote), entsteht ein dynamischer Auftrieb, der in zwei Komponenten zerlegt wird: Dynamischer Auftrieb (orthogonal zur Strömungsrichtung) und Widerstandskraft (parallel zur Strömungsrichtung). Diese Kräfte sind zum Beispiel für die typische Bewegung von Motorbooten verantwortlich. Entsprechend der Oberflächennormalen des Objektes können diese Kräfte berechnet werden. Eine effiziente Methode zur Berechnung der Kräfte wird zum Beispiel in [YHK07] gegeben.

2.4. Optik

Dieser Abschnitt beschreibt wichtige optische Effekte, die für die Repräsentation von Flüssigkeiten in der Computergraphik zu berücksichtigen sind. Auf die Darstellung weitergehender Effekt, wie zum Beispiel Polarisation oder Beugung wird an dieser Stelle verzichtet, da diese schon im Alltag kaum zu beobachten sind und somit im Rahmen der Computergraphik vernachlässigt werden können.

Das visuelle Erscheinungsbild transparenter Flüssigkeiten ist in hohem Maße durch das Wechselspiel mit Licht gekennzeichnet. Bei dem Begriff *Optik* bezieht sich diese Arbeit auf die klassische Optik, die als Lehre vom Licht in dem Wellenbereich der elektromagnetischen Strahlung aufgefasst werden kann, der vom menschlichen Auge wahrgenommen werden kann. Diese befasst sich mit Vorgängen, die bei der Wechselwirkung von Licht mit Medien auftreten. Die optischen Effekte, die bei der

Wechselwirkung von Licht und Objekten auftreten, die wesentlich größer sind als die Wellenlänge des Lichtes selbst, werden im Teilgebiet der Strahlenoptik und der geometrischen Optik behandelt. Dazu gehört zum Beispiel die Wechselwirkung von Licht mit Linsen, Spiegeln, Prismen und Blenden und somit auch die Wechselwirkung mit transparenten dynamischen Materialien, wie es die meisten Flüssigkeiten sind. Der folgende Abschnitt behandelt optische Reflexion und Brechung, gefolgt von einem Abschnitt über Absorption. Schließlich werden die Fresnel-Gleichungen kurz diskutiert, die die Intensitätsverteilungen dieser Phänomene beschreiben.

Reflexion und Brechung Nachfolgend werden die Berechnungen von Reflexions- und Brechungsstrahlen separat beschrieben. Diese werden verwendet, um die Richtungsänderung von Sicht- und Lichtstrahlen zu bestimmen. *Sichtstrahlen* bezeichnen dabei die vom Blickpunkt ausgehenden Strahlen und *Lichtstrahlen* die von der Lichtquelle ausgehenden Strahlen. Letztere sind für die Erzeugung von Kaustiken vonnöten, die ein markantes Merkmal transparenter Flüssigkeiten darstellen.

Das *Reflexionsgesetz* besagt, dass einfallender und reflektierter Strahl mit dem Einfallslot gleiche Winkel bilden und mit diesem in einer Ebene liegen. Eine Berechnung des Reflexionsvektors unter Verwendung analytischer Geometrie, unter Verzicht auf trigonometrische Funktionen, ermöglicht eine effiziente Berechnung des Reflexionsvektors $\hat{\mathbf{r}}^{refl}$:

$$\hat{\mathbf{r}}^{refl} = 2\hat{\mathbf{n}} \cdot (\hat{\mathbf{n}} \cdot \hat{\mathbf{e}}) - \hat{\mathbf{e}}. \tag{2.11}$$

Dabei bezeichnet $\hat{\mathbf{n}}$ die normierte Oberflächennormale und $\hat{\mathbf{e}}$ den normierten einfallenden Vektor.

Lichtbrechung tritt auf, wenn ein Lichtstrahl von einem Medium 1 in ein anderes Medium 2 mit unterschiedlichen Brechungsindizes n_1, n_2 übergeht. Die resultierende Winkelveränderung von einfallendem und ausfallendem Strahl wird mit dem Snelliusschen Brechungsgesetz beschrieben: $n_1 \cdot \sin\Theta_{in} = n_2 \cdot \sin\Theta_{out}$. Die geometrische Berechnung des Brechungsvektors $\hat{\mathbf{r}}^{br}$ wird nachfolgend gegeben (Einfallender Strahl $\hat{\mathbf{e}}$, normierte Oberflächennormale $\hat{\mathbf{n}}$):

$$\hat{\mathbf{r}}^{br} = \left(\frac{n_1}{n_2} \cdot (\hat{\mathbf{n}} \cdot \hat{\mathbf{e}}) - \sqrt{1 - \frac{n_1^2}{n_2^2}(1 - (\hat{\mathbf{n}} \cdot \hat{\mathbf{e}})^2)}\right) \cdot \hat{\mathbf{n}} - \frac{n_1}{n_2} \cdot \hat{\mathbf{e}}. \tag{2.12}$$

Absorption Beim Durchgang elektromagnetischer Strahlung durch eine Schicht treten Absorptionseffekte auf, die die Intensität der reflektierten und gebrochenen Strahlung verringern. Lediglich ein Teil des einfallenden Strahlungsflusses Φ_e wird hinter der Schicht als durchgelassener Strahlungsfluss Φ_t nachgewiesen. Dabei hängt die Absorption vom Schichtmaterial und von der Wellenlänge λ der Strahlung ab. Der Strahlungsfluss im Inneren einer Schicht nimmt exponentiell mit der Eindring-

tiefe x ab und wird mit folgender Gleichung beschrieben:

$$\Phi_\lambda(x) = e^{-a(\lambda)x}. \tag{2.13}$$

Der Absorptionskoeffizient $a(\lambda)$ $[m^{-1}]$ charakterisiert dabei das Absorbermaterial.

Fresnel-Gleichungen Die Komposition aus Reflexion, Brechung und Absorption wird durch die Fresnel-Gleichungen beschrieben. In Abhängigkeit der Brechungsindizes der beteiligten Materialien n_1, n_2 und der Winkel der reflektierten und gebrochenen Strahlen θ^{refl}, θ^{br} ergeben sich folgende Intensitäten:

$$I^{refl} = \frac{1}{2}\left(\left(\frac{n_1\cos\theta^{refl} - n_2\cos\theta^{br}}{n_1\cos\theta^{refl} + n_2\cos\theta^{br}}\right)^2 + \left(\frac{n_1\cos\theta^{br} - n_2\cos\theta^{refl}}{n_1\cos\theta^{br} + n_2\cos\theta^{refl}}\right)^2\right) \tag{2.14}$$

$$I^{br} = 1 - I^{refl}. \tag{2.15}$$

Soll zusätzlich Absorption berücksichtigt werden, so gilt der Energiesatz [Stö07], der besagt, dass

$$I^{refl} + I^{br} + I^{abs} = 1. \tag{2.16}$$

Um Dispersion zu behandeln, können alle Berechnungen frequenzabhängig durchgeführt werden. In der Regel werden in der Computergraphik Approximationen der Fresnel-Gleichungen verwendet, die wesentlich effizienter zu berechnen sind und kaum wahrnehmbare Unterschiede aufweisen. Es sei bedacht, dass diese Berechnung später für jedes Fragment der darzustellenden Oberfläche durchgeführt werden muss. Die Schlick-Approximation zum Beispiel reduziert den Aufwand zur Berechnung der Fresnel-Gleichungen [Sch94]:

$$I^{refl}_{Schlick} = R_0 + (1 - R_0)(1 - \cos\theta^{refl})^5. \tag{2.17}$$

R_0 bezeichnet dabei den Reflexionsgrad bei orthogonalem Einfallswinkel.

3. Flüssigkeiten in der Computergraphik — ein Überblick

In den folgenden Abschnitten wird ein Überblick über existierende Ansätze zur Repräsentation von Flüssigkeiten in der Computergraphik gegeben. Begonnen wird mit einer kurzen Übersicht über erste historische und approximative Ansätze zur Flüssigkeitssimulation und -animation. Anschließend erfolgt die separate Behandlung der Simulations-, Oberflächenextraktions- und Darstellungsverfahren. Die Simulationsverfahren können klassifiziert werden als

- Höhenfeld-basierte Verfahren,
- Dreidimensionale Verfahren (Euler- und Lagrange Verfahren) und
- Hybride Verfahren.

Dementsprechend ist der Abschnitt der Simulationsverfahren strukturiert. Bei Präsentation der Oberflächenextraktionsverfahren wird zwischen Höhenfeld-basierten und dreidimensionalen Methoden unterschieden. Schließlich erfolgt eine Zusammenfassung, in der die Möglichkeiten und Grenzen einer interaktiven Flüssigkeitsrepräsentation behandelt werden.

Es sei angemerkt, dass dieses Kapitel einen allgemeinen Überblick über die Flüssigkeitssimulation in der Computergraphik gibt, wobei die interaktiven Techniken eingegliedert sind. Erlauben erwähnte Techniken Interaktivität, so wird dies explizit erwähnt. Obwohl der Fokus dieser Arbeit auf interaktiven Verfahren liegt, ist dieses Vorgehen sinnvoll, da die Methoden der interaktiven Verfahren eng mit denen nicht-interaktiver Verfahren verknüpft sind. So kann ein ganzheitlicher Überblick über die Flüssigkeitsrepräsentation gegeben werden.

3.1. Historische Entwicklung

Bereits 1822 und 1845 formulierten Claude Navier und George Stokes unabhängig voneinander die Navier-Stokes Gleichungen, die das physikalische Fundament der Strömungsmechanik darstellen (siehe Abschnitt 2.1). Harlow und Welch präsentierten 1965 den ersten Ansatz, diese numerisch unter Berücksichtigung freier Oberflächen im zweidimensionalen Raum zu lösen [HW65]. Es dauerte jedoch bis in die 1990er Jahre, bis eine dreidimensionale Variante des Algorithmus in der Computergraphik verwendet wurde, um dreidimensionale Flüssigkeiten

zu repräsentieren [FM96]. Doch zunächst wurden approximative Methoden und Animationstechniken verwendet.

Erste Methoden zur Darstellung von Ozeanoberflächenwellen verwendeten Überlagerungen dynamischer, trigonometrischer Funktionen [Max81, Sch80]. Ts'O und Barsky erweiterten diese Modelle, um die durch Brechung hervorgerufene Richtungsänderung der Wellen bei Wassertiefenänderungen entlang einer Küstenlinie zu modellieren [TB87]. Ihre Methode schießt ähnlich dem Ray Tracing-Verfahren [Whi80] Strahlen entlang der Wellenlaufrichtung in Richtung Küste. Die endgültige Wellenform wird mit Splines interpoliert. Mit Hilfe von Texture-Mapping werden optische Effekte wie Reflexion und Brechung approximiert. Peachey beschreibt eine vollständige Küstenumgebung, die ebenfalls auf der Verwendung überlagerter trigonometrischer Funktionen basiert [Pea86]. Seine Methode modelliert zusätzlich die Wellenausbreitungsgeschwindigkeit in Abhängigkeit der Wassertiefe und eine Wellenbrechung unter Verwendung eines Partikelsystems [Ree83]. Fournier and Reeves präsentierten im gleichen Jahr eine vergleichbare Umgebung [FR86]. Diese verwendet jedoch Gerstner-Wellen zur Repräsentation der Wellenzüge. So kann die typische Zykloidenform von Wasserwellen beschrieben werden. Ein Partikelsystem repräsentiert dabei brechende Wellen. Goss benutzt Partikel, um die Ausbreitung einer Bugwelle von Schiffen zu modellieren [Gos90]. Miller und Pearce adaptieren ein Partikelsystem mit wirkenden Abstoßungskräften zwischen einzelnen Partikeln [MP89]. So gelingt es ihnen, viskose Flüssigkeiten zu repräsentieren. Perlin beschreibt im Rahmen seines *Image-Synthesizer* die Verwendung von Rauschen auf Pixelebene zur Erzeugung von Ozeanwellen ([Per85],[EMP+04]).

Die historische Entwicklung der Flüssigkeitssimulation in der Computergraphik basiert somit auf der Verwendung von Animationstechniken. Mit der gewachsenen Prozessor-Leistung können solche Verfahren durchaus in Echtzeit ausgeführt werden. Die Ergebnisse erzielen in der Regel jedoch keinen Photorealismus, da die Komplexität realer Flüssigkeiten mit Animationsansätzen schwerlich repräsentiert werden kann. Im folgenden Abschnitt werden aktuelle Ansätze präsentiert, die im Rahmen einer Approximation eine realistische Repräsentation bestimmter Effekte erlauben.

3.2. Approximationsverfahren

Reeves führte die Verwendung von Partikelsystemen in die Computergraphik ein [Ree83]. Die Partikel bewegen sich unabhängig voneinander und können somit sehr schnell integriert werden. Bekannt wurde seine Arbeit durch den so genannten Genesis-Effekt. Dieser Effekt animiert die Ausbreitung einer Explosion in dem Film *Star Trek II: The Wrath of Khan* (1982, Paramount Pictures) und stellt die erste Verwendung eines Partikel-Systems mit Bewegungsunschärfe in Filmen dar.

Partikel-Systemen können auch das Verhalten von Fontänen und Wasserfällen approximieren [Sim90]. In einem solchen System wird die hohe Geschwindigkeit der Wasserpartikel ebenfalls durch Bewegungsunschärfe visualisiert. Derartige Methoden können GPU-basiert einen hohen Grad an Realismus in Echtzeitumgebungen

erzielen [Tat06]. Jedoch können lediglich fallende Wassermassen auf diese Art approximiert werden, die sich hauptsächlich unter dem Einfluss der Gravitation bewegen und bei denen das Strömungsverhalten vernachlässigt werden kann.

Zahlreiche weitere Flüssigkeitseffekte, wie zum Beispiel Regen oder Pfützen, können realistisch repräsentiert werden, ohne Verwendung einer komplexen unterliegenden physikalischen Simulation. Die Anwendung von Animationsverfahren und Videosequenzen kann für solche Situationen überzeugende und realistische Szenarien erzeugen. Tatarchuk zum Beispiel beschreibt ein interaktives Stadt-Szenario bei Regen, das unter Verwendung derartiger Verfahren einen hohen Grad an Realismus erzielt [Tat06].

3.3. Simulationsverfahren

Im Gegensatz zu den im vorherigen Abschnitt beschriebenen Approximationsverfahren behandelt dieser Abschnitt die physikalisch-basierte Simulation von Flüssigkeiten. Im folgenden Abschnitt werden Höhenfeld-basierte Verfahren behandelt. Anschließend werden dreidimensionale Methoden vorgestellt: Zunächst Gitterbasierte (Euler-) Verfahren und dann Partikel-basierte (Lagrange-) Verfahren. Abschließend werden existierende hybride Techniken behandelt.

3.3.1. Höhenfeld-basierte Verfahren

Höhenfeld-basierte Verfahren können in Echtzeit in der Regel detailliertere bzw. größere Flüssigkeitsmengen darstellen als dreidimensionale Verfahren. Sie basieren auf zweidimensionalen Simulationen, die einen geringeren Aufwand besitzen als dreidimensionale Simulationen: $O(n^2) \leftrightarrow O(n^3)$. So können feinere Simulations-Auflösungen genutzt werden, die in detaillierteren Flüssigkeiten resultieren. Höhenfeld-basierte Verfahren können jedoch keine dreidimensionalen Effekte repräsentieren, wie zum Beispiel spritzendes Wasser oder brechende Wellen. In Szenarien, in denen derartige Effekte jedoch nicht notwendig sind, können Höhenfeldbasierte Ansätze die Qualität und Geschwindigkeit erheblich verbessern.

Im Folgenden werden zunächst ambiente Oberflächen behandelt. Anschließend erfolgen Ausführungen zu Oberflächenwellen und zweidimensionalen Strömungssimulationen.

Ambiente Oberflächenwellen *Ambiente Wellen* bezeichnen die grundlegende Oberflächenbewegung, die durch Wind und Strömungen hervorgerufen wird. Somit können sie zum Beispiel keine Wellenausbreitung bewegter Objekte repräsentieren. Daraus resultiert jedoch auch der große Vorteil der ambienten Oberflächenwellen: Sie können vorberechnet werden (nicht aber die blickabhängige Oberflächenextraktion und die Darstellung) und der zugehörige Aufwand der Simulation kann zur Laufzeit vernachlässigt werden. Somit sind ambiente Wellen gut geeignet für interaktive Umgebungen.

Mastin et al. verwenden einen Ansatz, um Höhenfeld-basierte Ozeanwellen mit Hilfe der Fourier-Transformation zu beschreiben [MWM87]. Anhand eines empirischen Wellenspektrums in Abhängigkeit des Windes (modifiziertes Pierson-Moskowitz-Spektrum [PM64]) wird eine Faltung im Fourier-Raum durchgeführt. Im Ortsraum resultiert diese in einem Wellenbild. Anhand verschiedener Phasenmanipulationen können die Wellen animiert werden. Vergleichbare Ansätze verwenden andere Wellenspektren (JONSWAP-Spektrum [PA00], Phillips-Spektrum [Tes01]).

Die letztgenannte Arbeit von Tessendorf führt den Begriff *ambiente Wellen* ein. Der Ansatz behandelt zusätzlich die Repräsentation rauhen Wassers (*"choppy waves"*) und Darstellung optischer Effekte wie Kaustiken und *God-Rays*. Die Technik wird in [Tes04] erweitert, so dass Objekte Wellen erzeugen können. Eine Darstellung einer echtzeitfähigen Ozean-Umgebung, basierend auf dem Algorithmus von Tessendorf, wird in [Bel03], [Mit04] und [Fre06] präsentiert. Unter anderem werden LOD-Techniken verwendet, um Aliasing-Artefakte zu vermeiden. Eine virtuelle Ozean-Umgebung zur Untersuchung verschiedener Wellen-Spektren wird in [Lac07] beschrieben. Szecsi und Arman verwenden einen schnellen prozeduralen Ansatz zur Repräsentation von Oberflächenwellen [SA08]. Dabei modellieren sie die Wellenabhängigkeit in Abhängigkeit der Wassertiefe und Küstenform. Derartige prozedurale Ansätze sind jedoch den Spektrum-basierten Ansätzen in Bezug auf Realismus ambienter Wellen unterlegen, da sie im Prinzip einfache Animationsverfahren darstellen und der Komplexität realer Flüssigkeiten nicht genüge tragen.

Oberflächenwellen Die Wellengleichung ist eine häufig genutzte Methode für die Repräsentation von Oberflächenwellen. Sie simuliert die Propagation radialer Wellen. Durch Überlagerung zahlreicher derartiger Wellen in zwei Dimensionen kann eine realistische Wellenausbreitung realisiert werden. Allerdings können keine Dispersionseigenschaften mit der Wellengleichung repräsentiert werden — alle Wellen besitzen die gleiche Ausbreitungsgeschwindigkeit. In der Regel wird sie mit Hilfe einer FD-Methode gelöst (zum Beispiel [Gom00]). Die Verwendung der Wellengleichung in Verbindung mit gekachelten, animierten Oberflächenwellen wird in [JH03] vorgestellt.

Yuksel et al. präsentieren eine approximierende Lösungsmethode für die Wellengleichung, die auf Partikeln basiert [YHK07]: *Wave Particles*. Die radial propagierenden Wellen werden mit Hilfe von Partikeln dargestellt, die sich im Zweidimensionalen bewegen. Bei Überschreitung von Schwellwerten der Nachbarschaftsdistanz werden neue Partikel eingefügt, um eine Unterabtastung bei Verwendung einer statischen Partikelanzahl zu vermeiden. So kann eine minimale Diskretisierung gewährleistet werden (siehe Abbildung 3.1). Die Partikel bewegen sich unabhängig voneinander und können somit schnell integriert werden. Die Oberfläche wird als Höhenfeld von Kerneln generiert, die an den jeweiligen Positionen der Partikel positioniert sind. Yuksel et al. präsentieren des Weiteren eine Kopplung mit einer starren Körper Simulation. Aufgrund der Ähnlichkeit der beschriebenen Umgebung zu der im Rahmen dieser Arbeit entwickelten Methode, die in Abschnitt 5.1.1 beschrieben wird,

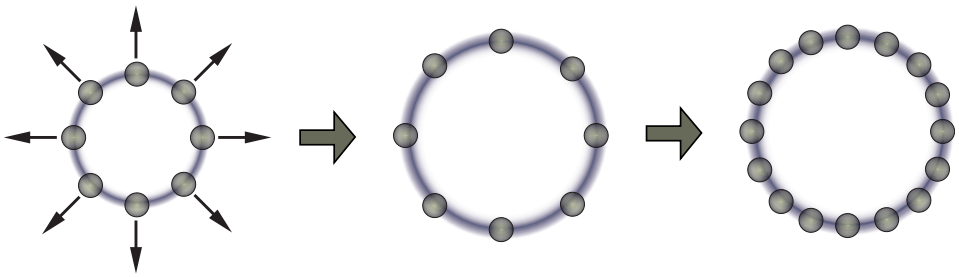


Abb. 3.1.: Prinzip der *Wave Particles*-Methode [YHK07]. Radial propagierende Wellen werden mit Hilfe von Partikeln repräsentiert. Wird ein Schwellwert der Distanz benachbarter Partikel überschritten, so werden neue Partikel eingefügt.

erfolgt in Abschnitt 5.1.1.4 ein direkter Vergleich beider Methoden.

Ein anderer Ansatz ist die Verwendung eines FFT-basierten Ansatzes (*Fast Fourier Transformation*) zur Repräsentation von Oberflächenwellen, die auf dem Huygensschen Prinzip basieren [Lov03]. Das System inkludiert die Approximation von Kaustiken. Aufgrund der Verwendung der aufwendigen FFT ist die Methode jedoch wesentlich langsamer, als die übrigen genannten.

Strömungssimulation 2D Für gewisse Szenarien sind dreidimensionale Flüssigkeitseffekte nicht unbedingt vonnöten. Aus diesem Grund — und zur Laufzeit-Verbesserung — wurden weitere Verfahren vorgestellt, die eine zweidimensionale Strömungssimulation adaptieren. Die Darstellung erfolgt mit Hilfe eines Höhenfeldes. Dieses kann aus Dichtewerten extrahiert werden [CdVL95, CLHM97, KEWE03] — zum Beispiel unter Verwendung der Bernoulli-Gleichung. Derartige Methoden können auch als Säulen-basiert betrachtet werden [HW04, MFC06, MFC07]. Die Vorteile solcher Simulationen liegen darin, dass sie eine Strömung und zugleich Oberflächenwellen repräsentieren können, wenngleich diese oftmals grob erscheinen. O'Brien und Hodgins beschreiben eine ähnliche Simulation, die aus hydrostatischer Druck- und Oberflächensimulation [OH95] besteht. Sie koppeln das System zusätzlich mit einem Partikelsystem, um spritzendes Wasser zu repräsentieren.

Die *Flachwassergleichung* ergibt sich aus den dreidimensionalen Navier-Stokes-Gleichungen indem die vertikale Strömungsgeschwindigkeit als vernachlässigbar angesehen wird (siehe auch Abschnitt 2.1). Kass und Miller verwenden eine vereinfachte Flachwassergleichung [KM90]. Ihre Vereinfachung kann jedoch keine keine Wirbel darstellen. Layton und Panne beschreiben eine implizite semi-Lagrange Integration zur Lösung der Flachwassergleichung, die in hoher Stabilität resultiert [LvdP02]. Kallin verwendet eine Quadtree-Datenstruktur, um eine adaptive Simulation der Flachwassergleichung zu erzielen [Kal08]. Jensen und Goliáš beschreiben eine interaktive Ozeanumgebung, die neben ambienten Wellen auch die Flachwassergleichung verwendet [JG01]. Sie beschreiben zudem Reflexions- und Brechungseffekte, die Darstellung von *God-Rays* mit Hilfe von *Image Slices* und die Abbildung von Kaustiken

und spritzendem Wasser. So gelingt es ihnen, eine überzeugende Ozeanumgebung zu modellieren.

3.3.2. Dreidimensionale Euler Verfahren

Die Simulation der Navier-Stokes Gleichungen kann mit expliziten und impliziten Verfahren erfolgen. Die expliziten Methoden besitzen den Vorteil einer schnellen Integration, das System wird jedoch bei großen Zeitschritten instabil. Implizite Methoden sind aufwendiger und dämpfen das System zusätzlich. Jedoch können wesentlich größere Simulationszeitschritte verwendet werden. Beide Integrationsmethoden werden in der Computergraphik verwendet. Deshalb werden die Methoden im Folgenden separat behandelt.

Explizite Methoden Harlow und Welch präsentierten den ersten Ansatz einer Strömungssimulation unter Berücksichtigung freier Oberflächen [HW65]. Die vorgestellte zweidimensionale *Marker-and-Cell*-Methode (MAC, auch: *Particle-in-Cell*, PIC) simuliert die Navier-Stokes Gleichungen mit einer expliziten FD-Methode. Die freie Oberfläche wird mit Hilfe von Marker-Partikeln detektiert, die sich in der Strömung bewegen, jedoch keine physikalischen Informationen tragen. Anhand der Marker-Partikel wird entschieden, welche Zellen Flüssigkeit beinhalten und wie die Randbedingungen zu setzen sind. Adaptive Gitterunterteilung und die Verwendung von Marker-Partikeln lediglich in der Umgebung der Oberfläche stellen zum Beispiel später vorgestellte Verbesserungen dar [CJRF97]. Die erste Umsetzung der MAC-Methode in drei Dimensionen wird in [HC72] gezeigt. Hirt und Nichols beschreiben zudem eine Methode, um die freie Oberfläche zu behandeln [HN81], die *Volume of Fluid*-Methode. Diese erhält das Volumen explizit, wenngleich sie in der Darstellung keine glatten Oberflächen erzielen kann. Eine andere Alternative ist die Diskretisierung der freien Oberfläche mit einem irregulären Gitter, welches mit der Strömung advektiert [UT92]. Entsprechend der Größenveränderung von Dreiecken werden neue Dreiecke eingeführt oder Existierende vernichtet. Mit Hilfe einer Unterteilungsmethode für langsam fließende Flüssigkeiten kann aus einem groben Vektorfeld ein feines erzeugt werden, um die Performanz zu verbessern [WW99]. Foster und Metaxas demonstrierten als Erste die Verwendung dreidimensionaler, regulärer Gitter zur Repräsentation dreidimensionaler Flüssigkeiten in der Computergraphik [FM96]. Die Arbeit stellt eine Erweiterung der von Harlow und Welch vorgestellten Methode dar. Gleichzeitig verwenden sie eine unilaterale Kopplung zwischen Flüssigkeit und schwimmenden Objekten: Objekte bewegen sich mit der Strömung, induzieren jedoch keine Wellen. Vorteile der expliziten Ansätze liegen in der guten Ausführungsgeschwindigkeit und der intuitiven Implementierung. Explizite *Solver* und zugehörige Quellcodes werden zum Beispiel in [GDN95, GDN98] gegeben.

Implizite Methoden Stam ersetzte die expliziten Ansätze mit einer implizite Semi-Lagrange Methode zur Simulation von Fluiden in der Computergraphik

[Sta99, Sta03]: *Stable Fluids*. Diese Methode bestimmt die Konvektion in den Gitterzellen für die Mittelpunkte der Zellen. Implizit wird dabei die Position bestimmt, die zum nächsten Zeitpunkt auf die Mittelpunkte der Gitterzellen bewegt wird. Da diese Punkte als mit der Strömung bewegte Partikel betrachtet werden können, wird eine derartige Methode als *Semi-Lagrange*-Methode bezeichnet. Die Methode behebt das Hauptproblem der hohen Instabilität bei großen Zeitschritten expliziter Methoden. Die Methode ist aufgrund des impliziten Ansatzes numerisch gesehen ausgesprochen stabil. Das *Stable Fluids*-Verfahren hat die Flüssigkeitssimulation in der Computergraphik maßgeblich beeinflusst und zahlreiche spätere Arbeiten verwenden einen derartigen Solver. Die Methode separiert die Lösung der Navier-Stokes Gleichung in vier sukzessive Schritte: Externe Kraft, Konvektion, Diffusion und Projektion. Die ersten drei Schritte ergeben sich aus den Navier-Stokes Gleichungen. Der letzte Schritt dient dazu, ein divergenzfreies Geschwindigkeitsfeld zu erzeugen, um Massenerhaltung zu gewährleisten. Eine freie Oberfläche wird in [Sta99] zunächst nicht betrachtet. Stam erweiterte die Methode in [Sta01], indem die beiden letzten Schritte der *Stable Fluids* Methode (Diffusion, Projektion) im Fourier-Raum ausgeführt werden. Entsprechend dem Helmholtz-Theorem kann ein Vektorfeld als Summe einer divergenzfreien Funktionen und eines Gradientenfeldes geschrieben werden. Im Fourier-Raum können die beiden Summanden direkt durch eine senkrechte Projektion bestimmt werden. Wird das Gradientenfeld vernachlässigt, wird mit Hilfe der inversen Fourier-Transformation ein divergenzfreies Strömungsfeld erzeugt. Die Dissipation innerhalb der von Stam beschriebenen Methode kann mit der Methode des *Constraint Interpolation Profile* (CIP) [YA91] verringert werden. Die Idee ist es, nicht nur die Funktionswerte der Gitterelemente zur Interpolation zu verwenden, sondern auch ihre Ableitungen.

Die Anwendung für die Flüssigkeitssimlation in der Computergraphik wird zum Beispiel in [SSK05] demonstriert. Kombiniert mit einer Octree-Datenstruktur können die Navier-Stokes Gleichungen adaptiv und effizient gelöst werden, wobei die Details erhalten bleiben [LGF04]. Bei Verwendung adaptiver, bewegter dreidimensionaler Gitter innerhalb einer Fluid-Simulation, kann im Rahmen der Galilei-Invarianz die Simulation der Fluid-Strömungen angepasst werden (zum Beispiel [SCP$^+$04]).

In interaktiven Umgebungen werden Euler Verfahren für dreidimensionale Flüssigkeiten kaum verwendet, da der Aufwand inklusive Behandlung freier Oberflächen und Oberflächenextraktion hoch ist. In jüngerer Zeit wurden auch GPU-basierte Implementierungen von FDM-Simulationen untersucht [Har04][CIS07]. Der hohe Grad an Parallelität heutiger GPUs motiviert dieses Vorgehen. Gerade solche Gitter-basierten Verfahren lassen sich intuitiv umsetzen, da sie lediglich von ihrer lokalen und gleichzeitig bekannten Umgebung abhängen. Derartige Implementierungen erzielen in der Regel die mehrfache Performanz einer vergleichbaren CPU-basierten Implementierung und ermöglichen interaktive Geschwindigkeiten für geringe Wassermengen (zum Beispiel Glas, Aquarium).

Kopplung mit starren Körpern Die Interaktion von Flüssigkeiten und starren Körpern kann verschiedene Kopplungen umfassen. So kann zwischen drei Kopplungsarten unterschieden werden [CMT04, OZH00]:

- Unilaterale Kopplung A: Flüssigkeit interagiert mit starrem Körper,
- Unilaterale Kopplung B: Starrer Körper interagiert mit Flüssigkeit,
- Bilaterale Kopplung: Flüssigkeit und starrer Körper interagieren.

Foster und Metaxas verwendeten den Ansatz von Stam zur Repräsentation von Flüssigkeiten in drei Dimensionen und erweitern ihn um die Behandlung bewegter Objekten (Unilaterale Kopplung A) [FF01]. Ihre Methode tastet die Oberfläche mit einem hybriden Ansatz ab, der aus der *Level-Set*-Methode und Partikeln besteht, die sich in der Flüssigkeit bewegen. Durch dieses Vorgehen können sie die Massen-Dissipation verringern und gleichzeitig glatte Oberflächen und detaillierte Flüssigkeitseffekte repräsentieren. Zudem erlauben sie der Flüssigkeit sich im Rahmen von Zwangsbedingungen tangential zu dem starren Körper zu bewegen. In [FM97] zeigten sie zuvor auch das später häufig verwendete Beispiel einer derartigen unilateralen Kopplung: Der Fall einer Kugel in ein Behältnis mit Flüssigkeit (siehe auch [EMF02, Wre03]). In einem derartigen Szenario, reagiert die Flüssigkeit auf die Kugel, jedoch nicht umgekehrt.

Takahasi et al. demonstrierten die Verwendung der *Volume of Fluid Methode*, um eine bilaterale Kopplung zu realisieren [TUKF02]. Dazu werden Elemente eines regulären Gitters genau dann als Randbedingung betrachtet, wenn sie mehr als zur Hälfte von einem starren Körper belegt sind. Eine bilaterale Kopplung zwischen Flüssigkeiten und elastischen Körpern kann auch mit Feder-Masse-Systemen modelliert werden [GHD03]. In dem Fall wird eine Euler-Flüssigkeitssimulation mit einer Lagrange-starren-Körper-Simulation gekoppelt. Eine bilaterale Kopplung mit beliebig geformten starren Körpern erfolgt in [CMT04]. Dabei werden die starren Körper innerhalb der Simulation behandelt als würden sie aus Flüssigkeit bestehen.

Irreguläre Gitter Die numerische Diffusion der MAC-Methode ist hoch aufgrund der permanenten Interpolation der Variablen. Brackbill und Ruppel führen die Fluid-Implicit-Particle Methode (FLIP) ein [BR86]. Diese verringert die numerische Diffusion erheblich, indem ein irreguläres Gitter in Abhängigkeit der Partikelbewegung adaptiert wird. Komplexe Oberflächen können zudem wesentlich besser in der Simulation berücksichtigt werden, wenn irreguläre Meshes zur Repräsentation verwendet werden. [FOK05] demonstriert dieses Vorgehen für Gase. Mit Hilfe adaptiver, unstrukturierter Tetrahedal-Meshes können auch komplexe Strömungen von Flüssigkeiten detailliert abgetastet werden und die Randbedingungen dynamischer starrer Körper können genauer gesetzt werden [KFCO06]. Diese Verfahren benötigen jedoch mehr Berechnungsaufwand und sind für interaktive Umgebungen ungeeignet.

Lattice-Boltzmann Verfahren Die Lattice-Boltzmann Methode stellt eine weitere Möglichkeit zur Lösung der Navier-Stokes Gleichungen dar. Die Navier-Stokes Gleichungen werden dabei nicht direkt gelöst, sondern die diskrete Boltzmann Gleichung der kinetischen Gastheorie wird gelöst, die die statistische Verteilung von Partikeln in Fluiden beschreibt. So kann der viskose Fluss bestimmt werden. In der Computergraphik ist die Methode noch jung; verwendet wird sie beispielsweise in [WZF+03] und [TR08]. In [TKR05] wird eine interaktive Umsetzung für Flüssigkeiten beschrieben, die jedoch verhältnismäßig undetailliert ist. Für einen Überblick der Methode sei auf [Suc01] verwiesen.

3.3.3. Dreidimensionale Lagrange Verfahren

Die in der Literatur am häufigsten erwähnte Lagrange Methode im Feld der Computergraphik ist die *Smoothed Particle Hydrodynamics*-Methode (SPH) — siehe auch Anhang A. Gingold und Monaghan führten die SPH-Methode in der Physik ein und benutzten sie, um astrophysikalische Phänomene zu beschreiben [GM77]. Ein allgemeiner Überblick ist in [Mon92] zu finden. Die Applikation für zweidimensionale Flüssigkeiten beschreibt Monaghan in [Mon94].

Desbrun und Gascuel beschreiben die Verwendung der SPH-Methode in der Computergraphik, um elastische und zähe Körper zweidimensional zu repräsentieren [DG96]. Stein und Max verwendeten die Methode erstmalig zur Repräsentation von Flüssigkeiten in der Computergraphik [SM98]. Der Ansatz wurde von Müller et al. in der wegweisenden Arbeit [MCG03] zur interaktiven Simulation von Flüssigkeiten erweitert. Oberflächenspannung modellieren sie mit der von Morris vorgestellten Methode [Mor00]. Die Darstellung erfolgt mit der *Surface-Splatting*-Methode [ZPvBG01] oder dem *Marching-Cubes*-Algorithmus [LC87]. Sie simulieren einige tausend Partikel und stellen zudem eine Verbesserung der Oberflächenextraktion vor: Der Beitrag jedes Partikels zur Isofläche hängt von der jeweiligen Dichte ab. So erscheint die Oberfläche glatter, wenngleich noch keine vollständig glatte Oberfläche erzeugt werden kann. Unter Verwendung von Partikeln verschiedener physikalischer Eigenschaften gelingt es Müller et al. auch, verschiedene Flüssigkeiten zu mischen [MSKG05]. Die Methode gestattet es unter anderem, Luftblasen innerhalb der Flüssigkeit zu repräsentieren. In einer anderen Arbeit demonstrieren Müller et al. die Möglichkeiten einer Kopplung von Fluid und zerbrechlichen Körpern [MST+04]. Diese werden mit einer FE-Methode repräsentiert. Sie erzielen interaktive Performanz für einige tausend Partikel. Adams et al. verwenden adaptive Partikel innerhalb einer SPH-Simulation [APKG07]. Es gelingt ihnen durch Verwendung verschiedener Partikel, die Anzahl der Partikel erheblich zu reduzieren und somit einen signifikanten Geschwindigkeitsvorteil zu erzielen, wenngleich ihr Ansatz nicht echtzeitfähig ist. SPH-basierte Ansätze zur Simulation unter Berücksichtigung starrer Körper, die mit der Flüssigkeit interagieren, werden in [Ama06] und [BTT09] beschrieben.

Die Portierung einer SPH-Simulation auf die GPU ist aufwendiger als bei FD-Ansätzen. Die Nachbarschaftssuche dynamischer Partikel unter Verwendung der

GPU zu realisieren, stellt dabei die Hauptschwierigkeit dar, da die notwendigen Datenstrukturen nicht direkt auf der GPU umgesetzt werden können. Zahlreiche mögliche Umsetzungen der Methode auf der GPU wurden vorgestellt: [AIY+04, KSW04, KC05, HKK07a, ZSP07, HTKK07]. Zehntausende Partikel können so unter Verwendung der GPU bei interaktiven Frameraten simuliert werden — die Geschwindigkeit realer Flüssigkeiten wird jedoch noch nicht erreicht. In der Regel werden dabei die Partikel als Punkte abgebildet. Eine Oberflächenextraktion und die Approximation optischer Effekte wird dabei nicht durchgeführt. Dies könnte aber effizient mit den in Abschnitt 6.2 und Kapitel 7 beschriebenen Verfahren erfolgen.

Ein grundsätzlicher Nachteil der Standard-SPH-Methode ist die Existenz von *Kompressibilität*. Die Methode kann keine inkompressiblen Fluide repräsentieren, so dass gerade bei großen Zeitschritten und hohen Geschwindigkeiten eine unrealistische Vibration entstehen kann.

Becker und Teschner verwenden die Tait-Gleichung [Mon94], um den Druck aus der Dichte zu bestimmen und können so die Kompressibilität signifikant verringern (*weakly compressible* SPH, [BT07]). Müller et al. verwenden in [MCG03] dagegen die ideale Gasgleichung, die im Gegensatz dazu in einer hohen Kompressibilität resultiert. Das Problem wird auch mit der *Moving-Particle Semi-Implicit-Methode* (MPS) behoben, die von Premoze et al. in die Computergraphik eingeführt wurde [PTB+03]. Die Methode ist der SPH-Methode ähnlich; sie führt jedoch einen Korrekturterm ein, der die Inkompressibilität sicherstellt. Losasso et al. beschreiben eine Kombination aus *Level Set* und SPH-Methode. Mit dieser Kombination gelingt es ihnen ebenfalls, die Kompressibilität gering zu halten. Diese Methoden zur Reduktion der Kompressibilität sind in der Regel ausgesprochen kostenintensiv. Die von Solenthaler und Parajola in [SP09] vorgestellte Methode iteriert nach jedem Simulationsschritt über die Dichte und aktualisiert entsprechend die Druckkräfte, um Inkompressibilität zu erhalten. Diese Methode ist wesentlich schneller als die vorhergehend genannten, dennoch beträgt die Berechnungszeit für einen Zeitschritt viele Minuten.

Der Vollständigkeit halber sei auch die Moving Least Squares Methode (MLS, auch Moving Least Squares Hydrodynamics, MLSPH, zum Beispiel [Dil99]) erwähnt. Diese ist der SPH-Methode ebenfalls ähnlich, doch werden nicht rein radiale Potentiale verwendet, sondern Potentiale, die entsprechend der Dichte der umliegenden Partikel adaptiert sind. So wird die Kompressibilität ebenfalls verringert — der numerische Aufwand ist jedoch ungleich höher — so dass die Methode für die Flüssigkeitssimulation in der Computergraphik bisher nicht verwendet wird.

3.3.4. Hybride Verfahren

Hybride Verfahren zur Flüssigkeitssimulation werden verwendet, um die Simulationsgeschwindigkeit oder die Qualität der repräsentierten Flüssigkeit zu verbessern. So kann eine Kopplung der MAC-Methode mit der SPH-Methode den Detailgrad

oder die Laufzeit simulierter Flüssigkeiten signifikant verbessern. Eine SPH-Methode kann zum Beispiel im Oberflächenbereich verwendet werden, um einen hohen Detailgrad an der Oberfläche zu erzielen. Die MAC-Methode kann dann im übrigen Volumen benutzt werden [SS06]. Eine andere Art der Kopplung besteht in der Erzeugung von Partikeln in turbulenten Situationen und Bereichen (zum Beispiel Gischt) [LTKF08].

Selle et al. benutzen einen hybriden Ansatz, um Wirbel besser repräsentieren und erhalten zu können [SRF05] — ansonsten werden Wirbel schnell und unrealistisch gedämpft. Sie verwenden Partikel, die die Wirbelinformationen advektieren. Diese werden rückgekoppelt auf ein Euler-basiertes Gitter. Irving et al. koppeln ein zweidimensionales, niedrig aufgelöstes Gitter mit einem dreidimensionalen Gitter in der Umgebung der Oberfläche [IGLF06]. Sie erzielen Geschwindigkeitsvorteile in entsprechenden Szenarien, wie zum Beispiel im Flachwasserbereich. Dennoch liegt die Berechnungszeit im Bereich von Minuten. Eine ähnliche adaptive Kopplung einer dreidimensionalen Lattice-Boltzmann Simulation und einer Flachwasser Simulation wird in [TRS06] beschrieben. Die Resultate aller genannten Verfahren erlauben jedoch keine echtzeitfähige Simulation.

3.3.5. Effekte

Die Flüssigkeitssimulation mit Hilfe der Navier-Stokes-Gleichungen führt zu einer realistischen Dynamik. Dennoch können gewisse Flüssigkeitseffekte nur eingeschränkt oder mit großem Aufwand realisiert werden. So zum Beispiel brechende Wellen, spritzendes Wasser oder Luftblasen. Die Repräsentation derartiger Effekte stellt einen weiteren Schwerpunkt der Forschung dar, da sie den Realismus virtueller Flüssigkeiten vergrößern.

Brechende Wellen Die Komplexität und das faszinierende physikalische Verhalten brechender Wellen zu repräsentieren, ist in der Computergraphik eine große Herausforderung, da es sich um ausgesprochen detaillierte Phänomene handelt. Die ersten Ansätze zur Animation brechender Wellen wurden in [Ree83] und [Pea86] vorgestellt. Dort werden Partikel verwendet, um die Brechung nachzubilden.

Zweidimensionale Simulationen brechender Wellen werden zum Beispiel in [Mon94] und [RO98] demonstriert. Eine zweidimensionale Datenbank brechender Wellen, mit unterschiedlichen, aber ähnlichen Startbedingungen kann zur Steuerung des Verhaltens dreidimensionaler brechender Wellen verwendet werden [MMS04]. Die Einträge werden Schicht-basiert dargestellt. Entsprechend der Auswahl des Animateurs kann das Erscheinungsbild der Welle gesteuert werden.

Enright et al. verwenden die *Particle-Level-Set*-Methode unter anderem zur Repräsentation einer brechenden Welle [EMF02]. Becker und Teschner benutzen hingegen eine SPH-Simulation, die die Kompressibilität erheblich verringert [BT07]. Losasso et al. verwenden eine Kopplung Gitter-basierter und Partikel-basierter Simulation, um ausgesprochen realistische Wellen zu erzeugen [LTKF08]. Diese Ansätze

erzielen überzeugende Ergebnisse, sind jedoch nicht Echtzeitfähig.

Wang et al. verwenden die *Moving Particle Semi-Implicit Methode* (MPS) [PTB+03], um brechende Wellen zu repräsentieren [WZC+06]. Sie verwenden ebenfalls eine zweidimensionale Simulation, aus der sie einen dreidimensionalen Datensatz erzeugen. Eine brechende Welle wird dabei von einer bewegten Ebene erzeugt. Die Simulationszeiten liegen im Bereich einer drittel Sekunde. Damit stellt die Methode die bisher schnellste vorgestellte Methode zur Repräsentation einer brechenden Welle unter Verwendung einer physikalisch-basierten Simulation dar.

Brechende Wellen bei interaktiver Geschwindigkeit wurden bisher lediglich unter Verwendung von Animationstechniken vorgestellt. So kann eine prozedurale Animationstechnik zur Repräsentation brechender Wellen verwendet werden, um die Wellen zu bewegen und zu brechen — bei interaktiven Geschwindigkeiten [JBS03]. Thürey et al. zeigen animierte brechende Wellen, die mit einer Flachwasser-Simulation gekoppelt sind und erzielen ebenfalls interaktive Geschwindigkeiten [TMSG07]. Die brechenden Wellen erscheinen jedoch bei beiden Verfahren aufgrund der verwendeten Animationstechniken relativ synthetisch.

Spritzende Flüssigkeiten Die Verwendung eines Partikelsystems zur Repräsentation von spritzendem Wasser in einer Höhenfeld-basierten Umgebung beschreiben O'Brien und Hodgins [OH95]. Partikel werden erzeugt, wenn fallende Objekte die Oberfläche durchschneiden. Eine Unterteilung der Partikel, die die Oberfläche verlassen, führt zu mehr Details bei spritzendem Wasser [IODN06]. Ein viskoelastisches Material kann mit Hilfe von Federn mit dynamischen Ruhelängen realisiert werden [CBP05]. Dieses Material kann auch Spritzeffekte repräsentieren. Ein anderer Ansatz ist die Anpassung der Partikelanzahl der Oberflächenpartikel innerhalb der *Particle-Level-Set* Methode [KCC+06]. Die Darstellung kann mit einem Ray Tracing-Verfahren erfolgen, wobei die Partikel dreidimensional geblendet werden [Bli82]. In interaktiven Umgebungen bietet sich die Überblendung von *Point-Sprites* an, da diese effizient unter Verwendung moderner GPUs dargestellt werden können.

Gas-Flüssigkeit- und Flüssigkeit-Flüssigkeit-Kopplungen Auch die Repräsentation von Luftblasen wurde eingehend untersucht. Die Schwierigkeit besteht darin, zwei Fluide mit sehr unterschiedlichen Dichten zu koppeln [KVG02, GH04, ZYP06, KLL+07]. Die Methoden lassen sich in der Regel nicht in Echtzeit realisieren. Hong und Kim verwenden die Ghost-Fluid-Methode (GFM) [FAMO99] zur Repräsentation von Gasen in Flüssigkeiten [HK05]. Sie demonstrieren ihr System an Luftblasen in Wasser. Die GFM bezeichnet die Verwendung von Ghost-Zellen innerhalb der Simulation. Diese werden verwendet, um die physikalische Eigenschaften aller Fluide in jeder Gitterzelle zu bestimmen. Weitere Untersuchungen beziehen sich auf die Darstellung kochender Flüssigkeiten [MUM+06, KC07]. Thürey et al. präsentieren eine Echtzeit-Methode, um aufsteigende Blasen in einem Bassin zu simulieren [TSS+07]. Dazu koppeln sie eine

Flachwassersimulation mit einer SPH-Simulation, die die Blasen repräsentieren.

Losasso et al. erweitern die *Level-Set*-Methode zur Repräsentation mehrerer Flüssigkeiten, die verschiedene Dichten und Viskositäten besitzen [LSSF06], erreichen jedoch keine interaktive Performanz.

Flüsse und Erosion Kipfer and Westermann beschreiben eine GPU-basierte Implementierung der SPH-Methode zur Repräsentation interaktiver Flüsse [KW06]. Zur Darstellung verwenden sie ein Höhenfeld. Nachteil der Methode ist die geringe Anzahl verwendeter Partikel, die die Oberfläche undetailliert erscheinen lassen. Andere Ansätze synthetisieren die Navier-Stokes Gleichungen mit Approximationen, um einen interaktiven Fluss darzustellen [TG02, YNBH09].

Ein weiteres Anwendungsgebiet der Flüssigkeitssimulation in der Computergraphik liegt in der Darstellung von Erosionen, zum Beispiel [ASA07, Ben07, KBKv09]. Ziel ist es, aus synthetischen Landschaften und Objekten mit automatischem Abtrag natürlich erscheinende zu erzeugen.

Flüssigkeiten und Gewebe Guendelman et al. koppeln Wasser und dünne deformierbare Objekte, wie zum Beispiel Gewebe [GSLF05]. Dazu verwenden sie eine Gitter-basierte Simulation. Das Pendant einer SPH-Simulation wird in [KSK04] vorgestellt. Beide Methoden geben den Dreiecken des Gewebes eine endliche Volumenausdehnung (*Thickened Triangle Volume/Wedges*). Eine interaktive Umsetzung in Interaktion mit Geweben wird in [HKK07b] präsentiert. Die Simulation wird bei interaktiven Bildraten durchgeführt, wobei einige tausend Partikel verwendet werden.

Weitere Effekte Lenaerts et al. verwenden eine SPH-Simulation, um das Fließen in porösen Materialien zu repräsentieren [LAD08]. Dazu variieren sie innerhalb von Materialien die Flusseigenschaften. Sie zeigen auch das Mischen von Flüssigkeiten mit granularen Materialien [LD09].

Zhu und Bridson verwenden eine Partikel-basierte Flüssigkeitssimulation, um das Verhalten von Sand zu repräsentieren [ZB05]. Dazu führen sie in einer Euler-Flüssigkeitssimulation einen Reibungsterm ein, der das Fließen verhindert. Unter Berücksichtigung der Temperatur kann unter anderem das Schmelzen von Objekten mit einer stabile Adaption des MAC-Verfahrens für Flüssigkeiten hoher Viskosität repräsentiert werden [CMVHT02]. Eine Erweiterung der Methode um Dehnbarkeit ermöglicht auch die Darstellung elastischer und plastischer Verformungen [GBO04].

Bargteil et al. präsentieren eine Texturierungsmethode für dreidimensionale, dynamische Flüssigkeiten [BSM+06]. Die Methode stellt eine Kopplung aus Oberflächen-Trackings [BGOS06], einer Methode zum überlappenden Texture-Mapping auf beliebige Objekte [PFH00] und einer Methode zur Textur-Synthese mit wenigen Artefakten [KEBK05] dar. So gelingt es ihnen, Texturen ohne Artefakte auf dynamische dreidimensionale Flüssigkeitsoberflächen abzubilden.

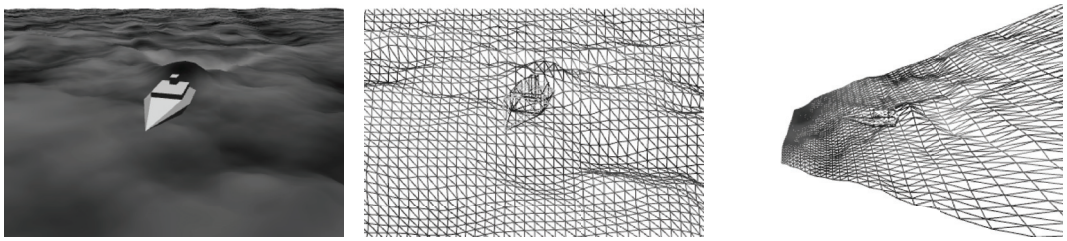
3.4. Oberflächenextraktion

3.4.1. Höhenfelder

Höhenfelder können effizient unter Verwendung der GPU dargestellt werden. Bei der Darstellung von Flüssigkeiten ist jedoch zu beachten, dass die Oberfläche dynamisch ist und sich in der Regel mit jedem dargestellten Zeitschritt ändert. So können die existierenden Darstellungsverfahren, die statische Höhenfelder effizient darstellen, dabei jedoch auf einem Vorberechnungsschritt basieren, in interaktiven Umgebungen nicht verwendet werden (zum Beispiel [LKR+96, DWS+97, Paj98, CGG+03]). Die Verwendung von *Geometry Image Warping* [DS04] zur Darstellung von Wasseroberflächen ist ungeeignet, da der Detailgrad ambienter Wellen in der Regel homogen verteilt ist, so dass eine Adaption in Abhängigkeit von Topologien kaum Verbesserungen erzielen kann. Für ambiente Ozeanflächen bieten sich für die Darstellung aufgrund der hohen Dynamik *Geometry Clipmaps* [LH04, YPZL05] und *Projected Grids* [HNC02, Joh04] an. Erstere besitzen den Nachteil von Übergängen zwischen verschiedenen LoD-Stufen. Im Gegensatz dazu projiziert die *Projected Grid*-Methode ein reguläres Gitter im Bildraum auf das Höhenfeld und erzielt so eine adaptive blickpunktabhängige Darstellung (siehe Abbildung 3.2). Aufgrund der Abtastung im Bildraum können bei der Methode jedoch Aliasing-Artefakte auftreten, die bei niedriger Abtastrate explizit behandelt werden müssen. Da jedoch Wasseroberflächen eine geringe Amplitude besitzen (zumindest im Vergleich zu Landschaften), ist der Nachteil der Aliasing-Artefakten bei Flüssigkeiten eher gering. Für dynamische Höhenfelder können die Höheninformationen in einer Textur gespeichert werden und zum Beispiel unter Verwendung von *Vertex Displacement Mapping* im *Vertex-Shader* verarbeitet werden [Kry05]. Unendlich ausgedehnte Höhenfelder können so effizient im Bildraum dargestellt werden.

3.4.2. Volumen

Gitter-basierte Simulationen In Euler-Flüssigkeitssimulationen wird die Oberfläche oftmals mit einer *Level-Set*-Methode bestimmt. Die Oberfläche wird als *Zero-Level-Set* der Skalarfunktion bestimmt und mit dem Geschwindigkeitsfeld advektiert (zum Beispiel [FF01]). Enright et al. führen eine Verbesserung der *Level-Set*-Methode als *Particle-Level-Set*-Methode ein [EFFM02]. Sie verwenden Marker-Partikel, um die Oberflächenrekonstruktion des *Level-Sets* zu verbessern. Um die Konvektion der Oberfläche besser zu modellieren, benutzen sie Partikel auf beiden Seiten der freien Oberfläche [EMF02] und extrapolieren die Geschwindigkeiten über die freie Oberfläche hinaus in den Luft-Bereich. So gelingt es ihnen, die Oberfläche detaillierter zu approximieren, die Volumenerhaltung besser zu gewährleisten und die Randbedingungen genauer definieren zu können als vorhergehende Verfahren [CJR95, FF01]. Es können auch gewichtete Durchschnittspositionen und Radien nah beieinander liegender Partikel verwendet werden, um die Oberfläche besser zu approximieren [ZB05]. Mit einem semi-Lagrange Ansatz kann eine vorzeichenbehaf-



(a) (b) (c)

Abb. 3.2.: Prinzip der *Projected Grid*-Methode. Szene (a) und zugehöriges Gitter (b) im Bildraum, rotiertes Gitter (c).

tete Abstandsfunktion der Oberfläche advektiert werden [BGOS06], um eine bessere Abtastung zu erzielen. Anhand der Abstandsfunktion wird dann der aktuelle Null-Level zu jedem Zeitschritt bestimmt.

Partikel-basierte Simulationen Ein erstes Verfahren, Punktwolken mit Hilfe eines gaussförmigen Potentials darzustellen, das die implizite Oberfläche definiert, wird in [Bli82] beschrieben. Diese Modelle werden als *Blobby Models* bezeichnet und mit Hilfe von Ray Tracing dargestellt. *Blobby Models* wurden später auch unter dem Namen *Metaballs* bekannt. Wyvill und Trotman verbessern die Präzision und Performanz für das Ray Tracing von *Blobby Models* [WT90].

Ein erster Ansatz zur Verwendung von Punkten als Render-Primitive wurde von Levoy und Whitted vorgestellt [LW85]. Verschiedene weitere Techniken zur Darstellung von Punktwolken ohne gegebene Konnektivität folgten [GD98, RL00, PZvBG00]. Diese Techniken verwenden vorberechnete hierarchische Datenstrukturen, um effizient über dargestellte Punkte zu entscheiden. Die von Zwicker et al. vorgestellte *Surface Splatting*-Technik kann ebenfalls zur Darstellung Partikelbasierter Flüssigkeiten verwendet werden [ZPvBG01]. Diese adaptiert einen *Elliptical Weighted Average Filter* (EWA) im Bildraum und kann zugleich Texturen auf die Partikel-Oberflächen abbilden [GH86]. Aufgrund der Verwendung einer Vorberechnung ist die Methode jedoch lediglich bedingt geeignet für interaktive Flüssigkeiten. Mittels der GPU können derartige Splatting-Verfahren beschleunigt werden [BK03, GBP04, IDYN06]. Eine Variation der Splatting-Methoden zur Darstellung von Iso-Surfaces wird in [CHJ03] beschrieben: *Iso-Splatting*. Partikel werden regulär im Dichtefeld positioniert und entsprechend des Gradienten in Richtung des gewünschten Iso-Wertes adaptiert. Ein anderer Ansatz ist es, Dreiecksnetze aus der Punktmenge zu extrahieren, diese zu filtern und darzustellen [HDD+92].

Eine Bildraum-basierte Methode für interaktive Umgebungen beschreiben Müller et al. in [MSD07]: *Screen Space Meshes*. Im Prinzip wird die *Projected Grid*-Methode, die für Höhenfelder entwickelt wurde, auf die sichtbare Oberfläche dreidimensionaler Objekte angepasst. Das darzustellende Tiefenbild wird aus einer Partikelwolke extrahiert. So wird ein im Bildraum reguläres Gitter erzeugt, welches die sichtbare

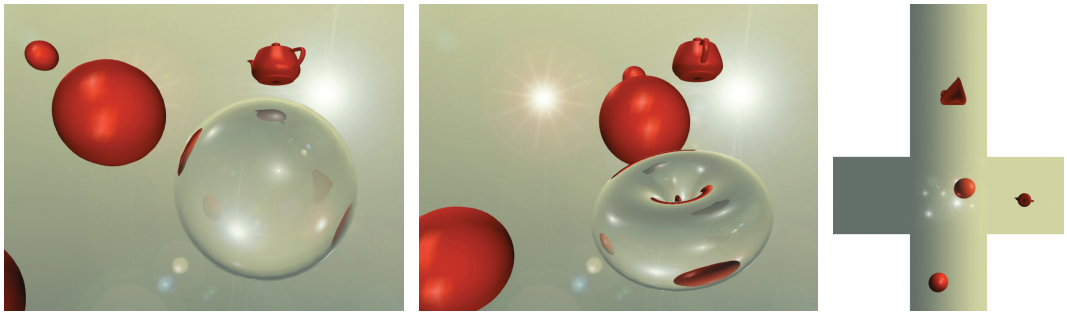
Flüssigkeitsoberfläche abtastet.

Ein anderes interaktives Bildraum-basiertes Verfahren ist die *Particle Splatting*-Methode [ALD06]. Partikel werden als Kugeln in den Frame-Buffer abgebildet und mittels *Alpha Blending*-Funktionen werden die Normalen interpoliert. Die Methode leidet jedoch unter Artefakten an der Silhouette, da individuelle Partikel dort noch sichtbar sind [RB08]. Zudem handelt es sich um ein Zwei-Pass Rendering-Ansatz, der für viele Partikel die Performanz stark sinken lässt. Die beiden letztgenannten Methoden werden in Abschnitt 6.2.2 mit der im Rahmen dieser Arbeit vorgestellten Methode verglichen, da sie vergleichbare Probleme lösen. Rosenberg beschreibt eine echtzeitfähige Methode, mit Hilfe von *Marching Slices* eine Oberfläche zu erzeugen [RB08]. Dazu werden zusammenhängende Flüssigkeitsvolumenbereiche innerhalb einzelner Schichten detektiert, deren Oberfläche anschließend polygonalisiert wird. Hoetzlein und Höllerer führen die Methode der *Sphere Scan Conversion* ein [HH09]. Diese basiert auf einer Geometriedeformation, die vortesselierte Zylinder an einen Partikelstrom (zum Beispiel Fontäne) anpasst. Sie erzielen interaktive Geschwindigkeiten für einige tausend Partikel. Nachteil der Methode ist, dass eine Hauptachse der Bewegung angegeben werden muss.

Gitter-basierte und Partikel-basierte Simulationen Oftmals wird in der Computergraphik der *Marching Cubes*-Algorithmus verwendet [LC87], um die dreidimensionale polygonale Iso-Oberflächen eines Dichtefeldes zu extrahieren (zum Beispiel in [MCG03, ZB05]). Im Falle von Partikelwolken, definieren die Partikelpositionen das Dichtefeld implizit. Als Dichtekernel werden in der Regel radiale Funktionen verwendet, wenngleich auch andere Funktionen verwendet werden können (siehe auch Abschnitt 5.1.2.3). Der Algorithmus diskretisiert das darzustellende Volumen mit Hilfe eines Gitters und entscheidet im Rahmen einer Fallunterscheidung, wie die Oberfläche das jeweilige Gitterelement schneidet. Entsprechend werden Polygone für jedes Gitterelement erzeugt. Variationen der *Marching Cubes*-Methode verbessern die Performanz (zum Beispiel [MSS94]) oder reduzieren Artefakte (zum Beispiel [NH91]). Eine GPU-basierte Umsetzung wird zum Beispiel in [GJD05] präsentiert.

3.5. Darstellung

Die Darstellung photorealistischer Flüssigkeiten in Nicht-Echtzeitumgebungen erfolgt in der Regel mit einem Ray Tracing-Verfahren [Whi80] in Kombination mit Photon Mapping [JC95, Jen01] oder einem *Light Ray Tracing*-Verfahren. So können optische Effekte wie Reflexionen, Brechungen, Absorptionen und Kaustiken akkurat repräsentiert werden. *Backward Beam Tracing*, als Erweiterung zur Beschleunigung von *Light Ray Tracing*, projiziert die Polygone der Oberfläche auf die Szene und beschleunigt so die Kaustikerzeugung [Wat90]. In [DCG07] wird die *Sphere Tracing*-Methode verwendet, um eine ambiente Wasseroberfläche darzustellen — jedoch können keine interaktiven Ergebnisse erzielt werden.



(a) (b) (c)

Abb. 3.3.: Prinzip des *Environment-Mappings*. Eine transparente Kugel und ein transparenter Torus (b) innerhalb einer dynamischen Szene. Die zugehörige, dynamische *Environment-Map* (c) wird aus Sicht des Objektes erzeugt.

In interaktiven Umgebungen ist die Verwendung eines Ray Tracing-Verfahrens auf heutigen Standard-PCs zu aufwendig. Schnelle Ray Tracing-Verfahren basieren auf umfangreichen vorberechneten Datenstrukturen, die die Schnittberechung erheblich beschleunigen. Bei dynamischen Objekten wie Flüssigkeiten müsste diese Datenstruktur jedoch in jedem Frame neu bestimmt werden.

Baboud und Decorét beschreiben einen vereinfachten *Ray Casting*-Ansatz zur Darstellung von Wasseroberflächen unter Verwendung der GPU [BD06a]. Sie verwenden zwei Höhenfelder. Das Erste repräsentiert die Wasseroberfläche und das Zweite den Untergrund. Die rasterisierte Schnittberechnung stellt eine Vereinfachung der in [BD06b] beschriebenen Methode dar, die Vorberechnungen zur Beschleunigung der Schnittbestimmung verwendet. Baboud und Decorét repräsentieren Reflexionen, Brechungen, Kaustiken und Absorption für das beschriebene Szenario, welches aus zwei Höhenfeldern besteht. Polygonale Objekte die im Volumen liegen, können mit der Methode nicht dargestellt werden. Insofern wäre eine Kombination dieser Methode mit der in Abschnitt 7.2 beschriebenen Methode sinnvoll, da dann sowohl der Untergrund als auch polygonale Objekte reflektiert und gebrochen werden könnten.

Environment Mapping ist eine häufig verwendete Methode zur schnellen Darstellung spekularer Reflexionen auf kurvigen Objekten (zum Beispiel [RKL+06]). Spekulare Reflexionen und einfache Brechungen können mit Hilfe dynamischer *Environment Maps* approximiert werden [BN76]. Die *Environment Map* wird als im Unendlichen liegend angenommen und die Zugriffsvektoren werden im Ursprung berechnet. Die Map wird generiert, indem die Szene aus Sicht des reflektierenden bzw. brechenden Objekts dargestellt wird. Die Zugriffsvektoren werden in Abhängigkeit der Blickrichtung und der Objekt-Oberflächennormalen bestimmt — in der Regel GPU-basiert [NC02, RKL+06]. Der Ansatz erzielt realistische Ergebnisse für kleine Objekte mit großen Krümmungen (Abb. 3.3). Bei großen bzw. flachen Objekten (wie zum Beispiel Wasseroberflächen) kommt es jedoch zu Artefakten, denn der Zugriff auf die *Environment Map* erfolgt immer vom Ursprung aus [Gre86]. Diese Tatsa-

che ist für *Environment Maps*, die im Unendlichen liegen, nicht von Bedeutung. Je näher ein Objekt jedoch an der transparenten Ebene liegt, um so wahrnehmbarer werden die Artefakte, da die geringen Variationen der Zugriffsvektoren alle nahezu auf die gleichen Punkte in der *Environment Map* zeigen. Für *spekulare Reflexionen* kann dieses Problem mit einem Algorithmus behoben werden, der die projizierten Reflexions-Vertices iterativ approximiert [RH06][EMDT06].

Planare Reflexionen können im Rahmen eines zweiten *Rendering Passes* realisiert werden. Die Szene wird aus Sicht des an der Ebene gespiegelten Blickpunktes dargestellt (zum Beispiel [BMG+99]). *Planare Brechungen* können theoretisch ähnlich dargestellt werden, sind jedoch in der Praxis komplizierter, da die Szene durch die Brechung verzerrt wird. Es kann jedoch eine Übergangsfunktion bestimmt werden, die die gebrochene Szene in den Bildraum transformiert [DB94].

Reflexions- und Brechungseffekte Höhenfeld-basierter Flüssigkeiten in interaktiven Umgebungen werden oftmals ebenfalls mit Mapping-Techniken effizient approximiert [Bel03, Joh04, Sou05]. Die Szene wird in eine *Environment Map* gerendert, einmal aus Blickposition und einmal aus der gespiegelten Blickposition. Entsprechend der Blickrichtung und der Oberflächennormalenvariation wird der Zugriffsvektor variiert. Dieser approximierende Ansatz erzielt überzeugende und schnelle Ergebnisse, wenngleich er typische optische Eigenschaften der Brechung, wie zum Beispiel Winkelveränderungen oder Größenskalierungen, nicht repräsentieren kann. *Environment Mapping* kann auch zur Darstellung optischer Eigenschaften eines quadratischen Pools verwendet werden [SW01]. Die Methode kann allerdings keine Objekte innerhalb des Pools optisch behandeln. Für rechteckige Pools, die mit Flüssigkeit gefüllt sind, kann auch ein vereinfachtes Ray Tracing-Verfahren mit den fünf beteiligten Flächen auf der GPU durchgeführt werden [VM00].

Alle bisher genannten, auf *Environment Mapping* basierende Ansätze unterstützen lediglich einfache Reflexionen und Brechungen — es wird also lediglich an der ersten sichtbaren Oberfläche reflektiert oder gebrochen. Ein Ansatz für interaktive *doppelte* Brechungen mit Hilfe einer *Environment Map* an statischen Objekten wird in [Wym05a] vorgestellt. Unter Berücksichtigung der *Front-* und *Back-Faces* ergeben sich akkurate Doppel-Brechungen für ein Objekt inmitten einer *Environment Map*, die als im Unendlichen liegend betrachtet werden kann. Das Verfahren wird in [Wym05b] um einen iterativen Prozess erweitert, um auch nahe gelegene Objekte optisch brechen zu können. Hu und Qin verbessern die Technik in [HQ07] unter Verwendung von Impostorn [Ger98]. Ihre Methode verbessert die Darstellung von Reflexionen und Brechungen nah beieinander liegender Objekte.

Environment Maps können auch benutzt werden, um das Ray Tracing-Verfahren zu beschleunigen [HS01]. Dazu werden mehrere radiale *Environment Maps* verwendet, die für weiter entfernte Objekte verwendet werden — so werden erheblich weniger Dreiecksvergleiche benötigt, wenngleich das Verfahren nicht interaktiv ist.

Das wohl bekannteste Verfahren zur Repräsentation von Kaustiken ist das Photon Mapping [JC95]. Die Methode verteilt Energie mit Hilfe von Partikeln und akkumuliert diese in der Photon Map. Ein anderer Ansatz zur Repräsentation von Licht-

effekten unter Wasser (Kaustiken und *God-Rays*) ist die Verwendung projizierter Oberflächenpolygone [NN94, IND01, IDN03]. Wand und Strasser stellen in [WS03] eine schnelle Methode zur Erzeugung von Kaustiken vor, die Abbilder der Lichtquelle für jeden abgetasteten Punkt auf die Szene projiziert. Mit Hilfe von Filtern werden kontinuierliche Kaustiken erzeugt. Kaustiken können auch effizient dargestellt werden, wenn keine globalen Photon Maps erzeugt werden, sondern die Photonen im Bildraum auf der GPU erzeugt werden [KBW06]. Ein anderer Ansatz ist das *Caustic Mapping* [SK07]. Dieses stellt ein Pendant zum *Shadow Mapping* dar und erzeugt Kaustik-Texturen, die auf die Szene abgebildet werden. Ein GPU-basierter Ansatz zur Erzeugung von Kaustiken bei Höhenfeld-basierten Flüssigkeiten wird in [TS00] vorgestellt. Dazu werden die Lichtintensitäten ebenfalls in einer Textur akkumuliert.

3.6. Anwendungen

Nicht-Interaktive Verfahren — Flüssigkeiten in Filmen Computergenerierte Flüssigkeiten, die einen Anspruch auf Photorealismus besitzen, lassen sich nicht in Echtzeit simulieren und benötigen in der Regel viele Minuten bis Stunden für die Berechnung eines einzigen Bildes. Fokus der Forschung lag und liegt vornehmlich auf solchen Nicht-Echtzeitverfahren. Ohne einer Einschränkung in Bezug auf die Rechenzeit können wesentlich detailliertere Simulationsauflösungen verwendet werden und komplexere und realistischere Flüssigkeiten repräsentiert werden. Derartige Methoden werden vornehmlich für Flüssigkeiten in Filmen und Animationen verwendet. Für solch realistische Flüssigkeiten, wie sie in Kinofilmen benötigt werden, ist zusätzlich noch ein umfangreicher Postprozess (in der Regel manuell) vonnöten, um überzeugende und photorealistische Ergebnisse zu erzielen.

Die allgemeine Komplexität computeranimierter Flüssigkeiten kann an dem Film *Shrek* (2001, Dreamworks) verdeutlicht werden, der seinerzeit der bis dahin aufwendigste computeranimierte Film war. Auf die Frage, was die schwierigste Szene des ganzen Films gewesen sei, antwortete der Produzent des Films Jeffrey Katzenberg *"It's the pouring of milk into a glass."* [HP01]. Zahlreiche folgende Forschungsarbeiten haben später Verwendung in verschiedenen Filmen gefunden. Vor allem *Level-Set*-Methoden haben sich in Filmen etabliert: Beispiele sind *Terminator 3* (2003, Warner Bros.,[REN+04]), *Pirates of the Caribbean* (2003, Walt Disney Pictures, [REN+04]), *The Day After Tomorrow* (2004, 20th Century Fox, [IS04]) und *Scooby Doo 2* (2004, Warner Bros., [WH04]). Eine SPH-Methode wurde zum Beispiel in *Pirates of the Caribbean 3* (2007, Walt Disney Pictures, [MCZ07]) verwendet. Der Algorithmus von Tessendorf zur Darstellung ambienter Höhenfelder wurde zum Beispiel in *Waterworld* (1995, Universal Pictures) und *Titanic* (1997, Paramount Pictures, 20th Century Fox) verwendet [Tes04] — mit einer jeweiligen Auflösung von 2048×2048. Eine zweidimensionale Simulationsumgebung wird verwendet, um Rauch- und Flüssigkeitseffekt mit für Animateure einfachen Mitteln in dem Trick-Film *The Prince of Egypt* (1998, DreamWorks SKG) darzustellen [Wit99]. Die zweidimensionalen Simulationen werden dabei als senkrechte Ebenen im Raum

abgebildet, um Tiefe zu repräsentieren.

In der Computergraphik sind nicht nur physikalische Flüssigkeitsrepräsentationen sondern auch Flüssigkeitseffekte gefragt, die kein reales Pendant besitzen. Ein Beispiel ist die Animation liquider Charaktere, die zum Beispiel in den Filmen *The Abyss* (1989, 20th Century Fox) oder *Terminator 2* (1991, Tri-Star Pictures) verwendet wurden. Ziel ist es, den liquiden Charakter plausibel zu animieren, dabei jedoch den fließenden Eindruck zu erhalten. Dazu kann eine Flüssigkeitssimulation in Kombination mit zusätzlichen externen Kraftfeldern verwendet werden, die die Strömung der Flüssigkeit entsprechend der Charakter-Bewegung steuern [SY05].

Interaktive Verfahren Die meisten existierenden Techniken zur Flüssigkeitsrepräsentation sind nicht interaktiv oder in Echtzeit ausführbar. Interaktive Ansätze rückten erst in den letzten Jahren in den Fokus der Forschung — als Folge der großen Leistungsverbesserung von PC-Hardware und der Nachfrage aus der Computerspiele-Industrie. Die Ergebnisse interaktiver Verfahren sind wegen des hohen Aufwandes und des geringen zur Verfügung stehenden Zeitfensters nicht vergleichbar mit nicht-interaktiven Verfahren. Die große Beschränkung der Rechenzeit lässt die simulierten Flüssigkeiten in der Regel vergleichsweise undetailliert erscheinen.

Aufgrund wirtschaftlicher Interessen von Herstellerfirmen ist in der Regel nicht bekannt, wie die konkreten Simulationen durchgeführt werden. In älteren Computerspielen wurden in der Regel einfache Animationstechniken verwendet, um Wasseroberflächen zu bewegen. In jüngerer Zeit haben sich Spektrum-basierte Ansätze wie die Methode von Tessendorf zur Repräsentation ambienter Wellen durchgesetzt (z.B. *Crysis* (2007, Electronic Arts), *Grand Theft Auto IV* (2008, Rockstar Games)). Da jedoch in der Regel keine Simulationen verwendet werden, kann mit den Oberflächen nicht oder sehr eingeschränkt interagiert werden. In *Tomb Raider Underworld* (2008, Eidos Interactive) werden animierte Videosequenzen verwendet, um das spritzende Wasser zu repräsentieren, welches entsteht, wenn die Protagonistin durch eine Wasserlache läuft — mit sehr überzeugenden Ergebnissen.

Verwendung der GPU Ein andauernder Trend in der Echtzeit-Simulation physikalischer Phänomene ist die Verwendung der GPU nicht nur zur Darstellung, sondern auch zur Simulation oder zur Oberflächenextraktion. Aufgrund der parallelen Architektur heutiger GPUs können angepasste Probleme zum Teil um ein Vielfaches schneller ausgeführt werden, als auf der CPU. Auf den aktuellen Graphikkarten der Hersteller Nvidia and ATI sind zahlreiche Shader-, Textur- und Render-Einheiten integriert, so dass ein hoher Grad an Parallelität in Shader-Programmen erzielt werden kann (siehe Tabelle 3.1).

Aktuelle GPUs sind in der Lage, innerhalb eines einzigen *Renderpasses* Millionen von Fragment-Programmen auszuführen. Die Komplexität und der Befehlsumfang dieser Programme ist selbstverständlich beschränkt und Probleme müssen in der Regel angepasst werden, um eine Implementierung auf der GPU zu ermöglichen.

GPU	USU	TMU	ROU
Nvidia Geforce GTX 280	240	80	32
ATI Radeon HD 4870	800	40	40

Tab. 3.1.: Konfigurationskern aktueller GPUs (August 2008). Unified Shader Units (USU), Texture Mapping Units (TMU) und Render Output Units (ROU).

Vor allem Probleme, die in unabhängige Prozesse parallelisiert werden können, profitieren von einer Umsetzung auf aktuellen GPUs. Für hohe Performanz in der Computergraphik sollte deshalb bei zeitintensiven Problemen eine Implementierung auf der GPU in Betracht gezogen werden — sofern das jeweilige Problem sich mit den Möglichkeiten der Parallelisierung einer GPU umsetzen lässt.

3.7. Zusammenfassung

Existierende Arbeiten zur Flüssigkeitssimulation in der Computergraphik sind Thema des vorliegenden Kapitels. Es behandelt die wichtigsten Arbeiten im Bereich der Simulation, der Oberflächenextraktion und der Darstellung. Der Großteil existierender Arbeiten fokussiert jedoch auf Nicht-Echtzeitverfahren, da die Komplexität realer Flüssigkeiten hoch ist und jeder der drei auszuführenden Schritte dementsprechend zeitaufwendig in der Ausführung ist. Allgemein können die existierenden Methoden als Höhenfeld-basiert oder dreidimensional klassifiziert werden. Merkmal der dreidimensionalen Methoden ist der hohe Aufwand der Simulation und der Oberflächenextraktion. Deshalb sind die Ergebnisse in interaktiven Umgebungen noch verhältnismäßig undetailliert bzw. repräsentieren wenig Flüssigkeit, da lediglich niedrig aufgelöste Diskretisierungen verwendet werden können. Auch die akkurate Darstellung dreidimensionaler Flüssigkeiten ist aufwendig, da die Transparenz die Repräsentation optischer Eigenschaften wie Reflexion, Brechung oder Kaustiken erfordert — für realistische Ergebnisse bietet sich somit ein teures Ray Tracing-Verfahren an — zum Beispiel in Kombination mit dem ebenfalls teuren Photon Mapping-Verfahren. In interaktiven Umgebungen muss jedoch für eine schnelle Darstellung auf approximative Methoden zurückgegriffen werden.

Höhenfeld-basierte Verfahren basieren in der Regel auf zweidimensionalen Simulationen und können somit wesentlich effizienter ausgeführt werden. Auch die Oberflächenextraktion gestaltet sich für Höhenfelder effizienter, da keine dreidimensionalen Effekte berücksichtigt werden müssen. Aus diesem Grund können mit Höhenfeldbasierten Verfahren in interaktiven Umgebungen wesentlich detailliertere Ergebnisse erzielt werden. Nachteil der Höhenfeld-basierten Verfahren ist jedoch die Tatsache, dass keine dreidimensionale Effekte, wie zum Beispiel spritzendes Wasser oder brechende Wellen repräsentiert werden können, und die Flüssigkeit somit immer ausgesprochen flach erscheint.

Um diese Nachteile zu verringern, gleichzeitig jedoch interaktive Ergebnisse

erzielen zu können, wird im nachfolgenden Kapitel ein allgemeiner Ansatz zur Flüssigkeitssimulation in interaktiven Umgebungen vorgestellt, der die Verwendung sowohl zwei- als auch dreidimensionaler Simulationen in einer einzigen Simulationsumgebung propagiert. Zusätzlich wird die Verwendung empirischer Methoden diskutiert, um auch komplexe Effekte repräsentieren zu können, die für eine physikalische Simulation in interaktiven Umgebungen zu aufwendig wäre. Konkrete Realisierungen unter Verwendung verschiedener Simulationsmethoden werden dann in Kapitel 5–7 vorgestellt.

4. Ein allgemeiner Ansatz zur Repräsentation interaktiver Flüssigkeiten

Wie im vorherigen Kapitel herausgestellt wurde, lag der Fokus der Forschung zur Flüssigkeitssimulation in der Computergraphik bisher vornehmlich auf realistischen Ergebnissen in Nicht-Echtzeitumgebungen. Erst in jüngerer Zeit wurden Ansätze zur interaktiven Simulation vorgestellt, die jedoch oftmals punktuelle Lösungen darstellen oder nur einen bestimmten Effekt repräsentieren. Zudem sind die Ergebnisse wegen der geringen zur Verfügung stehenden Rechenzeit meistens undetailliert.

Die wichtigste wahrnehmbare Komponente realer Flüssigkeiten ist die *Oberfläche*. Gerade bei transparenten Flüssigkeiten wie Wasser kann die innerhalb eines Volumens liegende *Strömung* nicht erkannt werden. Erst wenn turbulente Situationen auftreten, in denen sich zum Beispiel die Oberfläche stark verformt oder sich verschiedene Flüssigkeiten vermischen, kann die Strömung indirekt wahrgenommen werden. Verformungen von Oberflächen hingegen sind bei allen Flüssigkeiten direkt erkennbar. Wird dieser Eigenschaft in einer Simulation nicht Genüge getragen, so erscheint das Ergebnis unrealistisch glatt. Diesem Sachverhalt widmet sich das vorliegende Kapitel.

Zunächst wird ein allgemeiner Ansatz zur Flüssigkeitsrepräsentation in interaktiven Umgebungen vorgestellt. Ziel ist es, eine problemangepasste und damit effiziente Repräsentation gewünschter Eigenschaften in interaktiven Umgebungen zu realisieren. Denn nicht alle Phänomene benötigen eine interaktive dreidimensionale Strömungssimulation. Als Beispiel sei die von Wind und Strömungen erzeugte, grundlegende Wellenbewegung eines Ozeans genannt — diese erfolgt ohne jegliche Nutzer- oder Objektinteraktion und kann somit ohne Einschränkung vorberechnet werden. Somit ist es gerade in interaktiven Umgebungen in Angesicht des zur Verfügung stehenden Zeitfensters sinnvoll, an geeigneten Stellen mit dimensionsreduzierten Simulationen oder Approximationen zu arbeiten. Aufgrund des Aufwandes einer Strömungssimulation kann es zum Beispiel sinnvoll sein, eine zweidimensionale Oberflächensimulation für Details hinzuzufügen oder gar auf eine Strömungssimulation zu verzichten, wenn Strömungen für ein Szenario nicht notwendig sind. Und obwohl die Plausibilität durch ein derartiges Vorgehen eventuell verringert wird, kann mit einem solchen Vorgehen eine überzeugende Repräsentation allgemeiner Flüssigkeitseffekte in interaktiven Umgebungen erzielt werden.

Auf Basis dieser Feststellung, wird in dieser Arbeit im Rahmen eines Schemas die Verwendung verschiedener Simulationsmethoden vorgeschlagen, um verschiedene

Phänomene effizient simulieren zu können. Vorteil einer derartigen allgemeinen Sicht sind die Möglichkeiten einer

- strukturierten Auswahl geeigneter Simulationsmethoden für verschiedene, räumlich getrennte Flüssigkeitsphänomene,
- strukturierten Kombination geeigneter Methoden um Möglichkeiten oder Details einer einzigen Simulationsmethode zu erweitern,
- Klassifikation der im Rahmen dieser Arbeit entwickelten Methoden und
- Klassifikation existierender hybrider Methoden, die verschiedene Simulationsmethoden adaptieren.

Im nachfolgenden Abschnitt 4.1 wird zunächst die Liquid-Pipeline behandelt, die in der Einleitung bereits kurz eingeführt wurde. Es schließt sich eine kurze Diskussion der Möglichkeiten und Grenzen unterschiedlicher Dimensionalitäten in der Flüssigkeitssimulation an (Abschnitt 4.2). Die weiteren Abschnitte folgen inhaltlich der Liquid-Pipeline, da die einzelnen Schritte unabhängig voneinander betrachtet werden können und die Reihenfolge dem Vorgehen einer Simulationsumgebung in der Praxis entspricht. Entsprechend dieser Abfolge stellt Abschnitt 4.3 anschließend das allgemeine Schema zu Echtzeit-Simulation von Flüssigkeiten in interaktiven Umgebungen vor. Daraus resultierende Anforderungen an die Oberflächenextraktion und die Darstellung werden in Abschnitt 4.4 und 4.5 behandelt. Eine Überblick über die Möglichkeiten und Grenzen aller Verfahren wird in Abschnitt 4.6 gegeben. Abschließend erfolgt eine Diskussion.

4.1. Liquid-Pipeline

Die Problematik der Flüssigkeitssimulation in der Computergraphik besteht in der hohen Komplexität der Eigenschaften von Flüssigkeiten. Die Hauptschwierigkeiten bestehen in der Repräsentation der

1. dreidimensionalen Strömungseigenschaften,
2. hohen Dynamik der Oberflächen und der
3. Vielzahl optischer Materialeigenschaften.

Diese drei Komponenten gilt es bei der Flüssigkeitsrepräsentation in der Computergraphik zu modellieren, um einen realistischen Eindruck der dargestellten Flüssigkeiten zu vermitteln. Entsprechend ergeben sich die drei in Abbildung 4.1 gezeigten, grundlegenden Schritte, die im Rahmen dieser Arbeit als *Liquid-Pipeline* bezeichnet werden. Die *Liquid-Pipeline* stellt somit die drei grundlegenden Komponenten dar, die jede Simulationsumgebung zur Darstellung realistischer Flüssigkeiten besitzen muss.

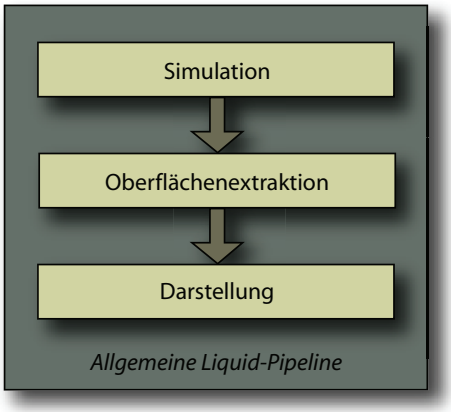
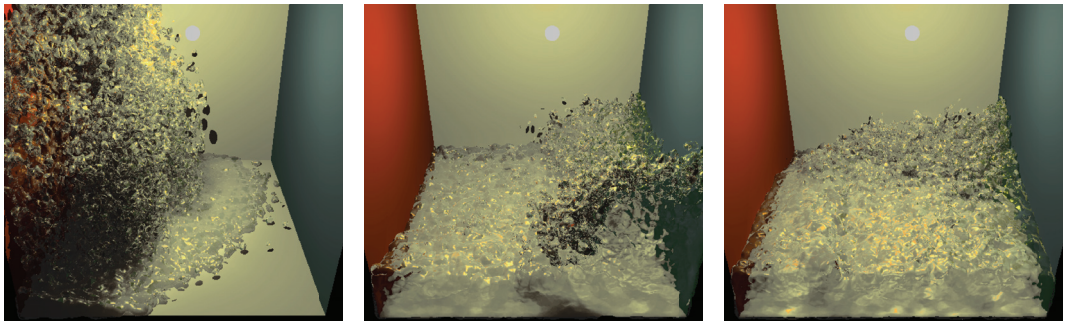


Abb. 4.1.: Die allgemeine Liquid-Pipeline (Wdh., siehe Abb. 1.1).

Im Folgenden werden die drei Schritte der *Liquid-Pipeline* in Hinblick auf ihre jeweilige Problematik in interaktiven Umgebungen separat betrachtet. Dieses Vorgehen trägt zum Verständnis der individuellen Problematik jedes Schrittes bei. Aufgrund der hohen Dynamik von Flüssigkeiten müssen alle drei Schritte für jedes darzustellende Bild durchgeführt werden. Ein grundsätzliches Ziel ist es, jeden Schritt möglichst effizient auszuführen, um eine interaktive Repräsentation zu ermöglichen.

Simulation Mit *Simulation* wird die Erzeugung des Datensatzes bezeichnet, der die Dynamik der Flüssigkeit beschreibt. Dieser Schritt kann physikalisch-basiert sein oder durch eine empirische Approximation realisiert werden. Eine physikalisch-basierte, dreidimensionale Flüssigkeitssimulation auf Basis der Navier-Stokes-Gleichungen benötigt viel Rechenzeit, führt aber zu realistischeren Animationen. Die dafür notwendige Lösung von Differentialgleichungssystemen bei einer feinen Abtastung ist numerisch aufwendig. Zudem wächst der Aufwand für dreidimensionale Simulationen kubisch mit dem Volumen — unabhängig davon, welches konkrete Verfahren zur Simulation verwendet wird.

Oberflächenextraktion Die *Oberflächenextraktion* bezeichnet den Prozess, der aus dem physikalischen Datensatz die freie Oberfläche für die Darstellung erzeugt. Die sich permanent ändernden dreidimensionalen freien Oberflächen beispielsweise polygonal zu repräsentieren, ist ein aufwendiger Prozess, da das gesamte Volumen abgetastet werden muss. Zudem muss dieser Schritt bei dynamischen Flüssigkeiten für jedes erzeugte Bild erneut durchgeführt werden. Somit stellt die Entwicklung bzw. Verwendung geeigneter und effizienter Algorithmen zur dreidimensionalen Oberflächenextraktion eine weitere Notwendigkeit in der Repräsentation von Echtzeit-Flüssigkeiten dar.



(a) (b) (c)

Abb. 4.2.: Beispiel: Nicht-Echtzeit. Die implizite Oberfläche einer interaktiven SPH-Simulation (40.000 Partikel, Simulation: $0,125\,s$/Zeitschr.) ist unter Verwendung von Ray Tracing und Photon Mapping (50.000 Photonen) dargestellt (3-6 Minuten/Frame, Auflösung: 512×512).

Darstellung Zur realistischen Darstellung von Flüssigkeiten müssen ihre optischen Eigenschaften repräsentiert werden. Gerade für transparente Flüssigkeiten stellt die realistische Darstellung in Echtzeitumgebungen eine weitere Schwierigkeit dar. Globale optische Effekte wie Absorption, Reflexion und Brechung und den daraus resultierenden Kaustiken sollten approximiert werden. Akkurat können diese Phänomene mit globalen Beleuchtungsmethoden dargestellt werden. Ray Tracing und Photon Mapping sind prinzipiell gut geeignet — siehe Abbildung 4.2. Aufgrund der hohen Laufzeit dieser Methoden auf Standard-PC-Hardware, sind auch an dieser Stelle schnelle, approximierende oder vereinfachende Verfahren für interaktive Bildraten notwendig.

4.2. Repräsentationsdimensionen

Allgemein kann zwischen zweidimensionalen und dreidimensionalen Flüssigkeitsrepräsentationen unterschieden werden. Für Oberflächen eignen sich vor allem zweidimensionale Simulationen, die effizient ausgeführt werden können. Sie eigenen sich zum Beispiel gut für Meeresoberflächen, Pools, etc. — also Szenarien mit in der Regel größeren Flüssigkeitsflächen, in denen dreidimensionale Flüssigkeitseffekte in Hinblick auf Echtzeitumgebungen vernachlässigt werden können. Für realistische dreidimensionale Effekte, wie zum Beispiel das Einschenken eines Glases Wassers, ist jedoch eine aufwendige dreidimensionale Simulation notwendig — wenngleich auch aus zweidimensionalen Simulationen unter bestimmten Bedingungen dreidimensionale Daten konstruiert werden können (zum Beispiel Abschnitt 5.1.4 — brechende Wellen). Entsprechend der Dimensionalität der Simulation ergibt sich in der Regel die Dimensionalität der Oberflächenextraktion. So können aus zweidimensionalen Simulationen effizient Höhenfelder extrahiert werden. Die Extraktion einer dreidimensionalen Oberfläche aus dreidimensionalen

Simulationsdaten hingegen ist wesentlich aufwendiger. Im folgenden Abschnitt wird nun die Simulation betrachtet — also Schritt 1 der Liquid-Pipeline.

4.3. Simulationsschema

Dieser Abschnitt präsentiert das allgemeine Schema zur Flüssigkeitssimulation in interaktiven Umgebungen. Dieses stellt eine strukturierte Sicht auf mögliche Simulationsverfahren verschiedener Komplexität zur Darstellung unterschiedlicher Flüssigkeitseffekte zur Verfügung.

Ausgangspunkt ist die folgende Erkenntnis: Eine überzeugende dreidimensionale Flüssigkeitssimulation sowohl großer Mengen (zum Beispiel Ozean) als auch komplexer strömungsmechanischer Effekte ist nicht in Echtzeit zu erreichen. Doch wie im vorhergehenden Abschnitt beschrieben wurde, beinhalten viele Situationen keine oder kaum wahrnehmbare dreidimensionale Effekte. Diese Tatsache motiviert das Vorgehen geeignete Simulationen zu koppeln. Komplexe Simulationen werden lediglich dort vorgenommen, wo sie tatsächlich benötigt werden. An anderen Orten wird lediglich eine vereinfachte oder dimensionsreduzierte Simulation vollzogen, wenn diese für das entsprechende Problem ausreichend ist. So kann der Rechenaufwand entsprechend der gewünschten Effekte minimiert und eine interaktive Geschwindigkeit erzielt werden. Für Flüssigkeitsanimationen, die keinerlei Interaktion mit dem Nutzer oder der Umgebung unterliegen, sollte innerhalb einer interaktiven Umgebung keine explizite Simulation durchgeführt werden. Vorberechnungen reduzieren den Aufwand auf einen fast vernachlässigbaren Anteil.

Zusätzlich kann eine direkte Kopplung verschiedener Verfahren die Qualität der dargestellten Flüssigkeiten wesentlich verbessern. So kann zum Beispiel die Kopplung einer niedrig aufgelösten Strömungssimulation und einer detaillierten Oberflächensimulation den visuellen Detailreichtum der repräsentierten Flüssigkeit aufgrund des unterschiedlichen Aufwandes effizient und wesentlich erweitern (siehe auch Abschnitt 5.1.2 und 5.1.3).

Grundkomponenten der Simulation Die Grundkomponenten einer allgemeinen Flüssigkeitssimulation sind in Abbildung 4.3 gegeben. Sie bestehen aus einer Komponente zur Flüssigkeitssimulation und einer zur starren Körper-Simulation.

Die Flüssigkeitssimulation ist unterteilt in *physikalisch-basierte Simulation* — auf der in dieser Arbeit der Fokus liegt — und *empirisch-basierte Emulation*. Erstere applizieren physikalisch-basierte Verfahren und sind somit in der Lage, reale Flüssigkeitseigenschaften zu repräsentieren. Letzte bezeichnen empirische Methoden oder auch Animationen, deren Ziel es ist, das Verhalten von Flüssigkeiten nachzubilden, ohne dabei eine aufwendige Simulation durchzuführen.

Erst die Kombination einer Flüssigkeitssimulation mit einer *starren Körper Simulation* ermöglicht zahlreiche, in interaktiven Umgebungen sinnvolle Anwendungen. So können mit der Kombination beider Verfahren dynamische Interaktionen

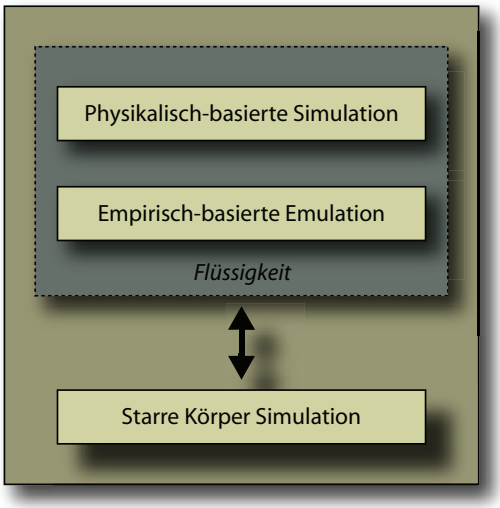


Abb. 4.3.: Grundkomponenten der Flüssigkeitssimulation.

der Flüssigkeit mit Objekten stattfinden, wie zum Beispiel Booten oder schwimmende Kisten. Statische Kollisionsobjekte, wie zum Beispiel ein Pfeiler, stellen numerisch gesehen Zwangsbedingungen in der Simulation dar und müssen somit nicht explizit simuliert zu werden. Die drei Grundkomponenten einer allgemeinen Flüssigkeitssimulation werden in den drei folgenden Abschnitten separat betrachtet.

4.3.1. Physikalisch-basierte Simulation

In Hinblick auf dreidimensionale virtuelle Umgebungen sind Flüssigkeitssimulationen mit zwei und mehr Dimensionen zweckmäßig. Die Flüssigkeiten können entsprechend ihrer Dimensionalität klassifiziert werden (Tabelle 4.1). Mit Hilfe dieser Klassifikation wird später über die Verwendung verschiedener Simulationsverfahren entsprechend der jeweiligen Anforderungen entschieden. Entsprechend der Dimensionalität ergeben sich die in Tabelle 4.2 gezeigten Simulationen.

Anhand der gewünschten physikalischen Effekte kann entschieden werden, welche Dimension für das jeweils darzustellende Problem am geeignetsten ist. So sind für große Wasserflächen zum Beispiel nicht notwendigerweise abgelöste Wassertropfen

Dim.	Beispiel
3D	Volumen — *z.B. Einschenken eines Glases, Br. Welle, Volumenströmung.*
2,5D	Oberflächen — *z.B. Wasseroberfläche, Wellenausbreitung,*
2D	Strömung in dünnen Schichten (Oberfl. im Gleichgewicht) — *z.B. Pfütze,*

Tab. 4.1.: Dimensionalitäten von Flüssigkeiten

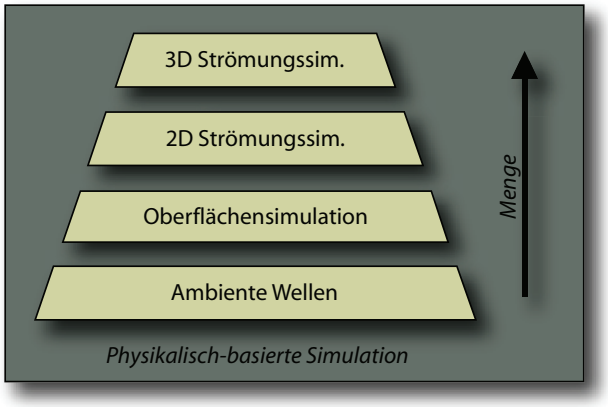


Abb. 4.4.: Stufenmodell der physikalisch-basierten Simulationsmethoden.

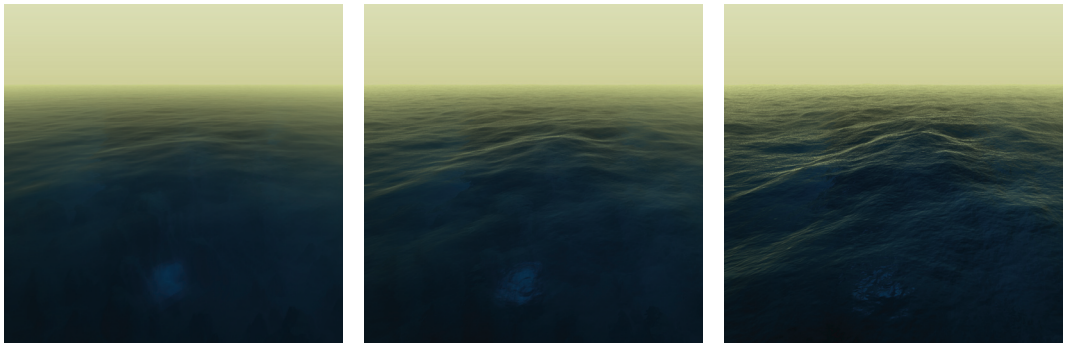
notwendig. Dementsprechend kann zur Reduktion der Simulationsdimensionalität auf ein zweidimensionales Simulationsverfahren mit Höhenfeld-basierter Darstellung zurückgegriffen werden.

Entsprechend des Aufwandes ergibt sich das Stufenmodell für die physikalisch-basierte Simulationskomponenten. Es ist in Abbildung 4.4 dargestellt. Die Reihenfolge der Stufen ergibt sich aus ihrer jeweiligen Komplexität, anhand derer, den gewünschten Effekten und der zur Verfügung stehenden Rechenzeit entschieden werden kann, welche Methode für den jeweiligen Fall am geeignetsten ist. Die algorithmische Komplexität der zweidimensionalen Strömungssimulation und der Oberflächensimulation ist gleich — wenngleich der numerische Aufwand der Strömungssimulation in der Praxis höher ist. Aus diesem Grund ist die 2D Strömungssimulation in Abb. 4.4 über der Oberflächensimulation angeordnet.

Die Anordnung des Stufenmodells repräsentiert gleichzeitig schematisch die Menge der in interaktiven Szenarien simulierbaren Flüssigkeit. So kann mit Hilfe ambienter Wellen ein quasi unendlicher Ozean dargestellt werden, der jedoch nicht interaktiv ist. Auf der anderen Seite kann mit einer dreidimensionalen Strömungssimulation in interaktiven Umgebungen eher die Menge eines Wasserglases oder Aquariums dargestellt werden — jedoch bei voller Interaktivität. Im Folgenden werden die vier Schritte des physikalischen Stufenmodells detailliert behandelt.

Dim.	Simulation	Aufwand
3D	3D Strömungssimulation	$O(n^3)$
2D	2D Strömungssimulation	$O(n^2)$
2D	Oberflächensimulation	$O(n^2)$
2D	Ambiente Wellen (vorberechnet)	$O(1)$

Tab. 4.2.: Dimensionalitäten und Aufwand von Simulationen



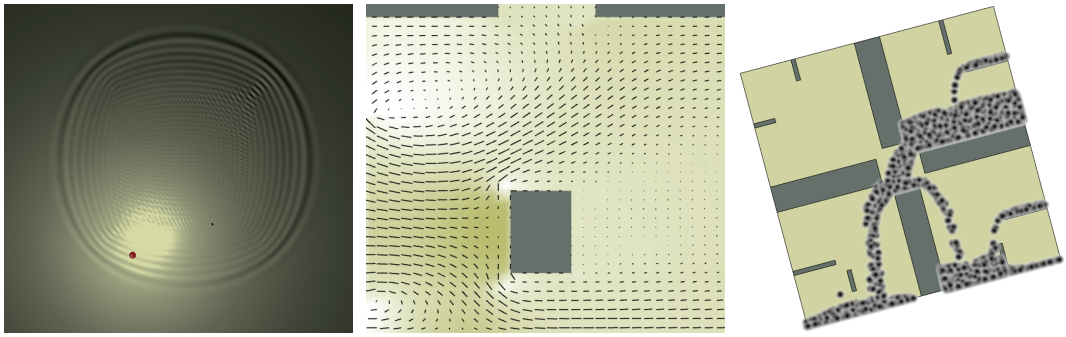
(a) (b) (c)

Abb. 4.5.: Beispiel: Ambiente Wellen — erzeugt mit dem Algorithmus von Tessendorf [Tes01]. Drei Layer werden verwendet. Von (a) bis (c) wird die Amplitude zweier Layer vergrößert.

Ambiente Wellen Der Begriff *ambiente Wellen* wird in Anlehnung an [Tes01] verwendet. So wie im Phong'schen Beleuchtungsmodell die Existenz einer diffusen Grundhelligkeit mit dem ambienten Term beschrieben wird, so wird die aus Wind und Strömung resultierende grundlegende, ohne Interaktion hervorgerufene Bewegung von Flüssigkeiten als *ambient* bezeichnet. Denn eine Wasserfläche besitzt schon bei geringen Windgeschwindigkeiten eine grundlegende Wellenbewegung, die innerhalb einer virtuellen Umgebung einen hohen Anteil zum Realismus der Simulation beiträgt (Abb. 4.5).

Ambiente Wellen erlauben keinerlei Interaktivität und sind unter Verwendung einer Vorberechnung zur Laufzeit effizient dargestellbar. Es existieren verschiedene, empirische und physikalisch-basierte Methoden, um derartige Wellen zu erzeugen (siehe auch Abschnitt 3.3.1). Die Vorberechnung erlaubt es, den damit verbundenen Aufwand während der interaktiven Repräsentation der Flüssigkeit zu vernachlässigen. Deshalb bietet sich ein kostenintensives physikalisches Verfahren zur Erzeugung an, um ein hohes Maß an Realismus erzielen zu können. Aus diesem Grund werden die ambienten Wellen in dem vorgestellten Stufenmodell der physikalisch-basierten Simulationen eingeordnet.

Oberflächensimulation Mit *Oberflächensimulation* werden im Folgenden Höhenfeld-basierte Simulationen von Oberflächenwellen bezeichnet. Diese resultieren zum Beispiel von Objekten, die sich durch die Flüssigkeit bewegen oder in die Flüssigkeit eintauchen. Ein Beispiel ist Abbildung 4.6a gegeben. Mit einer Oberflächensimulation können zudem Wellen an beliebigen Positionen erzeugt werden. Da Oberflächensimulationen zweidimensionale Simulationen darstellen und lediglich in der Repräsentation eines Höhenfeldes resultieren, können sie effizient ausgeführt werden. Somit kann eine feine Diskretisierung der Oberfläche — auch in interaktiven Szenarien — erzielt werden, was im Vergleich zu dreidimensionalen Simulationen in hohen Details der repräsentierten Wellen resultiert.



(a) (b) (c)

Abb. 4.6.: Beispiele: Zweidimensionale Simulationsmethoden. Oberflächensimulation mit der Wellengleichung (a), Strömungssimulation mit einer FD-Methode (b), Strömungssimultion mit einer SPH-Methode (c).

2D und 3D Strömungssimulation Schließlich werden mit *Strömungssimulation* zwei- und dreidimensionale Simulationen bezeichnet, die den Fluss entsprechend der Strömungsmechanik bestimmen. Diese Simulationen resultieren aus der Lösung der Navier-Stokes-Gleichungen. Dabei besitzen zweidimensionale Simulationen ein wesentlich besseres Laufzeitverhalten als dreidimensionale Simulationen — können jedoch lediglich zweidimensionale Strömungen repräsentieren. Bei dreidimensionalen Simulationen ist die Komplexität hoch, da ein Differentialgleichungssystem in drei Dimensionen gelöst werden muss. Zudem ändern sich die freien Oberflächen dreidimensionaler Flüssigkeiten permanent und müssen dabei in der Simulation berücksichtigt werden. Mit freien Oberflächen wird dabei die Grenzschicht zwischen einer Flüssigkeit und einem Gas bezeichnet. Auf der anderen Seite erhält die simulierte Flüssigkeit erst durch die Lösung der dreidimensionalen Navier-Stokes-Gleichungen ein plausibles dreidimensionales Verhalten. Beispiele zweidimensionaler Strömungssimulationen werden in Abbildung 4.6b,c gegeben. Abbildung 4.6b zeigt das Geschwindigkeits- und Dichtefeld bei Verwendung einer FD-Methode und Abbildung 4.6b die überblendeten Partikel einer SPH-Simulation.

4.3.2. Empirisch-basierte Emulation

Empirisch-basierte Emulationen basieren auf der Idee, den jeweiligen Effekt darzustellen und nicht die Ursache des Effektes zu simulieren, die in der Regel wesentlich aufwendiger zu simulieren ist. Nachteil ist oftmals, dass eine realistische Interaktion nicht möglich ist.

Derartige Methoden werden im Rahmen des vorgestellten Schemas im Stufenmodell der empirisch-basierten Flüssigkeiten zusammengefasst — siehe Abbildung 4.7. Die Einbettung von *Videosequenzen*, ist wohl die kostengünstigste Methode zur Flüssigkeitsrepräsentation. *Plausible Animationen* bezeichnen hier die Verwendung klassischer Animationstechniken, wie zum Beispiel dem *Keyframing*. Schließlich be-

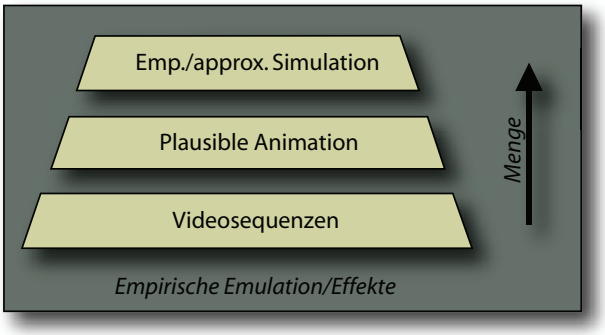


Abb. 4.7.: Stufenmodell der empirisch-basierten Emulationsmethoden.

zeichnen *empirische/approximative Emulationen* die Repräsenation ohne eine direkte physikalische Anlehnung.

Die Anordnung innerhalb des Stufenmodells folgt dem prinzipiellen Aufwand der Methoden. Vorteil derartiger empirischer Ansätze ist es, dass auf eine teure physikalische Simulation verzichtet werden kann und damit die Oberflächenextraktion wesentlich vereinfacht bzw. unnötig werden kann. Die einzelnen Komponenten des empirischen Stufenmodells werden im Folgenden separat behandelt.

Videosequenzen Vom Standpunkt des Berechnungsaufwands aus betrachtet, ist die Integration von Videosequenzen in die virtuelle Umgebung die kostengünstigste Darstellung von Flüssigkeiten. So kann eine einzige Videosequenz eingebettet werden, die zum Beispiel einen Wasserfall darstellt — und gleichzeitig, aus dem nötigen Abstand betrachtet — photorealistisch wirkt. Eine Projektion der Videosequenzen auf dreidimensionale Objekte realisiert auch scheinbar dreidimensionale Flüssigkeiten.

Eine andere Anwendung ist die Verwendung zahlreicher Videosequenzen zur Repräsentation bestimmter, häufig auftretender Effekte. In [Tat06] wird die Verwendung von Videosequenzen vorgestellt, um das typische Wasserspritzen zu repräsentieren, welches beim Eintreffen eines Regentropfens auf eine Wasseroberfläche entsteht. Durch die Verwendung dieser Videosequenz für jeden einschlagenden Tropfen wird ein hoher Grad an Realismus erzielt.

Plausible Animationen Animationen zur Repräsentation von Flüssigkeiten können unter Verwendung klassischer Animationsverfahren wie dem Bild-für-Bild Verfahren oder dem Keyframing erzeugt werden.

In Videospielen werden oftmals plausible Animationen verwendet, um bewegte Flüssigkeiten zu repräsentieren. Gerade bei älteren Systemen mit geringer Rechenleistung gab es lange keine Alternative zur Verwendung von Animationen zur Repräsentation von Flüssigkeiten. So kann zum Beispiel eine einfache, ambiente Wasseroberfläche mit der Überlagerung dynamischer trigonometrischer Funktionen

imitiert werden. Des Weiteren können Animationen verwendet werden, um eine eigentlich nicht darstellbare Interaktivität zu imitieren. So wird in einigen aktuellen Videospielen das spitzende Wasser, das von einem bewegten Motorboot erzeugt wird, mit dynamischen Texturen dargestellt (siehe Kapitel 3). Ein Grundproblem der Animationsverfahren ist jedoch die Tatsache, dass die komplexen Eigenschaften von Flüssigkeiten oftmals nicht realistisch mit einfachen Approximation animiert werden können — und die entsprechenden Ergebnisse in Bezug auf Realismus in der Regel den physikalisch-basierten Simulationen weit unterlegen sind.

Neben einer reinen Simulation der Bewegung von Flüssigkeiten können zudem weitere Animations-Effekte, wie zum Beispiel Schaum oder Gischt, in einem Postprozess hinzugefügt werden. Derartige Effekte sind in der Regel stark turbulent und besitzen einen hohen Detailgrad. Für diese Phänomene bieten sich Animationen an, da eine physikalisch-basierte Simulation zu komplex wäre.

Empirische/approximative Simulationen Schließlich können zur Flüssigkeitsrepräsentation auch approximative Simulationen verwendet werden. Dabei ist die Idee, die teure Simulation der Navier-Stokes-Gleichungen für spezielle Repräsentationen durch eine einfachere Approximation zu ersetzen. Der Vorteil dieses Vorgehens liegt darin, dass eine Interaktion mit der Flüssigkeit möglich ist, da eine Simulation verwendet wird — auch wenn es sich nicht um eine physikalisch-basierte Flüssigkeitssimulation handelt.

Ein populäres Beispiel ist die Verwendung von Partikeln, die sich unabhängig und lediglich unter Einfluss der Gravitations- und Reibungskräfte bewegen und zum Beispiel fallende Wassermassen und -tropfen repräsentieren. Diese gehorchen im freien Fall weniger den Gesetzen der Strömungsmechanik als den Gesetzen der Gravitation, so dass eine derartige Approximation für fallende Flüssigkeiten genutzt werden kann. Mit diesem Ansatz können zum Beispiel Wasserfälle oder Fontänen dargestellt werden, die gleichzeitig jedoch interaktiv sind und mit Objekten ihrer Umgebung verhältnismässig plausibel interagieren können.

4.3.3. Starre Körper

Die oben beschriebenen Verfahren ermöglichen es, das Verhalten einer Flüssigkeit adaptiv in Abhängigkeit der Komplexität der gewünschten Phänomene zu simulieren. Eine wichtige Anwendung in der Computergraphik ist jedoch nicht nur die Simulation und Interaktion mit Flüssigkeiten, sondern zusätzlich die Modellierung der Wechselwirkung zwischen Flüssigkeiten und Objekten. Bei der Integration von Objekten ist zwischen zwei prinzipiellen Typen zu unterscheiden, die die jeweilige Simulation bzw. Kollisionsbehandlung bestimmen:

- Statische Körper und
- dynamische Körper.

Als *statische Körper* werden nachfolgend Objekte bezeichnet, die in ihrer Position fix sind und nicht von der Flüssigkeit beeinflusst werden. Damit können sie als Zwangsbedingungen innerhalb der jeweiligen Flüssigkeitssimulation modelliert werden. Als *dynamische Körper* werden im Folgenden Objekte bezeichnet, die sich entsprechend den Gesetzen der Mechanik bewegen. Um eine Kopplung mit der Flüssigkeit zu erzielen, werden Kräfte einbezogen, die durch Auftrieb und Strömung verursacht werden. So kann eine realistische Dynamik der Objekte simuliert werden. Eine Rückkopplung der starren Körper-Simulation zur Flüssigkeitssimulation ermöglicht die Erzeugung von Wellen.

4.3.4. Zusammenfassung

Mit den in den vorherigen Abschnitten beschriebenen Stufenmodellen der physikalisch-basierten Simulation und der empirisch-basierten Emulation ergibt sich aus den Grundkomponenten der Flüssigkeitssimulation (Abb. 4.3) das *allgemeine Schema der interaktiven Flüssigkeitssimulation*, welches in Abbildung 4.8 dargestellt ist. Es stellt eine strukturierte Sicht auf Simulationsverfahren verschiedener Komplexität zur Darstellung von Flüssigkeiten zur Verfügung.

Die in Abbildung 4.8 dargestellten Komponenten können miteinander interagieren. So kann zum Beispiel ein Körper, der in ein Gewässer fällt, gleichzeitig radial propagierende Wellen innerhalb einer Oberflächensimulation und die zugehörigen Wasserspritzer innerhalb einer empirischen Simulation erzeugen. Umgekehrt beeinflussen Oberflächenwellen zum Beispiel die Bewegung eines schwimmenden Objektes. Die genaue Kopplung hängt von den verwendeten Simulationsmethoden und den gewünschten Effekten ab, die repräsentiert werden sollen. Die konkreten Möglichkeiten derartiger Kopplungen werden im anschließenden Kapitel 5 behandelt.

4.4. Oberflächenextraktion

Die hohe Dynamik von Flüssigkeiten erfordert die Erzeugung einer Oberfäche für jeden darzustellenden Simulationsschritt. Auch die Extraktion der Oberfläche kann anhand ihrer Dimension klassifiziert werden, wobei der Aufwand mit der Dimension wächst — siehe Tabelle 4.3.

Dim.	Oberflächenextraktion	Aufwand
3D	3D Oberfläche	$O(n^3)$
2,5D	Höhenfeld	$O(n^2)$
2D	Ebene	$O(1)$

Tab. 4.3.: Aufwand polygonaler Oberflächenextraktionsmethoden (n: Diskretisierung der räumlichen Dimensionen).

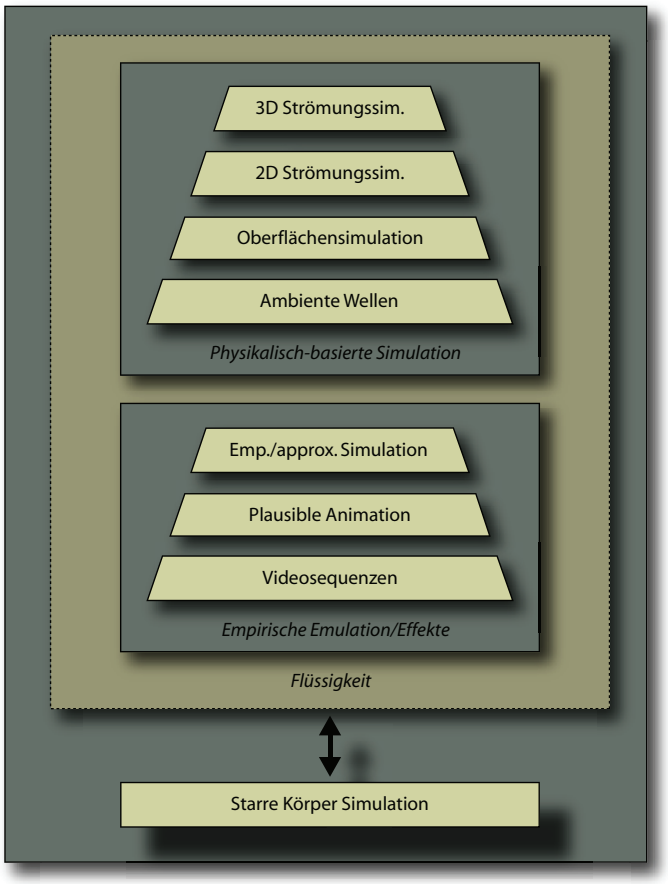


Abb. 4.8.: Allgemeines Schema zur interaktiven Flüssigkeitssimulation.

Der *zweidimensionale Fall* des Spezialfalles Ebene ist der Vollständigkeit halber aufgeführt, da lediglich die Normale einer Ebene bestimmt werden muss. Dieser Fall würde entsprechend des allgemeinen Schemas zur Flüssigkeitssimulation (Abb. 4.8) zum Beispiel für eine dreidimensionale Strömungssimulation ohne Betrachtung freier Oberflächen zutreffen.

Im *zweieinhalb-dimensionalen* Fall, also zum Beispiel bei der Simulation von Oberflächenwellen, kann die Oberfläche mit Hilfe eines Höhenfeldes extrahiert werden. Höhenfelder sind gut geeignet für interaktive Umgebungen, da sie verhältnismäßig effizient generiert und dargestellt werden können.

Dreidimensionale Flüssigkeiten besitzen eine dreidimensionale Oberfläche, deren Extraktion aufwendig sein kann. Dieser Fall trifft für die jeweils oberste Stufe der physikalisch-basierten Simulation und der empirisch-basierten Emulation zu (Abb. 4.8). Also im Falle einer *dreidimensionalen physikalisch-basierten Simulation* mit Betrachtung freier Oberflächen und einer *dreidimensionalen empirischen Simulation.*

Die Darstellung von Flüssigkeiten kann allgemein klassifiziert werden als

- blickpunktunabhängig und
- blickpunktabhängig.

Eine blickpunktunabhängige Darstellung hat den Vorteil einer einfachen Implementierung — jedoch können zahlreiche Polygone außerhalb des sichtbaren Bereiches erzeugt werden, so dass die Rechenzeit nicht optimal genutzt wird. Vorzuziehen sind in interaktiven Umgebungen blickpunktabhängige Verfahren, die die Oberfläche lediglich im sichtbaren Bereich extrahieren, so dass zum Beispiel bei gleicher Anzahl verwendeter Polygone wesentlich mehr Details dargestellt werden können.

4.5. Darstellung

Fokus dieser Arbeit liegt auf der realistischen Repräsentation virtueller Flüssigkeiten. Für die Darstellung transparenter Flüssigkeiten wie zum Beispiel Wasser, ist die realistische Repräsentation optischer Effekte wie Reflexionen und Brechungen essentiell. Zudem sollten Absorptionen und Kaustiken betrachtet werden. Somit liegt der Schwerpunkt der Darstellung auf der Nachbildung folgender optischer Eigenschaften:

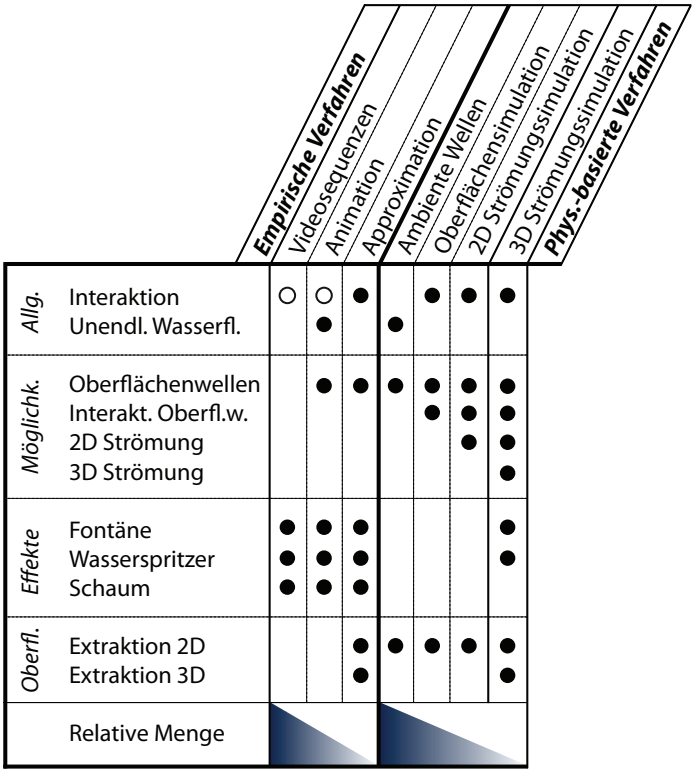
- Beleuchtung,
- Reflexion und Brechung,
- Absorption,
- Kaustiken (resultierend aus Lichtbrechung und -reflexion).

Weitere optische Effekte, wie zum Beispiel Beugung und die daraus resultierenden Interferenzen oder Polarisation (bei monochromatischem Licht), sind im Alltag kaum zu beobachten und können daher in einer computergraphischen Anwendung vernachlässigt werden.

Auch bei der Darstellung kann zwischen Höhenfeld-basierten und dreidimensionalen Oberflächen unterschieden werden. Die Höhenfeld-basierten Verfahren nutzen in der Regel eine planare Approximationen der gesamten Oberfläche, um optische Effekte zu repräsentieren. Bei dreidimensionalen Oberflächen werden in der Regel geschickte empirische Approximationen verwendet, um den Schein realer Optik zu wahren.

4.6. Überblick

Die Möglichkeiten und Grenzen der in den vorherigen Abschnitten diskutierten Methoden sind entsprechend dem allgemeinen Schema zur Flüssigkeitssimulation (Abschnitt 4.3.4) in Tabelle 4.4 dargestellt. Die folgenden Ausführungen beziehen sich auf diese Tabelle.



Tab. 4.4.: Möglichkeiten und Grenzen der verschiedenen Methoden. Bezeichnung: ○ - eingeschränkt möglich, ● - möglich.

Abgesehen von den ambienten Wellen ermöglichen alle Methoden eine Interaktion. Die Interaktion mit Videosequenzen und Animationen wurde als eingeschränkt klassifiziert, da es sich weniger um Interaktion handelt, als um ein ereignisabhängiges Abspielen einer Bildsequenz. Unendliche Wasserflächen können lediglich mit Hilfe von Animationen und ambienten Wellen erzielt werden, wenngleich mit der in Abschnitt 5.1.1 vorgestellten Methode zweidimensionale Simulationen adaptiert werden, um in wichtigen und dynamischen Regionen Simulationen durchzuführen. So kann der Anschein einer unendlichen Simulationsumgebung erreicht werden.

Im Feld der *Möglichkeiten* sind die empirischen Methoden unterrepräsentiert, da sie keine Simulation durchführen. Animationen und Approximationen können jedoch zur Darstellung von Oberflächenwellen verwendet werden, mit denen nicht interagiert wird. Videosequenzen können realistische Abbildungen erzeugen, doch eine dynamische und interaktive Repräsentation ist schwerlich zu erzielen. Die physikalischbasierten Verfahren nehmen in ihren Möglichkeiten mit wachsender Komplexität zu. Dies ist an der diagonalen Möglichkeitenklassifizierung zu erkennen.

Die empirischen Verfahren können detaillierte *Effekte* effizient erzeugen, wie zum Beispiel Fontänen oder spritzendes Wasser. Ansonsten sind derartige Effekte lediglich unter Verwendung einer dreidimensionalen Simulation zu erzielen. Schaum stellt

zum Beispiel eine Ausnahme dar, da die komplexen physikalischen Zusammenhänge nicht durch die Navier-Stokes Gleichungen beschrieben werden und somit nicht wie die erstgenannten Effekte mit einer dreidimensionalen Strömungssimulation erzeugt werden können. Die Simulation von Schaumbildung ist aufwendig und wird vornehmlich im Bereich der Nicht-Echtzeit untersucht (siehe auch Abschnitt 3.3.5).

Zur *Oberflächenextraktion* können bei den zweidimensionalen Verfahren Höhenfelder verwendet werden. Für dreidimensionalen Repräsentationen ist jedoch eine dreidimensionale Oberflächenextraktion vonnöten. Schließlich ist die relative, darstellbare Menge gegeben, die einen Indikator für die unterschiedlichen Komplexitäten und die damit verbundenen Detailgrade darstellt.

4.7. Zusammenfassung

In diesem Kapitel wurde ein allgemeiner Blick auf die Repräsentation von Flüssigkeiten in Echtzeitumgebungen beschrieben. Die vorgestellte *Liquid-Pipeline* fasst die drei notwendigen Schritte für eine Flüssigkeitsrepräsentation in der Computergraphik zusammen: Simulation, Oberflächenextraktion und Darstellung.

Für die *Simulation* wurde ein allgemeines Schema vorgestellt, welches eine strukturierte Sicht auf mögliche Simulationsverfahren verschiedener Komplexität darstellt. Ziel ist es nicht nur, physikalisch-basierte Flüssigkeiten in vielen Erscheinungsformen in interaktiven Szenarien zu simulieren, sondern auch unter Zuhilfenahme verschiedenster Simulationen gleichzeitig detaillierte Ergebnisse zu ermöglichen — trotz der zeitlichen Einschränkung. Welche Stufen des Modells in einer jeweiligen Umgebung verwendet werden, hängt dabei von den gewünschten Eigenschaften, aber auch von der gewünschten Performanz ab.

Die *Oberflächenextraktion* und die *Darstellung* wurden unter Berücksichtigung der Dimensionalität des aus der Simulation resultierenden Datensatzes diskutiert. Der vorhergehende Abschnitt stellt eine Entscheidungsgrundlage zur Verfügung, anhand derer die entsprechenden Methoden für eine konkrete Realisierung ausgewählt werden können. Genauere Details der Realisierung und Kopplung verschiedener, konkreter Repräsentationsverfahren werden in den drei nachfolgenden Kapiteln behandelt, wobei die drei Kapitel entsprechend der Struktur der allgemeinen Liquid-Pipeline gegliedert sind.

5. Simulation

Die dreidimensionale Flüssigkeitsrepräsentation in Echtzeitumgebungen führt aufgrund der hohen Komplexität der einzelnen Schritte der Liquid-Pipeline zu verhältnismäßig undetaillierten Ergebnissen. Somit sind neue Verfahren erforderlich, um realistischere Resultate zu erzielen. Dieses Kapitel zur *Simulation* und die beiden folgenden Kapitel *Oberflächenextraktion* und *Darstellung* widmen sich dieser Problematik im Konkreten und basieren dabei auf den im vorhergehenden Kapitel präsentierten allgemeinen Überlegungen.

Dieses Kapitel der Simulation orientiert sich an den *Grundkomponenten der Flüssigkeitssimulation*, die in Abschnitt 4.3 behandelt wurden. Somit werden im Folgenden zunächst Methoden für die physikalisch-basierte Simulation und anschließend Methoden für die empirisch-basierte Emulation präsentiert. Abschließend wird die Inklusion starrer Körper in die jeweiligen Simulationsmethoden thematisiert.

5.1. Physikalisch-basierte Simulation

Die unterschiedlichen Kopplungsmöglichkeiten im Rahmen des physikalisch-basierten Stufenmodells (siehe Abschnitt 4.3.1) sind Thema dieses Abschnitts. Zugleich werden die Qualität und zu erwartenden Ausführgeschwindigkeiten der Resultate demonstriert. Abbildung 5.1 veranschaulicht die in den nächsten Abschnitten kombinierten Simulationstechniken. Folgende Kombinationen werden im weiteren Verlauf des Textes behandelt:

- Ambiente Wellen, Oberflächensimulation,
- Ambiente Wellen, Oberflächensimulation, 2D Strömungssimulation,
- Ambiente Wellen, Oberflächensimulation, 3D Strömungssimulation,
- Schichtung von 2D Strömungssimulationen.

Die Kopplung zwei- und dreidimensionaler Strömungssimulationen wird in dieser Arbeit nicht thematisiert, da diese bereits untersucht wurde und keine echtzeifähige Repräsentation ermöglicht. So demonstrieren Thürey et al. eine adaptive Kopplung zwei- und dreidimensionaler Strömungssimulationen in [TRS06]. Die nahe Umgebung eines Bootes wird dreidimensional simuliert, während die Repräsentation der weiteren Umgebung mit einer Flachwassersimulation erfolgt. Irving et al. koppeln ein zweidimensionales, niedrig aufgelöstes Gitter mit einem dreidimensionalen Gitter

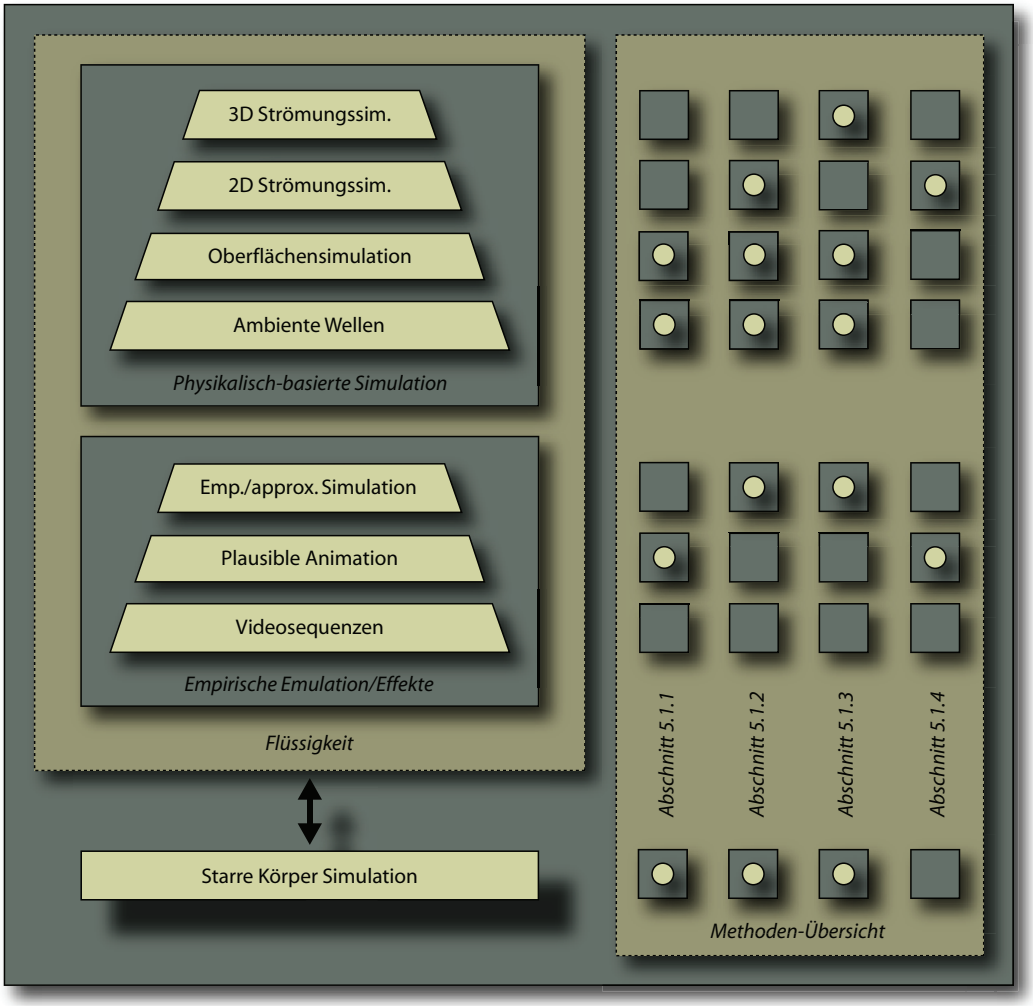


Abb. 5.1.: Übersicht über die in den folgenden Abschnitten präsentierten Kombination verschiedener Techniken.

in der Umgebung der Oberfläche [IGLF06] und erzielen so Geschwindigkeitsvorteile in der Berechnung.

Begonnen wird in Abschnitt 5.1.1 mit einer ambienten Wellensimulation in Verbindung mit frei translierbaren Oberflächensimulationen. Mit der Methode können adaptive und detaillierte Wellen im Rahmen einer beliebig großen Flüssigkeitsumgebung realisiert werden. Die ambiente Simulation wird dann im weiteren Verlauf nicht weiter erwähnt, da sie prinzipiell jeder Höhenfeld-basierten Simulation hinzugefügt werden kann. Anschließend erfolgt die Präsentation von Methoden zur Kopplung von Oberflächensimulationen mit 2D-Strömungssimulationen (Abschnitt 5.1.2) und 3D-Strömungssimulationen (Abschnitt 5.1.3). Ziel dieser

Verfahren ist es detaillierte Oberflächen innerhalb Echtzeitumgebungen zu erzielen und gleichzeitig die Möglichkeiten einer Strömungssimulation zur Verfügung zu stellen. So können strömende Flüssigkeiten dargestellt werden ohne auf Oberflächendetails verzichten zu müssen. Den Abschluss dieses Abschnitts zur interaktiven Simulation bildet eine Methode zur Erzeugung dreidimensionaler brechender Wellen aus zweidimensionalen Strömungsimulationen, die geeignet kombiniert werden (Abschnitt 5.1.4).

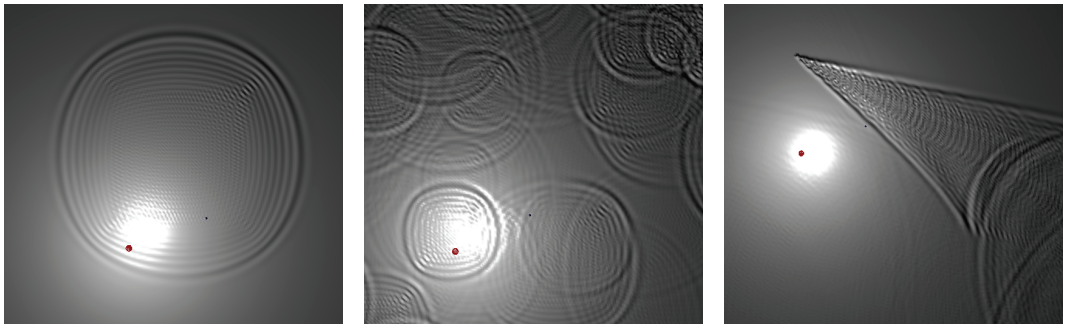
An dieser Stelle sei angemerkt, dass Laufzeitmessungen für die einzelnen Methoden in den einzelnen Abschnitten gegeben werden. Diese beziehen sich jedoch auf die Ausführung der gesamten Liquid-Pipeline, also inklusive der Oberflächenextraktion und der Darstellung — obwohl diese im Detail erst in den nachfolgenden Kapiteln 6 und 7 behandelt werden. Dieses Vorgehen ist zweckmäßig, um die Laufzeiten den jeweiligen Verfahren eindeutig und verständlich zuordnen zu können.

5.1.1. Kombinierte Oberflächensimulation und ambiente Wellen: Infinite Flüssigkeitsumgebungen mit Objektinteraktion

Die im Folgenden beschriebene Simulationsumgebung zielt auf die interaktive Repräsentation offener, scheinbar unendlich großer Flüssigkeitsflächen mit Objektinteraktion ab. Es sind jedoch auch beschränkte Oberflächen mit der vorgestellte Methode darstellbar, wobei die Performanz noch gesteigert werden kann. Die Grundidee ist eine adaptive Simulation, die lediglich in einem definierten Bereich um den Betrachter oder einer *Region of Interest* (RoI) simuliert. Dadurch kann zum Beispiel ein fahrendes Boot realistisch mit der Flüssigkeit interagieren – gleichzeitig aber auch beliebige Strecken zurücklegen. Eine Reduktion von Berechnungen ist unumgänglich, um dieses Ziel auf gegenwärtiger Hardware zu realisieren. Eine existierende Strömung kann in derartigen Szenarien zunächst vernachlässigt werden, da das visuelle Erscheinungsbild im Wesentlichen von Oberflächenwellen geprägt ist. Somit kann die physikalische Simulation auf eine zweidimensionale Wellenrepräsentation beschränkt werden, was zu einer wesentlichen Beschleunigung führt. Die Behandlung von Strömungen erfolgt jedoch prinzipiell ähnlich und wird in Abschnitt 5.1.1.4 diskutiert. Zur Repräsentation bietet sich ein Höhenfeld an, da lediglich eine Oberfläche simuliert wird. Damit kann eine Oberflächenextraktion ebenfalls effizient unter Verwendung der GPU durchgeführt werden (siehe dazu Kapitel 6). Aus Geschwindigkeitsgründen wird in diesem Ansatz auf die Darstellung dreidimensionaler Effekte wie zum Beispiel von Gischt verzichtet. Konzeptuell können derartige Effekte entsprechend Kapitel 4 integriert und mit der Wellensimulation gekoppelt werden. Zur Repräsentation von Oberflächenschaum wird in Abschnitt 5.2.2.1 eine Animationstechnik vorgestellt.

5.1.1.1. Simulation

Zur Darstellung infiniter Oberflächen wird lediglich in notwendigen Gebieten zur Laufzeit *simuliert*, in anderen werden Vorberechnungen verwendet. Als notwendige



(a) (b) (c)

Abb. 5.2.: Beleuchtete Lösung der zweidimensionalen Wellengleichung. Eine radiale Welle propagiert (a), mehrere radiale Wellen propagieren (b) und die Welle eines schwimmenden, bewegten Objektes (c). Aufgrund der Verwendung eines effizienten, zweidimensionalen FD-Ansatzes können hohe Auflösungen in Echtzeit simuliert werden.

Gebiete werden Gebiete betrachtet, in denen eine Interaktion zwischen Objekten und der Flüssigkeit oder eine Nutzerinteraktion stattfindet. Zusätzlich werden ambiente Wellen verwendet, so dass eine Simulation entsprechend dieser zwei Wellentypen separiert werden kann:

Typ	Dim.	Methode	Simulation
Ambiente Wellen	2D	Vorberechnung	Frequenzspektrum
Interaktive Oberflächenwellen	2D	FDM	Wellengleichung

Zur Simulation ambienter Wellen wird im Folgenden der Algorithmus von Tessendorf verwendet [Tes01]. Interaktive Wellen werden mit der Wellengleichung beschrieben (Abschnitt 2.2). Zur Berechnung wird im Folgenden eine FD-Methode verwendet — siehe Abbildung 5.2.

5.1.1.2. Diskrete Wellengleichung

Zur Diskretisierung der Wellengleichung wird im Folgenden die von Gomez [Gom00] vorgeschlagene Methode der zentralen Differenzen verwendet. In Hinblick auf eine GPU-basierte Implementierung wird ein zweidimensionales Feld $z^t_{i,j} = f^t(i,j)$ mit $i,j \epsilon \mathbb{N}$ und $0 \leq i,j \leq size$, Schrittweite $h = 1/size$ zum Zeitpunkt t verwendet. Damit ergibt sich mit $a = \frac{c^2 \Delta t^2}{h^2}$ folgende Diskretisierung der Wellengleichung (siehe auch Abschnitt 2.2):

$$z^{t+1}_{i,j} = a \cdot \sum_{\substack{k=i\pm1, l=j; \\ k=i, l=j\pm1}} z^t_{k,l} + (2-4a) \cdot z^t_{i,j} - z^{t-1}_{i,j}, \tag{5.1}$$

wobei Δt die Zeitschrittweite und c die Ausbreitungsgeschwindigkeit beschreiben. Auf diese Weise kann die Wellengleichung in einem quadratischen Gebiet simuliert

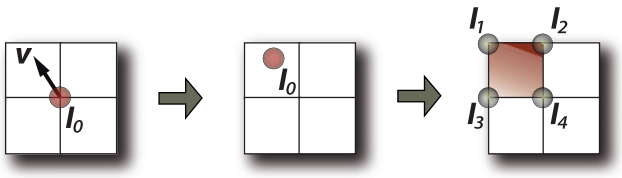


Abb. 5.3.: Numerische Diffusion. Eine beliebige Intensität $\mathbf{I}_0$ bewegt sich mit Geschwindigkeit $\mathbf{v}$. Nach der Translation wird der Intensitätswert auf die anliegenden Gitterpunkte verteilt.

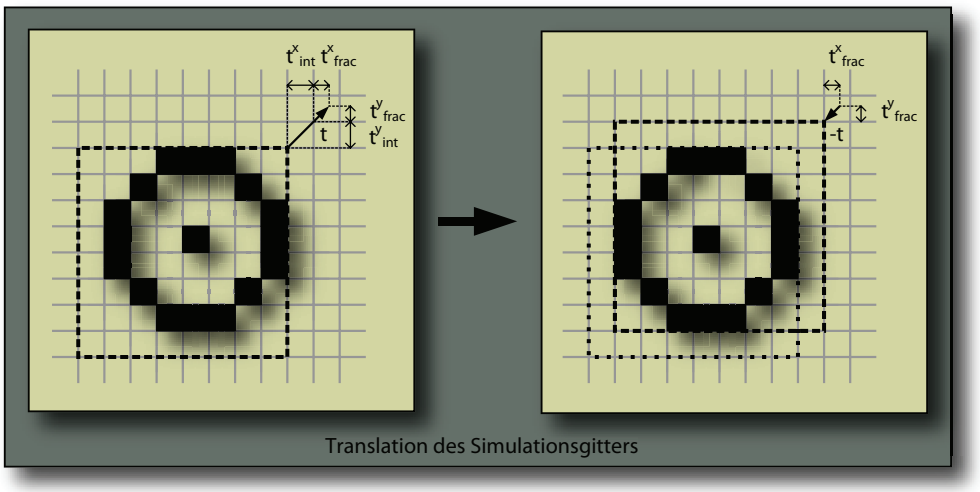


Abb. 5.4.: Das Simulationsgitter wird relativ zur unendlichen Ozeanoberfläche verschoben. Dazu wird der Translationsvektor $\mathbf{t}$ in einen diskreten und einen realen Anteil unterteilt.

werden.

Um das diskrete Gitter kontinuierlich zu verschieben, wird ein Translationsvektor $\mathbf{t}\epsilon\mathbb{R}^2$ definiert. Eine direkte Translation in $\mathbb{R}^2$ des diskreten Gitters würde aufgrund des notwendigen Glättungsschrittes in hoher numerischer Diffusion resultieren (Abb. 5.3). Die Translation $\mathbf{t}$ erfolgt in zwei Schritten. Zunächst wird eine Translation des Gitters im diskreten Raum durchgeführt, wobei keinerlei numerische Diffusion auftreten kann. Es folgt eine kontinuierliche Translation im Objektraum, wobei das Simulationsgitter nicht verändert wird. So wird eine numerische Diffusion vollständig vermieden.

Dafür wird $\mathbf{t}$ in den diskreten Anteil $\mathbf{t}_{int} = (t^x_{int}, t^y_{int})^T \epsilon \mathbb{N}^2$ und den kontinuierlichen Anteil $\mathbf{t}_{frac} = (t^x_{frac}, t^y_{frac})^T \epsilon \mathbb{R}^2$ mit $0 \leq \mathbf{t}^x_{frac}, \mathbf{t}^y_{frac} \leq 1$ unterteilt (Abb. 5.4). Die numerische Lösung der Wellengleichung hängt entsprechend Gleichung 5.1 vom aktuellen und vorhergehenden Zeitschritt ab. Der vorhergehende Schritt im diskreten Raum wird im Folgenden mit $\mathbf{t}^{old}_{int}$ bezeichnet.

Mit $m = i - t^x_{int}$, $n = j - t^y_{int}$, $o = m - t^{x\ old}_{int}$ und $p = n - t^{y\ old}_{int}$ ergibt sich folgende

modifizierte, diskrete Wellengleichung:

$$z_{i,j}^{t+1} = a \cdot \sum_{\substack{k=m\pm1,l=n;\\ k=m,l=n\pm1}} z_{k,l}^{t} + (2-4a) \cdot z_{m,n}^{t} - z_{o,p}^{t-1}. \tag{5.2}$$

Bei einem Zugriff auf das Simulationsgitter, wird die kontinuierliche Translation $\mathbf{t}_{frac}$ dem Zugriffsvektor hinzugefügt. Obwohl die eigentliche Translation des Simulationsgitters im diskreten Raum erfolgt, kann so dennoch eine kontinuierliche Translation des Gitters in Weltkoordinaten erzielt werden. Eine Skalierung des Simulation-Gitters in x oder y Richtung um s_x bzw. s_y wird mit einer Skalierung des Translationsvektors um $(1/s_x, 1/s_y)^T$ behandelt.

Stabilität Für explizite Lösungen der Wellengleichung, ergibt sich ein Stabilitätskriterium der Simulation. Die Simulation wird instabil und Wellenamplituden wachsen exponentiell wenn die folgende Bedingung verletzt wird [Gom00]:

$$\frac{c^2 \Delta t^2}{h^2} \leq \frac{1}{2}. \tag{5.3}$$

5.1.1.3. Verwendung mehrerer Gitter

Mit der im vorhergehenden Abschnitt beschriebenen Methode kann ein Simulationsgitter an jeder gegebenen Position innerhalb einer unendlichen Ebene positioniert werden. Zusätzlich kann das Gitter zum Beispiel einem schwimmenden Objekt folgen, so dass in der Umgebung eine permanente Wellensimulation erfolgen kann. Des Weiteren kann das Gitter auch dem View-Frustrum folgen, so dass in Blickrichtung immer eine Simulation stattfindet. Im Folgenden wird das beschriebene Verfahren auf die Verwendung mehrerer Gitter erweitert, um eine adaptive Simulation in mehreren Gebieten zu erreichen (Abb. 5.5).

Verschiedene Positionen In vielen Szenarien interagieren mehrere Objekte mit einer Flüssigkeitsoberfläche — wie zum Beispiel Boote. Zahlreiche Objekte mit einem einzigen Simulationsgitter zu simulieren, würde in der Regel bedeuten, ein sehr großes Gitter verwenden zu müssen. Dennoch besteht die Gefahr, dass einzelne Objekte dieses Gitter verlassen und somit keine Wellen mehr erzeugen können. Der oben beschriebene Ansatz zur Verwendung bewegter Gitter kann jedoch auf zahlreiche Objekte erweitert werden. Dabei wird jedes Gitter unabhängig bewegt. Verschiedene, bewegte Objekte können dann an verschiedensten Stellen interaktive Wellen erzeugen, ohne dass gleichzeitig eine globale Simulation durchgeführt werden muss. Die Höhe der überlagernden Wellen wird durch Addition übereinander liegender Gitter bestimmt, da physikalische Wellen interferieren.

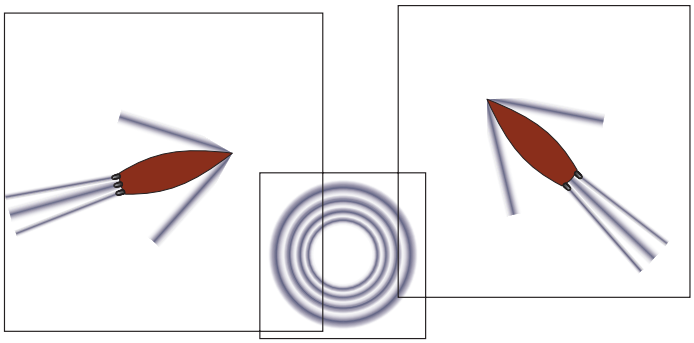


Abb. 5.5.: Prinzip der bewegten Simulationsgitter: Mehrere Wellensimulationen werden auf der unendlichen Wasserfläche verwendet.

Verschiedene Skalierungen Die Verwendung mehrere Simulationen kann nicht nur an verschiedenen Positionen genutzt werden. Es können auch verschiedene Skalierungen verwendet werden, um zum Beispiel verschiedene Detailstufen zu simulieren. So könnte zum Beispiel ein hochaufgelöster Regen-Layer (zweckmäßigerweise gekachelt) die entsprechenden filigranen, radial propagierenden Wellen erzeugen. Andere Layer können dennoch den im Vergleich dazu groben Wellenschlag eines bewegten, schwimmenden Objektes simulieren — unter Verwendung gröberer Gitter.

Eine weitere Anwendung verschiedener Skalierungen ist die Umsetzung einer LoD-Simulation. Die jeweiligen Gittergrößen können entsprechend der Bedeutung der Objekte für die aktuelle Szene gewählt werden. Objekte mit einem geringen Abstand zur Kamera sollten somit ein eher feineres Gitter erhalten, um mehr Details darstellen zu können. Objekte, die sich kamerafern bewegen, können eher ein gröberes Gitter erhalten, um Rechenleistung zu sparen. So wird eine adaptive Simulation und eine gute Performanz erzielt.

5.1.1.4. Ergebnisse und Diskussion

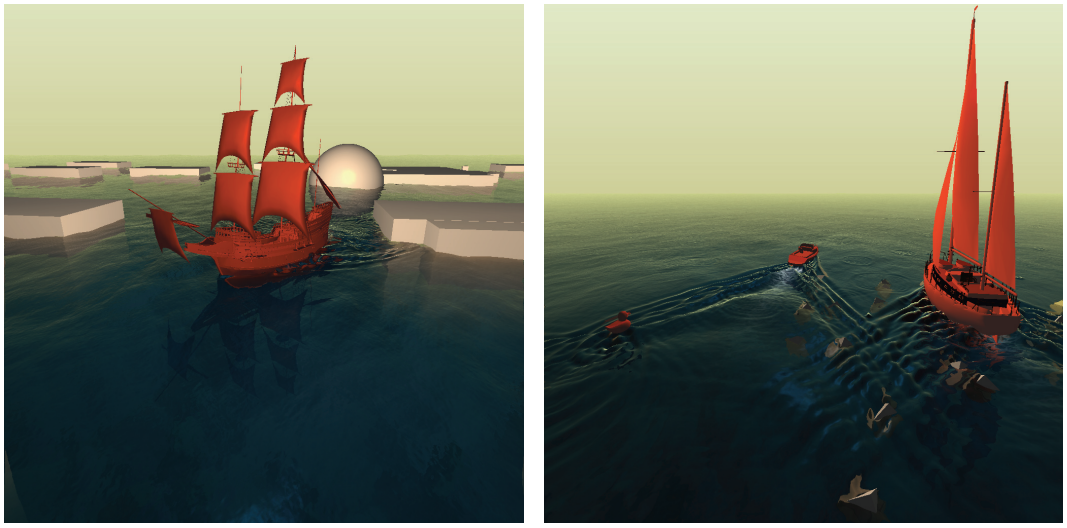
Die im Folgenden präsentierten Ergebnisse wurden auf einem Quad-Core Desktop PC mit einer 2,4 GHz Intel Q6600 CPU, 4 GB RAM und einer Graphikkarte basierend auf einer GeForce GTX 280 GPU gemessen. Die gezeigte, unoptimierte prototypische Implementierung benutzt lediglich einen CPU-Core, so dass die Performanz mit einer parallelisierten Implementierung noch erheblich verbessert werden kann. Die gemessenen Zeiten beinhalten die Flüssigkeitssimulation, die Oberflächenextraktion und die Darstellung. Die Messungen erfolgten bei einer Bildschirmauflösung von 1024×1024. In allen Beispielen wurden zwei Wellengleichungssimulationen verwendet, mit Auflösungen von 2048×2048 und 1024×1024, die auf der GPU simuliert werden. Die Schaum-Simulation (wird in Abschnitt 5.2.2 behandelt) wird in der prototypischen Implementierung mit einer Auflösung von 1024×1024 realisiert und ebenfalls auf der GPU ausgeführt. Alle verwendeten Texturen sind in Abbildung 5.8 am Ende dieses Abschnitts dargestellt.

Abb. 5.6.: Interaktives Motorboot.

Performanz Frameraten für verschiedene *Projected Grid*-Abtastungen (siehe Abschnitt 3.4.1) sind in Tabelle 5.1 gegeben. Die Verteilung der Laufzeit des entwickelten Prototyps für die verschiedenen Schritte bei einer hohen *Projected Grid*-Abtastung von 1×1 sind in Tabelle 5.2 dargestellt. Die gemessenen Frameraten erlauben eine vollständige Interaktion mit der Flüssigkeitsoberfläche, wie zum Beispiel die Wellenerzeugung bewegter Objekte. Ein Großteil der Rechenzeit wird für die Oberflächenextraktion mit der *Projected Grid*-Methode benötigt, da die Simulation selbst effizient auf der GPU durchgeführt werden kann. Der Aufwand der *Projected Grid*-Methode wächst quadratisch entsprechend der Viewport-Auflösung, da es sich um ein Bildraumverfahren handelt. Dementsprechend muss ein Kompromiss zwischen der abgetasteten Bildraumauflösung und den dazugehörigen Aliasing-Artefakten und der Laufzeit getroffen werden (vgl. dazu auch Abbildung 6.1, Kapitel 6). So kann die gleiche Simulation durch Änderung der Bildraumauflösung verschiedene Geschwindigkeiten annehmen. Entsprechend kann eine adaptive Darstellung bezüglich des Zielsystems erfolgen, ohne die Simulation ändern zu müssen. So können konstante Frameraten auf beliebigen Rechnern gewährleistet werden — in Abhängigkeit der dargestellten Qualität.

	1×1	2×2	5×5
Motorboot (3)	40,3	56,3	69,4
Korsar (3)	35,2	46,7	58,8
3 Boote (43)	32,2	43,0	56,8
Bojen (41)	26,6	34,9	40,9
Armada (40)	22,7	27,6	30,3

Tab. 5.1.: Laufzeitmessungen in *FPS* für verschiedene *Projected Grid*-Abtastungen. Die Anzahl der jeweilig verwendeten starren Körper ist in Klammern angegeben. Die Beispiele sind in den Abbildungen 5.6, 5.7 und 5.28 dargestellt.



(a) (b)

Abb. 5.7.: Beispiele: Korsar (a), 3 Boote (b).

Qualität Die beschriebene Echtzeit-Methode ermöglicht die Darstellung realistischer und großer Flüssigkeitsszenarien, mit Oberflächeninteraktion. Um eine effiziente Simulation zu erzielen, konzentriert sich die Methode auf die wichtigste visuelle Erscheinung von Flüssigkeiten: Oberflächenwellen. So kann eine verhältnismäßig detaillierte Simulation erzielt werden. Aufgrund der Beschränkung auf Oberflächenwellen, bietet sich die Methode für alle Szenarien an, in denen Strömungen und andere dreidimensionale Flüssigkeitseffekte vernachlässigt werden können. Im anschließenden Abschnitt wird erläutert, wie die Methode für zweidimensionale Strömungen erweitert werden kann. Die beschriebene Methode ist blickpunktabhängig und Simulationen können in spezifischen, wichtigen Regionen — die dynamisch sein können — definiert werden. Des Weiteren können verschiedene Simulationsauflösungen oder aber verschiedene Skalen verwendet werden. Die beschriebene Kopplung mit schwimmen-

	SKS	L&S	O&D	Gesamt
Motorboot	1,8	35,1	63,1	100,0
Korsar	2,0	29,3	68,7	100,0
3 Boote	2,7	23,7	73,6	100,0
Bojen	12,3	23,6	64,1	100,0
Armada	1,3	36,0	62,7	100,0

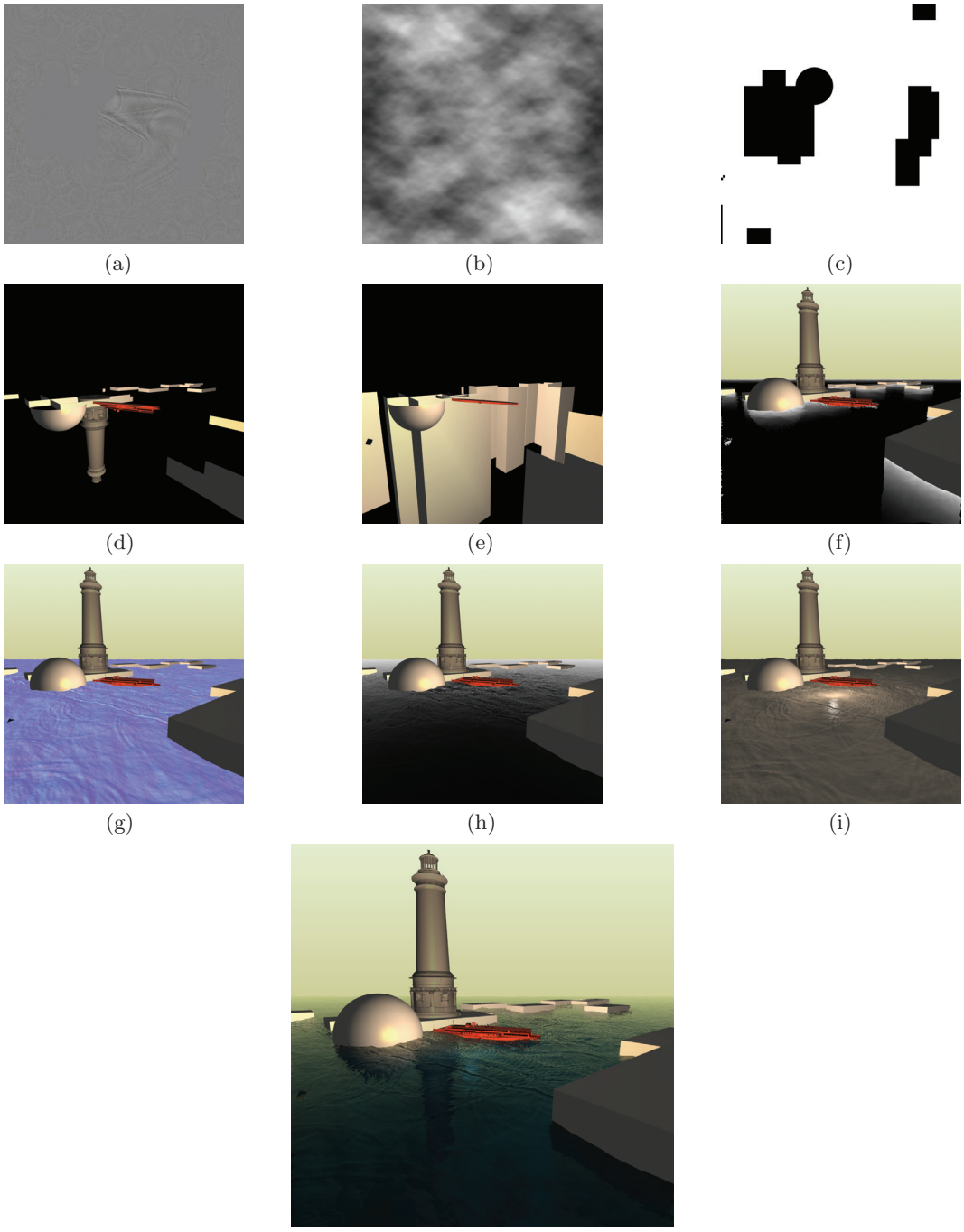
Tab. 5.2.: Verteilung der Berechnungszeit in Prozent für eine *Projected Grid*-Auflösung von 1×1. Starre Körper-Simulation (SKS), Flüssigkeits- und Schaumsimulation (L&S) und Oberflächenextraktion und Darstellung (O&D).

den Objekten ist physikalisch-basiert und resultiert in einer effizienten Simulation und Wellenerzeugung — verschiedene Objekte können verschiedene Wellen erzeugen.

Flüsse oder Seen können direkt mit der beschriebenen Methode dargestellt werden, da in Uferbereichen lediglich Kollisionsbedingungen gesetzt werden müssen. Die Simulation und Oberflächenextraktion kann in derartigen Szenarien noch erheblich effektiviert werden, da beide Prozesse lediglich in eingeschränkten Bereichen vonnöten sind — und nicht wie oben beschrieben, für eine scheinbar unendliche Fläche. Für Ozeane ist die Verwendung der Wellengleichung physikalisch lediglich eine Approximation, da die Wellenausbreitung in tiefen Gewässern aufgrund von Dispersion komplexer ist und die Methode Wellen konstanter Wellenlänge beschreibt. Dispersion kann nicht mit der beschriebenen zweidimensionalen Simulationsmethode beschrieben werden. Diese Einschränkung ist in Hinblick auf interaktive Umgebungen jedoch akzeptabel. Die beschriebene Methode ist für Echtzeitumgebungen entwickelt worden und soll keine hochdetaillierte Offline-Simulation ersetzen — wenngleich sie auch in einer Offline-Produktion im Rahmen eines Preview-Modes von Nutzen sein kann.

Vergleich mit der Wave Particles-Methode Da die *Wave Particles*-Methode [YHK07] ebenfalls eine Umgebung für große Flüssigkeitsflächen vorstellt, wird die Arbeit an dieser Stelle kurz mit der hier beschriebenen Methode verglichen (siehe auch Abschnitt 3.3.1). Ein fundamentaler Unterschied ist die verwendete Simulationsmethode: Anstatt der Verwendung von Wellenpartikeln verwendet die hier vorgestellte Methode einen FD-Ansatz, der das Huygenssche Prinzip simuliert. Unter Verwendung von Wellenpartikeln wird lediglich die Ausbreitung einer radial propagierenden Wellenfront beschrieben, in deren Inneren die Oberfläche glatt ist. Sie stellt also lediglich eine Approximation der Wellengleichung dar. Zudem kann Wellenbeugung nicht direkt mit der *Wave Particles*-Methode beschrieben werden. Aus diesen Gründen können mit der hier beschriebenen Methode detailliertere Oberflächen dargestellt werden. Die *Wave Particles*-Methode ist performant, jedoch ist die Performanz direkt abhängig von der Anzahl der dargestellten Wellen — die hier vorgestellte Methode besitzt bei der Verwendung von statischen Gittergrößen eine konstante Laufzeit und kann zudem effizient auf der GPU ausgeführt werden.

Verwendung einer Strömungssimulation Das hier beschriebene Verfahren der bewegten Gitter kann ebenso für zweidimensionale Gitter-basierte Strömungssimulationen verwendet werden. Diese Gitter müssen entsprechend der hier beschriebenen Methode ebenfalls um den integralen Translationsanteil verschoben werden. Bei der Überlagerung von Gittern muss der überlagerte Teil der beteiligten Gitter äquivalent sein, damit die Strömungen interagieren können. Dazu wird der jeweilige Randbereich aus dem jeweils anderen Gitter kopiert. So können auch Strömungen mit der beschriebenen Methode behandelt werden. Die gezeigte prototypische Implementierung unterstützt derzeit jedoch noch keine Strömungen.



(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

(j)

Abb. 5.8.: Die verwendeten Texturen zur Erzeugung des finalen Bildes. Wellensimulation (a), ambiente Wellen (b), Kollisionen-Map (c), Reflexions- und Brechungs-Map (d,e), Absorptions-Map (f), Normalen-Map (g), Fresnel-Term (h), Beleuchtung (i). Die Verwendung dieser Texturen führt zu dem gezeigten Ergebnis (j).

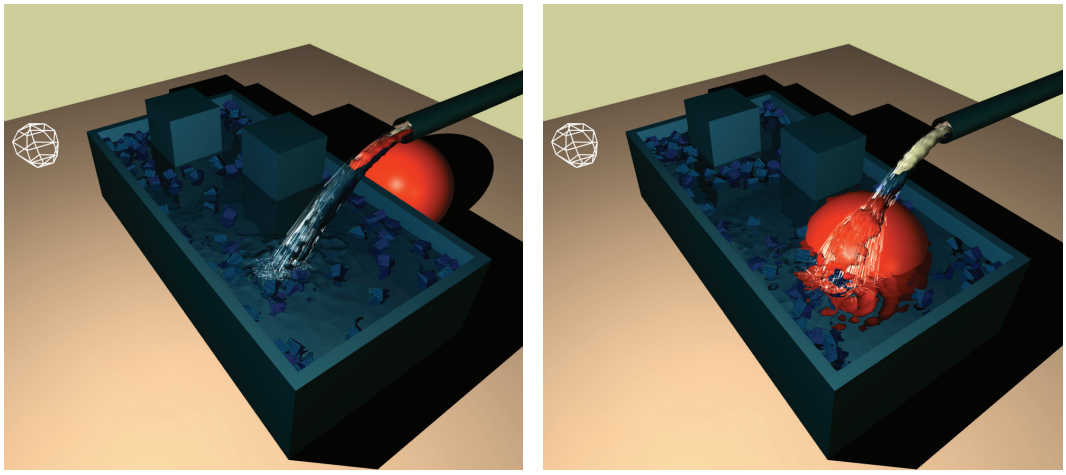
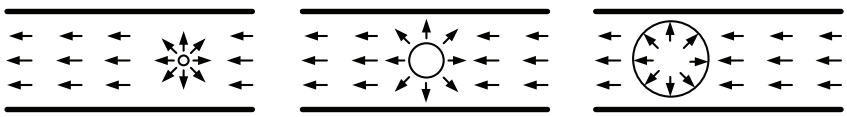


Abb. 5.9.: Interaktive Flüssigkeit, simuliert und dargestellt bei 33 FPS. Simulationsbasis ist die Verwendung einer Oberflächenwellensimulation und einer zweidimensionalen Strömungssimulation.

5.1.2. Kombinierte Oberflächen- und 2D Strömungssimulation: Wave Particles in strömenden Flüssigkeiten

Dieser Abschnitt beschreibt ein System, welches sowohl eine Oberflächenwellensimulation als auch eine 2D Strömungssimulation anwendet — zusätzlich zu einer ambienten Wellensimulation. Motivation für das im Folgenden beschriebene Verfahren ist die Erweiterung einer direkten Wellensimulation um einen Strömungsfluss. Diese Kombination ist geeignet für Umgebungen, in denen eine Strömungssimulation benötigt wird, jedoch eine dreidimensionale Strömung nicht notwendig ist — hingegen explizite Oberflächenwellen erfordert werden (wie zum Beispiel in einem Bach). Da dreidimensionale Strömungen für einen Betrachter kaum zu erkennen sind, wenn sich kein sichtbares Material innerhalb der Strömung bewegt, ist die Reduktion von drei auf zwei Dimensionen für viele Szenarien sinnvoll, da der Aufwand der Strömungssimulation erheblich gesenkt wird. Umgekehrt kann bei gleicher Performanz eine interaktive 2D Simulation eine wesentlich höhere Abtastung des Simulationsraumes und damit mehr Details erzielen, als eine 3D Simulation mit gleichgroßer Oberfläche. Des Weiteren kann ein derartiges System die Ausbreitung von Wellen in schnell strömenden Flüssigkeiten beschreiben — inklusive des Effektes der Deformation von Wellen bei starken Strömungen. Bei einem strömenden Fluss würde sich zum Beispiel die radial propagierende Welle eines eintauchenden Objektes mit der Strömung bewegen:



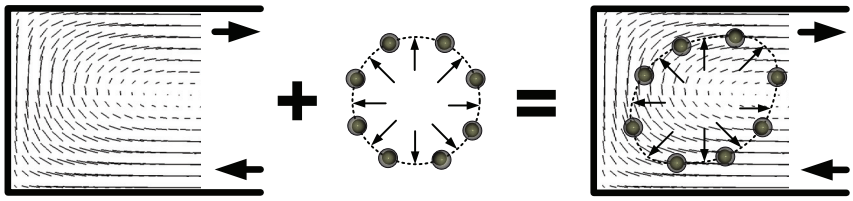


Abb. 5.10.: Die Grundidee: Eine 2D Strömungssimulation (hier: Ein- und Ausfluss, links) wird kombiniert mit einer Oberflächenwellensimulation (hier: Zirkulare Welle, mitte) — es ergeben sich Wellendeformationen entsprechend der Strömung (rechts).

Zusätzlich zur Wellensimulation können schwimmende Objekte innerhalb einer Strömung bewegt und externe Strömungen eingebracht werden. So können Flüssigkeitsoberflächen vollständig dargestellt werden. Unter Verwendung einer Massepunkt-Simulation können zusätzlich dreidimensionale Effekte approximiert werden, wie zum Beispiel eine Fontäne (Abb. 5.9).

5.1.2.1. Simulation

Die Strömung wird mit Hilfe einer finiten Differenzenmethode simuliert. Zur Lösung der zweidimensionalen Navier-Stokes Gleichungen kann prinzipiell jede Simulationsmethode verwendet werden. Die *Stable Fluids*-Methode von Jos Stam [Sta99] bietet jedoch eine ausgezeichnete Balance zwischen Genauigkeit, Stabilität und Performanz (siehe Abschnitt 3.3.2). Die zu lösende Poisson Gleichung kann mit dem iterativen Gauss-Seidel Verfahren gelöst werden. Kollisionsobjekte innerhalb des Flusses können über Zwangsbedingungen als *slip*, *no slip* oder hybrid definiert werden.

Die Oberflächensimulation erfolgt entsprechend der Wellengleichung. Die konkrete Simulation kann mit Hilfe der *Wave Particles*-Methode erfolgen, die Yuksel et al. eingeführt haben [YHK07]. Die Methode stellt einen Ansatz dar, um eine zweidimensionale Wellengleichung Partikel-basiert zu approximieren (siehe auch Abschnitt 3.3.1). Vorteil einer Partikel-basierten Simulation in diesem Fall ist, dass keine Diffusion aufgrund der Strömung auftreten kann. Die verwendeten Simulationen des im Folgenden beschriebenen Ansatzes sind zusammenfassend in folgender Tabelle dargestellt:

Typ	Dim.	Simulationsmethode	Simulierte Gleichung
Strömung	2D	FDM	N.-S. Gleichungen
Oberflächenwellen	2D	Wave Particles	Wellengleichung

Mit Hilfe einer Kopplung der Oberflächenwellen- und der Strömungssimulation können jegliche, als Höhenfeld darstellbare Flüssigkeitseffekte repräsentiert werden. Zusammenfassend ist die Grundidee der Kopplung beider Simulationen in Abbildung 5.10 dargestellt.

5.1.2.2. Kopplung der Simulationen

Ziel der Kopplung ist es in schnell fliessenden Strömungen die Wellenbewegung auf dem bewegten Medium der Flüssigkeit darzustellen. Die Oberflächenwellen repräsentierenden Partikel bewegen sich unabhängig voneinander — lediglich die jeweils zugehörigen Höhenwerte addieren sich aufgrund von Interferenz. Da sich die Oberflächenwellen mit der Strömung bewegen, jedoch selber keinerlei Strömung generieren, muss lediglich eine Einweg-Kopplung realisiert werden:

$$\text{Strömung} \rightarrow \text{Oberflächenwellen.}$$

Somit kann die Strömungssimulation autonom ausgeführt werden und die Wellensimulation kann anschließend unter Verwendung der Ergebnisse der Strömungssimulation erfolgen. Vorteil eines derartigen Ansatzes, der auf zwei aufeinanderfolgenden Simulationen basiert, ist die effiziente Simulation mit Hilfe zweier optimaler Simulationsmethoden. So können zum Beispiel für beide Simulationen unterschiedliche Auflösungen verwendet werden — in Abhängigkeit des gewünschten Detailgrades der jeweiligen Simulation. Bei Strömungen ist in der Regel nicht notwendigerweise eine feine Diskretisierung vonnöten, so dass die Strömung mit niedriger Auflösung simuliert und zwischen den Gitterpunkten interpoliert werden kann. So können die wichtigsten visuellen Eigenschaften von Flüssigkeiten — die Oberflächenwellen — mit hoher Auflösung berechnet und dargestellt werden. Nachfolgend wird die Kopplung im Detail beschrieben.

In der *Wave Particles*-Methode werden Oberflächenwellen von autonomen Partikeln beschrieben. Ein Kopplung mit der Strömungssimulation kann somit erzielt werden, indem die Geschwindigkeiten der Wellenpartikel entsprechend der umliegenden Strömungen angepasst werden. Da der Strömungsfluss aufgrund der verwendeten FD-Methode in einem diskretisierten Gitter der Größe $n \times m$ vorliegt, kann die Strömungsgeschwindigkeit $\mathbf{v}(\mathbf{x})$ für jeden gegebenen Ort $\mathbf{x} \in \mathbb{R}^2$ mit $i < x_x < (i+1)$ und $j < x_y < (j+1)$ mit Hilfe einer bilinearen Interpolation bestimmt werden:

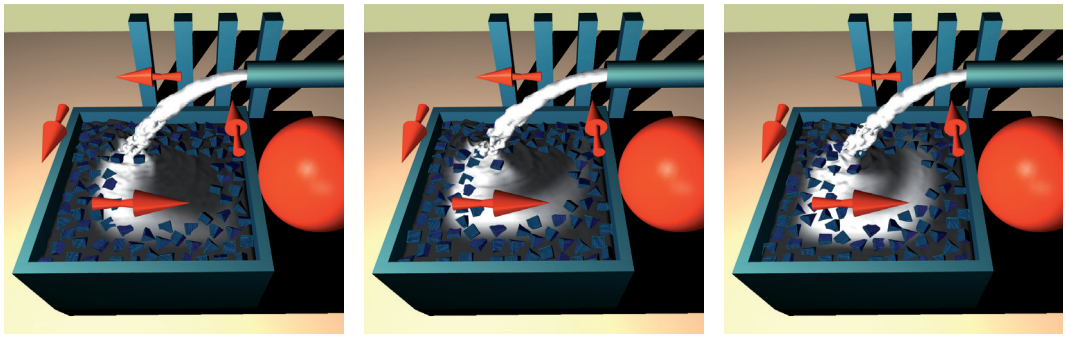
$$\mathbf{v}(\mathbf{x}) \approx \begin{pmatrix} i+1-x_x \\ x_x - i \end{pmatrix}^T \begin{pmatrix} \mathbf{v}_{i,j} & \mathbf{v}_{i,j+1} \\ \mathbf{v}_{i+1,j} & \mathbf{v}_{i+1,j+1} \end{pmatrix} \begin{pmatrix} j+1-x_y \\ x_y - j \end{pmatrix}. \tag{5.4}$$

Mit diesem Vorgehen werden Aliasing-Artefakte bei der Bewegung von Wellenpartikeln wegen der Strömung verhindert. Die Position $\mathbf{p}_n$ jedes Partikels n wird explizit in Abhängigkeit der so bestimmten Strömungsgeschwindigkeit adaptiert ($\triangle t$: Zeitschritt):

$$\mathbf{p}_{n+1} = \mathbf{p}_n + \triangle t \cdot \mathbf{v}(\mathbf{p}_n). \tag{5.5}$$

5.1.2.3. Unterschiedliche Flüssigkeiten

Das beschriebene Vorgehen ermöglicht die Repräsentation von Flüssigkeiten, die sich vermischen. Abbildung 5.11 demonstriert dies an einem Wirbel schwarzer



(a) (b) (c)

Abb. 5.11.: Beispiel zur Mischung von Flüssigkeiten an unterschiedlichen Zeitpunkten: Eine weiße Flüssigkeit wird mit einer schwarzen Flüssigkeit vermischt. Die Flüssigkeit im Becken strömt dabei zirkular.

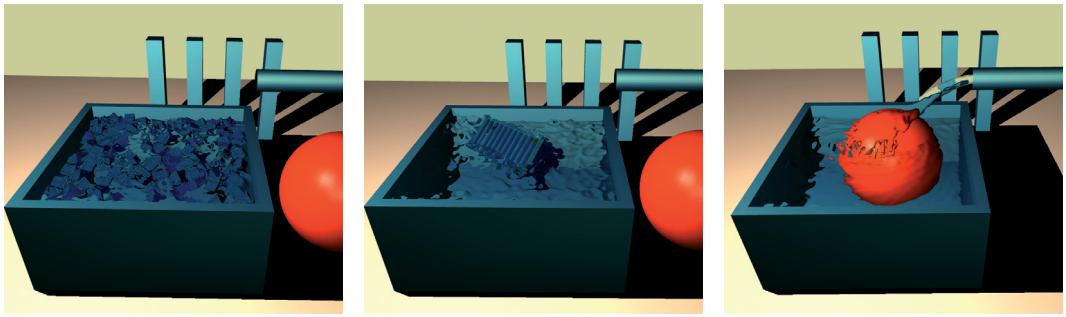
Flüssigkeit, in das eine weiße Flüssigkeit eingelassen wird. Zum einen entstehen am Eintreffpunkt der Fontäne Oberflächenwellen, die mit der im vorhergehenden Absatz erwähnten Methode erzeugt werden. Zum anderen propagieren die zu vermischenden Anteile entsprechend der Strömungssimulation. Da die beschriebene Methode jedoch eine zweidimensionale Simulation verwendet, handelt es sich lediglich um eine zweidimensionale Mischung, so dass zum Beispiel die Mischung einer transparenten und einer opaken Flüssigkeit nicht in drei Dimensionen dargestellt werden kann.

5.1.2.4. Ergebnisse und Diskussion

Die vorgestellte Methode wurde unter Verwendung von OpenGL 2.0 und der zugehörigen Shadersprache GLSL in C++ implementiert. Laufzeitmessungen werden in Tabelle 5.3 gegeben. Die Messungen erfolgten auf einer Dual-Core 2,6 GHz AMD Athlon 64 CPU mit 2 GB RAM und einer auf dem ATI Radeon x1900 Chip basierenden GPU (512MB). Die Implementierung erfolgte unter Verwendung von *Multi-Threading*, wobei die Parallelisierung folgendermaßen realisiert wurde:

Core 1:	Oberflächenextraktion, Interaktion, Darstellung
Core 2:	N.-S. Gleichungen, Wellengleichung, Starre Körper, Fontäne

Während der erste Core den aktuellen Zeitschritt darstellt, simuliert der zweite Core den nächsten Zeitschritt. Die Simulationsmethoden wurden CPU-basiert realisiert, so dass die Performanz wesentlich verbessert werden kann, wenn sowohl für die Oberflächensimulation als auch für die Strömungssimulation eine GPU-basierte Implementierung verwendet werden würde. So könnten auch höhere Gitterauflösungen verwendet werden. Jedoch sei erwähnt, dass bereits die verhältnismäßig niedrigen Auflösungen der verwendeten Strömungssimulation aufgrund der zusätzlichen Verwendung einer Wellensimulation zu relativ detaillierten Ergebnissen führen.



(a) (b) (c)

Abb. 5.12.: Beispiele der vorgestellten Methode mit Objekt-Interaktion.

Die Vorteile der beschriebenen Kombination der Simulationsmethoden und werden im Folgenden zusammengefasst dargestellt:

- Physikalisch-basierte Strömungssimulation,
- Physikalisch-basierte Wellensimulation,
- Strömungsinduktion (zum Beispiel Boot, Turbine),
- Strömungsreaktion (zum Beispiel Objekte, Blätter auf einem Fluß),
- Automatischer, globaler Fluss (zum Beispiel Bach),
- Interaktive Wellenerzeugung (zum Beispiel Regen, bewegte Objekte).

Verwendung einer FD-Methode zur Lösung der Wellengleichung Die beschriebene Kopplung von Strömung und Oberflächenwellen lässt sich auch mit Hilfe einer FDM statt der *Wave Particles*-Methode zur Lösung der Wellengleichung realisieren. Entsprechend der Geschwindigkeiten des Strömungsgitters werden die Amplituden der Oberflächenwellen bewegt. Der entscheidende Nachteil des Gitter-basierten Ansatzes im Vergleich zu dem beschriebenen Partikel-basierten Ansatz besteht in der hohen numerischen Diffusion. Partikel hingegen können nicht diffundieren. Da die Intensitätswerte der Wellensimulation mit reelen Geschwindigkeiten verschoben werden, muss anschließend eine Extrapolation durchgeführt werden, um die Intensitäten wieder auf die benachbarten Gitterpunkte zu verteilen. Damit werden die Intensitäten numerisch diffundiert Da dieses Vorgehen für jeden einzelnen Simulationsschritt notwendig ist, ist die numerische Diffusion in der Simulation hoch — vor allem in interaktiven Umgebungen, in denen ein großer Gitterabstand verwendet wird. Somit laufen die simulierten Wellen auseinander. Des Weiteren könnte ein direkter Vergleich mit der Flachwassergleichung Aufschluss über die Qualität der Ergebnisse der Methode geben.

Abbildung	NS-GG	WP (max)	O-GG	FPS
5.9	32×64	40000	128×256	33
5.11	32×32	20000	128×128	23
5.12a	32×32	20000	128×128	74
5.12a	64×64	20000	128×128	70
5.12a	128×128	20000	128×128	27
5.12a	256×256	40000	256×256	4
5.12b	32×32	20000	128×128	76
5.12c	32×32	20000	128×128	23

Tab. 5.3.: Laufzeitmessungen der vorgestellten Methode in verschiedenen Szenarien (in FPS). Gittergröße der Navier-Stokes Simulation (NS-GG), maximale Anzahl verwendeter *Wave Particles* (WP), Oberflächengittergröße (O-GG), Beispiele 5.11 und 5.12a verwenden 100 starre Körper.

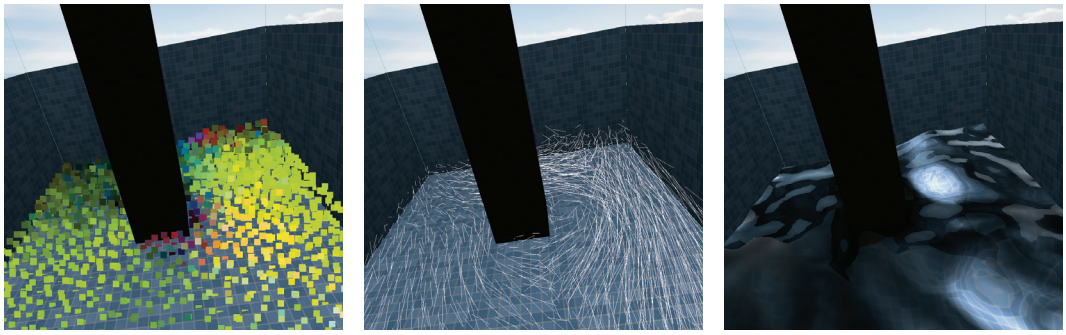
Verwendung einer 3D-Strömungssimulation Falls dreidimensionale Strömungen für ein gegebenes Szenario erforderlich sind, wie zum Beispiel das Vermischen zweier Flüssigkeiten in 3D, so kann auch eine dreidimensionale Strömungssimulation verwendet werden. Dabei muss jedoch nicht notwendigerweise eine Oberflächenextraktion erfolgen, wie es bei im nächsten Abschnitt 5.1.3 beschriebenen Methode erfolgt. Wenn zum Beispiel ein Aquarium mit internen Strömungen und keiner bewegten freien Oberfläche dargestellt werden soll, so kann die Strömungssimulation im Volumen erfolgen und die Oberflächenwellen werden anschließend wie oben beschrieben hinzugefügt.

5.1.3. Kombinierte Oberflächen- und 3D Strömungssimulation

Eine dreidimensionale Strömungssimulation von Flüssigkeiten in Echtzeitumgebungen führt in der Regel zu verhältnismäßig groben Ergebnissen. Aufgrund der Komplexität können lediglich niedrige Gitterauflösungen in Euler-Verfahren oder wenige Partikel in Lagrange-Verfahren verwendet werden. Somit erscheint die resultierende Oberfläche oftmals sehr glatt, mit nur geringen Oberflächendetails.

In der Praxis ist die wahrnehmbare Oberfläche für den Betrachter die wichtigste Eigenschaft der visuellen Repräsentation. Eine dreidimensionale Strömung hingegen kann nicht direkt wahrgenommen werden. Erst wenn sich etwas innerhalb der Strömung bewegt oder sich die Oberfläche verändert, kann sie indirekt wahrgenommen werden. Diese Tatsache ist in Abbildung 5.13 dargestellt: Der dreidimensionale Fluss (Abb. 5.13a,b) kann nach der Oberflächenextraktion nicht direkt wahrgenommen werden (5.13c). Um der visuellen Wichtigkeit der Flüssigkeitsoberfläche Rechnung zu tragen, sollte die Oberfläche angemessen simuliert und abgetastet werden, um detaillierte Ergebnisse zu erhalten.

In diesem Abschnitt wird ein Verfahren beschrieben, welches die Oberflächendetails einer dreidimensionalen Simulation in Echtzeit wesentlich verfeinert — durch Verwendung einer zusätzlichen Oberflächenwellensimulation. Des



(a) (b) (c)

Abb. 5.13.: Prinzip einer SPH-Simulation. Das Flüssigkeitsvolumen wird mit Partikeln diskretisiert. Diese werden zur Lösung der Navier-Stokes Gleichungen verwendet. In (a) sind die Geschwindigkeiten und ihre Orientierungen entlang der x,y und z-Achse zur Verdeutlichung farbkodiert. In (b) sind die Geschwindigkeiten entsprechend ihres Betrages und der Richtung gezeigt. Die extrahierte Oberfläche ist in (c) dargestellt. Aufgrund der geringen Anzahl verwendeter Partikel ist die SPH-generierte Oberfläche undetailliert und glatt.

Weiteren erscheinen Flüssigkeitssimulationen in Echtzeitumgebungen aufgrund größerer Simulationszeitschritte oft unnatürlich gedämpft oder viskos. Auch diese unerwünschte Eigenschaft wird mit der vorgestellten Methode verringert. Die Methode verwendet eine Höhenfeld-basierte Repräsentation der dreidimensionalen Strömungsdaten, auf die die Oberflächenwellen projiziert werden (siehe Beispiel in Abbildung 5.14). Somit müssen Partikel, die das Volumen verlassen, explizit behandelt werden, da sie nicht im Rahmen eines Höhenfeldes repräsentiert werden können.

5.1.3.1. Simulation

Zur dreidimensionalen Strömungssimulation wird im Folgenden eine SPH-Simulation verwendet. Die Verwendung einer FDM-Simulation ist jedoch ebenfalls möglich und wird in Abschnitt 5.1.3.4 diskutiert. Zur Simulation der Oberflächenwellen wird die Wellengleichung verwendet und mit einer FD-Methode gelöst.

Die Simulation mit der SPH-Methode führt in Echtzeitumgebungen zu Flüssigkeiten, die aus verhältnismäßig wenigen Partikeln bestehen. So müssen zum Beispiel bei der Darstellung geeignete Potentialfunktionen gewählt werden, damit der Abstand zwischen Partikeln visuell unsichtbar bleibt. Dennoch führt diese Unterdiskretisierung zu dem Effekt, dass detaillierte Flüssigkeitseigenschaften prinzipiell nicht repräsentiert werden können. Dazu gehören physikalische und visuelle Eigenschaften. So können Wellen geringer Amplitude oder Wellenlänge nicht mit wenigen Partikeln repräsentiert werden. Zudem führt die Dämpfung der SPH-Simulation in interaktiven Umgebungen dazu, dass Oberflächenwellen unrealistisch schnell gedämpft werden. Des Weiteren kann eine Kopplung einer SPH-Simulation

Abb. 5.14.: Beispiel: Echtzeit-Wasser-Umgebung — simuliert und dargestellt mit der präsentierten Methode (30 FPS). Mit dem Volumen der Flüssigkeit kann interagiert werden.

bestehend aus wenigen Partikeln und zum Beispiel einem schwimmenden Objekt in Echtzeitumgebungen lediglich tieffrequente Wellenzüge darstellen. Hochfrequente Wellen könnten mit der SPH-Methode lediglich unter Verwendung eines Vielfachen der Partikel simuliert werden. Derartige Simulationen könnten dann jedoch nicht mehr in Echtzeit ausgeführt werden. Die Repräsentation hochfrequenter Wellen werden jedoch mit einer zweidimensionalen Lösung der Wellengleichung ermöglicht. Durch die Verwendung zweier Simulationen kann so eine detaillierte Flüssigkeitssimulation erzielt werden, die eine vollständige Strömungssimulation mit einschließt – dabei jedoch ebenfalls detaillierte Oberflächenwellen darstellen kann. Die verwendeten Verfahren sind zusammenfassend in Abbildung 5.15 und in folgender Tabelle dargestellt — siehe zu den einzelnen Verfahren auch Abschnitte 2.1, 2.2, 3.3.1 und 3.3.3:

Typ	Dim.	Methode	Simulierte Gleichung
Strömung	3D	SPH	Navier-Stokes Gleichungen
Oberflächenwellen	2D	FDM	Wellengleichung

5.1.3.2. Zeitschritte

Da die Gitter-basierte Oberflächensimulation auf Basis der Wellengleichung der in Gleichung 5.3 gegebenen Stabilitätsbedingung für eine stabile Simulation unterliegt, können lediglich kleine Zeitschritte $\triangle t_{\mathrm{WG}}$ verwendet werden. Andererseits erzielen erst hohe Gitterauflösungen detaillierte Ergebnisse, so dass die Wellen lediglich langsam propagieren. Aus diesem Grund ist die Simulation mehrerer Zeitschritte innerhalb eines Darstellungsschrittes sinnvoll. Die dreidimensionale Strömungssimulation unter Verwendung der SPH-Methode hingegen ist sehr stabil — dabei jedoch aufwendiger als die Oberflächensimulation. Somit können größere Zeitschritte $\triangle t_{\mathrm{N\text{-}S}}$ für diese Simulation gewählt werden, so dass beide Simulationen im Rahmen ihrer Möglichkeiten effizient und adaptiv ausgeführt werden können.

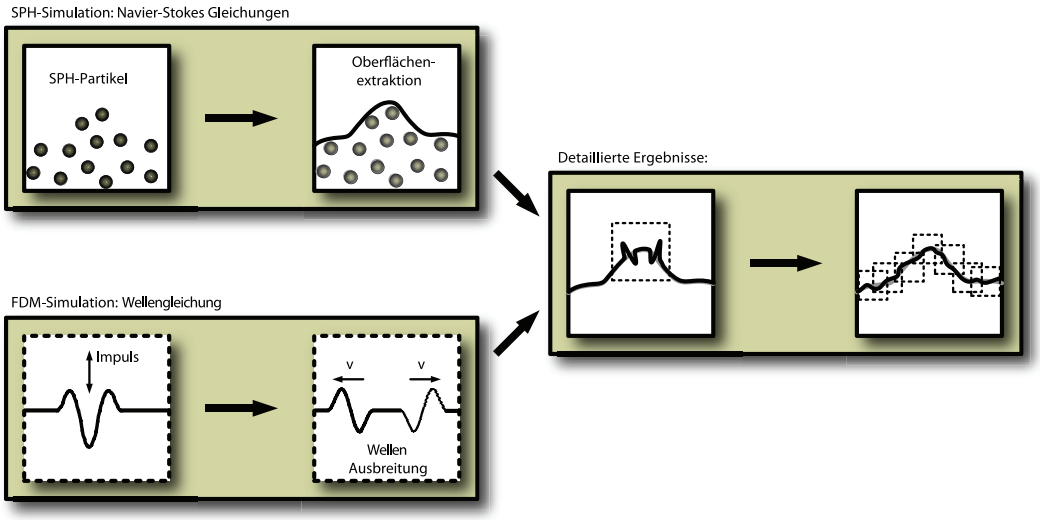


Abb. 5.15.: Prinzip: Die globale Strömung wird mit Hilfe einer SPH-Simulation bestimmt (oben, 2D Querschnitt). Eine zusätzliche Oberflächensimulation ergänzt Oberflächendetails (unten, 1D Beispiel). In der Kombination ergeben sich detailliert Oberflächen (rechts).

Zur zeitlichen Synchronisation beider Simulationen wird das Verhältnis beider Simulationszeitschritte $\triangle t_{\mathrm{WG}}$ und $\triangle t_{\mathrm{N\text{-}S}}$ auf ein integrales Verhältnis N_t eingeschränkt:

$$N_t = \frac{\triangle t_{\mathrm{N\text{-}S}}}{\triangle t_{\mathrm{WG}}}. \tag{5.6}$$

Somit werden N_t Oberflächensimulationsschritte pro Volumensimulationsschritt ausgeführt:

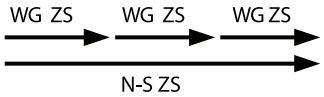


Abb. 5.16.: Beispiel zur Simulations-Synchronisation: Zeitschritt (ZS), $N_t = 3$. In diesem Beispiel wird die Wellengleichung für drei Zeitschritte simuliert, während die Strömungssimulation lediglich einen Zeitschritt simuliert.

5.1.3.3. Oberflächenkombination

Die Flüssigkeitsoberfläche muss aus beiden Simulationen extrahiert werden. Dazu wird ein Höhenfeld-basierter Ansatz verwendet. Da die Kombination der Oberflächen explizit für die hier beschriebene Methode entwickelt wurde, erfolgt die Beschreibung aus Gründen der Systematik hier und nicht in Kapitel 6 (Oberflächenextraktion). Die Erzeugung der Höhenfelder selbst hingegen wird in Kapitel 6 beschrieben, da diese allgemeingültig ist.

Die zweidimensionale Wellengleichung besitzt typischerweise eine höhere Oberflächenauflösung als die dreidimensionale Strömungssimulation. Prinzipiell könnte die Flüssigkeitsoberfläche der SPH-Simulation mit der gleichen Auflösung wie die der Oberflächensimulation extrahiert werden. Da die Oberfläche der Strömungssimulation jedoch verhältnismäßig undetailliert ist, ist es aus Effizienzgründen sinnvoll, die SPH-Oberfläche mit einer niedrigeren Auflösung zu generieren — entsprechend den repräsentierten Details.

Zunächst wird die Einschränkung eingeführt, dass die Oberflächenauflösung der Lösung der Wellengleichung ein ganzzahlieges Vielfaches der Oberflächenauflösung der Lösung der SPH-Simulation ist — vergleichbar zu dem Vorgehen im vorhergehenden Abschnitt zur Oberflächenerzeugung. So kann eine schnelle Kombination erfolgen, da keine aufwendigen Fließkommazahlenoperationen durchgeführt werden müssen. Somit ergibt sich für die aus der SPH-Simulation generierte Oberfläche der Größe $X_{\mathrm{SPH}} \times Y_{\mathrm{SPH}}$ und der detaillierteren Oberfläche der Wellengleichung der Größe $X_{\mathrm{WG}} \times Y_{\mathrm{WG}}$ im Verhältnis der diskrete Wert $N_{\mathrm{Oberfläche}}$:

$$N_{\mathrm{Oberfläche}} = \frac{X_{\mathrm{WG}}}{X_{\mathrm{SPH}}} = \frac{Y_{\mathrm{WG}}}{Y_{\mathrm{SPH}}}. \tag{5.7}$$

Diese Einschränkung ist keine große Beschränkung: Aufgrund der erfolgenden Glättung, ist die genaue Größe des Höhenfeldes frei wählbar und kleine Veränderungen in der Größe sind nicht oder zumindest kaum erkennbar. Anschließend wird das niedrig aufgelöste SPH-Höhenfeld bilinear auf die Größe des Höhenfeldes der Wellengleichung extrapoliert und beide können direkt überlagert werden — siehe Abbildung 5.17.

5.1.3.4. Ergebnisse und Diskussion

Die präsentierte Methode wurde unter Verwendung von OpenGL 2.0 und der zugehörigen Shadersprache GLSL in C++ implementiert. Laufzeitmessungen werden in Tabelle 5.4 gegeben und erreichen in der Regel Echtzeitperformanz bei einer Darstellungsauflösung von 950×950. Die Messungen erfolgten auf einer Dual-Core 2,6 GHz AMD Athlon 64 CPU mit 2 GB RAM und einer auf dem ATI Radeon x1900 Chip basierenden GPU (512MB). Die Implementierung erfolgte unter Verwendung von *Multi-Threading*. Dabei simuliert ein Core die Physik mit SPH und behandelt Nutzereingaben und der andere Core löst die Wellengleichung, extrahiert die Oberfläche, kreiert Kaustiken mit dem in Abschnitt 7.1 beschriebenen Verfahren und stellt die Szene dar. Die Parallelisierung ist somit folgendermaßen realisiert:

Core 1:	SPH-Simulation, Interaktion
Core 2:	Wellengleichung, Oberflächenextraktion, Kaustiken, Fontäne, Darstellung

Die Performanz kann unter Verwendung von GPU-basierten Implementierungen noch verbessert werden - so wie es bei der in 5.1.1 beschriebenen Methode reali-

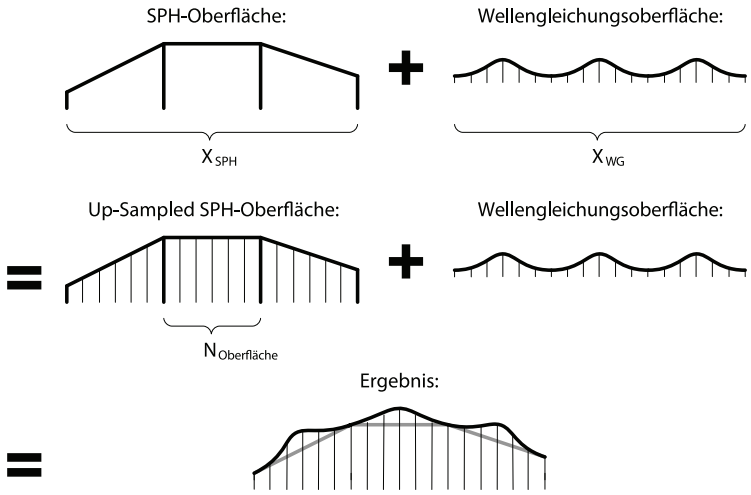


Abb. 5.17.: Kombination der aus der SPH-Simulation und der Oberflächensimulation resultierenden Höhenfelder verschiedener Auflösungen.

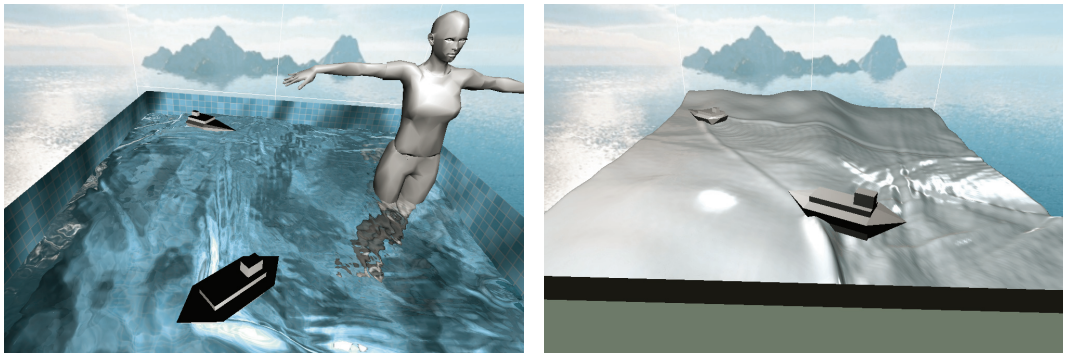
siert wurde. Allgemein hängt die Laufzeit von folgenden Parametern ab:

- Anzahl verwendeter SPH-Partikel,
- $X_{\mathrm{SPH}} \times Y_{\mathrm{SPH}}$ (Oberflächenauflösung),
- $X_{\mathrm{WG}} \times Y_{\mathrm{WG}}$ (Simulations- und Oberflächenauflösung).

Messungen zeigen, dass 40–70% der Laufzeit für die Simulation mit der SPH-Methode verwendet werden — bei weniger als 4000 verwendeten Partikeln in der gezeigten prototypischen Implementierung. Somit lassen sich Oberflächen-Details prinzipiell kaum darstellen. Die hier vorgestellte Technik der Kombination aus 3D Strömungssimulation und 2D Oberflächensimulation löst dieses Problem und erzielt detaillierte Oberflächen bei guter Performanz — siehe zum Beispiel Abbildung 5.19 und 5.18. Zusammenfassend liegen die Vorteile der präsentierten Methode in

Beispiel (Abb.)	N-S (SPH) (Anz. Part.)	N-S (SPH) ($X_{\mathrm{SPH}} \times Y_{\mathrm{SPH}}$)	WG (FDM) ($X_{\mathrm{WG}} \times Y_{\mathrm{WG}}$)	Cores (Anz.)	FPS (1/s)
5.2a,b,c	-	-	300 × 300	1	46
5.13c	2000	50 × 50	-	2	102
5.14	2000	50 × 50	400 × 400	2	30
5.18a	2000	50 × 50	200 × 200	2	85
5.18b	2000	50 × 50	200 × 200	2	93
5.19b,c	2000	50 × 50	200 × 200	2	75

Tab. 5.4.: Laufzeitmessungen des präsentierten Algorithmus für die gegebenen Beispiele (in FPS). Gegeben sind die Anzahl der SPH-Partikel, die Gittergrößen des SPH- und des Wellengleichungshöhenfeldes und die Anzahl verwendeter *Cores*.



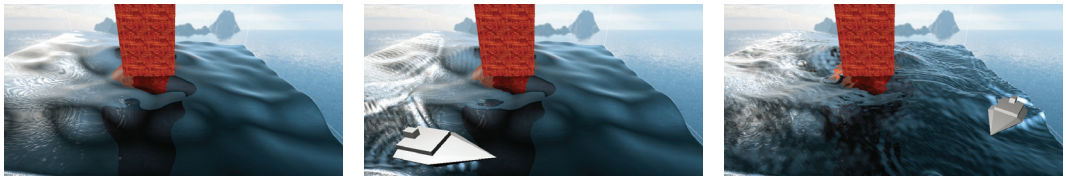
(a) (b)

Abb. 5.18.: Pool Szene mit Kaustiken (a). Verwendung eines Phong-Shaders zur Darstellung (b).

- Volumen-Interaktionen (zum Beispiel Bewegung eines Wasserglases, Objekte),
- Oberflächen-Interaktionen (zum Beispiel Regen, bewegte Objekte),
- globale Strömungen (zum Beispiel Bach) und in der
- Bewegungen von Objekten mit der Strömung (zum Beispiel Blätter),

Die niedrige Viskosität realen *Wassers* kann zum Beispiel kaum in Echtzeit mit einer Navier-Stokes-basierten Simulation erzielt werden — wegen großer Zeitschritte und dementsprechend hohen Dämpfungen. Deshalb ähneln Flüssigkeiten in Echtzeitumgebungen in ihren Bewegungen zum Beispiel oftmals eher Öl als Wasser. Dieser Nachteil wird mit der vorgestellten Methode verringert, da die Viskosität aufgrund der detaillierten Oberflächenwellen mit niedriger Dämpfung niedrig erscheint. Auf der anderen Seite können Flüssigkeiten mit hoher Viskosität direkt repräsentiert werden, indem der Simulationszeitschritt verringert und die Dämpfung vergrößert wird. Ein Einschränkung liegt in der Höhenfeld-basierten Darstellung der dreidimensionalen Simulation — so lassen sich dreidimensionale Effekte, obwohl sie simuliert werden, nicht direkt darstellen. Ablösende Flüssigkeitspartikel zum Beispiel müssen explizit behandelt werden.

Verwendung anderer Simulationsmethoden Die hier beschriebene Methode verwendet eine SPH-Simulation zur Strömungsrepräsentation und eine FDM-Simulation zur Lösung der Wellengleichung. Doch die Methode ist unabhängig der konkreten verwendeten Simulationsmethoden. So kann entsprechend des in Kapitel 4 vorgestellten Schemas auch eine FD-Methode zur dreidimensionalen Strömungssimulation verwendet werden. Wenn eine dementsprechende Oberfläche generiert wird, kann dann die detaillierte Oberflächensimulation auf diese Oberfläche abgebildet werden. Für interaktive Umgebungen bietet sich jedoch eine SPH-Simulation an, da sie effizient ausgeführt werden kann — gerade für



(a) (b) (c)

Abb. 5.19.: Ein niedrig aufgelöstes Höhenfeld (a) wird aus einer dynamischen SPH-Simulation extrahiert. Mit der hier beschriebenen Methode können der Simulation detaillierte Oberflächenwellen hinzugefügt werden (Boot (b), Regentropfen (c)).

geringe Flüssigkeitsmengen. Auch für die Wellengleichung muss nicht notwendigerweise ein FDM-Ansatz gewählt werden. So kann zum Beispiel auch die *Wave Particles*-Methode (Abschnitt 3.3.1, 5.1.2) zur Approximation verwendet werden.

5.1.4. Brechende Wellen

Das Simulation einer brechenden Welle ist seit langem ein Thema in der Computergraphik. Die Repräsentation der großen, beteiligten Flüssigkeitsmenge und der hohe Detailgrad stellen dabei die Hauptschwierigkeiten dar. Für interaktive Anwendungen wurden bisher prozedurale Ansätze vorgestellt ([JBS03], [TMSG07]). Die vollständige dreidimensionale Simulation der notwendigen Flüssigkeitsmenge in Echtzeit ist mit heutiger Rechentechnik schwerlich zu erreichen. Die starke Selbstähnlichkeit einer realen brechenden Welle ist Ausgangspunkt für die hier präsentierte Schichtung zur virtuellen Repräsentation. Lediglich eine zweidimensionale SPH-Simulation wird ausgeführt, was eine erhebliche Beschleunigung der Simulation darstellt. Dabei werden die Mengen aller Partikelpositionen der letzten n Zeitschritte der Simulation als Partikelschichten gespeichert $(P(t_j), j = -n, \ldots, 0)$. Eine brechende Welle wird aus diesem zweidimensionalen Datensatz konstruiert. Dafür werden die Schichten s_i entlang der z-Achse angordnet (i: z-Positions Index, Abb. 5.20). Die Form der Welle wird mit der Funktion $f(i)$ und

$$s_i = P(t_{f(i)}), \tag{5.8}$$

beschrieben, die die Anordnung der letzten n Partikelschichten im Objektraum definiert. Mit Hilfe der Abbildungsfunktion $f(i) \in \{-n, \ldots, 0\}$ können folglich verschiedenste Wellenfronten beschrieben werden. $f(i) = 0$ beschreibt zum Beispiel eine gerade und $f(i) = -i$ eine spitze Wellenfront. In den hier gezeigten Beispielen wurden Überlagerungen trigonometrischer Funktionen verschiedener Frequenzen verwendet. Auf diese Weise kann im Prinzip jede beliebige Wellenform beschrieben werden.

Da die beschriebene Methode lediglich eine zweidimensionale Flüssigkeit simuliert, reichen wenige Partikel zur Diskretisierung der Flüssigkeit aus, um die Welle brechen

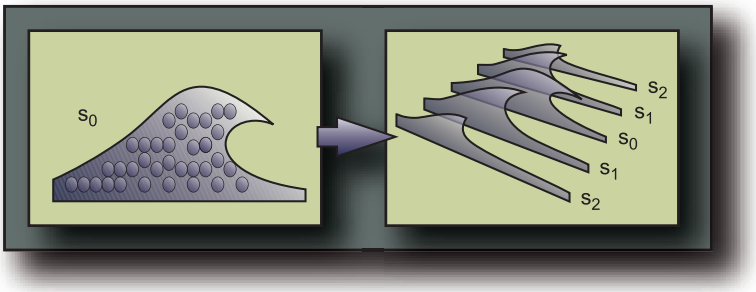


Abb. 5.20.: Prinzip der Simulation brechender Wellen. Lediglich eine 2D SPH-Simulation wird durchgeführt. Die letzten n Zeitschritte (hier: $n = 3$) der Simulation werden als Schichten angeordnet. So wird das symmetrische Erscheinungsbild einer dreidimensionalen brechenden Welle nachgebildet.

zu lassen. In der Summe vieler Schichten ergeben sich jedoch pro Frame viele Partikel, die zur Erzeugung einer dreidimensionalen Oberfläche berücksichtigt werden müssen. So sind zum Beispiel mehr als 100.000 dargestellte Partikel in Echtzeitumgebungen möglich — siehe zum Beispiel Abbildung 5.21. Die Oberflächenextraktion kann zum Beispiel mit der im später folgenden Abschnitt 6.2 beschriebenen Methode erfolgen, die trotz der hohen Partikelanzahl eine Oberfläche in Echtzeit generiert (siehe dazu Abbildung 5.25, Abschnitt 5.2.2).

Erzeugung einer brechenden Welle Die Erzeugung einer brechenden Welle in der Computergraphik basiert in der Regel auf der Erzeugung einer Welle, die entlang einer schiefen Ebene läuft. Aufgrund von Dispersion wächst die Amplitude der Welle und bei Erreichen eines Schwellwertes wird die Welle instabil und bricht. Die Welle selbst wird mit einer bewegten Ebene an einem Ende des Reservoirs erzeugt. Diese bewegt sich horizontal und erzeugt so aufgrund des Druckes eine Welle. In

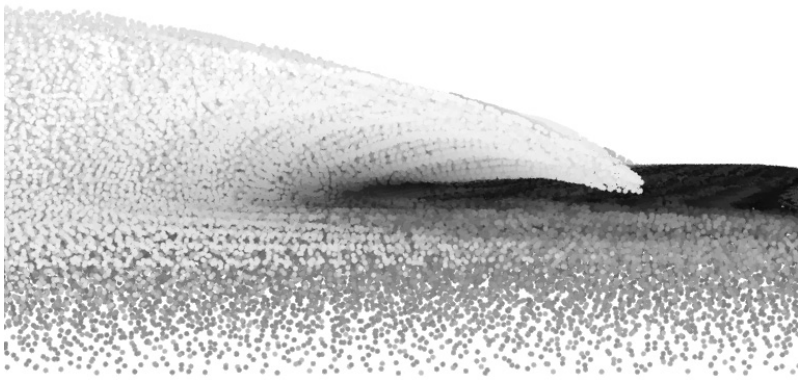


Abb. 5.21.: Simulation brechender Wellen: Partikeldarstellung.

dieser Arbeit wurde ein anderer Ansatz zur Erzeugung brechender Wellen gewählt, der stabiler als das oben beschriebene Vorgehen ist, da nicht mit hohen, extern herbeigeführten Drücken gearbeitet wird und so den Anforderungen in Echtzeitumgebungen gerecht wird. Eine zusätzliche externe Kraft wird eingeführt, die die Flüssigkeit beschleunigt. Da die Flüssigkeit sich in einem Pool bewegt, wird sie somit in Richtung der externen Kraft beschleunigt. Wird das Kraftfeld dann umgekehrt oder deaktiviert, erfolgt eine Brechung an der Poolwand. In der Praxis kann auch das Gravitationsfeld variiert werden. Die so entstehenden Wellen entsprechen physikalisch gesehen den Wellen, die in einem bewegten Gefäß erzeugt werden. Zudem kann ein virtueller Pool so auch interaktiv bewegt werden und die Welle somit interaktiv zur Brechung gebracht werden.

5.2. Empirisch-basierte Emulation

Im Folgenden werden beispielhaft mögliche Anwendungen auf Basis des Stufenmodells der empirisch-basierten Flüssigkeiten behandelt (siehe Abschnitt 4.3.2). Zunächst werden Approximationen und dann Animationen von Flüssigkeiten präsentiert. Die beschriebenen Methoden wurden mit den physikalisch-basierten Simulationsmethoden gekoppelt, die im vorherigen Abschnitt beschrieben wurden. Auf die Behandlung von Videosequenzen wird an dieser Stelle verzichtet, da sie im Rahmen dieser Arbeit nicht zum Einsatz kamen. Ihre Verwendung kann den Realismus in interaktiven Umgebungen jedoch zusätzlich verbessern.

5.2.1. Empirische/approximative Simulationen

Die beispielhaften Möglichkeiten *empirischer/approximativer Simulationen* im Rahmen des empirisch-basierten Stufenmodells (siehe Abschnitt 4.3.2) sind Thema dieses Abschnitts. Zunächst erfolgt die Erläuterung der Verwendung einer Massepunkt-Simulation. Anschließend wird die Wellenerzeugung von Antrieben innerhalb von Oberflächensimulationen präsentiert.

5.2.1.1. Massepunkt-Simulation

Die seit Langem in der Computergraphik verwendete Nutzung von Massepunkten zur Approximation von Flüssigkeiten basiert auf der Annahme, dass die Interaktion zwischen Flüssigkeitspartikeln (oder Tropfen) im freien Fall vernachlässigt werden kann. So können die Partikel unabhängig und lediglich unter Einfluss der Gravitation simuliert werden. Damit entfällt die kostenintensive Nachbarschaftssuche und Lösung der Navier-Stokes Gleichungen, so dass wesentlich mehr Partikel simuliert werden können als bei einer physikalischen Strömungssimulation. So können zum Beispiel Fontänen, Wasserfälle, fallende Tropfen oder spritzendes Wasser effizient repräsentiert werden. Im weiteren Verlauf dieser Arbeit werden diese Flüssigkeitsphänomene stellvertretend mit dem Begriff *Fontäne* bezeichnet.

Der chaotische Charakter einer Fontäne wird im Rahmen des Partikelerzeugunsprozess modelliert. Die Verwendung geeigneter zufälliger Erzeugungsfunktionen tritt an die Stelle realer physikalisch-basierter Simulationen. Jedes Partikel besitzt eine Masse, sowie einen Geschwindigkeits- und einen Positionsvektor. Zudem unterliegt jedes Partikel der Gravitationskraft $\mathbf{F}_g = m\mathbf{g}$. Das chaotische Verhalten realer Flüssigkeiten wird durch zufällig gewählte Anfangspositionen und -geschwindigkeiten innerhalb eines definierten Bereiches erzielt. Dazu werden Partikel in einem diskreten kubischen Volumen $(\triangle x, \triangle y, \triangle z)$ um $\mathbf{x}_0$ mit zufälligen Positionen erzeugt. r und r_{neg} bezeichnen zufällige Funktionen $(0 \leq r \leq 1;\ -1 \leq r_{neg} \leq 1)$.

$$\mathbf{x} = \mathbf{x}_0 + \begin{pmatrix} r_{neg}\triangle x \\ r_{neg}\triangle y \\ r_{neg}\triangle z \end{pmatrix}. \tag{5.9}$$

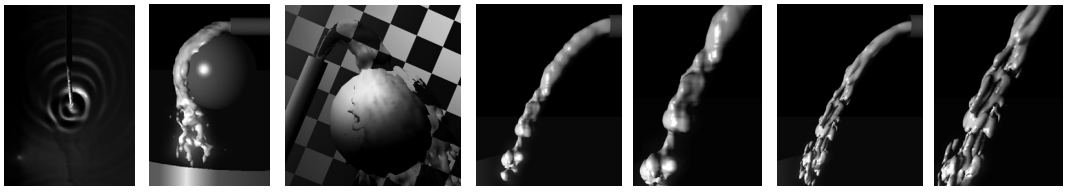
Die initialen Geschwindigkeiten werden ebenfalls zufällig generiert. Partikelgeschwindigkeiten werden um eine gegebene Hauptdirektion $\mathcal{R} \cdot (1,0,0)^T$ mit zugehöriger Streuung $v_{str\ 1}, v_{str\ 2}$ verteilt:

$$\mathbf{v}_{init} = \mathcal{R} \cdot \begin{pmatrix} v_0 + r \cdot v_{var} \\ r_{neg} \cdot v_{str\ 1} \\ r_{neg} \cdot v_{str\ 2} \end{pmatrix}. \tag{5.10}$$

$\mathcal{R}$ ist eine Rotationsmatrix, die die Hauptflussrichtung definiert, v_0 ist die minimale Ausflussgeschwindigkeit und v_{var} beschreibt die Variation der Geschwindigkeiten. Mit Hilfe dieser Formeln kann das Ausgangsvolumen und die Richtung einer beliebigen Fontäne intuitiv definiert bzw. modelliert werden. Dennoch sind selbstverständlich andere Erzeugungsfunktionen denkbar. Da reale Fontänen nicht notwendigerweise eine konstante Ausflussgeschwindigkeit besitzen, kann durch eine Variation von v_0 der Realismus in diesen Fällen verbessert werden. Entsprechend des zu simulierenden Effektes kann diese Variation zum Beispiel zufällig oder periodisch erfolgen.

Ein weiterer Kontrollparameter der Fontäne ist die Anzahl generierter Fontänen-Partikel n_F pro Zeitschritt. So kann die Intensität bzw. Flüssigkeitsmenge der Fontäne definiert werden. Eine natürliche Fontäne besitzt in der Regel Intensitäts-Variationen, die zum Beispiel folgendermaßen dargestellt werden können: $n_F = n_0 + r \cdot n_{var}$. n_0 ist die minimale Anzahl generierter Partikel pro Zeitschritt und n_{var} beschreibt die Intensität der Variation. Instantane Veränderungen von n_F modelliert die Veränderung der Flussintensität. Für $n_F = 0$ kommt der Fluss zum Erliegen.

Erreicht ein Fontänen-Partikel die Oberfläche der Flüssigkeit, so kann es vernichtet werden und löst einen Event aus. Dieser kann in einer Oberflächensimulation zum Beispiel zur Initialisierung einer radial propagierenden Welle genutzt werden. In einer Strömungssimulation kann er im Rahmen einer Druckänderung oder Strömungsinitiierung verarbeitet werden. Die Kollisionsbehandlung von Fontänen-

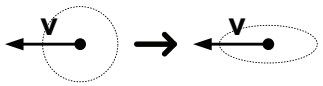


(a) (b) (c) (d) (e) (f) (g)

Abb. 5.22.: Beispiel: Empirische Fontänen. Ansicht von oben (a), Kollisionen (b,c). Oberflächenextraktion: Kugelpotentiale (d,e), elliptische Potentiale (f,g).

Partikeln an Objekten kann mit dem Reflexionsprinzip (siehe auch Abschnitt 2.4) und einer Dämpfung erfolgen. So kann eine realistische und schnelle Interaktion auch mit bewegten Objekten realisiert werden.

Die Darstellung der so erzeugten Partikelwolke kann zum Beispiel durch die Verwendung von Unschärfe entsprechend der Geschwindigkeit geschehen (zum Beispiel [Tat06]). In der hier gezeigten Methode wird ein *Marching Cubes*-Algorithmus zur Darstellung verwendet (Abb. 5.9, 5.11 und 5.22). Die Struktur der Fontäne kann besser bei der Oberflächenextraktion berücksichtigt werden, wenn die implizite Oberfläche nicht durch radialsymmetrische Potentiale definiert wird, sondern durch elliptische Potentiale, die entsprechend der Geschwindigkeit adaptiert sind:



So kann die Strömungsrichtung besser repräsentiert und abgetastet werden und die Flüssigkeit erscheint detaillierter (siehe dazu auch Abbildung 5.22d–g). Im später folgenden Abschnitt 6.2 wird eine effiziente Methode zur Oberflächendarstellung dreidimensionaler Partikelwolken präsentiert, die für interaktiven Umgebungen besser geeignet ist als die *Marching-Cubes*-Methode. Der dort beschriebene Algorithmus wird in dem Abschnitt unter anderem auch an Fontänen demonstriert.

5.2.1.2. Modellierung eines Antriebs

Im Folgenden wird die Erzeugung von Oberflächenwellen beschrieben, die zum Beispiel vom Antrieb eines Motorboots resultieren können und in der markanten Heckwelle resultieren. Innerhalb von Oberflächensimulationen kann eine physikalische Strömungsmodellierung von Antrieben nicht erfolgen. Deshalb wird im Bereich der jeweiligen Schiffsschraube ein Wellengenerator positioniert.

Da Wasserfahrzeuge nicht auf einen Motor beschränkt sind, werden die Generatoren symmetrisch zur Längsachse des Objektes positioniert. Somit wird Generator i an folgender Position in Abhängigkeit der maximalen Anzahl von Motoren m und

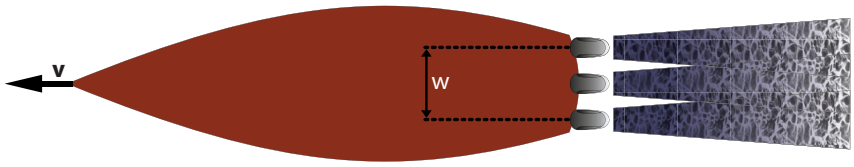


Abb. 5.23.: Motorwellen- und Schaumerzeugung (drei Maschinen in diesem Beispiel).

der Breite w des Fahrzeugs positioniert (Abb. 5.23):

$$y = -\frac{w}{2} + i \cdot \frac{w}{m} + \frac{w}{2m}. \tag{5.11}$$

Der Wellengenerator ist generell und kann beliebige Formen annehmen. Kreisförmige oder zumindest abgerundete Formen erzielen jedoch aufgrund fehlender Kanten bessere Ergebnisse. Eine Welle wird durch Abbildung der Form in das Simulationsgitter erzeugt, da dann jedes eingeschlossene Gitterelement Ausgangspunkt einer radial propagierenden Elementarwelle ist. Größe und Intensität der Wellengeneratoren hängen linear von der Objektgeschwindigkeit (bzw. der Motorgeschwindigkeit) ab.

5.2.2. Animation dynamischer Phänomene

Dieser Abschnitt beschreibt zwei Techniken zur Animation von Schaum auf Oberflächen und Gischt brechender Wellen. Die Methoden stellen damit Beispiele für Animationstechniken im Rahmen der *empirisch-basierten Emulation* dar.

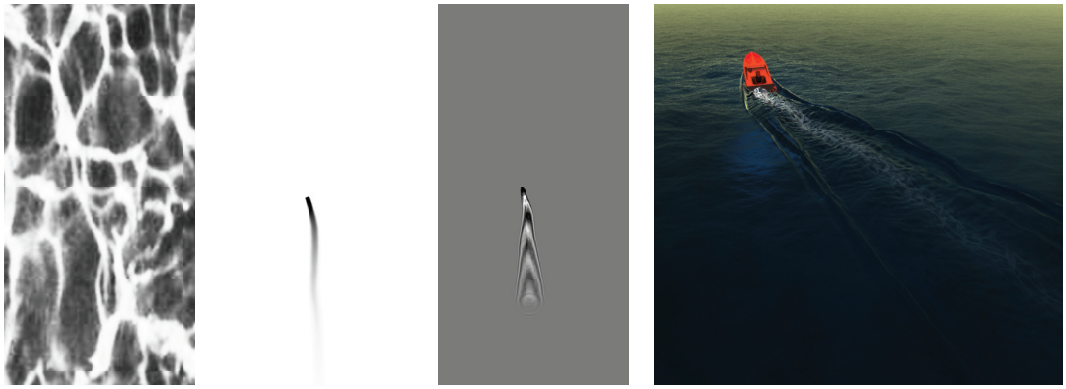
5.2.2.1. Schaum

Oftmals ist es von Interesse, neben der Wellenerzeugung einer Objektbewegung die Schaumbildung auf einer Oberfläche zu repräsentieren. Die starken Strömungen und Verwirbelungen einer Schiffsschraube führen zum Beispiel zu der markanten Heckwelle und der typischen Schaumbildung im Kielwasser eines Bootes. Im Folgenden wird eine Methode zur Modellierung dieses Effekts im Rahmen der beschriebenen Flüssigkeitssimulation vorgestellt. Aufgrund der hohen Komplexität einer physikalischen Simulation dieses Effektes wird das Resultat modelliert und nicht die Ursache.

Der folgende Ansatz zur Repräsentation von Schaum basiert auf der Beobachtung, dass Schaum typische Eigenschaften besitzt, die im Folgenden gegeben werden:

- Die Menge/Dichte des erzeugten Schaumes hängt von der Motorgeschw. ab.
- Die Menge/Dichte des Schaumes verringert sich in Abhängigkeit der Zeit.
- Schaum verweilt an seiner Position und breitet sich sehr langsam aus.

Diese Eigenschaften werden mit einem zellulären Automaten repräsentiert. Dazu wird an der gewünschten Position ein Schaumgenerator positioniert, der im Prinzip dem in Abschnitt 5.2.1.2 beschriebenen Wellengenerator entspricht. Von Antrieben generierter Schaum wird dementsprechend im Bereich des Motors erzeugt.



(a) (b) (c) (d)

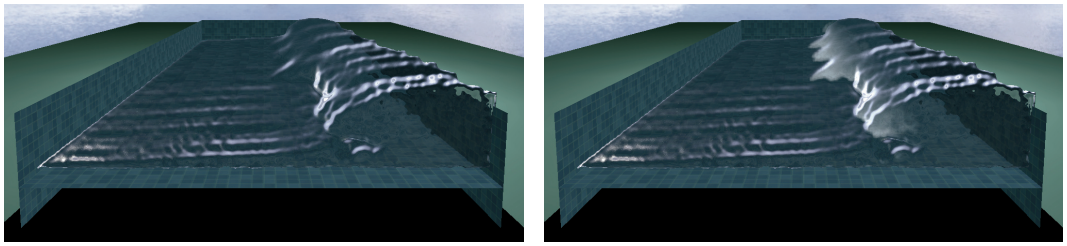
Abb. 5.24.: Schaum und Wellenerzeugung. Schaumtextur (a), Schaumgenerator und Filterfunktion (b), Simulationsgitter der Wellengleichung (c), Ergebnis (d).

Innerhalb des Schaumgeneratorbereiches wird neuer Schaum der Intensität f mit $f \epsilon [0, 1]$ in Abhängigkeit der Motorengeschwindigkeit in einer *foam map* generiert. Diese beinhaltet die aktuelle Schaumintensität für jedes Gitterelement. Anhand der Intensitätswerte wird später pro Gitterzelle eine Schaumtextur eingeblendet. Die Ausbreitung von Schaum wird durch langsames Diffundieren der Intensitäten erreicht. Diese Diffusion kann durch eine Filterfunktion effizient umgesetzt werden (Abb. 5.23):

$$f_{i,j}^{new} = d \cdot f_{i,j} + \frac{1-d}{4} \cdot (f_{i+1,j} + f_{i,j+1} + f_{i-1,j} + f_{i,j-1}). \tag{5.12}$$

Die Geschwindigkeit des Ausblendens und Ausbreitens wird über den Parameter $d \epsilon [0, 1]$ beschrieben. Die Ausdehnung des Schaumgenerators kann bei bewegten Objekten — wie bereits beim Heckwellengenerator beschrieben wurde — von der Geschwindigkeit des Objektes abhängen. Der beschriebene Zustandsautomat kann neben der Erzeugung von Schaum in einer Heckwelle auch zur Erzeugung von Schaum an jeder beliebigen Position verwendet werden, zum Beispiel wenn ein Objekt ins Wasser fällt.

Schließlich wird für eine überzeugende Schaumdarstellung eine Textur verwendet, die entsprechend der Intensitätswerte innerhalb des Schaumgenerators eingeblendet wird (Abb. 5.24). So hat ein Designer die größtmögliche Freiheit, das visuelle Erscheinungsbild des Schaumes zu parametrisieren. Die Verwendung verschiedener Schaumtexturen in Verbindung mit verschieden parametrisierten Generatoren ermöglicht auch die Darstellung verschiedener Arten von Schaum innerhalb einer Umgebung.



(a) (b)

Abb. 5.25.: Brechende Wellen. Grundansatz (a), Verwendung einer zusätzlichen Geschwindigkeits-Map zur Darstellung von Gischt (b). 200 Schichten mit jeweils 700 Partikeln, 53, 1 FPS. Die Oberfläche ist mit der in Abschnitt 6.2 beschriebenen Methode dargestellt.

5.2.2.2. Gischt

Zur realistischeren Darstellung der brechenden Welle (Abschnitt 5.1.4) wird in Abbildung 5.25b eine zusätzliche Geschwindigkeits-Map der Partikel verwendet. Konkret wird die Geschwindigkeit in y-Richtung verwendet, um eine Farbverschiebung in Richtung Weiß durchzuführen. Somit entsteht in Abhängigkeit der Geschwindigkeiten der betroffenen Partikel ein Gischteffekt.

5.3. Starre Körper

Die Inklusion starrer Körper in die vorgestellten Simulationsmethoden ist Thema dieses Abschnitts. Zunächst wird die diskrete Representation behandelt, die im Rahmen dieser Arbeit Verwendung fand. Anschließend wird die Wellenerzeugung und -reflexion innerhalb von Oberflächenwellensimulationen und die Strömungserzeugung und -reaktion innerhalb von Strömungssimulationen behandelt.

5.3.1. Repräsentation

Physikalisch werden schwimmende oder bewegte Objekte simuliert, wie es in Abschnitt 2.3 beschrieben wurde. Die Objekte werden im Rahmen dieser Arbeit mit Partikeln abgetastet und mit Hilfe dieser werden die physikalischen Größen berechnet. Die Diskretisierung der Objekte erfolgt entsprechend Abbildung 5.26. Die Partikel werden adäquat auf und innerhalb des Rumpfes positioniert. Ziel ist es, die Anzahl der modellierenden Partikel möglichst gering zu halten, um damit eine schnelle Berechnung der Objektbewegungen zu ermöglichen. Jedes Partikel repräsentiert die umgebende Masse des Objektes. Auftriebskräfte werden lediglich für unter der Oberfläche liegende Partikel bestimmt. Anhand aller wirkenden Kräfte wird das Drehmoment jedes Partikels in Bezug auf den Schwerpunkt bestimmt. Das gesamte wirkende Drehmoment ergibt sich aus der Summe der einzelnen Momente. Die Verwendung von Partikeln erlaubt damit die schnelle Simulation *beliebiger* Objektformen.

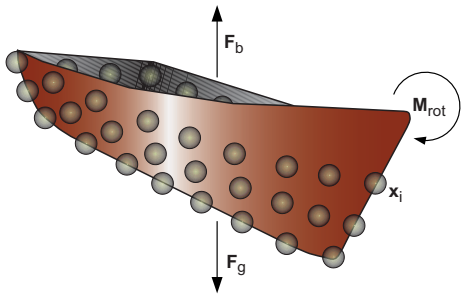


Abb. 5.26.: Starre Körper, wie zum Beispiel Boote, werden unter Verwendung von Partikeln diskretisiert.

Glättung der Auftriebskraft Um die Anzahl von Partikeln pro Objekt gering zu halten, werden verhältnismäßig grobe Abtastungen verwendet. Dies kann für Partikel im Bereich der Oberfläche zu Sprüngen in der Auftriebskraft führen, wenn diese die Oberfläche durchschreiten. Damit diese Situationen nicht in sprunghaften Bewegungen resultiert, wird die Auftriebskraft im direkten Bereich unterhalb der Oberfläche linear skaliert, so dass auf der Oberfläche die Auftriebskraft verschwindet. So kommt es auch bei grob abgetasteten Objekten nicht zu unrealistischen, sprunghaften Bewegungen.

Kollisionsbehandlung zwischen Objekten Auf der starren Körper-Oberfläche liegende Partikel werden in der gezeigten prototypischen Umsetzung auch zur Kollisionsdetektion verwendet. Dazu wird zwischen Partikeln unterschiedlicher Objekte, die einen Schwellwert in der Distanz unterschreiten, eine Kollisionskraft eingeführt, die direkt auf die beteiligten Partikel wirkt (siehe Abbildung 5.27). Dieses Vorgehen bietet sich aufgrund der Verwendung der Partikel zur Bestimmung physikalischer Eigenschaften an, da die Datenstruktur direkt weiterverwendet werden kann — zudem kann diese Methode effizient ausgeführt werden. Dennoch kann es bei diesem Vorgehen und bei ungünstigen Diskretisierungen zu Überschneidungen von Objekten kommen. Ist dieser Effekt unerwünscht, kann auf eine der zahlreichen existierenden, exakten Kollisionsbehandlungsmethoden zurückgegriffen werden.

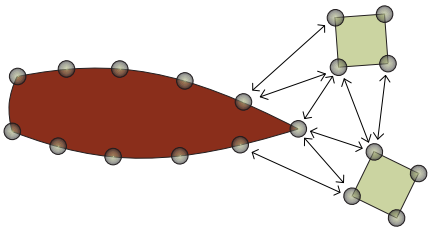
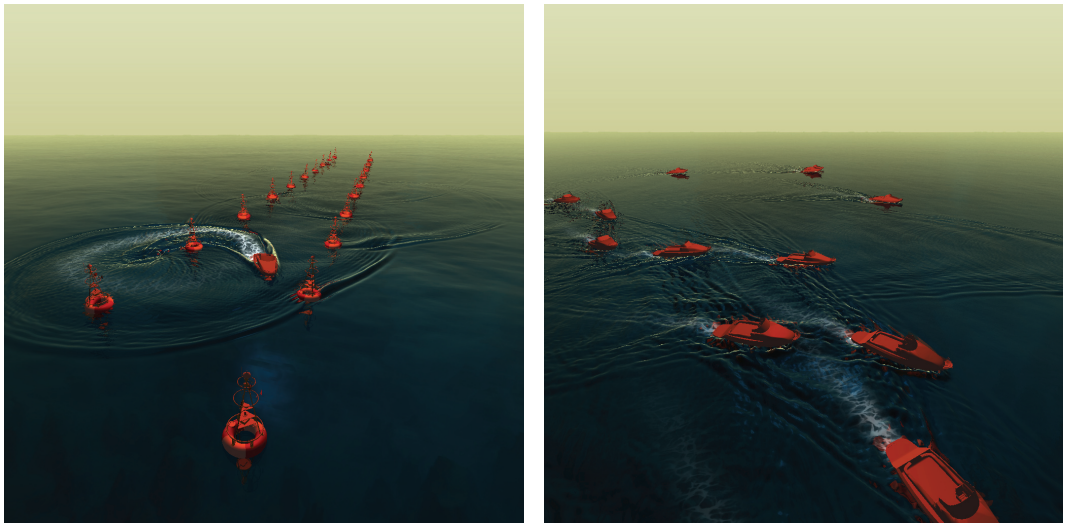


Abb. 5.27.: Kollisionspartikel werden für eine effiziente Kollisionsbehandlung eingeführt. Zwischen ihnen wirken bei Unterschreitung eines Schwellwertes Abstoßungskräfte.



(a) (b)

Abb. 5.28.: Beispiele: Bojen (a) und Armada (b).

Engpass innerhalb einer GPU-basierten Oberflächensimulation Die starre Körper-Simulation besitzt einen Engpass, wenn die starre Körper-Simulation auf der CPU und die Oberflächensimulation auf der GPU erfolgt. Dann ist ein *Memory-Readback* der Oberflächeninformationen notwendig, der auf heutiger Standard-PC-Hardware ausgesprochen kostenintensiv ist, damit die schwimmenden Objekte mit den Oberflächenveränderungen der Flüssigkeit interagieren können. Die Wellenerzeugung bewegter Objekte kann maßgeblich beschleunigt werden, indem niedrig aufgelöste Modelle der relevanten Objekte zur Wellenerzeugung verwendet werden. Aufgrund der anschließend erfolgenden Gitterprojektion und dem notwendigen *Multipass-Rendering* der Objekte, ist dieses Vorgehen sinnvoll — jedoch in der gezeigten prototypischen Implementierung noch nicht umgesetzt, wodurch gerade unter Verwendung vieler schwimmender Objekte die Performanz stark reduziert wird (Abb. 5.28).

5.3.2. Oberflächenwellenerzeugung und -reflexion

Wellenerzeugung Innerhalb von Oberflächensimulationen (FDM oder *Wave Particles*), muss die Wellenerzeugung explizit modelliert werden, da keine Strömungseigenschaften repräsentiert werden. In zwei Situationen werden Wellen von dynamischen Objekten erzeugt:

- Fallende und eintauchende Objekte und
- bewegte, schwimmende Objekte.

Für den ersten Fall wird eine Welle in Form der Silhouette des eintauchenden Objektes erzeugt, während das Objekt die Flüssigkeitsoberfläche durchschneidet. Dazu wird die Silhouette des Objektes dem Simulationsgitter hinzugefügt. So gilt jeder zur Silhouette gehörige Punkt als Ausgangspunkt einer radial propagierenden Elementarwelle. Entsprechend dem Huygensschen Prinzip ergibt sich aus der Überlagerung der resultierenden Elementarwellen die propagierende Wellenfront. Um den Zeitpunkt der Wellenerzeugung zu bestimmen, besitzt jedes Objekt einen Zustand q: $(q = 1)$ unter der Oberfläche, $(q = 0)$ über der Oberfläche. Der Zustand wird in jedem Zeitschritt aktualisiert und die Welle wird erzeugt, wenn ein Partikel eines Objektes mit Position $\mathbf{x}_k$ zum Zeitpunkt t die Oberfläche durchschneidet:

$$\begin{aligned} \textit{erzeuge\_welle} = &\Big[(q^{t-1} = 0) \wedge ((\mathbf{x}_k^t)^z < 0)\Big] \vee \\ &\Big[(q^{t-1} = 1) \wedge ((\mathbf{x}_k^t)^z > 0)\Big]. \end{aligned}$$

Wenn *erzeuge_welle* für ein gegebenes Objekt wahr ist, hat das Objekt die Oberfläche durchschritten und eine Welle wird erzeugt. Dazu wird das Objekt mit der Oberflächenebene geschnitten und auf das Simulationsgitter projiziert. Die in das Simulationsgitter eingefügten Werte sind dabei proportional zur Eintauchgeschwindigkeit.

Die Wellenerzeugung bewegter Objekte, wie zum Beispiel Booten, erfolgt nach dem Prinzip, dass bewegte Objekte an der Vorderseite in Bewegungsrichtung den Druck der Flüssigkeit vergrößern und auf der Rückseite den Druck verringern. Somit können auf der Vorderseite Wellen mit positiver Amplitude und auf der Rückseite mit negativer Amplitude erzeugt werden. Die Wellenintensität ist dabei abhängig von der Bewegungsgeschwindigkeit des bewegten Objektes.

Da die Simulation in einem diskreten Zeitraum stattfindet, ist somit Ziel, die Fläche zu extrahieren, die während des letzten Zeitschrittes überschritten wurde. Dies kann über die Objektpositionen des aktuellen und des vorhergehenden Zeitschrittes erfolgen. Das genaue Vorgehen wird im Folgenden beschrieben. Ziel ist eine effiziente Methode, die auf der GPU ausgeführt werden kann.

Zunächst wird für den aktuellen und den vorhergehenden Zeitschritt der Schnitt zwischen Objekt und Flüssigkeitsoberfläche bestimmt. Es ist möglich, diesen Schnitt exakt zu bestimmen, indem die jeweiligen Polygonmengen geschnitten werden. Da er jedoch in jedem Zeitschritt und für jedes Objekt neu bestimmt werden muss, werden im Folgenden zwei Annahmen zur Beschleunigung getroffen: Zunächst approximiert eine Ebene mit $z = 0$ die Oberfläche. Diese Approximation trifft für moderat bewegte Flüssigkeitsoberflächen zu, da die Wellen in der Regel klein sind im Vergleich zur Ausdehnung der Oberfläche. Bei sehr stürmischer See und dementsprechend starker Bewegung des Objektes kann es jedoch zu Fehlern kommen. Allerdings sinkt der Realismus bei stürmischer See ohnehin, da überschlagende Wellen und spritzendes Wasser nicht dargestellt werden. In der Regel resultiert diese Einschränkung allerdings nicht in visuellen Artefakten. Des Weiteren wird angenommen, dass je-

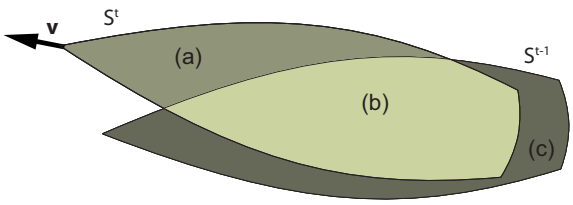


Abb. 5.29.: Wellen werden mit Hilfe von Mengenoperationen auf S^t und S^{t-1} zum aktuellen und vorherigen Zeitschritt generiert. Bugwellen entstehen in (a) und Heckwellen in (c).

der Schnitt des schwimmenden Objektes mit $z = const., z < 0$ eine Teilmenge der Schnitte bei $z = 0$ ist. Diese Annahme ist gültig für die meisten Boote und Schiffe — zum Beispiel auch für Katamarane. Falls die Annahme für gegebene, komplexere Objekte nicht zutreffen sollte, können ein oder mehrere veränderte Modelle zur Schnittbestimmung verwendet werden, welche in Abschnitten der beschriebenen Bedingung genügen. Eine andere Möglichkeit ist ein Culling des aktuellen Objektes entlang der beiden Ebenen $z = \pm\varepsilon$ mit kleinem ε in Kombination mit einem Flächenfüllungs-Algorithmus. So können auch komplexe Objektformen mit großer Dynamik korrekte Wellen erzeugen — auf Kosten der Performanz.

Sind die Schnittmengen zum aktuellen und vorhergehenden Zeitschritt S^t und S^{t-1} bestimmt, können sie mit Hilfe von Mengenoperationen kombiniert werden. Drei verschiedene Mengen, die verschiedene Wellenerzeugungen repräsentieren, ergeben sich (Abbildung 5.29):

1. Bugwelle: $B^t = S^t \backslash S^{t-1}$,
2. Rumpf: $R^t = S^t \bigcap S^{t-1}$,
3. Heckwelle: $H^t = S^{t-1} \backslash S^t$.

$\bigcap$ bezeichnet den Durchschnitt und $\backslash$ die Differenz der Mengen. Wellen werden erzeugt, indem innerhalb der extrahierten Flächen B^t und H^t zu den aktuellen Simulationswerten Wellenimpulse der entsprechenden Orientierung hinzugefügt werden (siehe dazu Abbildung 5.18, Abschnitt 5.1.3).

Wellenreflexion Die generierte Menge R^t kann direkt zur Reflexion von Wellen am Rumpf verwendet werden. Im Ruhezustand entspricht $S^t = R^t$.

Bei einer *FD-Methode* wird das Simulationsgitter innerhalb von R^t zurückgesetzt. Somit werden interaktive Wellen an jedem Objekt entlang der Wasserlinie korrekt reflektiert. Die Wellenreflexion kann jedoch nur anstatt der im vorhergehenden Abschnitt beschriebenen Wellenerzeugung verwendet werden. Ansonsten würden die erzeugten Wellen von der Reflexionsmethode überschrieben werden. Wellenreflexion an bewegten, wellenerzeugenden Objekten wäre jedoch kaum sichtbar. Somit können schwimmende aber unbewegte Objekte Wellen reflektieren und bewegte Objekte Wellen erzeugen.

Bei der *Wave Particles-Methode* können die Partikel direkt an R^t reflektiert werden. Die Kollisionsbehandlung entsprechend der *Wave Particles*-Methode arbeitet dabei unverändert. Aufgrund der expliziten Integration der Geschwindigkeiten, kann jedoch für ein Wellenpartikel nicht garantiert werden, dass es nicht in ein Kollisionsobjekt eindringt. Dieser Effekt tritt vor allem in Bereichen scharfer Kanten auf. In diesem Fall muss das Partikel auf die nächstliegende Oberfläche des Kollisionsobjektes verschoben werden. Da dieser Effekt jedoch maximal für wenige Partikel pro Zeitschritt auftritt, reduziert diese Berechnung die Performanz lediglich unwesentlich. Aufgrund der hohen Anzahl von Partikeln, kann ein derartiges Partikel, welches sich nach einer Kollisionsbehandlung noch innerhalb eines Kollisionsobjektes befindet, auch direkt vernichtet werden. Innerhalb einer interaktiven Simulation resultiert dieses Vorgehen in der Regel nicht in wahrnehmbaren Nebeneffekten. Eine implizite Integration würde das Problem des Eindringens reduzieren. Aufgrund hoher Partikelanzahlen ist dies nicht direkt sichtbar. Zudem ist eine eine implizite Methode rechenintensiver, so dass sich dennoch die explizite Methode empfiehlt.

5.3.3. Strömungserzeugung und -reaktion

In den folgenden beiden Abschnitten wird zunächst die Strömungserzeugung und Strömungsreaktion statischer und dynamischer Objekten innerhalb einer SPH-Simulation und dann innerhalb einer FDM-Simulation behandelt.

5.3.3.1. Starre Körper innerhalb einer SPH-Simulation

Statische Körper Kollisionen von Flüssigkeitspartikeln mit statischen Objekten werden mit Hilfe eines Kraftfeldes behandelt, das Kollisionsobjekte umgibt (Abb. 5.30):

$$\mathbf{F}_k \sim \frac{1}{d+1} \cdot \hat{\mathbf{n}}_{Objekt}. \tag{5.13}$$

d beschreibt die Distanz zwischen Objekt und Partikel und $\hat{\mathbf{n}}_{Objekt}$ den geglätteten Normalenvektor des Objektes. Dieser kann zum Beispiel in einem Vorberechnungsschritt mit Hilfe einer Glättung Gitter-basiert erzeugt werden. Somit kann die Kollisions- oder Abstoßungskraft $\mathbf{F}_k$ auf jedes Partikel in der Umgebung des Kollisionsobjektes wirken. Dieser explizite Ansatz behandelt auch sehr schnelle Partikel: Ein Partikel, welches innerhalb eines Kollisionsobjektes liegt, wird zusätzlich eine Kraft erfahren, die es entsprechend der Normalenrichtung wieder aus dem Objekt heraus bewegt. Somit können Objekte auch durch die Flüssigkeit bewegt werden. Aufgrund des beschriebenen Kraftfeldes wird der Kollisionsbehandlungsprozess auf mehrere Zeitschritte verteilt (siehe Abbildung 5.30) und resultiert in einer stabilen Simulation.

Dynamische Körper Die Behandlung dynamischer starrer Körper innerhalb einer SPH-Simulation kann unter Zuhilfenahme von Partikeln, die den starren Körper

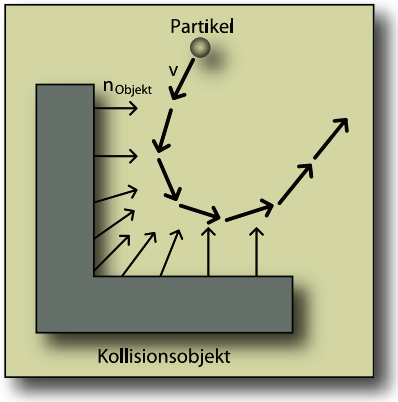


Abb. 5.30.: Prinzip der Kollisionsbehandlung.

diskretisieren, direkt realisiert werden. Die Diskretisierung und Berechnung der physikalischen Kräfte, die auf den starren Körper wirken, erfolgt wie in den Abschnitten 2.3 beschrieben. Die Partikel, die den starren Körper diskretisieren, werden innerhalb der SPH-Simulation als SPH-Partikel behandelt, so dass die Flüssigkeit direkt mit dem starren Körper interagiert. Diese Partikel werden jedoch nicht entsprechend der Dynamik der Flüssigkeit simuliert, sondern ihre relativen, statischen Positionen werden entsprechend der Orientierung des starren Körpers nach jedem Zeitschritt wieder hergestellt, um Deformationen zu unterdrücken. Lediglich die umliegenden Flüssigkeitspartikel behandeln diese Partikel als SPH-Partikel.

5.3.3.2. Starre Körper innerhalb einer FDM-Simulation

Eine Interaktion mit Objekten kann auch mit einer FDM-Strömungssimulation erfolgen — siehe zum Beispiel Abbildung 5.12. Dazu muss lediglich die Objektposition in Weltkoordinaten in den Simulationsraum transformiert werden. *Statische Objekte* können als Randbedingungen innerhalb der Simulation behandelt werden. Die auf die Oberfläche *dynamischer Objekte* wirkenden Kräfte können entsprechend Abschnitt 2.3 bestimmt werden. So kann sich ein schwimmendes Objekt entsprechend der Strömung bewegen. Gleichzeitig kann ein dynamisches Objekt jedoch auch Strömungen erzeugen, indem um das Objekt herum Strömungen initiiert werden.

5.4. Zusammenfassung

Dieses Kapitel stellt neue Beiträgen für den Schritt der Simulation vor. Schwerpunkt der vorgestellten Ansätze liegt auf der effizienten und problemangepassten Ausführung der Simulationen. Entsprechend dem vorgestellten allgemeinen Schema zur Simulation werden physikalisch-basierte und empirisch-emulierte Flüssigkeiten behandelt. Zudem erfolgt die Präsentation von Kopplungen der

Flüssigkeitssimulationen mit starren Körpern.

Für die physikalisch-basierte Simulation wird die Kombination von Simulationstechniken beschrieben, um die Qualität oder Möglichkeiten virtueller Flüssigkeiten zu verbessern. Mit Hilfe bewegter Simulationsgitter auf einer ambienten Oberfläche kann beispielsweise mit scheinbar unendlichen Wasserflächen interagiert werden. Die Methode ist auch zur Darstellung von Flüssen und Seen gut geeignet. Die gezeigte Kombination einer Oberflächenwellen- und Strömungssimulation in zwei Dimensionen baut auf der *Wave Particles*-Methode auf und erweitert diese um die Repräsentation von Strömungen. So können zusätzlich bewegte Flüssigkeiten repräsentiert werden. Mit Hilfe der Kopplung einer dreidimensionalen Strömungssimulation und einer Oberflächensimulation können wesentlich detaillierte Oberflächen in Echtzeit repräsentiert werden. Vergleichbare Details nur unter Verwendung einer Strömungssimulation zu erzielen, ist in Echtzeit heutzutage nicht möglich. Schließlich kann eine Schichtung zweidimensionaler Simulationsdaten in Kombination mit der vorgestellten Methode zur Oberflächendarstellung dynamischer Punktwolken eine realistische Darstellung brechender Wellen in Echtzeit erreichen.

Beispielhaft für empirisch-emulierte Flüssigkeiten wurden Punktmasse-Simulationen, Antriebswellenerzeugung und Animationen von Schaum und Gischt behandelt. So können zusätzlich Phänomene repräsentiert werden, die für eine physikalisch-basierte Simulation in interaktiven Umgebungen zu aufwendig wären.

Die Extraktion von Oberflächen aus den Daten der hier vorgestellten Simulationsmethoden ist Thema des nächsten Kapitels.

6. Oberflächenextraktion

Nachdem im vorherigen Kapitel mögliche Simulationsmethoden behandelt wurden, erfolgt in diesem Kapitel die Präsentation von Oberflächenextraktionsmethoden. Die Oberflächenextraktion von Flüssigkeiten kann in Höhenfeld-basierte und dreidimensionale Oberflächenextraktion entsprechend der Dimensionalität der zugrundeliegenden Simulation unterteilt werden (siehe Kapitel 4). Während die Extraktion von Höhenfeldern effizient geschehen kann, verhindert in interaktiven Umgebungen der Aufwand einer dreidimensionalen polygonalen Oberflächenextraktion aus dreidimensionalen Partikelwolken oftmals eine schnelle Oberflächendarstellung. Deshalb werden schnelle Methoden benötigt, um dreidimensionale Flüssigkeitsoberflächen bei interaktiven Frameraten extrahieren zu können.

Im Folgenden wird zunächst auf die im Rahmen dieser Arbeit verwendeten Methoden zur Höhenfeldextraktion eingegangen. Anschließend wird eine neue, schnelle Methode zur Extraktion von Oberflächen aus dreidimensionalen, dynamischen Partikelwolken beschrieben.

6.1. Höhenfelder

Die Darstellung von Höhenfeldern ist in vielen Anwendungsgebieten essentiell und dementsprechend existieren viele Methoden, die zumeist zur Darstellung statischer Höhenfelder (zum Beispiel Landschaften) entwickelt wurden und in der Regel einen Vorberechnungsschritt verwenden (siehe auch Abschnitt 3.4.1). Da Flüssigkeitsoberflächen jedoch dynamisch sind, können keine Verfahren verwendet werden, die eine Vorberechnung voraussetzen. Im Folgenden werden zunächst unendliche und beschränkte Höhenfelder behandelt. Dann erfolgt die Präsentation der Extraktion eines Höhenfeldes aus einer dreidimensionalen Partikelwolke.

Unendliche Oberflächen Für unendliche Oberflächen — wie zum Beispiel die eines Ozeans — sind die *Projected Grid*-Methode [HNC02, Joh04] und das Verfahren der *Geometry Clipmaps* [LH04, YPZL05] geeignet (vgl. Abschnitt 3.4.1). In dieser Arbeit wird die *Projected Grid*-Methode für die Erzeugung unendlicher Oberflächen verwendet, da diese eine blickpunktabhängige Abtastung der Oberfläche im Bildraum durchführt und damit ausschließlich Polygone im Bildbereich und einen weichen Übergang des LoDs generiert, entsprechend dem Abstand zur Kameraposition. Zudem stellt die *Projected Grid*-Methode die am häufigsten verwendete Methode zur Darstellung unendlicher Flüssigkeitsflächen dar. Die Artefakte, die aufgrund der

(a) (b) (c)

Abb. 6.1.: Mit abnehmender Abtastung im Bildraum nehmen die daraus resultierenden Artefakte zu. Abtastrate in Pixeln: 1×1 (a), 2×2 (b), 4×4 (c).

Abtastung im Bildraum entstehen können, sind beispielhaft in Abbildung 6.1 dargestellt. Die Amplitude von Flüssigkeitsoberflächen sind jedoch verhältnismäßig gering (im Vergleich zu Berglandschaften), so dass prinzipiell wenig Artefakte entstehen. Unter Verwendung von *Mip-Mapping* oder zusätzlichen Filtern können diese reduziert werden. Zusätzlich werden in dieser Arbeit die Normalen mit zunehmenden Kameraabstand in Richtung $(0, 1, 0)^T$ adaptiert, um ein Rauschen in den Lichtreflexen zu vermeiden.

Räumlich beschränkte Oberflächen Bei räumlich beschränkten Oberflächen — wie zum Beispiel die eines Pools — können die beiden oben genannten Verfahren ebenfalls genutzt werden. In der Regel können die Oberflächen jedoch auch performant mit einem *brute force*-Ansatz erzeugt werden, wobei alle Oberflächenpolygone dargestellt werden. Dieser Ansatz wird für die in dieser Arbeit gezeigten Pool-Beispiele verwendet. Bei Bedarf kann mit zunehmendem Abstand zur Kamera die Abtastung des Höhenfeldes halbiert werden — entsprechend dem Verfahren der *Geometry Clipmaps*.

Extraktion eines Höhenfeldes aus einer Partikelwolke Die Extraktion eines Höhenfeldes aus einer Partikelwolke (zum Beispiel aus einer SPH-Simulation resultierend) basiert auf einer impliziten Oberfläche, die als Höhenfeld extrahiert wird — verwendet wird dieser Ansatz in der im Rahmen dieser Arbeit entwickelten Methode zur Kopplung einer dreidimensionalen Strömungs- und einer zweidimensionalen Oberflächensimulation (Abschnitt 5.1.3).

Radialsymmetrische Potentiale ϕ_i definieren pro Partikel i die implizite Oberfläche. Für n Partikel mit den Positionen $\mathbf{x}_i$ $(i = 1 \dots n)$ kann das Potential $\phi(\mathbf{x})$ an

jeder Position $\mathbf{x}$ bestimmt werden (h_0: Kernel-Radius):

$$\phi(\mathbf{x}) = \sum_{i=1}^{n} \phi_i(\mathbf{x}), \tag{6.1}$$

$$\phi_i(\mathbf{x}) = \begin{cases} \sqrt{1 - \frac{\|\mathbf{x}-\mathbf{x}_i\|^2}{h_0^2}}, & \text{gdw. } \|\mathbf{x} - \mathbf{x}_i\| \leq h_0, \\ 0, & \text{sonst.} \end{cases} \tag{6.2}$$

Entsprechend eines Schwellwerte kann aus $\phi(\mathbf{x})$ effizient ein Höhenfeld erzeugt werden, indem der Schwellwert für jeden Höhenfeldeintrag entlang der z-Achse gefunden wird. Mit einer anschließenden Glättung kann die *Blobbyness* der Oberfläche verringert werden. Im anschließenden Abschnitt wird nun die Extraktion einer dreidimensionalen Oberfläche aus einer dreidimensionalen Partikelwolke behandelt.

6.2. Volumen: Geometriefreie Oberflächendarstellung dynamischer Partikelwolken im Bildraum

Die Oberflächenextraktion dreidimensionaler Flüssigkeiten erfolgt in der Regel mit einem *Marching-Cubes*-Algorithmus oder bei Gitter-basierten Verfahren mit einer *Level-Set*-Methode (vgl. Abschnitt 3.4.2). Für interaktive Umgebungen wurde die *Screen Space Meshes*-Methode präsentiert [MSD07], die ein polygonales Netz im Bildraum erzeugt. Im Folgenden wird eine neue, schnellere Methode vorgestellt, welche eine Extraktion freier Oberflächen aus Partikelwolken im Sichtfeld ermöglicht, ohne dabei ein polygonales Netz zu erzeugen. Stattdessen wird die Oberfläche Bildbasiert aus der Partikelwolke generiert. Das Verfahren ist unter Berücksichtigung der Möglichkeiten aktueller Graphikkartenhardware entwickelt worden und kann vollständig auf der GPU ausgeführt werden, so dass der hohe Grad an Parallelität direkt ausgenutzt wird. Die Methode kann eine Oberfläche für Millionen von Partikeln im Sichtfeld bei interaktiven Frameraten erzeugen. Der Algorithmus kann für beliebige Partikelwolken verwendet werden, wenngleich der Schwerpunkt der Anwendung im Bereich der dynamischen und interaktiven Flüssigkeiten liegt.

Die Grundidee des Algorithmus liegt in einer Dimensions-Reduktion des dreidimensionalen Problems: Die Oberflächenextraktion eines dreidimensionalen Volumendatensatzes wird auf ein zweidimensionales Problem reduziert. Der in der Regel zur Oberflächenextraktion verwendete Marching Cubes Algorithmus oder einer seiner Derivate zum Beispiel erzeugt die dreidimensionale polygonale Oberfläche einer implizit gegebenen Oberfläche. Dieser Vorgang muss bei dynamischen Objekten, wie es zum Beispiel dynamische Flüssigkeiten sind, in jedem Zeitschritt wiederholt werden. Der große Nachteil dieses Vorgehens liegt darin, dass zahlreiche Polygone in Bereichen erzeugt werden, die im nachfolgenden *Rendering*-Pass nicht dargestellt werden oder nicht sichtbar sind und damit den Aufwand unnötigerweise erheblich vergrößern. Ziel der dynamischen Oberflächenextraktion in interaktiven Umgebun-

gen ist die Darstellung im Sichtbereich, so dass idealerweise lediglich im sichtbaren Bereich eine Oberfläche erzeugt wird.

Unter Berücksichtigung dieser Anforderungen bietet sich eine Oberflächenextraktion im Bildraum an. Oberflächen werden lediglich im Sichtbereich erzeugt. Diesem Paradigma folgt der hier beschriebene Algorithmus, wobei keinerlei Polygone erzeugt werden, hingegen die Oberfläche Bild-basiert aus einer Tiefen-Map erzeugt wird. Damit ergeben sich die Vorteile einer

- geometriefreien Oberflächenextraktion im
- Sichtbereich bei
- hoher Performanz.

Die Grundidee besteht darin, jegliche Operationen (zum Beispiel Glättungen oder Beleuchtungen) im Bildraum durchzuführen, um die Geschwindigkeitsvorteile aktueller Graphikkartenhardware optimal einsetzen zu können und keinerlei Berechnungen für nicht sichtbare Objekte/Objektteile durchführen zu müssen. Die Geschwindigkeit der Methode in interaktiven Umgebungen – vor allem für viele Partikel (Dimension > 100.000) – ist der Hauptvorteil gegenüber anderen Ansätzen (wie zum Beispiel den *Screen Space Meshes* [MSD07]), die insbesondere daraus resultiert, dass keinerlei Polygone erzeugt werden und kein Datenaustausch von der GPU zur CPU stattfindet.

6.2.1. Prinzip

Die präsentierte Methode benötigt als Eingabe die N Positionen einer unstrukturierten 3D Partikelwolke $\mathbf{x}_1, \ldots, \mathbf{x}_N \in \mathbb{R}^3$ und die homogene Projektionsmatrix $\mathcal{P} \in \mathbb{R}^{4\times4}$. Anhand des *Splat*-Radius r_p, der die Größe der *Splats* beschreibt, und des Filterkernels der Größe k, der die Glättungsstärke beschreibt, wird das visuelle Verhalten der Oberfläche gesteuert. Der Basis-Algorithmus besteht aus folgenden vier Schritten (Abb. 6.2), die im weiteren Verlauf dieses Abschnitts separat behandelt werden:

1. Erzeuge Tiefen-Map $\mathbf{Z}$ aus 3D Partikelwolke,
2. Erzeuge geglättete Tiefen-Map $\mathbf{Z}_{\text{smooth}}$,
3. Bestimme Normalen-Map $\mathbf{N}$ der geglätteten Tiefen-Map,
4. Per-Fragment Beleuchtung im Bildraum unter Verwendung von $\mathbf{Z}_{\text{smooth}}$, der Bildraumkoordinate, $\mathbf{N}$ und der Position der Lichtquelle im Bildraum.

Zusammengefasst ergibt sich somit folgender Ablauf:

$$\text{Partikelwolke} \rightarrow \mathbf{Z} \rightarrow \mathbf{Z}_{\text{smooth}} \rightarrow \mathbf{N} \rightarrow \text{Beleuchtung}. \tag{6.3}$$

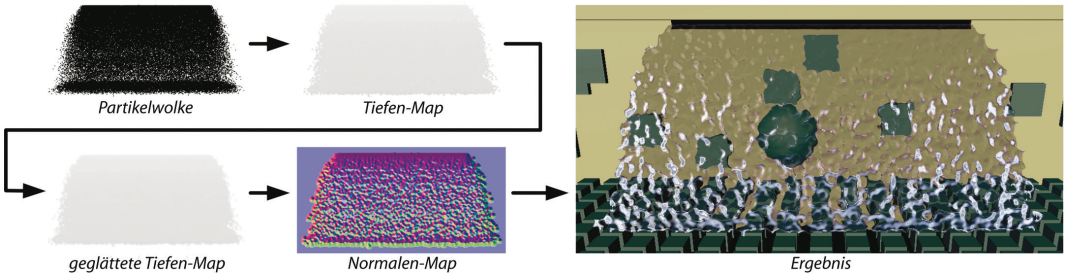


Abb. 6.2.: Hauptschritte der vorgestellten Methode (Wasserfall, Frontalansicht).

6.2.1.1. Erzeugung der Tiefen-Map

Zunächst wird eine Tiefen-Map $\mathbf{Z} \in \mathbb{R}^{W \times H}$ mit den Tiefenwerten $z_{i,j}$ in der gewünschten Auflösung $W \times H$ erzeugt (siehe Abbildung 6.2). Dazu wird jedes Partikel mit Position $\mathbf{x}_i$ als zirkularer *Point Splat* im Bildraum dargestellt. Wegen der homogenen Transformation sind die z-Werte in Bildschirmkoordinaten bei perspektivischen Projektionen nicht linear verteilt. Lediglich im Spezialfall orthogonaler Projektionen können die z-Werte ohne weitere Vorverarbeitung genutzt werden. Aus diesem Grund werden die homogenen Koordinaten $\mathbf{x} = (x, y, z, w)^T$ folgendermaßen in die Bildraumkoordinaten $(x_p, y_p, z_p)^T$ transformiert:

$$\begin{pmatrix} x_p \\ y_p \\ z_p \end{pmatrix} = \begin{pmatrix} W \cdot (\frac{1}{2} + \frac{x/w}{2}) \\ H \cdot (\frac{1}{2} + \frac{y/w}{2}) \\ z \end{pmatrix}. \tag{6.4}$$

Dabei beschreiben H und W die horizontale und vertikale Auflösung des Bildraumes. Somit wird die z-Koordinate linearisiert und beschreibt den Abstand zur Kamera.

Der *Point Splat*-Radius r_p beeinflusst maßgeblich die visuelle Erscheinung der Flüssigkeit. Der Radius kann prinzipiell in Abhängigkeit benachbarter Partikel im Bildraum abgeschätzt werden. Dazu kann zum Beispiel die in [ZPvBG01] vorgestellte Methode genutzt werden. Da die hier beschriebene Methode allerdings für dynamische Partikelwolken entwickelt wurde, müsste dieser Schritt für jeden einzelnen Simulationsschritt explizit durchgeführt werden. Für viele Partikel ist diese Abschätzung jedoch nicht in Echtzeit möglich, da die notwendige topologische Nachbarschaftssuche für hohe Anzahlen von Partikeln aufwendig ist. Innerhalb einer SPH-Simulation kann r_p innerhalb einer Vorberechnung nach oben abgeschätzt werden. Dazu wird der *Splat*-Radius r_p während eines Simulationslaufes als durchschnittlicher Abstand benachbarter Partikel im Bildraum bestimmt. Die projezierten Radien r_x, r_y, r_z von r_p in Weltkoordinaten können folgendermaßen berechnet werden [MSD07]:

(a) (b) (c)

Abb. 6.3.: Beispiel: Glas. Verschiedene Glättungsfilter zur Oberflächenextraktion: Konstanter Filterkernel (a), Dreiecksfilter (b), Binomialfilter (c).

$$\begin{pmatrix} r_x \\ r_y \\ r_z \end{pmatrix} = \begin{pmatrix} r_p W \sqrt{p_{1,1}^2 + p_{1,2}^2 + p_{1,3}^2}/w \\ r_p H \sqrt{p_{2,1}^2 + p_{2,2}^2 + p_{2,3}^2}/w \\ r_p \end{pmatrix}, \tag{6.5}$$

$p_{i,j}$ sind die Elemente der Projektionsmatrix $\mathcal{P}$. Somit werden die *Splats* entsprechend ihres Abstandes zur Kamera skaliert. Bei einem Bildschirmverhältnis entsprechend W/H werden Verzerrungen verhindert und die Partikel erscheinen kreisförmig im Bildraum ($r_p = r_x = r_y$). Die Tiefen-Map $\mathbf{Z}$ wird schließlich durch derartige Darstellung aller Partikel im Bildraum erzeugt.

6.2.1.2. Oberflächenerzeugung durch Glättung

Zur Erzeugung einer geglätteten Oberfläche im dreidimensionalen Raum, wird der Tiefen-Buffer mit einem Tiefpassfilter geglättet. Verschiedene Filter wurden im Rahmen dieser Arbeit untersucht und die beschriebene Methode hat sich als ausgesprochen robust gegenüber verschiedener Filterkernel herausgestellt (Abb. 6.3). In dieser Arbeit wird ein Binomialfilter [AL96] verwendet. Dieser ist eine gute Approximation eines Gaußfilters, dessen Kernel extrem effizient mit ganzzahligen Operationen berechnet werden kann:

$$b_{i,j} = \frac{1}{2^{n+m-2}} \begin{pmatrix} m-1 \\ i \end{pmatrix} \begin{pmatrix} n-1 \\ j \end{pmatrix}, \tag{6.6}$$

mit $i = 0 \ldots m-1$, $j = 0 \ldots n-1$ und $\begin{pmatrix} r \\ q \end{pmatrix} = \frac{r!}{q!(r-q)!}$.

Die quadratische Kernelgröße $k = m = n$ wird abgeschätzt über die *Point Splat*-Größe r_p: $k = r_p$. Beginnend mit dieser Abschätzung kann das visuelle Ergebnis mit

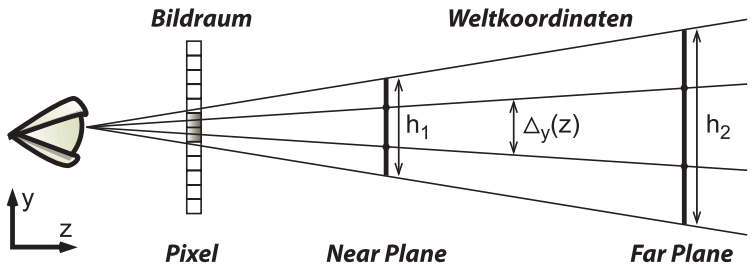


Abb. 6.4.: Berechnung des Gitterabstandes auf Pixelbasis im Bildraum für $\triangle_y(z)$.

dem Parameter k variiert werden. Eine geeignete Kernelgröße ist essentiell für die Qualität der Ergebnisse. Wird der Kernel zu klein gewählt, können Löcher in der Oberfläche entstehen — wird er hingegen zu groß gewählt, wird das Ablösen einzelner Partikel erschwert und die Oberfläche wird intensiv geglättet, so dass Details verschwinden.

6.2.1.3. Normalenberechnung

Anhand des geglätteten Tiefenbilds $\mathbf{Z}_{\text{smooth}}$, welches die Tiefeninformationen der Oberfläche in Bildkoordinaten darstellt, kann die Normalen-Map $\mathbf{N} \in \mathbb{R}^{W \times H}$ erzeugt werden. Dazu wird das normierte Kreuzprodukt der Tangentenvektoren für jeden Bildpunkt bestimmt. Das geglättete Tiefenbild kann als Höhenfeld betrachtet werden, welches aus Kamerasicht von oben betrachtet wird. Jede Koordinate im Bildraum x_p, y_p, z_p bildet Koordinaten in Weltkoordinaten ab. Im Fall perspektivischer Projektionen hängt der Abstand benachbarter Koordinaten in Weltkoordinaten vom Abstand zur Kamera Position in Bildkoordinaten ab. Somit können benachbarte Pixel im Bildraum aufgrund von Parallaxe verschiedene Abstände besitzen — in Abhängigkeit des Abstandes zur Kamera.

Werden zwei Objekte verglichen, eines nah an der *Front-Clipping-Plane* und eines nah an der *Back-Clipping-Plane*, so erscheint das nah liegende Objekt wesentlich größer. Somit besitzen benachbarte Pixel auf beiden Objekten verschiedene Entfernungen in Abhängigkeit des Tiefenwertes zueinander. Die aktuellen Abstände zum benachbarten Höhenfeldeintrag können anhand der z Koordinate und der linearen Skalierung zwischen der *Near-Plane* w_1, h_1 und *Far-Plane* w_2, h_2 bestimmt werden (Abb. 6.4):

$$\triangle_x(z) = w_1 + (w_2 - w_1)z \tag{6.7}$$

$$\triangle_y(z) = h_1 + (h_2 - h_1)z. \tag{6.8}$$

Unter Berücksichtigung dieser Werte können die partiellen Ableitungen der geglätteten Tiefen-Map diskretisiert werden. Es ergeben sich die Tangenten $(\triangle_x(z), 0, \triangle z_x)^T$ und $(0, \triangle_y(z), \triangle z_y)^T$ — mit $\triangle z_x = \mathrm{Z}_{\text{smooth}}(x+1, y) - \mathrm{Z}_{\text{smooth}}(x, y)$ und $\triangle z_y = \mathrm{Z}_{\text{smooth}}(x, y+1) - \mathrm{Z}_{\text{smooth}}(x, y)$. Die Tangenten ergeben im normierten

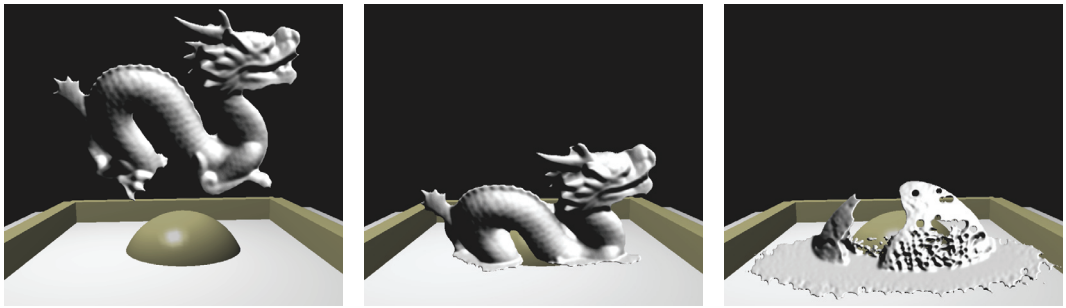


Abb. 6.5.: Stanford Dragon, dargestellt mit dem Phong'schen Beleuchtungsmodell bei Simulations-Zeitschritten t_i $(i = 0, 5, 10)$.

Kreuzprodukt die Normalen-Map **N**.

Die oben beschriebene Methode setzt voraus, dass die Normalen der Partikel nicht gegeben sind. Falls sie jedoch gegeben sein sollten, so können diese direkt als Normalen-*Splats* in die Normalen-Map gerendert, geglättet und verwendet werden.

6.2.1.4. Beleuchtung

Alle Beleuchtungsschritte werden im Bildraum durchgeführt. Somit können sie auf einer per-Fragment-Basis mit Hilfe der Tiefen-Map **Z** und der Normalen-Map **N** realisiert werden. Mit diesen beiden Maps und den bekannten xy-Koordinaten im Bildraum sind alle notwendigen Oberflächeninformationen für eine Beleuchtungsberechnung gegeben. Zunächst wird die Lichtposition l_p und die Kameraposition c_p mit Gleichung 6.4 im Bildraum bestimmt. Anhand dieser Positionen kann der Lichtvektor $\hat{\mathbf{l}}$ und Blickvektor $\hat{\mathbf{v}}$ für jeden Eintrag in **Z** über Differenzen berechnet werden. Damit sind die notwendigen Informationen gegeben, um ein Beleuchtungsmodell anwenden zu können (zum Beispiel das Phong'sche Beleuchtungsmodell — Abb. 6.5), oder um optische Effekte zu approximieren.

6.2.1.5. GPU-basierte Implementierung

Der beschriebene Algorithmus ist für die effiziente Ausführung auf der GPU entworfen worden. Die beschriebenen vier Schritte (Abb. 6.2) werden Bild-basiert auf Texturbasis implementiert. Somit stellt jeder einzelne Schritt eine Filterverarbeitung dar, die lediglich auf umliegende Fragmente zugreift. Mit Hilfe der *Render to Texture* Funktionalität, werden die entsprechenden Texturen direkt auf der GPU erzeugt und es findet kein teurer Datentransfer zwischen GPU und CPU statt. Jeder Schritt der Methode wird also im Rahmen eines einzigen *Texture Render Calls* ausgeführt.

6.2.1.6. Beispiele

Zur Bewertung des Laufzeitverhaltens und der Qualität der vorgestellten Methode zur polygonfreien Oberflächendarstellung wurden verschiedene Szenarien untersucht.

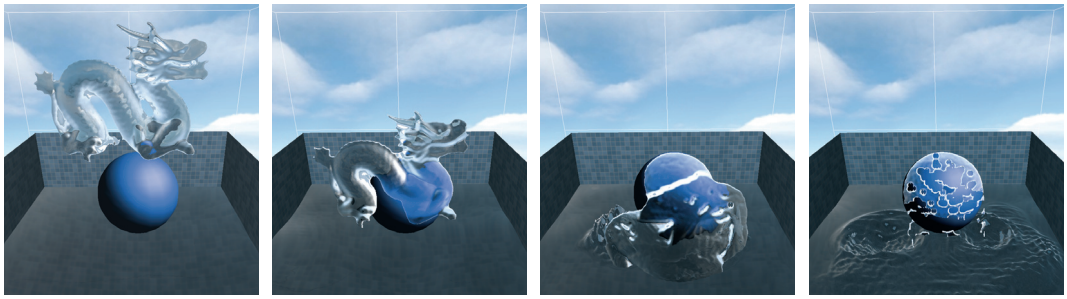


Abb. 6.6.: Liquider Drachen (64,474 Partikel) unter Einfluss der Gravitation im Pool bei interaktiven Frameraten (Zeitschritte t_i mit $i = 0, 10, 20, 30$).

Im Folgenden werden die einzelnen Experimente und ihre jeweilige Simulationsbasis beschrieben. Die Ergebnisse werden in den beiden anschließenden Abschnitten vorgestellt und diskutiert.

Basis sind jeweils dreidimensionale Echtzeit-Partikelsimulationen mit bis zu 140.000 Partikel. Nicht alle Simulationszenarien verwenden eine Navier-Stokes Simulation. Deshalb entspricht die Bewegung der jeweils dargestellten Flüssigkeit nicht notwendigerweise demjenigen realer Flüssigkeiten. Da jedoch noch keine Technik vorgestellt wurde, die eine physikalisch-basierte Simulation derartiger Partikelanzahlen in Echtzeit auf aktueller Standard-PC-Technik ermöglicht, wurden unter anderem auch andere Partikelsysteme untersucht. Die Art der Simulation ist in erster Linie für die beschriebene Extraktionsmethode nicht von Belang. Werden Navier-Stokes Simulationen mit mehr als 100.000 Partikel in Zukunft vorgestellt, können diese mit der hier beschriebenen Methode in Echtzeit dargestellt werden. Im Folgenden werden die einzelnen Experimente kurz vorgestellt.

Glas, Würfel (Abb. 6.3, 6.9) Es wird eine interaktive SPH-Simulation mit 2.000 bzw. 3.000 simulierten Partikeln genutzt. In der GPU-basierten Partikelsimulation werden 65.536 Partikel verwendet (Abb. 6.8).

Brechende Wellen Die brechenden Wellen (Abb. 5.25, Abschnitt 5.2.2) wurde mit 140.000 Partikeln und 200 Schichten simuliert. Als Simulationsverfahren wurde das im Rahmen dieser Arbeit entwickelte Verfahren zur Echtzeit-Simulation brechender Wellen genutzt (Abschnitt 5.1.4).

Wasserfall Der Wasserfall wurde mit einem Massepunkt-Simulator realisiert (siehe Abschnitt 5.1.2.3). Der Wasserfall wird dabei von 60.000–100.000 Partikeln repräsentiert.

Drache Der Dragon aus dem *Stanford Repository* (Abb. 6.6) wird von 64.474 Parikeln repräsentiert. Die Partikelpositionen entsprechen den Vertexpositionen des

Beispiel	Sim. [ms]	512 **Z** [ms]	512 Oberfl. [ms]	512 FPS [1/s]	1024 **Z** [ms]	1024 Oberfl. [ms]	1024 FPS [1/s]	512/1024 **Z** [ms]	512/1024 Oberfl. [ms]	512/1024 FPS [1/s]
Drachen	15,0	1,9	0,13	58,7	3,3	2,8	47,3	1,9	0,13	58,7
Wasserfall	22,1	2,9	0,53	39,1	3,1	2,6	35,9	2,9	0,61	38,7
Br.Welle	9,3	9,0	0,43	53,2	12,0	2,7	41,5	9,0	0,46	53,1
Glas	13,5	3,5	0,12	58,4	5,1	3,1	46,0	3,5	0,26	57,9
Würfel	13,3	5,3	0,57	52,9	9,0	3,8	38,7	5,3	0,73	52,5

Tab. 6.1.: Laufzeitmessungen der beschriebenen Methode in den verschiedenen Szenarien. Gegeben sind die Zeiten der Simulation (Sim), der Szenen- und Punktdarstellung (inklusive der Erzeugung der Tiefen-Map **Z**) und der vorgestellten Methode zur Oberflächendarstellung (Oberfl.) — bei verschiedenen Auflösungen.

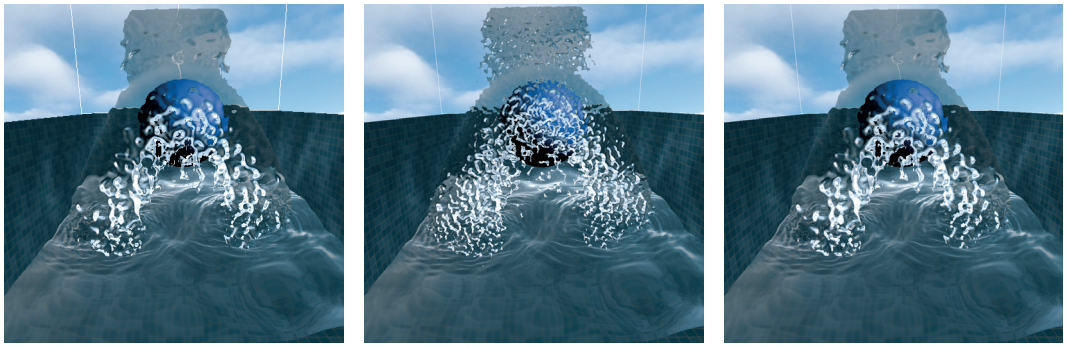
polygonalen Objektes. Somit wird lediglich die Oberfläche des Drachen mit Partikeln abgetastet. Das Fallen des Drachen und die Partikelkollisionen werden ebenfalls durch eine Massepunkt-Simulation realisiert.

Pool Szenen Zur realistischen Darstellung wurden einige der oben beschriebenen Experimente noch mit der dreidimensionalen Strömungssimulation in Verbindung mit einer Oberflächensimulation gekoppelt (Abschnitt 5.1.3). So kann zusätzlich zu den mit der beschriebenen Methode generierten dreidimensionalen Objekten eine interaktive Flüssigkeitsoberfläche simuliert werden, die mit den fallenden Flüssigkeiten interagieren kann. Dazu wird ein Impuls auf der Flüssigkeitsoberfläche gesetzt, wenn ein Partikel die Wasseroberfläche durchschneidet. So können detaillierte Wellenschläge der einschlagenden Partikel erzeugt werden.

Asian Dragon Neben den vorhergehenden dynamischen Partikelwolken wurde das beschriebene Verfahren auch an einem statischen Objekt mit großer Partikelzahl untersucht. Dazu wurde das statische Objekt des *Asian Dragon* aus dem *Stanford Repository* mit 3.609.455 Partikeln verwendet, wobei jede Vertexposition als Partikelposition verwendet wird und die Kanten vernachlässigt werden (Abb. 6.12).

6.2.2. Ergebnisse und Diskussion

Ergebnisse Die vorgestellten Ergebnisse wurden mit einer GPU-basierten Umsetzung der hier beschriebenen Oberflächendarstellungsmethode erzielt. Die jeweilige zugrundeliegenden Partikel-Simulationen wurde CPU-basiert (*Single Core*) implementiert. Laufzeitmessungen sind in Tabelle 6.1 dargestellt. Die Messungen erfolgten, soweit nicht anders angegeben, auf einer Dual-Core 2,6 GHz AMD Athlon 64 CPU, 2 GB RAM und einer auf dem ATI Radeon x1900 Chip basierende GPU (512MB). Es sind unter anderem Laufzeitmessungen für Bildraum-Auflösungen von 512×512 und 1024×1024 angegeben. Alle gemessenen Frameraten liegen im Bereich von 30–60 FPS und erlauben volle Interaktivität. Herauszustellen ist, dass



(a) (b) (c)

Abb. 6.7.: Wasserfall. Dieselbe Szene ist mit verschiedenen Screen-Space Auflösungen dargestellt: 512×512 (a), 1024×1024 (b), Szene: 1024×1024 – Flüssigkeit: 512×512 (c). Details werden in Abschnitt 6.2.2 gegeben.

die Simulationszeit in der Regel die Oberflächenextraktion und Darstellungszeit mit der beschriebenen Darstellungsmethode übersteigt. Bisher war die komplexe dreidimensionale Oberflächenextraktion oftmals die Komponente, die die Anzahl repräsentierbarer Partikel in Echtzeit maßgeblich verringerte.

Da es sich bei der vorgestellten Methode um ein Bild-basiertes Verfahren handelt, hängt der Aufwand der Methode direkt von der Anzahl zu verarbeiteter Fragmente ab. Aus diesem Grund kann es hilfreich sein, den Algorithmus bis zur Erzeugung der Normalen-Map in einem auflösungsreduzierten Bildraum durchzuführen. Der Brechungs- und Reflexionspass kann dann in der vollen Auflösung durchgeführt werden. So kann die Performanz erheblich verbessert werden, da die aufwendige Filterung in einem reduzierten Bildraum durchgeführt wird, die Reflexions- und Brechungseigenschaften jedoch bei voller Auflösung bestimmt werden. Die Messungen der Ausführungsgeschwindigkeiten für dieses Vorgehen sind ebenfalls in Tabelle 6.1 gegeben – dabei wird bis zur Normalen-Map-Erzeugung eine 512×512 Textur verwendet, die anschließend bilinear auf 1024×1024 skaliert wird. In Abbildung 6.7 sind graphische Beispiele der verschiedenen Auflösungen dargestellt. Das beschriebene Vorgehen kann ein gutes Verhältnis von Qualität zu Performanz erzielen. Alle anderen hier gezeigten Bilder zur beschriebenen Methode sind mit diesem Verfahren generiert (mit Ausnahme des *Asian Dragons*, der bei 1024×1024 erzeugt wurde). Wie zu erkennen ist, hängt die Topologie der extrahierten Oberfläche von der verwendeten Auflösung ab. Dies ist ein unerwünschter Effekt, liegt aber in dem Bildraum-Ansatz prinzipiell begründet und ist somit unvermeidbar. Wie in Tabelle 6.1 zu erkennen ist, reduziert sich die Performanz in der Auflösung von 1024×1024 erwähnenswert. Dieser Effekt liegt daran, dass der Aufwand prinzipiell quadratisch ist und in diesem Fall um das vierfache steigt, und somit der Fragmentshader stärker belastet ist.

Die vorgestellte Methode wurde mit der ebenfalls im Rahmen dieser Arbeit ent-

wickelten Methode zur Strömungs- und Oberflächensimulation gekoppelt (siehe Abschnitt 5.1.3). Die Ergebnisse sind in Abbildung 6.6 und 6.7 dargestellt und erzielen Geschwindigkeiten von etwa 25Hz. Beide Simulationen wurden auf nur einem Core durchgeführt. Lediglich die Oberflächenextraktion des Höhenfeldes wurde auf dem zweiten Core ausgeführt. Die Performanz der Oberflächenextraktionsmethode entspricht dabei den Werten, die in Tabelle 6.1 gegeben wurden.

Die Komplexität der Methode skaliert linear. Die Komplexität der Konstruktion der Tiefen-Map hängt linear von der Anzahl verwendeter Partikel n ab: $O(n)$. Dieser Schritt besteht in der Praxis daraus, die Partikel als Punktobjekte in den Framebuffer zu rendern. In den folgenden Schritten hängt die Komplexität lediglich von der Anzahl der verarbeiteten Fragmente ab: $O(W \times H)$. Somit skaliert der Aufwand linear in Abhängigkeit der Partikelanzahl und verwendeter Fragmente.

Das Beispiel *Asian Dragon* mit über 3,5 Millionen Partikeln demonstriert die Methode für dichte Punktwolken. Die Oberfläche wird im Bildbereich mit 13 FPS extrahiert und dargestellt (Graphikkarte: nVidia GeForce GTX 280) — bei einer Auflösung von 1024×1024. Bei derartig hohen Dichten kann die beschriebene Methode sehr detaillierte Ergebnisse erzielen, was für Nicht-Echtzeitumgebungen im Rahmen eines Vorschaumodus von Interesse sein kann. Dennoch liegt das potentielle Einsatzgebiet vornehmlich im Bereich der interaktiven, aber dynamischen Punktwolken. Dabei sei jedoch beachtet, dass die Partikelanzahl in interaktiven und zugleich dynamischen Umgebungen in der Regel wesentlich geringer ist, und folglich nicht so detaillierte Ergebnisse erzielt werden können, wie bei statischen Objekten.

Die Verwendung der GPU zur Simulation kann eine weitere wesentliche Beschleunigung erzielen (siehe Beispiele in Abb. 6.8). Zum einen kann die hohe Parallelität der Graphikkarte eine effiziente Simulation ermöglichen, zum anderen entfällt der Austausch der Partikelpositionen zwischen CPU und GPU. Dieser stellt einen weiteren hohen Kostenfaktor dar. So können wesentlich mehr Partikel in interaktiven Umgebungen simuliert und dargestellt werden, als es CPU-basierte Verfahren ermöglichen.

Diskussion Im Folgenden werden Möglichkeiten und Einschränkungen der beschriebenen Methode zur Oberflächenextraktion aus Partikelwolken diskutiert. Zudem erfolgt ein Vergleich mit existierenden Techniken.

Die Hauptvorteil der beschriebenen Methode ist die gute Performanz mit der Oberflächen im Sichtbereich erzeugt werden können — damit ist die Methode prädestiniert für interaktive Umgebungen. Wie in den Beispielen zu erkennen ist, sind die Ergebnisse jedoch verhältnismäßig undetailliert. Die Ursache liegt hauptsächlich in der verhältnismäßig geringen Partikelanzahl derartiger dynamischer Simulationen, die eine detaillierte Oberflächenextraktion per se verhindern, da die Oberflächeninformationen mit wenigen Partikeln schlecht aufgelöst sind — und weniger an dem Verfahren selbst. Bei dichten Partikelwolken jedoch, wie zum Beispiel dem *Asian Dragon*-Beispiel, können detaillierte Ergebnisse erzielt werden. Temporäre Inkohärenzen der Oberfläche können bei Partikelwolken geringer

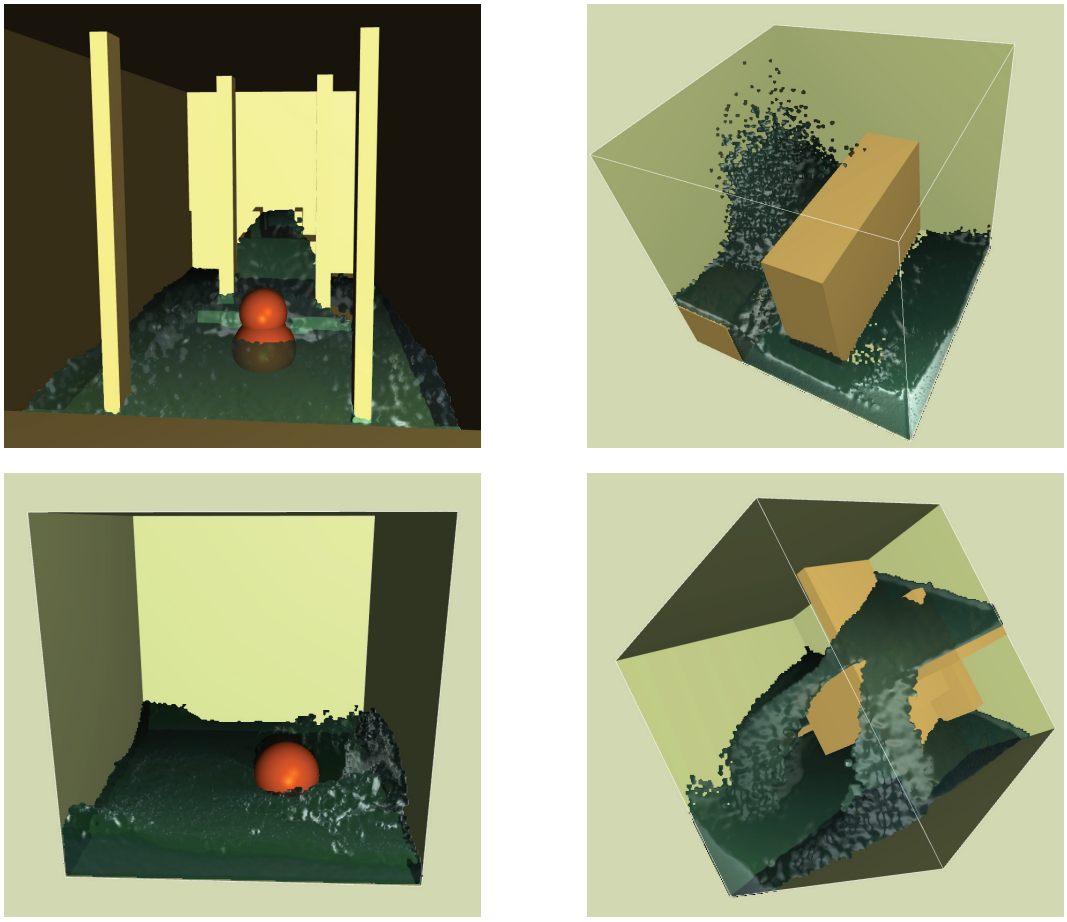
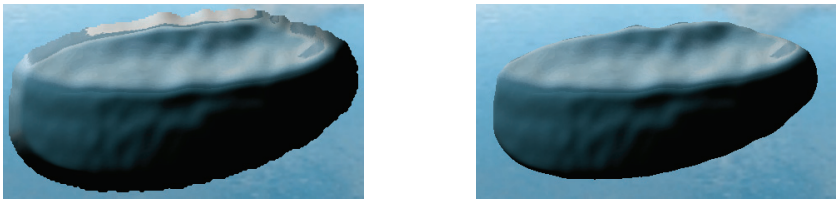


Abb. 6.8.: GPU-basierte Simulation. 65536 Partikel simuliert und dargestellt mit 50-90 Hz. Die Partikelpositionen müssen nicht zwischen CPU und GPU ausgetauscht werden. Dadurch kann die Performanz wesentlich verbessert werden.

Dichte während der Animation entstehen. Während der Darstellung dynamischer Flüssigkeiten sind diese aufgrund der hohen Dynamik jedoch kaum sichtbar.

Silhouetten-Artefakt Reduktion Die Glättung der Tiefen-Map zur Oberflächenerzeugung führt an der Silhouette des Flüssigkeitsvolumens zu Artefakten, da die harte Kante herausgefiltert wird. Somit werden für die geglättete Kante Normalen erzeugt, die unrealistische Beleuchtungseffekte nach sich ziehen (Abb. 6.9). An diesen Konturen können zudem unrealistische Beleuchtungseffekte auftreten, wenn das Objekt zwischen Lichtquelle und Kameraposition liegt. Es können keine *Backface*-Normalen repräsentiert werden, da es sich um ein Bildraum-Verfahren handelt und lediglich sichtbare Oberflächen behandelt werden. Zur Unterdrückung der unerwünschten Kanteneffekte kann die geglättete Silhouette unterdrückt werden. Der prinzipielle Ablauf ist im Folgenden dargestellt:



(a) (b)

Abb. 6.9.: Beispiel der Silhouetten-Artefakt Reduktion (Glas). Ohne (a) und mit Silhouetten-Artefakt Reduktion (b).

1. Erzeuge binäre Tiefen-Map $\mathbf{B}$,
2. Erzeuge geglättete Tiefen-Map $\mathbf{S}$ aus $\mathbf{B}$,
3. Stelle Flüssigkeit dar, wenn $\mathbf{S}_{x,y} < 1$.

Entsprechend der Kernelgröße k wird die Silhouette verkleinert. Dazu wird ein Binärbild $\mathbf{B}$ der Tiefen-Map erzeugt,

$$\mathbf{B}(x,y) = \begin{cases} 1 & : \quad \mathbf{Z}(x,y) = 1 \\ 0 & : \quad \mathbf{Z}(x,y) < 1 \end{cases}, \tag{6.9}$$

welches mit dem gleichen Filter geglättet wird, wie das Tiefenbild (siehe Abschnitt 6.2.1.2). Daraus resultiert die Silhouetten-Map $\mathbf{S}$, die im Bereich der Silhouette Intensitätswerte ungleich Null oder Eins besitzt. Innerhalb der Silhouette beträgt der Wert Null. Somit kann dieser Intensitätswert verwendet werden, um die Silhouette zu verkleinern. Die resultierende Flüssigkeit wird abschließend lediglich an den Fragment Positionen dargestellt, die einen Silhouetten-Map Eintrag ungleich eins besitzen (Abb. 6.9).

Prinzipiell kann die Methode auch für statische Partikelmengen (zum Beispiel gestoppte Animation) verwendet werden. Jedoch können bei wenigen Partikeln an den Kanten *Blobby-Artifacts* auftreten, die ihren Ursprung in der Glättung des Höhenfeldes besitzen – siehe Abbildung 6.10. Jedoch werden die Artefakte die innerhalb der äußeren Kontur konkaver Objekte liegen (vom Blickpunkt aus gesehen) oder aber an der Kante zwischen räumlich getrennten Flüssigkeitsvolumina nicht behandelt. Im Falle mehrerer unabhängiger Volumina kann ein *Multipass-Rendering*-Verfahren das Problem lösen. Die einzelnen Flüssigkeiten werden sukzessiv dargestellt, so dass die Konturen jeweils unabhängig voneinander generiert und vor allem geglättet werden können – so können keine Glättungsartefakte zwischen den Flüssigkeiten entstehen.

Für konkave Objekte jedoch kann dieses Problem schwerlich in Echtzeit gelöst werden. Das Problem selbst wird durch die Verwendung des Binomial-Filters und der damit verbundenen Glättung der Kanten verursacht. Diese jedoch ist der essentielle Schritt, um eine geschlossene und glatte Oberfläche zu erhalten. Durch

Abb. 6.10.: Artefakte können an statischen Objekten während einer Transformation entstehen. Ein statisches Flüssigkeitsobjekt innerhalb einer turbulenten Situation wird um die y-Achse gedreht. Dabei verändert sich die Kontur leicht — jedoch weich. Dieses Beispiel verwendet lediglich 2000 Partikel — für signifikant mehr Partikel reduziert sich dieser Nebeneffekt, da sowohl der *Splat*-Radius als auch der Glättungskernel verkleinert werden kann.

die Verwendung kantenerhaltender Filter kann dieses Problem prinzipiell behandelt werden. Einfache kantenerhaltende Filter wie Median oder Bilaterale Filter sind ungeeignet, da die Tiefen-Map hauptsächlich aus diskreten Werten besteht und diese Filter dann für Partikelwolken mit wenig Partikeln keine geglättene und geschlossene Oberfläche erzeugen können. Zudem sind sie kostenintensiv, da diese Filter nicht separabel sind. Komplexere kantenerhaltende Rekonstruktionsalgorithmen (zum Beispiel [GLQ06, ASHU05, Str97]) können dem beschriebenen Problem per Definition entgegenwirken. Derartige Rekonstruktionsalgorithmen sind jedoch aufwendig und können derzeit noch nicht in Echtzeit ausgeführt werden. Ihre Laufzeiten liegen bereits für ein Bild der Größe 256×256 im Bereich von Sekunden. Ein Hauptaspekt für eine gute Performanz des Filterungsprozessess ist die Separierbarkeit des Filterkerns. Dieses Eigenschaft ist bei dem im Rahmen dieser Arbeit verwendeten Binomial-Filter gegeben. Dennoch besitzt der Filter-Schritt den größten Aufwand aller Schritte innerhalb der beschriebenen Methode.

Brechung an Vorder- und Rückseite Die Verwendung lediglich einfacher Brechung der Sichtstrahlen (also Brechung lediglich an der ersten sichtbaren Oberfläche) kann nur in geschlossenen Behältern (zum Beispiel ein Pool) zu akkuraten Ergebnissen führen. Im Fall der brechenden Welle zum Beispiel können Sichtstrahlen die Flüssigkeit jedoch wieder verlassen. Wird in solchen Situationen eine einfache Brechung verwendet, kann eine unplausible Flüssigkeitsdarstellung die Folge sein.

Die Methode kann jedoch prinzipiell auf die Verwendung doppelter Brechung angepasst werden. Die vorgestellte Methode extrahiert lediglich die sichtbaren Oberflächen im Bildraum — dieses sind die *Front-Faces*. Für eine doppelte Brechung sind jedoch die *Back-Faces* vonnöten. Diese können für konvexe Objekte in einem zweiten Pass erzeugt werden, wobei der Tiefen-Buffer-Vergleich in Schritt 1 (Erzeugung der

Tiefen-Map) der Methode umgekehrt wird. So kann die Oberfläche der kameraabgewandten Back-Faces generiert werden. Liegen beide Oberflächen-Maps vor, kann ein Echtzeit Brechungsverfahren angewendet werden. So könnte zum Beispiel mit dem Verfahren von Wyman [Wym05a] eine akkurate doppelte Brechung von Objekten innerhalb einer im Unendlichen liegenden *Environment Map* in Echtzeit dargestellt werden.

Dynamisches LoD für Partikel-basierte Simulationen Die *Splat*-Größe r_p und die Kernel-Größe k können innerhalb Partikel-basierter Simulationen zur Anpassung der Glättung der Oberfläche in statischen Situationen genutzt werden. Kleine Werte vergrößern die Details, lassen die Oberfläche jedoch *blobby* erscheinen. Umgekehrt erzielen große Werte eine glatte Darstellung, sind jedoch zur Repräsentation von Details nicht geeignet. So kann die Flüssigkeit auch bei niedriger Abtastung mit wenigen Partikeln eine glatte Oberfläche erhalten. In Situationen mit hoher Dynamik ist jedoch ein hoher Detailgrad zweckmäßig. Eine adaptive Glättung in Abhängigkeit der kinetischen Gesamtenergie der Massepunkte

$$E_{kin} = \sum_{i=1}^{n} \frac{1}{2} m_i \mathbf{v}_i^2, \tag{6.10}$$

eines Partikel Systems mit n Partikeln, Geschwindigkeiten $\mathbf{v}_i$ und Massen m_i ($i = 1 \ldots n$) genügt dieser Anforderung und kann effizient berechnet werden. In Abhängigkeit der Gesamtenergie werden die *Splat*- und Kernel-Größen angepasst. Somit erscheinen Flüssigkeiten in Ruhe glatt und in dynamischen Situationen detailliert.

Vergleich mit existierenden Verfahren Das beschriebene Verfahren hat bis zur Erzeugung der Tiefen-Map Ähnlichkeiten mit den *Screen Space Meshes* [MSD07]. Dort wird jedoch ein polygonales Gitter im Bildraum mit einem Derivat des Marching Squares Algorithmus auf der CPU erzeugt. Die Anzahl der erzeugten Dreiecke hängt dabei quadratisch von der Bildraumabtastung ab. Für einen Vergleich wurde die *Screen Space Meshes*-Methode im Rahmen dieser Arbeit nachimplementiert (Abb. 6.11). Da diese Implementierung jedoch unoptimiert ist, soll ein direkter Performanzvergleich nicht durchgeführt werden. Aber wie in Abbildung 6.12 zu erkennen ist, konvergiert die *Screen Space Meshes*-Methode qualitativ in Abhängigkeit der Abtastrate gegen die hier vorgestellte Methode. Der große Unterschied ist, dass die hier beschriebene Methode keinerlei Polygone erzeugt und die *Screen Space Meshes*-Methode im ungünstigsten Fall für eine Abtastung von Eins und einer Auflösung von 1024×1024 über 2 Millionen Dreiecke auf der CPU erzeugen würde. Dazu erfolgt die Normalenberechnung und eine Gitterglättung, inklusive dem zugehörigen, teuren Datentransfer zwischen GPU und CPU. Der Performanzvorteil der hier beschriebenen Methode erscheint somit offenkundig. Ein Vorteil der *Screen Space Meshes*-Methode ist jedoch die Detektion von Unstetigkeitsstellen — ein vergleichbarer

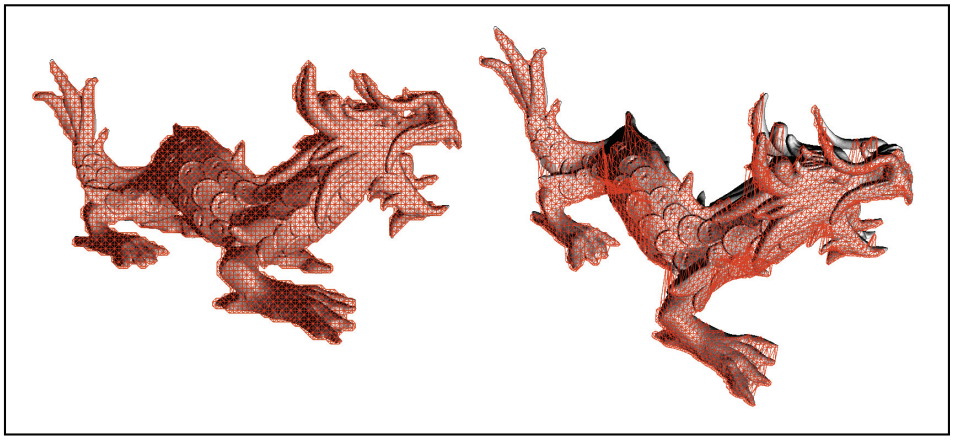


Abb. 6.11.: Prinzip der *Screen-Space Meshes*-Methode [MSD07]. Ein reguläres Gitter wird aus Kameraposition erzeugt (links), welches das Objekt im Screen-Space abtastet (rechts).

Ansatz für die hier gezeigte Technik wurde vorher im Rahmen der Silhouetten Artefakt Reduktion diskutiert, würde jedoch die Performanz wesentlich reduzieren.

Die hier beschriebene Methode erzielt auch bessere Performanz als die *Particle Splatting*-Methode [ALD06]. Diese Methode verwendet Alpha-Blending zur Interpolation von Normalen im Bildraum. Sie beschreiben Frameraten von 10 FPS in Umgebungen mit etwa 100.000 Partikeln und 100 FPS in Umgebungen mit 10.000 Partikeln. Diese Performanzangaben beziehen sich auf den reinen Render-Vorgang und beinhalten keinerlei Simulation. Dementsprechend ist das hier vorgestellte Verfahren diesem Verfahren in der Performanz überlegen. Dieser Vorteil resultiert nicht nur aus der besseren Graphikkartenhardware, die hier verwendet wurde, sondern vor allem aus der Tatsache, dass in [ALD06] ein Two-Pass-Rendering-Verfahren benutzt wird. Zudem resultiert die *Particle Splatting*-Methode in unrealistischen Silhouetten, an denen einzelne Partikel noch zu erkennen sind. Schließlich erscheinen die Oberflächen der hier vorgestellten Methode auch detaillierter und das hier verwendete Darstellungsmodell vergrößert den Realismus zusätzlich. Dieses wird um nun folgenden Kapitel präsentiert.

6.3. Zusammenfassung

Dieses Kapitel behandelte die Oberflächenextraktion aus Simulationsdaten. Zunächst wurde die Höhenfeld-basierte Oberflächenextraktion mit Hinblick auf dynamische Flüssigkeitsoberflächen thematisiert. Die Oberflächendarstellung dreidimensionaler Partikelwolken kann effizient mit dem hier vorgestellten Bild-basierten Ansatz erfolgen. Es handelt sich um einen GPU-basierten Ansatz, der im Bildraum arbeitet und somit direkt eine blickpunktabhängige LoD-Darstellung erlaubt. Die Methode erzeugt kein polygonales Gitter und kann deshalb ausgesprochen effizient

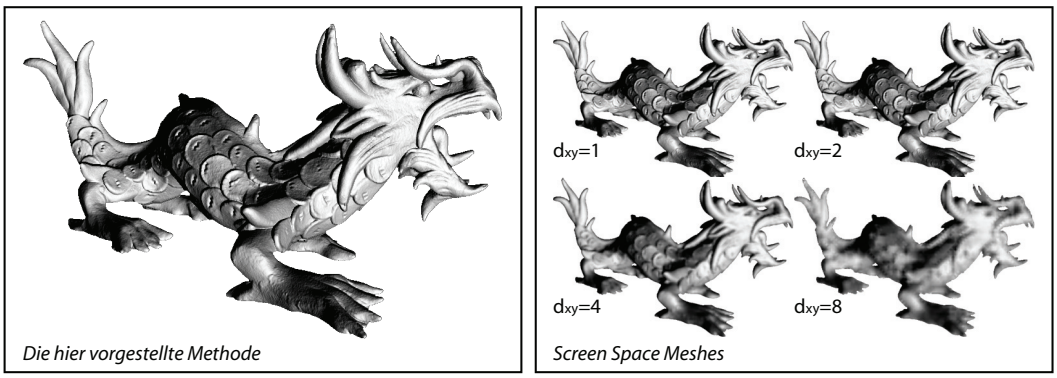


Abb. 6.12.: Asian Dragon (3,609,455 vertices). Vergleich der beschriebenen Methode (13 FPS, links) mit dem *Screen-Space Meshes*-Ansatz ohne Silhouetten Detektion bei verschiedenen Screen-Space Abtastraten *ss* (rechts). Die Ergebnisse konvergieren für kleine *ss* gegen die Ergebnisse, die mit dem hier beschriebenen Verfahren erzeugt wurden.

ausgeführt werden. Die Darstellung der extrahierten Oberflächen ist Thema des folgenden Kapitels.

7. Darstellung

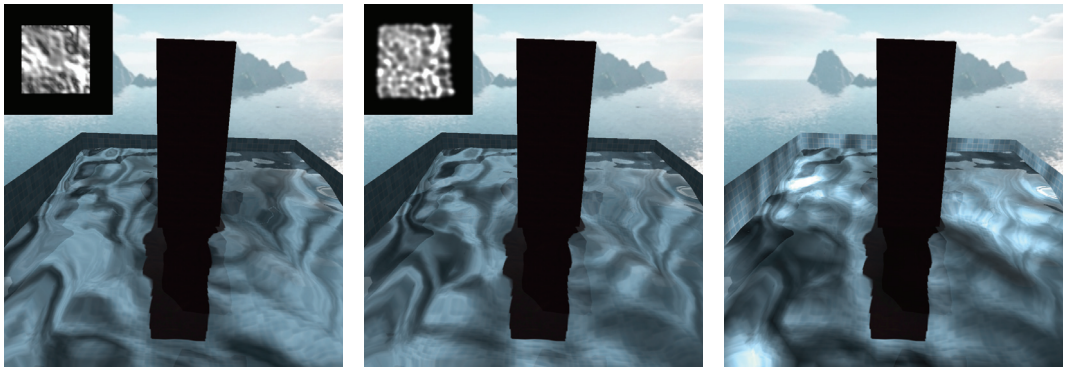
Nachdem in den beiden vorherigen Kapiteln mögliche Simulations- und Oberflächenextraktionsmethoden behandelt wurden, erfolgt nun die Präsentation von Methoden zur Darstellung optischer Eigenschaften. Die optischen Eigenschaften von Flüssigkeiten basieren vornehmlich auf der Darstellung von Reflexionen, Brechungen oder Absorptionen. Diese erfordern in Echtzeitumgebungen in der Regel die Verwendung von Approximationen. Kleinere Erweiterungen, auf existierenden Techniken basierend, werden im folgenden Abschnitt 7.1 gegeben. Ein Problem existierender Mapping-Techniken ist die Repräsentation optischer Reflexion und Brechung polygonaler Objekte, die die Flüssigkeitsoberfläche durchschneiden. Abschnitt 7.2 präsentiert eine Methode zur Reduktion von Artefakten für diese Ereignisse, mit Fokus auf den visuellen Übergang an der Grenzschicht der Flüssigkeit von Objekt zu gebrochenem Objekt und der brechungstypischen visuellen Skalierung und Winkeländerung.

Die nicht-realistische Darstellung von Flüssigkeiten liegt nicht im Fokus dieser Arbeit. Zur Repräsentation physikalischer Informationen kann allgemein aus dem Portfolio des Wissenschaftsbereiches der Strömungsvisualisierung geschöpft werden. Beispielhaft wird in Abschnitt 7.3 eine effiziente Beschriftungsmethode für Punktwolken vorgestellt. Mit dieser können zum Beispiel die physikalischen Eigenschaften Partikel-basierter Flüssigkeitssimulationen exploriert werden. Daneben liegen die zahlreiche Anwendungsmöglichkeiten in jeglichen Bereichen in denen eine interaktive Beschriftungsmethode gefragt ist.

7.1. Effiziente Approximationen

Approximationen von Kaustiken und Absorption werden im Folgenden behandelt, um den Realismus virtueller Flüssigkeiten zu vergrößern. Dann erfolgt ein kurzer Überblick über die schnelle Approximation von Reflexion und Brechung der Sichtstrahlen. Dieser stellt gleichzeitig die Überleitung zu dem anschließenden Abschnitt dar, der die optische Behandlung oberflächenschneidender, polygonaler Objekte thematisiert.

Kaustiken Kaustiken entstehen durch Reflexion und Brechung von Lichtstrahlen und resultieren in komplexen Lichterscheinungen. Sie tragen in hohem Maße zum Realismus virtueller Flüssigkeiten bei. Ein physikalisch akkurater Ansatz ist die Verwendung von *Light Ray Tracing*, also einem Ray Tracing der Lichtstrahlen (siehe



(a) (b) (c)

Abb. 7.1.: Kaustiken einer dynamischen Strömung um ein Objekt. Ein Funktion in Abhängigkeit der Oberflächenvariationen und -normalen kann Kaustiken approximieren (a). Physikalische Kaustiken (*Light Ray Tracing*) mit niedriger Abtastung ähneln der Approximation (b). Eine Überlagerung beider Verfahren resultiert in hohen Kontrasten (c).

auch Abschnitt 3.5). In interaktiven Umgebungen werden aus Geschwindigkeitsgründen jedoch oftmals niedrige Abtastungen gewählt, die extrapoliert werden. Die hohe Komplexität von Kaustiken im Allgemeinen führt dazu, dass die Korrektheit innerhalb dynamischer Szenen kaum vom Betrachter bewertet werden kann — so dass Vereinfachungen zur Darstellung von Kaustiken verwendet werden können.

Die Affinität zwischen Höhenfeld und niedrig abgetasteten Kaustiken wird im Folgenden verwendet, um eine ausgesprochen effiziente Kaustik-Map zu erzeugen. Diese kann anschließend auf die Szene projiziert werden. Da das Höhenfeld vollständig bestimmt ist — inklusive Normalen — kann eine Formel aufgestellt werden, die diese Informationen verwendet, um eine Kaustik-repräsentierende Map zu erzeugen, die aus Lichtposition auf die Szene projiziert wird. So kann mit den Höhenwerten $z_{min} < z < z_{max}$ und den zugehörigen Normalen $\mathbf{n} = (n_x, n_y, n_z)$ ein Kaustik-Intensitätswert berechnet werden. Im Rahmen dieser Arbeit wird folgende Formel verwendet:

$$I = a \cdot \frac{z - z_{min}}{z_{max} - z_{min}} + b \cdot \|n_x + n_y\| + c. \tag{7.1}$$

Mit den Konstanten a,b und c kann die Intensität der Kaustiken angepasst werden. Diese Approximation kann nicht die hochfrequenten Lichterscheinungen repräsentieren, die reale Kaustiken in der Regel besitzen. Dennoch bildet sie dynamische Kaustiken entsprechend der Oberflächenwellenbewegung ab und verbessert damit die Plausibilität der dargestellten Flüssigkeit. Der Vorteil liegt darin, dass die Kaustik-Map auf diese Weise GPU-basiert innerhalb einer einzigen Texturdarstellung, also sehr schnell, erzeugt werden kann (siehe dazu auch Tab. 7.1 im folgenden Abschnitt). Des Weiteren kann eine Überlagerung der Kaustik-Methode mit einer physikalisch-basierten Methode den Detailgrad verbessern (Abb. 7.1).

Absorption Die Inklusion von Absorption erzeugt zusätzlichen Realismus, da der Tiefeneindruck der Flüssigkeit verstärkt wird. Die dazu notwendige Berechnung erfordert Kenntnis über die Distanz, die ein Sichtstrahl innerhalb des absorbierenden Mediums zurücklegt. Diese Distanz kann effizient für jeden Sichtstrahl unter Verwendung der GPU im Bildraum approximiert werden. Im Falle Höhenfeld-basierter Oberflächen werden dazu die Tiefen-Map der Flüssigkeitsoberfläche und die der Szene ohne Flüssigkeit benötigt. Die Differenz der jeweiligen Tiefenwerte ist dann ein Maß für die Distanz, die die Sichtstrahlen in dem Medium zurücklegen (siehe dazu auch Abbildung 5.8, Abschnitt 5.1.1.4). Die Brechung von Sichtstrahlen an der freien Oberfläche wird bei der Bestimmung der Distanz nicht berücksichtigt. Die Absorption kann mathematisch bestimmt werden, wie es in Abschnitt 2.4 beschrieben wurde.

Im Falle dreidimensionaler Oberflächen kann im Prinzip die gleiche Methode angewendet werden. Um jedoch eine mögliche Rückseite der Flüssigkeit einzuschließen, wird in die Tiefen-Map der Szene noch die dem Betrachter abgewandte Seite der Flüssigkeit (zum Beispiel *Backface*-Polygone) hinzugefügt. So kann auch eine Absorption eines Volumens berücksichtigt werden.

Reflexion und Brechung Für die Darstellung von Reflexionen und Brechungen wird in dieser Arbeit eine Kombination aus *Environment Mapping* und Bild-basierten Ansätzen [Joh04, Sou05] verwendet. Dazu werden die reflektierten und gebrochenen Sichtstrahlen berechnet, wie es in Abschnitt 2.4 beschrieben wird. Anhand dieser wird auf die *Environment Map* zugegriffen. Für eine zusätzliche Bild-basierte Reflexion und Brechung von Objekten werden die berechneten Vektoren verwendet, um mit Hilfe einer Versetzung auf die Reflexions- oder Brechungs-Map zuzugreifen. Diese Maps werden in den gezeigten Beispielen mit unendlicher Ozeanoberfläche anhand der über der Oberfläche und der unter der Oberfläche liegenden Objekte erzeugt, wobei die über der Oberfläche liegenden Objekte an der approximierenden Oberflächenebene gespiegelt werden (vgl. dazu auch Abbildung 5.8, Abschnitt 5.1.1.4). Dieses Vorgehen basiert also auf einer dynamischen Abbildung der gespiegelten und unter der Oberfläche liegenden Objekte auf die Oberfläche und weniger auf einer akkuraten Reflexion oder Brechung. Um Artefakte hinter Objekten zu vermeiden, die aus dem Bild-basierten Ansatz resultieren, kann ein Tiefenwertvergleich zwischen Flüssigkeitsoberflächenpunkt und abgebildetem Objektpunkt das Problem lösen. Im Falle dreidimensionaler Flüssigkeiten kann zumindest die Brechung ebenso approximiert werden. Die Brechungs-Map wird dabei anhand der hinter dem Objekt liegenden Objekt generiert.

Die im Rahmen dieser Arbeit gezeigten Pool-Bilder verwenden die im nächsten Abschnitt beschriebene Methode der Reflexion und Brechung, die für Objekte entwickelt wurde, die die Flüssigkeitsoberfläche schneiden. Die Komposition der finalen Intensitäten erfolgt entsprechend der Fresnel-Gleichungen (siehe Abschnitt 2.4).

(a) (b) (c)

Abb. 7.2.: Beispiele der vorgestellten Methode zur schnellen Darstellung von Reflexionen und Brechungen oberflächenschneidender Objekte.

7.2. Optische Reflexion und Brechung oberflächenschneidender Objekte

Dieser Abschnitt beschreibt eine Methode zur schnellen Approximation von Reflexionen und Brechungen polygonaler Objekte, die sich nahe einer dynamischen Flüssigkeitsoberfläche befinden oder diese durchschneiden. Für gewellte Oberflächen ist gerade die Approximation optischer Brechung schwierig und führt mit einfachen Approximationsverfahren in der Regel zu Artefakten an der Grenzschicht: Das Objekt erscheint an der Grenzschicht zerrissen.

Ein *Ray Casting*-Ansatz, wie er in [BD06a] vorgestellt wurde, vermag einen Höhenfeld-basierten Untergrund darzustellen. Dieser Ansatz kann jedoch keine polygonalen Objekte optisch brechen, die zwischen Untergrund und Flüssigkeitsoberfläche liegen oder aber die Oberfläche durchschneiden. Diese Lücke wird im Folgenden mit einem Mapping reflektierter und gebrochener Abbilder polygonaler Objekte geschlossen. Fokus liegt auf der ununterbrochenen Objektdarstellung an der Grenzschicht und der brechungstypischen Größen- und Winkelveränderung der Bildobjekte in Abhängigkeit der Blickrichtung (siehe Beispiele in Abb. 7.2).

7.2.1. Prinzip

Die lokale Umgebung oberflächenschneidender Objekte wird pro Objekt planar approximiert, um die zugehörigen virtuellen Kameravektoren (siehe folgenden Abschnitt) zu bestimmen. Anhand dieser werden eine Brechungs-Map und eine Reflexions-Map erzeugt. Diese werden anschliessend unter Berücksichtigung des lokalen Normalenfeldes auf die Flüssigkeitsoberfläche projiziert. Die einzelnen Schritte des Verfahrens sind die Folgenden:

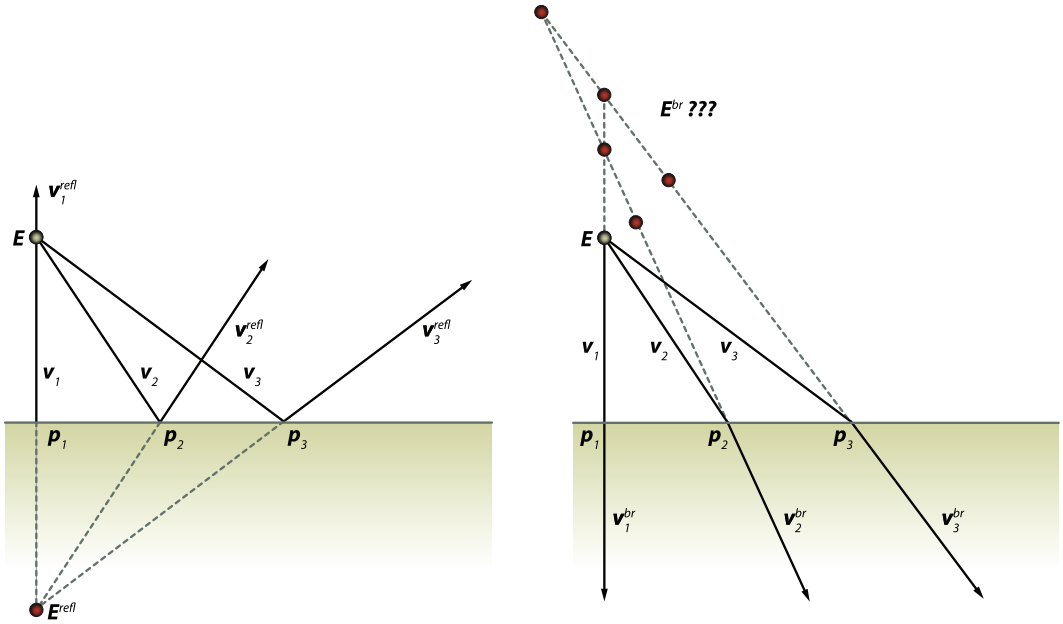


Abb. 7.3.: Prinzip der planaren Reflexion und Brechung. Ein planarer Spiegel mit Kameraposition **E**. Drei Sichtstrahlen werden beispielhaft reflektiert (links). Alle reflektierten Sichtstrahlen fokussieren in der virtuellen Kameraposition $\mathbf{E}^{refl}$. Die gebrochenen Sichtstrahlen bei einem Medienwechsel (hier: Luft/Wasser) fokussieren nicht (rechts).

1. Bestimme planare Approximation der Flüssigkeitsumgebung,
2. Bestimme virtuelle Kamerapositionen (Reflexion, Brechung),
3. Erzeuge Brechungs-Map, Reflexions-Map,
4. Projiziere Maps auf Flüssigkeitsoberfläche – unter Berücksichtigung der lokalen Flüssigkeitsumgebung.

Jedes Objekt wird dabei separat behandelt. Objekte, die die Flüssigkeitsoberfläche schneiden werden an der Schnittfläche in zwei Teile unterteilt: Einen über der Oberfläche liegenden und einen unter der Oberfläche liegenden. Zudem wird angenommen, dass der gebrochene Sichtstrahl das Flüssigkeitsvolumen nicht mehr verlässt. Somit ist eine einfache Brechung ausreichend. Diese Annahme ist für die meisten Situationen gültig.

7.2.1.1. Virtuelle Kamera

Aus Kamerasicht fokussieren alle reflektierten Sichtstrahlen in einem Punkt (Abb. 7.3a). Dieser Punkt wird im Folgenden als virtuell reflektierte Kameraposition bezeichnet und ist für eine Ebene über eine Spiegelung der Kameraposition an der Ebene definiert. Im Falle von Brechungen existiert ein derartiger Fokus nicht

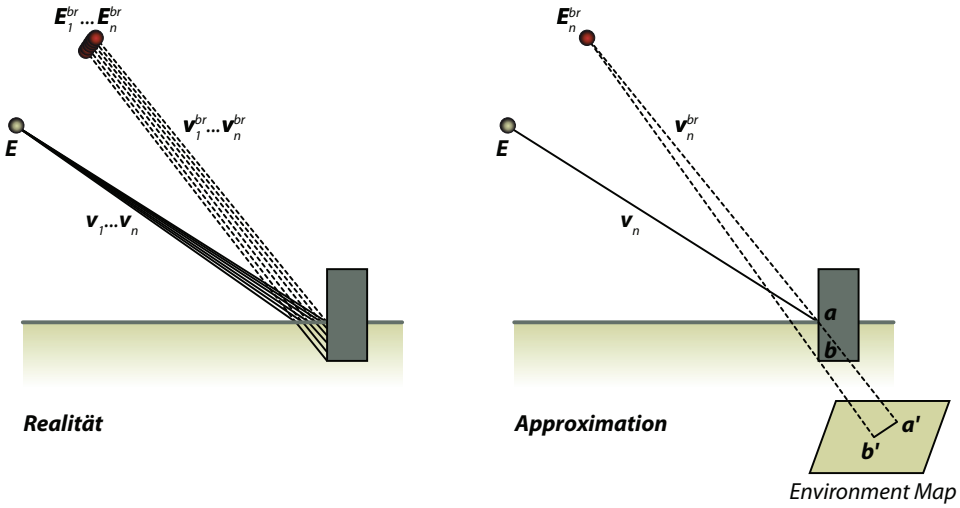


Abb. 7.4.: Approximation planarer Brechung: Die Menge virtuell gebrochener Kameravektoren $\mathbf{E}^{br}_{1\ldots n}$ (links) wird von dem Vektor $\mathbf{E}^{br}_{n}$ (rechts) repräsentiert.

(Abb. 7.3b,c). Insofern muss eine Approximation durchgeführt werden. Diese wird im Folgenden als virtuell gebrochene Kameraposition bezeichnet. Die Berechnung der virtuellen Kameravektoren wird im folgenden Abschnitt beschrieben.

7.2.1.2. Grundansatz

Zunächst wird die planare Reflexion behandelt, anschließend erfolgt die Behandlung der planaren Brechung. Im anschließenden Abschnitt wird dann die Erweiterung für gewellte Flüssigkeitsoberflächen gegeben. Reflexionen können nach Bestimmung der virtuell reflektierten Kamera $\mathbf{E}^{refl}$ durch Environment-Mapping bestimmt werden. $\mathbf{E}^{refl}$ ergibt sich dabei aus der Ebenen Normalen $\mathbf{n}_{\text{plane}}$, einem Punkt auf der Ebene $\mathbf{p}_{\text{plane}}$ und der Kamera Position $\mathbf{E}$ (Abb. 7.3a) — siehe auch Abschnitt 2.4:

$$\mathbf{E}^{refl} = \mathbf{E} - 2 \cdot ((\mathbf{E} - \mathbf{p}_{plane}) \cdot \hat{\mathbf{n}}_{plane})\, \hat{\mathbf{n}}_{plane}. \tag{7.2}$$

Anschließend wird die Szene aus virtuell reflektierter Kameraposition $\mathbf{E}^{refl}$ in eine Reflexions-Map dargestellt und im Bereich der reflektierenden Ebene dargestellt.

Im Gegensatz zur planaren Reflexion besitzt die planare Brechung keinen eindeutigen Fokus und damit keinen eindeutigen virtuell gebrochenen Kamera Vektor. Für einen lokalen Bereich jedoch, liegen die virtuellen Kamera Vektoren nah beieinander und können pro Objekt zusammengefasst werden und die durch die Brechung hervorgerufene Winkelverschiebung kann in einer Approximation vernachlässigt werden. Abbildung 7.3 b zeigt die drei virtuell gebrochenen Kameravektoren $\mathbf{E}^{br}$, die bei Brechung der Sichtstrahlen $\mathbf{v}^{br}_1$, $\mathbf{v}^{br}_2$ und $\mathbf{v}^{br}_3$ entstehen. Werden die Distanzen zwischen Brechungspositionen $\mathbf{p}_n$ $(n = 1, 2, 3, \ldots)$ und $\mathbf{E}$ bzw. $\mathbf{E}^{br}$ gleichgesetzt,

um der Realität zu entsprechen, so ergeben sich die anderen drei in Abbildung 7.3b gezeigten virtuellen Kameravektoren.

Diese Menge von Vektoren $\mathbf{E}^{br}_{1\ldots n}$ werden pro Objekt mit einer einzigen Kameraposition $\mathbf{E}^{br}_n$ approximiert (Abb. 7.4). $\mathbf{E}^{br}_n$ ist die virtuell gebrochene Kameraposition, die mit dem gebrochenen Sichtvektor $\mathbf{v}^{br}_n$ des Sichtvektors $\mathbf{v}_n$ gebrochen an $\mathbf{a}$ bestimmt wird:

$$\mathbf{E}^{br}_n = \mathbf{a} - \|\mathbf{E} - \mathbf{a}\| \frac{\mathbf{v}^{br}_n}{\|\mathbf{v}^{br}_n\|}. \tag{7.3}$$

Im Dreidimensionalen ist $\mathbf{a}$ der nächste Punkt zu $\mathbf{E}$, der auf der Ebene und dem zugehörigen Objekt liegt. Somit ist $\mathbf{E}^{br}_n$ für Blickpunkt $\mathbf{a}$ exakt. In der Praxis kann $\mathbf{a}$ über den Schnitt des *Bounding Cylinders* des Objektes und der Ebenen-Projektion der Sichtlinie von $\mathbf{E}$ auf den Schnitt der Objekt-Hauptachse mit der Ebene bestimmt werden.

Der Brechungsvektor $\mathbf{v}^{br}_n$ wird über das Snelliussche Gesetz beschrieben. In Abhängigkeit der Brechungsindizes $\eta = \frac{n_1}{n_2}$ und der Blickrichtung $\mathbf{v}$ ergibt sich entsprechend Abschnitt 2.4:

$$\begin{aligned} k &= 1 - \eta^2(1 - (\hat{\mathbf{v}} \cdot \hat{\mathbf{n}})^2) \\ \hat{\mathbf{v}}^{br} &= \begin{cases} \eta\hat{\mathbf{v}} - (\eta(\hat{\mathbf{v}} \cdot \hat{\mathbf{n}}) + \sqrt{k})\,\hat{\mathbf{n}} & : \quad k \geq 0 \\ 0 & : \quad k < 0. \end{cases} \end{aligned} \tag{7.4}$$

Objekte, die die Flüssigkeitsoberfläche schneiden, werden in zwei Teile geteilt, um die Brechung von über der Oberfläche liegenden Teilen zu verhindern. Die unter der Oberfläche liegenden Objektteile werden in eine *Environment Map* abgebildet – mit Kameraposition $\mathbf{E}^{br}_n$. Auf diese wird in einem Fragment Shader zugegriffen: Die korrekten Reflexions- und Brechungsvektoren werden auf per-Pixel Basis in Abhängigkeit der Oberflächenposition und -normalen bestimmt und zum Zugriff auf die *Environment Map* genutzt. Punkt $\mathbf{a}$ zum Beispiel erscheint an der korrekten Stelle. Das beschriebene Vorgehen führt jedoch zu einem Fehler in der visuellen Größe des gebrochenen Objektes (Abb. 7.5). Physikalisch resultieren die Sichtstrahlen $\mathbf{v}_1$ und $\mathbf{v}_2$ in den gebrochenen Sichtstrahlen $\mathbf{v}^{br}_1$ und $\mathbf{v}^{br}_2$. Dann jedoch würden die auf dem Objekt liegenden Punkte a und b an Positionen a' und b'_{real} erscheinen. Das bisher beschriebene Vorgehen projiziert a korrekt auf a', b jedoch wird auf b' projiziert. Damit würde die Größe des gebrochenen Objektteils nicht korrekt erscheinen. Eine Skalierung der unter der Flüssigkeitsoberfläche liegenden Objektteile behebt dieses Problem. Die Skalierung erfolgt mit dem Ursprung a' und bildet b' auf b'_{real} ab. Dazu wird der Skalierungsfaktor der Objektlänge iterativ bestimmt und das Objekt linear interpoliert. So entspricht die virtuelle Länge der gebrochenen Objektteile der Realität.

Approximation Zusammenfassend liegt die Approximation darin, dass b physikalisch korrekt aus der virtuellen Kameraposition $\mathbf{E}^{br}_1$ wahrgenommen wird — in-

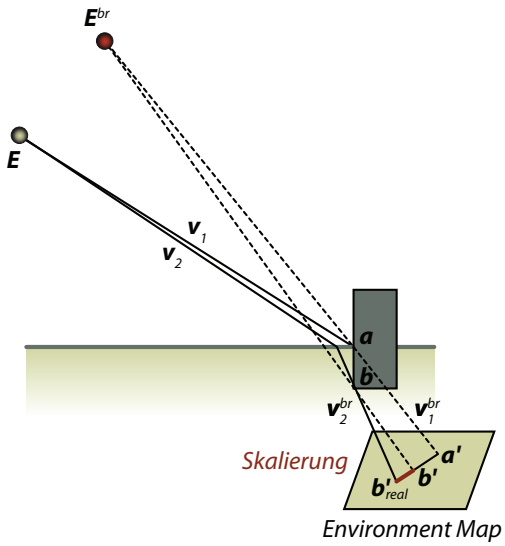
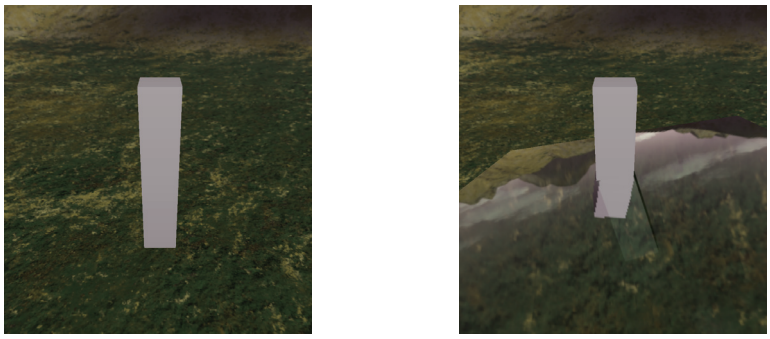


Abb. 7.5.: Lösung des Skalierungsproblems.

nerhalb der beschriebenen Approximation jedoch aus Kameraposition $\mathbf{E}_n^{br}$. Somit entsteht ein Fehler für die dargestellten Objektteile zwischen a und b: Physikalisch würde sich die Perspektive der Wahrnehmung ändern. In dem beschriebenen Ansatz wird lediglich eine Perspektive verwendet (aus Sicht von $\mathbf{E}_n^{br}$) und linear interpoliert. Da die Kamerapositionen $\mathbf{E}_1^{br}$ bis $\mathbf{E}_n^{br}$ für ein gegebenes Objekt nahe beieinander liegen, kann die Approximation jedoch kaum wahrgenommen werden. Beispiele der planaren Reflexion und Brechung sind in Abbildung 7.6 und 7.8a gegeben.



(a) (b)

Abb. 7.6.: Planare Reflexion und Brechung. Der Perspektivwechsel und die Skalierung des reflektierten und gebrochenen Bildes sind dargestellt. So können die nicht direkt sichtbaren, dunklen Seiten des Objektes auf der Oberfläche wahrgenommen werden.

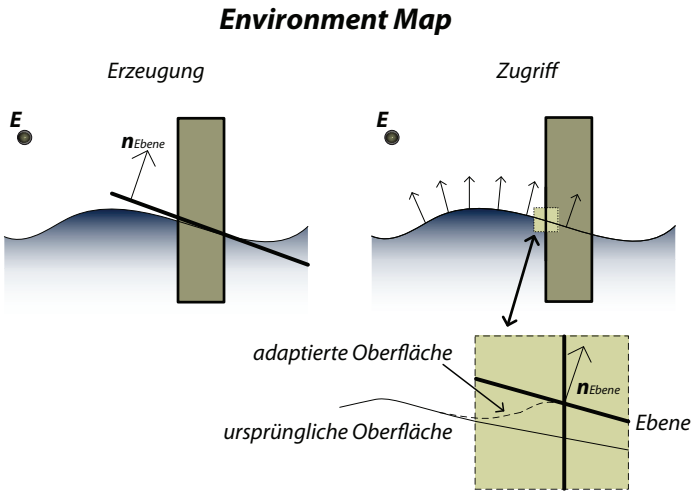
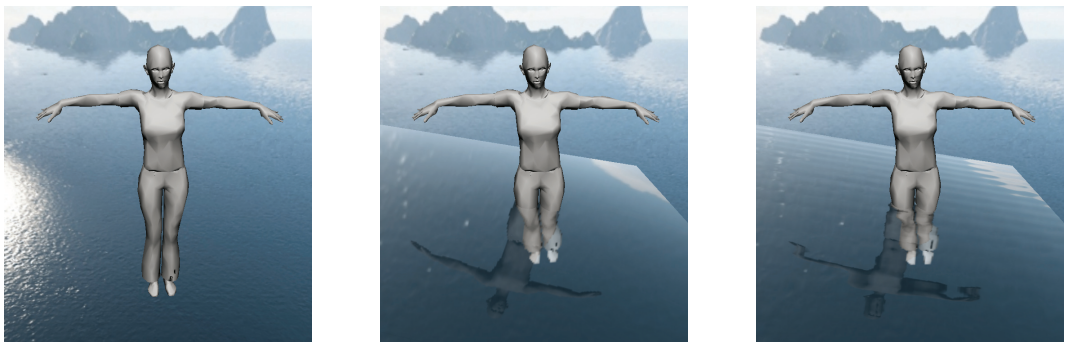


Abb. 7.7.: Erzeugung der *Environment Map* und der zugehörige Zugriff auf die *Environment Map*.

7.2.1.3. Erweiterungen

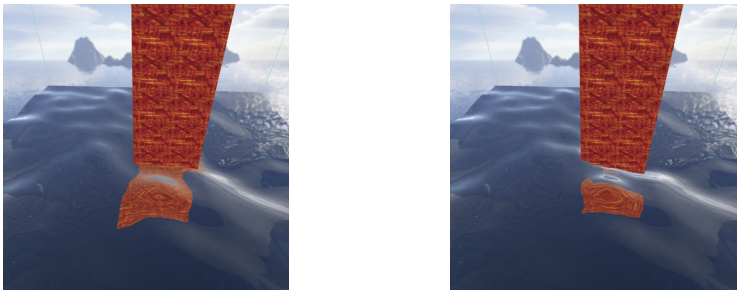
Die beschriebene planare Reflexion und Brechung wird im Folgenden für oberflächenschneidende, polygonale Objekte an dynamischen, welligen Flüssigkeitsoberflächen erweitert. Ein dynamisches Höhenfeld repräsentiert dabei die Oberfläche. Die Oberfläche in der Umgebung des Objektes wird planar approximiert (Fig. 7.7 links). Die Ebene E wird durch die durchschnittliche Höhenfeldwerte $\tilde{\mathbf{h}}$ der Schnittfläche von Objekt und Oberfläche und der zugehörigen Durchschnittsnormalen $\hat{\tilde{\mathbf{n}}}$ beschrieben:

$$E : \hat{\tilde{\mathbf{n}}}\tilde{\mathbf{h}} = \hat{\tilde{\mathbf{n}}}\mathbf{x} \tag{7.5}$$



(a) (b) (c)

Abb. 7.8.: Beispiel Reflexion und Brechung. In (c) werden überlagerte Wellenfunktionen zur Oberflächenvariation verwendet.



(a) (b)

Abb. 7.9.: Visueller Übergang der Brechung an der Grenzschicht mit der vorgestellten Methode (a). Der Übergang ist ununterbrochen. Aktuelle Techniken verwenden $\mathbf{E}$ statt $\mathbf{E}_n^{\text{br}}$ zur Generation der *Environment Map*. So ist der Übergang unterbrochen und eine Skalierung findet nicht statt (b).

Anhand dieser Approximation wird die dynamische *Environment Map* wie im vorhergehenden Abschnitt beschrieben erzeugt. Diese Map wird später anhand der Oberflächenpositionen und -normalen auf die Oberfläche abgebildet (Abb. 7.7 rechts). Die Zugriffsvektoren werden mit dem Reflexionsgesetz und dem Snelliusschen Gesetz berechnet. Da Flüssigkeitsoberflächen in der Regel sehr dynamisch sind, muss die *Environment Map* in jedem Frame aktualisiert werden. Statische *Environment Maps* können lediglich im Falle ruhiger Flüssigkeiten genutzt werden.

Visueller Übergang an Grenzschicht Da eine planare Oberflächenapproximation zur Erzeugung der *Environment Map* benutzt wird, können in der nahen Umgebung des Objektes Brüche in der Brechung auftreten, denn die Zugriffsvektoren treffen aufgrund der Oberflächendynamik in der Regel nicht exakt die projizierte Abbildung innerhalb der Environment-Map. Der Zugriff mit den am Höhenfeld re-

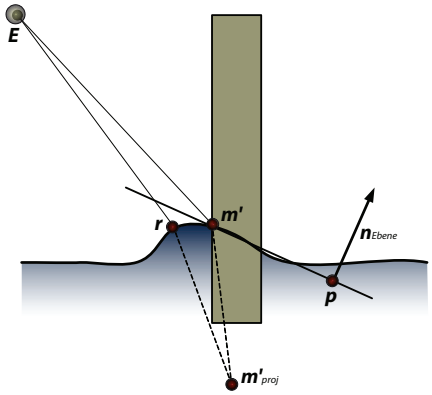


Abb. 7.10.: Bei großen Wellenamplituden können dennoch Artefakte auftreten. Die Approximation bricht **m'** zu $\mathbf{m'}_{proj}$. Somit kann **m'** auch an **r** wahrgenommen werden.

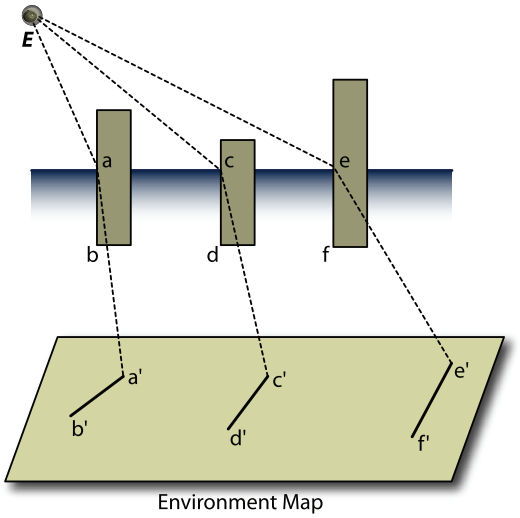


Abb. 7.11.: Erzeugung der *Environment Map* für mehrere Objekte.

flektierten/gebrochenen Sichtstrahlen ermöglicht zum Beispiel Zugriff auf Positionen oberhalb des Objektes. Um einen geschlossenen visuellen Übergang an der Grenzschicht der Flüssigkeit zu erhalten, müssen die Oberflächenpositionen und -normalen an der Kontur des Objektes denen der approximierenden Ebene entsprechen (Abb. 7.7 rechts). So stimmen die reflektierten/gebrochenen Sichtstrahlen mit denen der Erzeugung der *Environment Map* überein. Die Oberfläche und die Normalen werden in der Umgebung des Objektes linear an diejenigen der Ebene angepasst. Somit ist der visuelle Übergang zwischen gebrochenen Objektteilen unter der Oberfläche und dem Objekt über der Oberfläche ununterbrochen — siehe dazu Abbildung 7.9. Es sei jedoch auch erwähnt, das bei stark gewellten Flüssigkeitsoberflächen mit der beschriebenen Methode Artefakte entstehen können — siehe Abbildung 7.10. Aufgrund der hohen Dynamik derartiger Szenen jedoch sind diese inkorrekten Brechungen kaum als solche wahrnehmbar.

Mehrere Objekte Das vorhergehend beschriebene Prinzip kann auch zur Brechung und Reflexion mehrerer Objekte verwendet werden. (Fig. 7.11). Alle Objekte werden separat aus ihrer jeweiligen virtuell gebrochenen/reflektierten Kamerapositionen in die *Environment Map* abgebildet.

Wenn Objekte dicht beieinander liegen, können im Fall hoher Wellenamplituden Objekte in der *Environment Map* übereinander liegen. Starke Brechungswinkelvariationen verursachen diesen Effekt. In diesem Fall können benachbarte Elemente zusammengefasst werden und als ein Brechungsobjekt behandelt werden. Oder aber eine separate *Environment Map* wird für diesen Fall verwendet. Diese Effekte treten jedoch lediglich bei stark gekrümmten Wellenoberflächen auf, so dass in der Regel eine einzige *Environment Map* verwendet werden kann.

Aufgrund des Mappings aus Kameraposition können hinter und neben Objekten

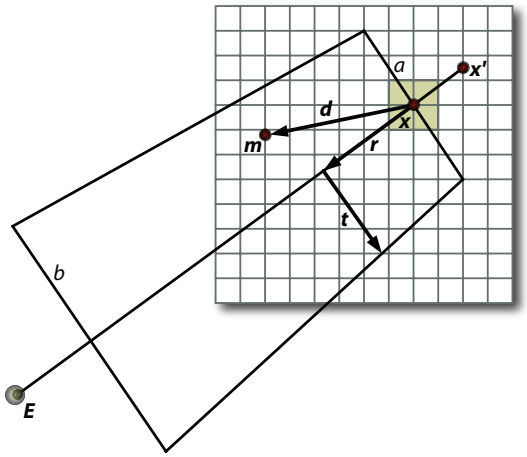


Abb. 7.12.: Berechnung des Trapezes, welches entgegen der Blickrichtung orientiert ist.

Brechungen erscheinen, die unrealistisch sind. So kann zum Beispiel eine Welle hinter einem Objekt zu einem *Environment Map* Zugriff führen, der Brechungsinformationen desselben Objektes zurückliefert. Aus diesem Grund werden zwei *Environment Maps* benutzt. Eine statische, die alle statischen Informationen ohne die zu brechenden Objekte enthält. Und eine dynamische, die die zu brechenden Objektdaten enthält. Diese Map wird wie im vorhergehenden Abschnitt beschrieben erzeugt. Der hauptsächliche Bereich, in dem Brechung auftritt liegt im Bereich zwischen Objekt und Kameraposition. Deshalb wird eine trapezförmige Fläche benutzt (Abb. 7.12), in deren Bereich die dynamische *Environment Map* benutzt wird. Außerhalb wird die statische Map genutzt. Das Trapez orientiert sich entlang der Geraden durch Objektmittelpunkt und Kameraposition projiziert auf die Ebene.

Welche Environment-Map benutzt wird, hängt von der Position $\mathbf{m}$ relativ zur Objektposition $\mathbf{x}$ ab (Fig. 7.12). Damit keine harten Übergänge zwischen beiden *Environment Maps* auftreten, wird linear zwischen beiden *Environment Maps* in Abhängigkeit des Abstandes $\mathbf{m}$ von der Orientierung $\mathbf{x} + s\mathbf{r}$ ($s > 0$) interpoliert. Abbildungen 7.2a,c zeigen Beispiele der Reflexion und Brechung mehrerer Objekte — jedes Tischbein in Abb. 7.2c wird dabei als einzelnes Objekt behandelt.

7.2.2. Ergebnisse und Diskussion

Die präsentierte Methode wurde unter Verwendung von OpenGL 2.0 und der zugehörigen Shadersprache GLSL in C++ implementiert. Laufzeitmessungen erfolgten auf einer Dual-Core 2,6 GHz AMD Athlon 64 CPU mit 2 GB RAM und einer auf dem ATI Radeon x1900 Chip basierenden GPU (512MB). Die Implementierung verwendet *Multi-Threading*. Dabei simuliert ein *Core* die Physik mit SPH und der andere *Core* extrahiert die Oberfläche, erzeugt die notwendigen Texturen und kreiert Kaustiken mit dem in Abschnitt 7.1 beschriebenen Verfahren. Laufzeitmessungen werden in Tabelle 7.1 gegeben und wurden bei einer Darstellungsauflösung

Beispiel (Abb.)	Simulation	Cores (Anz.)	Refl./Br. (FPS)	Approx. Kaust. (FPS)	Phys. Kaust. (FPS)
7.1	SPH (2000P)	2	102	102	77
7.1	SPH (3000P)	2	84	84	71
7.2a	SPH (2000P)	2	75	64	64
7.2b	SPH (2000P)	2	55	55	45
7.2b	SPH (3000P)	2	52	52	41
7.2c	SPH (2000P)	2	51	51	44
7.6	Trigonom.	1	91	91	-
7.8	Trigonom.	1	51	51	-

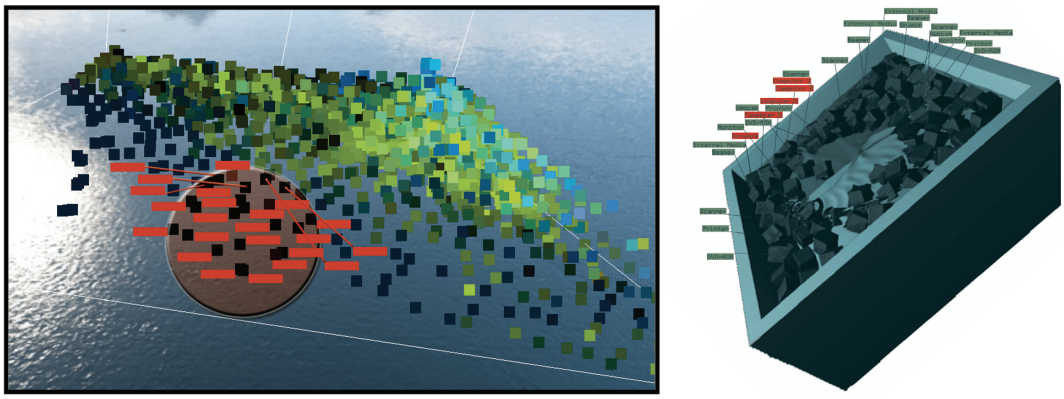
Tab. 7.1.: Laufzeiten des vorgestellten Algorithmus in verschiedenen Szenarien (Refl./Br.). Wegen des *Shader*-basierten Texturzugriffes, kann der Aufwand der approximativen Kaustiken vernachlässigt werden. Die physikalischen Kaustiken (*Light Ray Tracing*) sind jedoch nicht zu vernachlässigen. Die ambienten Wellen in den letzten beiden Beispielen wurden durch Überlagerung trigonometrischer Funktionen erzeugt. Die gegebenen Zeiten gelten für die SPH-Simulation, Oberflächenextraktion und die Darstellung.

von 950×950 erhoben. Die Geschwindigkeiten erlauben volle Interaktivität. Die Performanz der unoptimierten Implementierung kann noch verbessert werden, indem eine GPU-basierte Strömungssimulation und Oberflächenextraktion verwendet wird. Vorteil der beschriebenen Methode ist neben der performanten Ausführung die Reduktion von Artefakten an der Grenzschicht und die der Realität entsprechende perspektivische Skalierung. Das beschriebene Verfahren soll kein Ersatz für akkurate Nicht-Echtzeit-Verfahren wie Ray Tracing darstellen, sondern stellt eine gute Approximation für interaktive Umgebungen zur Verfügung.

7.3. Interaktive Partikel-basierte Beschriftung ohne Verdeckung von Hintergrundinformationen

Dieser Abschnitt präsentiert ein Konzept zur interaktiven und überdeckungsfreien Beschriftung graphischer Elemente. Dabei bezeichnet *Beschriftung* (oder auch *Label*) sowohl textuelle als auch graphische Elemente, die einer existierenden Graphik hinzugefügt werden und in der Regel weitere, über das eigentliche Bild hinausgehende, Informationen aufbereiten und präsentieren.

Die Methode ermöglicht eine Beschriftung dynamischer Punktwolken bei interaktiven Geschwindigkeiten. Zudem können Hintergrundinformationen in der Beschriftung berücksichtigt werden, so dass sie nicht verdeckt werden. Aufgrund dieser Eigenschaften ist die Methode auch für die Visualisierung Partikel-basierter Flüssigkeitssimulationen interessant. Gerade in Echtzeitumgebungen mit verhältnismäßig wenigen Partikeln können die physikalischen Werte in Labeln angegeben werden (Abb. 7.13a). Aber auch während der Entwicklung von Simulations-Systemen kann eine derartige Exploration der Daten nützlich sein. Des Weiteren könnten zum Beispiel dynamische Objekte innerhalb von Flüssigkeiten



(a) (b)

Abb. 7.13.: Beispiele der präsentierten, schnellen Beschriftungsmethode: Beschriftung von SPH-Partikeln (a) und Beschriftung von Objekten innerhalb einer Strömung (b).

beschriftet werden (7.13b).

Die beschriebene Beschriftungsmethode benötigt als Eingabe lediglich eine Liste von Punktkoordinaten, die zugehörigen Label und das Hintergrundbild, welches während der Beschriftung berücksichtigt werden soll. Somit ist der beschriebene Ansatz sehr allgemein und bietet zahlreiche Anwendungsmöglichkeiten neben der Beschriftung Partikel-basierter Flüssigkeitssimulationen. So ist Beschriftung in vielen Visualisierungsumgebungen eine wichtige Technik, um dargestellte Daten zu kommunizieren. Aufgrund dieser Allgemeinheit ist der vorliegende Abschnitt ebenfalls allgemein geschrieben und nimmt nicht explizit Bezug auf Flüssigkeiten.

Überblick Schwerpunkt der im Folgenden beschriebenen Methode liegt auf der schnellen Beschriftung beliebiger graphischer Elemente, ohne dabei andere visuelle Elemente zu überdecken. Die hier beschriebene Partikel-basierte Beschriftungsmethode kann tausende von Labeln in interaktiven Umgebungen positionieren — ohne der Notwendigkeit einer Vorberechnung. Überdeckungen werden mit Hilfe von Partikeln detektiert, die bereits vergebene Bildschirmbereiche markieren. Die vorgestellte Methode erzielt Ergebnisse, die denen existierender Methoden in der Regel in Bezug auf Performanz zumindest ebenbürtig sind (siehe Abschnitt 7.3.2 — Diskussion) — die vorgestellte Methode kann jedoch zusätzlich Hintergrundobjekte behandeln und führt eine Distanzbeschriftung ein, welche die Beschriftung in dichten Situationen ermöglicht, in denen rein adjazente Methoden (Label liegen an dem zu beschriftenden Objekt an) keine Beschriftung durchführen können. Die vorgestellte Beschriftungstechnik wurde in Zusammenarbeit mit Martin Luboschik und Heidrun Schumann entwickelt und veröffentlicht [LSC08, CLS09].

Die Überlappungsfreie Beschriftung allgemeiner graphischer Objekte ist ein NP-hartes Problem (zum Beispiel [MS91, KT96]). Deshalb verwenden aktuelle Ansätze,

die dieses Problem in interaktiven Umgebungen lösen wollen, im Allgemeinen Approximationen und Heuristiken zur Einschränkung des Lösungsraumes und somit zur Beschleunigung. Ein Ziel ist es in der Regel, möglichst viele Label zu positionieren, gleichzeitig jedoch auch visuelle Präferenzen zu berücksichtigen. Da die Beschriftung von Punktobjekten ein häufiges und zugleich eingeschränktes Problem ist, fokussieren viele Techniken auf dieses Problem unter Berücksichtigung von Label-Label Konflikten. Um eine interaktive Beschriftung in dynamischen Umgebungen zu erzielen, werden in der Regel intensive Vorverarbeitungsschritte (zum Beispiel [BDY06, PGP03, YCL02, YCL05]) oder Reduktionen möglicher Beschriftungspositionen (zum Beispiel [FW91, Mot07]) und/oder des zu beschriftenden Raumes vorgenommen (zum Beispiel [FLH+06]). Die hier vorgestellte Methode benötigt keine Vorberechnung, so dass auch dynamische Punktwolken interaktiv beschriftet werden können. Oftmals wird auch eine feste Größe der Label angenommen — dieses ist bei dem vorgestellten Algorithmus nicht der Fall. Die Methode zur Beschriftung von Punktobjekten wird schließlich um die Berücksichtigung anderer visueller Elemente und um die Beschriftung beliebiger Objekte erweitert.

7.3.1. Prinzip

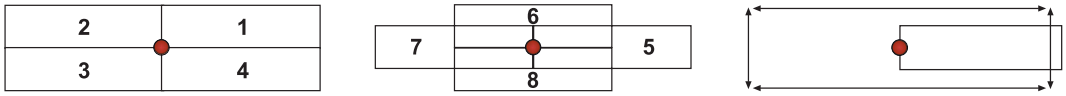
Der präsentierte Ansatz stellt eine stabile und schnelle Beschriftungsmethode dar, ohne existierende visuelle Elemente zu überdecken. Um eine effiziente Ausführung zu erzielen, wird der Beschriftungsprozess unterteilt und im Rahmen einer Beschriftungs-Pipeline bearbeitet (Abschnitt 7.3.1.1). Diese Pipeline verwendet komplexer werdende Beschriftungsalgorithmen für zugleich weniger zu beschriftende Partikel. Zur Vermeidung von Verdeckungen existierender Label oder auch existierender Hintergrundelemente wird ein Partikel-basierter Ansatz verwendet. Dazu werden Partikel für hinzukommende Label oder den existierenden Hintergrund eingefügt. Diese dienen später der effizienten Konflikterkennung. Zusätzlich zu adjazenten Beschriftungspositionen wird eine Distanzbeschriftung eingeführt, um dichte Punktwolken oder ausgedehnte graphische Objekte zu beschriften. Derartige Situationen könnten für adjazente Beschriftungen ansonsten nicht gelöst werden. Des Weiteren wird der Ansatz auf die achsenparallele Beschriftung beliebiger Objekte erweitert — so zum Beispiel Linien- oder Flächenobjekte.

Die grundlegenden Schritte zur überdeckungsfreien Beschriftung unter Berücksichtigung existierender graphischer Elemente sind:

1. Rastergraphik oder Vektorgraphiken repräsentieren den momentan belegten Raum, der von anderen Elementen belegt wird. Dieser wird abgetastet, um die Konfliktpartikelmenge zu erzeugen.

2. Die zu beschriftenden Punkte werde als *Label Partikel* der Konfliktmenge hinzugefügt.

3. Auf Basis der Konfliktmenge wird beim Durchlaufen der Beschriftungs-Pipeline effizient über Akzeptanz bzw. Ablehnung entschieden.



(a) (b) (c)

Abb. 7.14.: Übliche Positionierungsmodelle mit den möglichen Beschriftungspositionen (Rechtecke) und Bewertungen (1: hohe Priorität ... 8: niedrige Priorität): 4-Positionsmodell (a), Positionen 5–8 des 8-Positionsmodells (b), *Slider*-Modell (c).

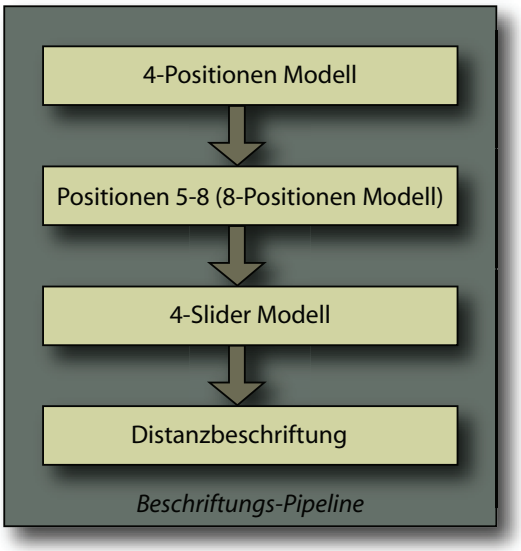
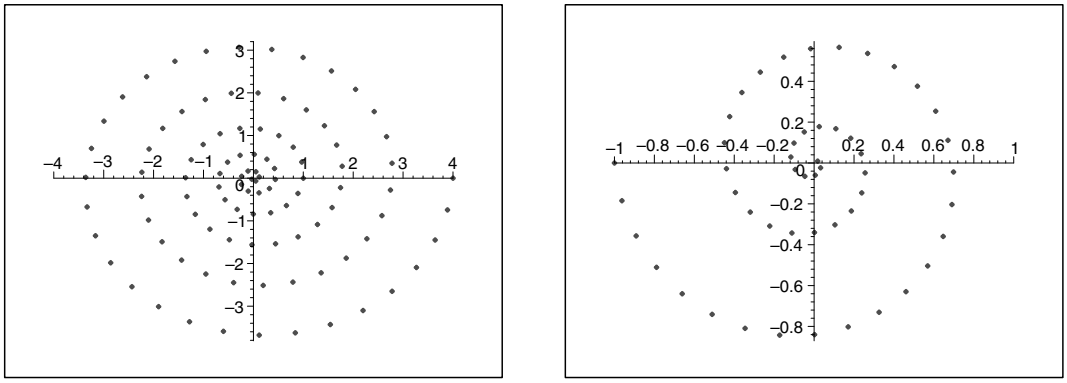


Abb. 7.15.: Die vier Schritte der Beschriftungs-Pipeline (siehe dazu auch Abb. 7.14).

Diese Schritte garantieren die Unabhängigkeit des Beschriftungsansatzes von den unterliegenden Bildinformationen und eine schnelle Detektion von Beschriftungskollisionen.

7.3.1.1. Beschriftungs-Pipeline

Die Beschriftungs-Pipeline ist entwickelt worden, um hohe Geschwindigkeiten zu erzielen. Die grundlegende Idee ist es, mit schnellen Beschriftungstechniken möglichst viele Label zu setzen — um Rechenleistung für komplexere Techniken bei möglichst wenig zu positionierenden Labeln zu bewahren. Die Beschriftungs-Pipeline wurde als Ergebnis dieser Erkenntnis und dem Experimentieren mit verschiedenen Pipelines definiert (Abb. 7.14 und Abb. 7.15). Jeder Schritt wird unabhängig für alle bis dahin unbeschrifteten Punkte angewendet. Schritte 1–3 stellen eine sukzessive Anordnung existierender Techniken dar. Diese benutzen in der Regel ein bis zwei dieser Schritte und versuchen die Ergebnisse zu optimieren — somit unterscheidet sich das hier beschriebene Vorgehen signifikant von existierenden Techniken.



(a) (b)

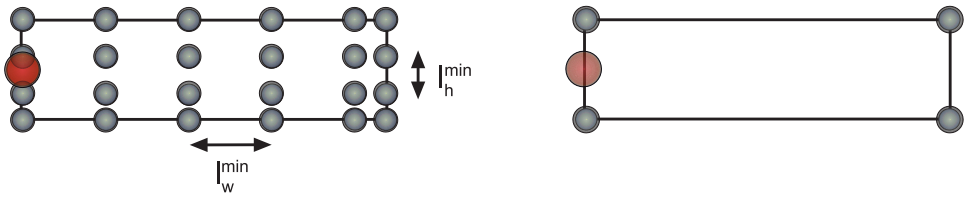
Abb. 7.16.: Abtastfunktion des Bildraumes: $r_{max} = 4$, $m_{max} = 100$, $n_{rot} = 6$, $d = 1$ (a); $r_{max} = 1$, $m_{max} = 50$, $n_{rot} = 3$, $d = -1$ (b).

Der vorgestellte Partikel-basierte Ansatz sucht nach verfügbarem Raum für Beschriftungen in der Umgebung unbeschrifteter Objekte, wenn adjazente Ansätze fehlschlagen (Pipeline-Schritte 1–3). Aus diesem Grund wird eine raumdiskretisierende Funktion definiert, die verfügbaren Raum nach der aktuelle Labelgröße in der Umgebung von Objekten untersucht. Diese selektiert die Position, die ohne Überdeckung möglichst nahe an dem zu beschriftenden Element liegt. In der Praxis muss die raumdiskretisierende Funktion nicht den gesamten zur Verfügung stehenden Bildraum abtasten — eine spärlich abtastende Funktion reduziert die Beschriftungsqualität kaum, steigert jedoch die Performanz erheblich. Hier wird die raumdiskretisierende Funktion zum Beispiel als Spriale $\mathbf{s}(m) \in \mathbb{R}^2$ mit variabler Abtastung konstruiert:

$$\mathbf{s}(m) = \begin{pmatrix} d \cdot \cos(2\pi\sqrt{\frac{m}{m_{max}}} \cdot n_{rot}) \\ \sin(2\pi\sqrt{\frac{m}{m_{max}}} \cdot n_{rot}) \end{pmatrix} \cdot \frac{m}{m_{max}} \cdot r_{max}, \quad m = 1 \ldots m_{max}.$$

m_{max} bezeichnet die Anzahl der Abtastungspunkte, r_{max} den maximalen Radius der Spirale und n_{rot} definiert die Anzahl der Rotationen der Spirale. Die Orientierung der Spirale wird mit dem Parameter $d \in \{1; -1\}$ bestimmt. Somit kann die raumabstastende Funktion intuitiv definiert werde (Abb. 7.16). Distanzbeschriftungen und korrespondierende Punktobjekte werden mit Linien verbunden, um die Zusammengehörigkeit zu repräsentieren.

Beschriftung nach Relevanz Die beschriebene Beschriftungs-Pipeline resultiert in einer Beschriftung entsprechend der Reihenfolge der zu beschriftenden Partikel. Während die ersten zu beschriftenden Punktobjekte noch den Großteil der zur Verfügung stehenden Fläche nutzen können, müssen die später zu positionierenden Label eventuell aus Platzgründen abgewiesen werden. Dieser Sachverhalt kann ge-



(a) (b)

Abb. 7.17.: Positionierung virtueller Partikel: Die vom Label eingenommene Fläche wird entsprechend der minimalen Labelgröße l_w^{min},l_h^{min} mit virtuellen Partikeln diskretisiert (a). Konstante Labelgrößen resultieren somit in lediglich vier virtuellen Partikeln (b).

nutzt werden, um eine Beschriftung entsprechend der Relevanz zu ermöglichen. In der Kartographie zum Beispiel ist der Name eines Staates in der Regel wichtiger als die Namen von Städten oder Flüssen. Wenn eine gegebene Relevanz angenommen werden kann, kann die beschrieben Beschriftungs-Pipeline eine Beschriftung entsprechend dieser Relevanz vornehmen. Dafür wird jede Relevanzstufe separat beschriftet — beginnend mit der Stufe höchster Relevanz. Somit erhalten wichtigere Elemente eher ein nahes oder gar adjazentes Label und vorhandene Fläche wird entsprechend der Relevanz verteilt.

7.3.1.2. Effiziente Konflikterkennung mit Partikeln

Ein entscheidender Schritt für gutes Laufzeitverhalten ist die schnelle Detektion von Konflikten, da dieser Test im ungünstigsten Fall für jede mögliche Labelposition aller Label und aller Pipeline-Schritte durchgeführt werden muss. Da die hier beschriebenen Label als achsenparallel angenommen werden, müssen lediglich Schnitte achsenparalleler Rechtecke detektiert werden. Zur schnellen und flexiblen Detektion wird die Menge der Konfliktpartikel eingeführt. Diese besteht aus *Label-Partikeln* und *virtuellen Partikeln*.

Label-Partikel repräsentieren die n zu beschriftenden Punktobjekte mit den Positionen $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^2$. Dreidimensionale Punktwolken werden in den zweidimensionalen Bildraum transformiert und können dann blickpunktabhängig beschriftet werden. *Virtuelle Partikel* werden dynamisch erzeugt oder vernichtet, um belegte Positionen im Bildraum zu markieren und mit ihrer Hilfe Konflikte zu erkennen. Eine mögliche Beschriftungsposition ist *konfliktfrei* und kann somit akzeptiert werden, wenn *kein* Konfliktpartikel innerhalb der zugehörigen rechteckigen Fläche existiert.

Somit kann der Konflikttest für ein gegebenes rechteckiges Label mit den Koordinaten $(x_{left}, y_{bottom}, x_{right}, y_{top})$ und einem gegebenen Konfliktpartikel mit Position (x_p, y_p) auf folgenden logischen Ausdruck reduziert werden:

$$\begin{aligned} accepted = \neg \big(& (x_p > x_{left}) \quad \wedge \quad (x_p < x_{right}) \quad \wedge \\ & (y_p > y_{bottom}) \quad \wedge \quad (y_p < y_{top}) \big). \end{aligned}$$

Neue virtuelle Partikel müssen bei Akzeptanz erzeugt werden, um eine spätere Verdeckung des hinzugekommenen Labels zu verhindern. Deshalb wird die Beschriftungsfläche mit der minimal existierenden Beschriftungsgröße l_w^{min},l_h^{min} abgetastet und entsprechend mit virtuellen Partikeln gefüllt (Abb. 7.17a). Im Sonderfall von lediglich einer existierenden Beschriftungsgröße ergeben sich lediglich vier hinzuzufügende virtuelle Partikel (Abb. 7.17b). Im folgenden Abschnitt wird beschrieben, wie mit Hilfe virtueller Partikel eine Verdeckung graphischer Hintergrundinformationen verhindert werden kann.

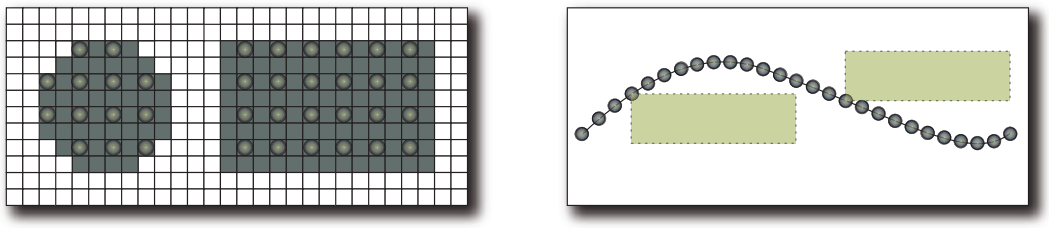
Zur Beschränkung des Konflikttests auf relevante Konfliktpartikel und damit zur Reduktion des Aufwands, wird eine Datenstruktur zur Nachbarschaftssuche eingefügt. Diese muss ein ausgewogenes Verhältnis von schneller Aktualisierung und Auswahl lediglich relevanter virtueller Partikel gewährleisten. Eine Gitter-basierte, orthogonale Bereichssuche erfüllt diese Bedingungen. Jedes Gitterelement kennt alle Partikel innerhalb der zugehörigen Fläche. Somit müssen lediglich Konfliktpartikel aus den betreffenden benachbarten Gitterelementen betrachtet werden, wenn die verschiedenen Beschriftungspositionen eines Punktobjektes getestet werden. Um den Speicherzugriff zu reduzieren und realistischen Szenarien gerecht zu werden, wird die Gitterelementgröße entsprechend der maximalen Beschriftungsbreite l_w^{max} und -höhe l_h^{max} gewählt.

Die möglichen Beschriftungspositionen werden entsprechend der aktuellen Beschriftungs-Pipeline gesetzt. Wenn kein Konflikt auftritt, wird die aktuell getestete Beschriftungsosition akzeptiert, andernfalls wird die nächste Beschriftungspositionen getestet. Falls keine gültige (konfliktfreie) Position gefunden wird, wird das aktuelle Partikel als unbeschriftet deklariert.

7.3.1.3. Abbildung graphischer Daten

Wie im vorhergehenden Abschnitt beschrieben wurde, können die virtuellen Partikel verwendet werden, um Konflikte zwischen Labeln effektiv zu detektieren. In diesem Abschnitt wird das Konzept erweitert, um beliebige existierende visuelle Objekte mit Hilfe virtueller Partikel in das Überlappungsproblem einzubeziehen. Um Unabhängigkeit von der konkreten, unterliegenden Visualisierungstechnik zu gewährleisten, wird die Diskretisierung der Objekte mit Partikeln separat sowohl für Rastergraphiken, als auch für Vektorgraphiken behandelt.

Bild-basierter Ansatz Als Eingabe wird ein rasterisiertes Bild angenommen — die Kollisionen-Map. Diese wird im Bildraum abgetastet und besitzt eine Farbe c_{hg}, die verfügbaren Beschriftungsraum anzeigt (diese entspricht in der Regel der Hintergrundfarbe). Virtuelle Partikel werden im Partikelraum eingeführt für alle abgetasteten Farbwerte $\neq c_{hg}$ (Abb. 7.18a). Ein Abtastrate von 1×1 Pixeln würde optimale Ergebnisse erzielen — höhere Abtastraten verbessern die Performanz erheblich und verringern die Beschriftungsqualität nicht, wenn keine Elemente existieren, die kleinere Ausdehnungen besitzen als die Abtastrate. So kann jedes beliebige visuelle



(a) (b)

Abb. 7.18.: Erzeugung virtueller Partikel: Ein Rasterbild wird mit virtuellen Partikeln abgetastet (a). Vektorobjekte (hier: Kurve) können direkt mit virtuellen Partikeln abgetastet werden (b). Die verwendeten Positionen können zusätzlich als mögliche Kandidaten für eine Beschriftung des Vektorobjektes verwendet werden.

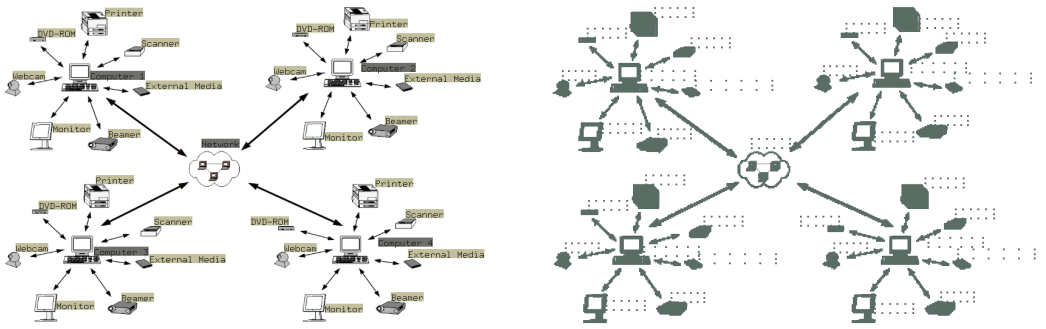
Element innerhalb der Beschriftung behandelt werden. Die Kollisionen-Map kann in vielen Anwendungen direkt innerhalb der *Rendering-Pipeline* extrahiert werden, so dass ein zusätzlicher *Rendering-Pass* nicht unbedingt notwendig sein muss. Beispiele werden in Abbildungen 7.19 und 7.22 gegeben. In Abbildung 7.19b sind die virtuellen Partikel beispielhaft dargestellt.

Vektor-basierter Ansatz Vektorgraphik kann auf ähnliche Weise in das Beschriftungsproblem mit einbezogen werden. Dazu muss lediglich die Kontur des Vektorobjektes ebenfalls im Bildraum abgetastet und als virtuelle Partikel in die Konfliktmenge eingefügt werden (Abb. 7.18b).

7.3.1.4. Erweiterungen

Der Partikel-basierte Beschriftungsansatz kann auf die Beschriftung beliebig geformter Objekte erweitert werden (flächige Punktobjekte, Linien und Flächen). Das zu beschriftende und nicht zu überdeckende Objekt wird zunächst mit virtuellen Partikeln abgetastet (wie im vorhergehenden Abschnitt beschrieben). Dann kann ein Label-Partikel innerhalb des zu beschriftenden Objektes positioniert werden. Aufgrund der darin positionierten virtuellen Partikel wird mit Hilfe des vierten Pipeline-Schrittes, der Distanzbeschriftung, eine mögliche Beschriftungsposition außerhalb des zu beschriftenden Objektes bestimmt (die anderen drei Schritte können in diesem Fall nicht zum Erfolg führen, da adjazente Techniken aufgrund der innerhalb des Objektes positionierten virtuellen Partikel fehlschlagen). Liegt die gefundene Position am Objekt an, so kann die Linie verzichtet werden, die die Zusammengehörigkeit signalisiert (Abb. 7.19). Mit Hilfe einer Invertierung der Kollisionen-Map können Flächen auch intern beschriftet werden (Label liegen innerhalb eines Objektes und nicht außerhalb).

Der beschriebene Ansatz ermöglicht zudem eine einfache Beschriftung mit beliebig geformten Labeln und eine Berücksichtigung von Flächen unterschiedlicher Relevanz.



(a) (b)

Abb. 7.19.: Beschriftung beliebig geformter Objekte. Die zu beschriftenden Punktobjekte liegen innerhalb der graphischen Objekte. Mit Hilfe der Distanzbeschriftung ergeben sich die Beschriftungspositionen (a). Die gezeigten virtuellen Partikel verhindern Verdeckung von Labeln und Hintergrundinformationen (b) — die Partikel sind in dieser Abbildung vergrößert dargestellt.

Für genauere Details sei an dieser Stelle auf [LSC08] verwiesen.

Die vorgestellte Methode beschriftet alle Label-Partikel unabhängig von den vorhergehenden Beschriftungspositionen. In dynamischen Umgebungen oder bei einer Transformation der Positionen kann sich die jeweilige Beschriftungsposition in jedem dargestellten Bild ändern, was zu visuellem Flimmern der Beschriftungen führen kann. Um dennoch fließende Bewegungen der Beschriftungen zu erzielen kann ein Animationsverfahren verwendet werden. Die Label bewegen sich träge in Richtung der aktuellen Beschriftungsposition und werden mit einer Fading-Funktion entsprechend ihrer Sichtbarkeit ein- und ausgeblendet. Für mehr Details sei an dieser Stelle auf [CLS09] verwiesen.

7.3.2. Ergebnisse und Diskussion

Die Experimente, die im Folgenden vorgestellt werden, wurden auf einem Dual-Core Desktop PC mit 2,6 GHz AMD Athlon 64 CPU, 2 GB RAM und einer auf der GeForce 8800 GTX GPU basierenden Graphikkarte durchgeführt. Alle gegebenen Messzeiten beziehen sich auf die Zeit, die zur Beschriftung und zur Darstellung benötigt wird.

Im Allgemeinen ist ein direkter Vergleich von Beschriftungsmethoden schwierig, da in der Regel verschiedene Voraussetzungen gelten. Die meisten existierenden Ansätze für interaktive Beschriftungen nutzen eine adjazente Beschriftungsstrategie, kennen somit keine Distanzbeschriftungen, und behandeln keine Hintergrundelemente. Einige Methoden nehmen auch statische Labelgrößen an. Die hier vorgestellte Methode ist frei von derartige Einschränkungen, was — neben der guten Performanz — einen Hauptvorteil der Methode darstellt.

Dennoch wurden Standarddatensätze und zufällige Konfigurationen, die in ande-

Beschriftungsmethode	*Anzahl Label* 500 in %	750 in %	1000 in %	1500 in %
mit Point-Selection				
Sim. Annealing	99	95	88	74
ohne Point-Selection				
FALP	100	97	90	—
Tabu Search	99	97	90	—
Sim. Annealing	98	92	82	—

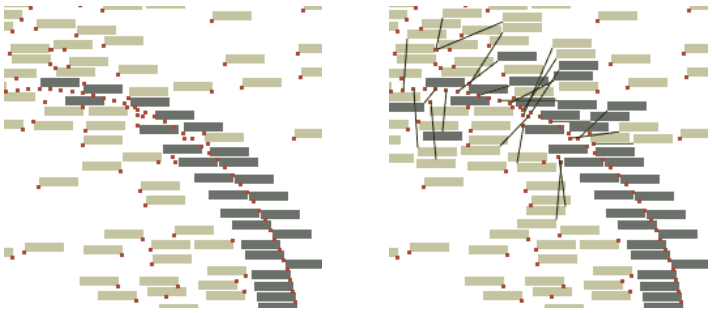
Tab. 7.2.: Beschriftungsergebnisse aus [CMS95] [YCL05] mit und ohne *Point-Selection*. Die Konfiguration wird in Tabelle 7.3 gegeben.

ren Veröffentlichungen verwendet wurden, mit der beschriebenen Beschriftungsmethode modelliert und beschriftet — so sind die präsentierten Ergebnisse zumindest teilweise vergleichbar. Für jede zufällig, uniform verteilte Punktwolke wurden 100 Messungen durchgeführt, deren Durchschnitt die gegebene Beschriftungsquote und Performanz beschreibt.

Anzahl beschrifteter Elemente Ein wichtiges Ziel der Beschriftung ist eine hohe Beschriftungsrate (möglichst viele Label ohne Überdeckungen zu positionieren). Der beschriebene Ansatz beschriftet möglichst viele Objekte adjazent. Erst, wenn die Anzahl von Punktobjekten zum verfügbaren Bildraum wächst, werden mehr Distanzbeschriftungen benötigt. Die Literatur unterscheidet zwei Zählweisen von Beschriftungskonflikten: Mit und ohne *Point-Selection*. Mit Selektion werden lediglich die Punktobjekte gezählt, die nicht konfliktfrei beschriftet werden können (sie werden *selektiert*) — ohne Selektion werden alle Punktobjekte gezählt, die an einem Konflikt beteiligt sind (vgl. [CMS95] für mehr Details). Tabelle 7.2 zeigt Referenzwerte aus [CMS95] und [YCL05]. Bei bis zu 750 Punktobjekten auf der gegebenen Fläche beschriftet die hier beschriebene Methode alle Punkte und es existieren somit keine Konflikte (Tabelle 7.3, Abb. 7.20 und 7.21). Damit ist die Zählweise von Konflikten in diesen Fällen irrelevant und die gezeigten Ergebnisse können direkt mit den Literaturwerten verglichen werden. Die hier beschriebene Methode beschriftet in diesen Fällen mehr Objekte als existierende Methoden. Im Falle von 1000 Punkten beschriftet die beschriebene Methode $99,96\%$ — eine untere Abschätzung für *Point-Selection allowed* liefert 99%, so dass auch in diesem Fall *Simulated Annealing* übertroffen wird.

Ergebnisse der *Beschriftung nach Relevanz* (Vgl. Abschnitt 7.3.1.1) werden in Tabelle 7.4 gegeben. 10% der Punktobjekte werden mit höher Relevanz deklariert (Dunkelgrüne Label in Abbildung 7.21). Mit Hilfe der Distanzbeschriftung können alle Elemente mit größerer Relevanz beschriftet werden. Somit ist die Methode auch für hierarchische Datensätze geeignet.

Die beschriebene Beschriftungs-Pipeline liefert in der Regel mehr adjazente Be-



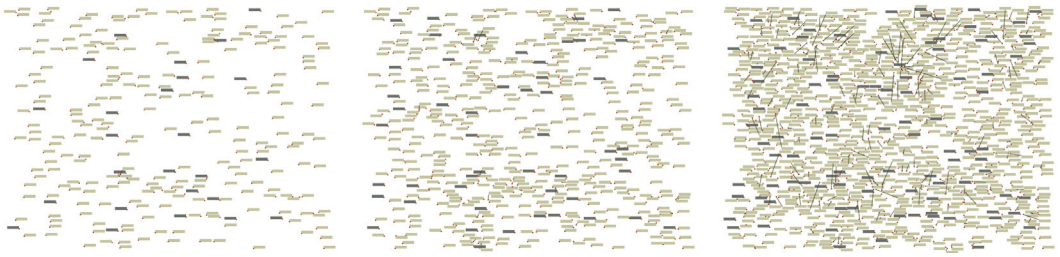
(a) (b)

Abb. 7.20.: Vergleich adjazente Beschriftung und Distanzbeschriftung. Pipeline-Konfiguration A (a) und Pipeline-Konfiguration D (b) — siehe dazu Tabelle 7.3.

PK	*Anzahl Label*							
	500		750		1000		1500	
	%	ms	%	ms	%	ms	%	ms
1	91,54	1	82,57	1	72,93	2	55,66	2
A	95,50	1	89,40	2	81,61	4	65,52	7
D	100,00	2	100,00	4	99,96	10	85,21	59
$\Delta_{D,A}$	4,50	1	10,60	2	18,35	6	19,69	52

Tab. 7.3.: Verschiedene Pipeline-Konfigurationen: Prozentsatz der gesetzten konfliktfreien Label und zugehörige Beschriftungszeiten (in *ms*) für verschiedene Anzahlen von Punktobjekten. Pipeline-Konfiguration (PK): (1) Beschriftungsergebnisse nach dem ersten Pipeline-Schritt, (A) adjacente Beschriftungsergebnisse nach den Pipeline-Schritten 1–3, (D) Beschriftungsergebnisse nach allen Pipeline-Schritten (mit Distanzbeschriftung), ($\Delta_{D,A}$) zeigt den Unterschied zwischen Reihe (A) und (D). Spiralen-Parameters für (D): $r_{max} = 150$, $c = 20$, $d = -1$, $m_{max} = 500$. Labelgröße ist 30×7 und Bildgröße ist 792×612 (entsprechend [CMS95]).

schriftungen als Distanzbeschriftungen. Bei zunehmender Anzahl zu beschriftender Objekte steigt auch die Anzahl der Distanzbeschriftungen. Distanzbeschriftungen verringern — auch wegen der zusätzlichen Verbindungslinien von Objekt zu Label (teilweise kreuzend) — die Lesbarkeit. Jedoch wird in realistischen Szenarien eine gute Lesbarkeit erzielt (Abb. 7.21). Zudem ist das Labelproblem ab einer gewissen Punktdichte nicht lösbar und der prinzipielle Ansatz der Distanzbeschriftung ermöglicht in Szenarien, in denen adjazente Beschriftungen fehlschlagen, noch vollständige Beschriftungsergebnisse. Der Unterschied zwischen den Pipeline-Schritten $1-3$ und dem zusätzlichen Schritt 4 der Distanzbeschriftung sind in Tabelle 7.3 dargestellt. Mit Distanzbeschriftung ist die Konfiguration erkennbar und alle Punktobjekte können beschriftet werden. Zudem ist die beschriebene Technik der erste interaktive Ansatz, der auch dynamische Hintergrundelemente, ohne vorhergehende Vorberechnung berücksichtigen kann.



(a) (b) (c)

Abb. 7.21.: Beschriftungsbeispiele für verschiedene Anzahlen zufällig verteilter Punktobjekte: 250 (a), 500 (b), 1000 (c). Die dunklen Label zeigen Punktobjekte mit höherer Relevanz (siehe Tab. 7.4) — sie werden also zuerst beschriftet. Parameter und Laufzeiten werden in Tabelle 7.3 gegeben.

Anzahl relevanter Label	*Relevante Label beschriftet mit:*	
	Adjazente Beschriftung	*Distanzbeschriftung*
10 aus 100	100,00 %	0,00 %
25 aus 250	99,84 %	0,16 %
50 aus 500	99,64 %	0,36 %
75 aus 750	99,29 %	0,71 %
100 aus 1000	98,33 %	1,67 %
150 aus 1500	95,63 %	4,37 %

Tab. 7.4.: Beschriftung nach Relevanz: Anzahl der relevanten Label mit adjazenter Beschriftung und Distanzbeschriftung. In allen Fällen werden alle relevanten Label (100 %) gesetzt. Die Konfiguration wird in Tabelle 7.3 gegeben.

Performanz Eine weitere, wichtige Anforderung an eine interaktive Beschriftungsumgebung ist die Ausführungsgeschwindigkeit. Der beschriebene Partikel-basierte Ansatz ist schnell und erlaubt eine vollständige Aktualisierung aller Beschriftungspositionen bei interaktiven Geschwindigkeiten — ohne der Verwendung einer Vorberechnung.

Laufzeitmessungen der beschriebenen Methode werden in Tabelle 7.3 gegeben. Innerhalb weniger Millisekunden können bis zu 1000 Punktobjekte beschriftet werden — inklusive Distanzbeschriftung. Nach Erkenntnis des Autors wurden schnellere Ergebnisse ohne der Verwendung eines Vorverarbeitungsschrittes noch nicht veröffentlicht.

Distanzbeschriftung ist der aufwendigste Schritt des beschriebenen Verfahrens. Im Fall von 1500 Punktobjekten auf der gegebenen Fläche müssen bereits $\approx 35\%$ aller Punktobjekte mit der Distanzbeschriftung beschriftet werden. Deshalb sinkt die Beschriftungsperformanz für 1500 Partikel auf 59 ms bei Pipeline-Konfiguration D. Dennoch werden auch in diesem Fall interaktive Frameraten erreicht. Auch in großen Umgebungen bietet das Partikel-basierte Beschriftungsverfahren eine gute Performanz. So können zum Beispiel 100.000 Partikel in weniger als einer Sekunde be-

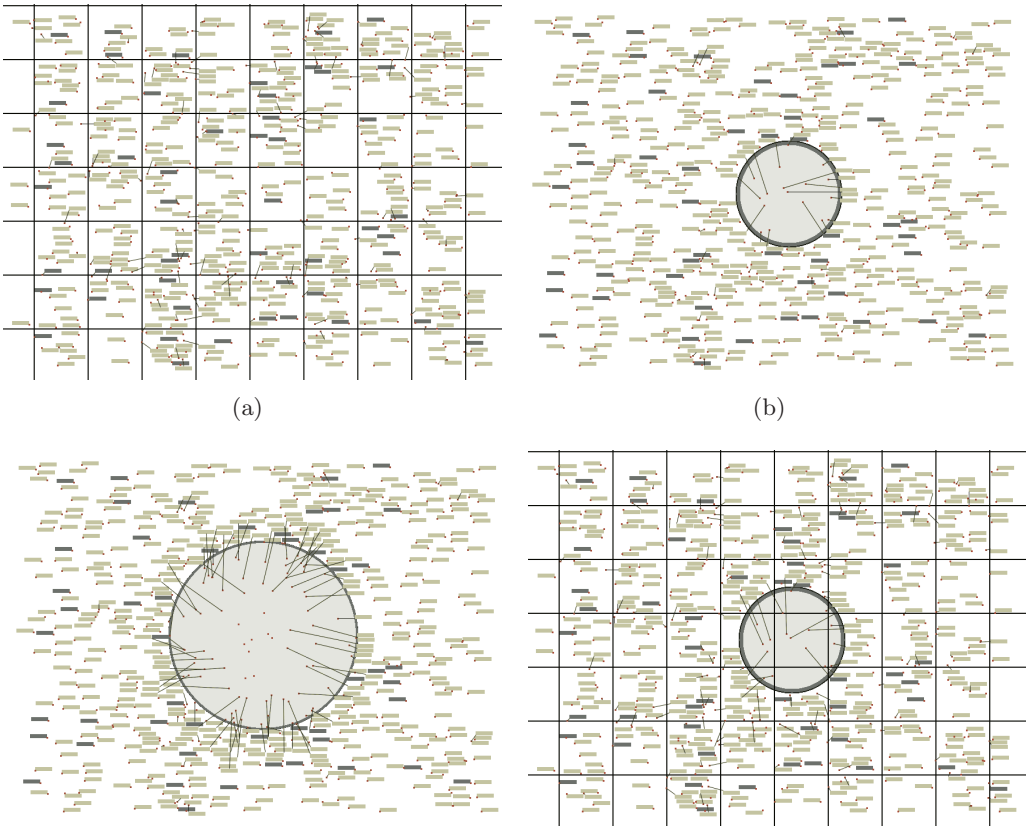


Abb. 7.22.: Beispiel: 500 Punktobjekte werden beschriftet (Konfiguration wie in Abbildung 7.21b). Dabei werden verschiedene Kollisionen-Maps verwendet: Gitter (a), kleine Linse (b), große Linse (c), Gitter & kleine Linse (d). Laufzeiten werden in Tabelle 7.5 gegeben. Die Beschriftungslinsen (vgl. [FP99]) können mit dem vorgestellten Verfahren durch Positionierung virtueller Partikel realisiert werden.

Kollisionen-Map	VP	%	ms
ohne	1587	100,0 (4)	2
Gitter	8339	100,0 (25,4)	20
kleine Linse	3353	100,0 (7,4)	17
große Linse	7109	98,2 (19,8)	34
Gitter & kleine Linse	9760	100,0 (29,2)	24

Tab. 7.5.: Ergebnisse für die Beispiele aus Abbildung 7.22 unter Verwendung von Relevanz und einer Kollisionen-Map: Anzahl virtueller Partikel (VP), Prozentsatz gesetzter Label, Prozentsatz der Distanzbeschriftungen in Klammern und Laufzeit in *ms*. Die Konfiguration wird in Tabelle 7.3 gegeben.

schriftet werden (für genauere Details, siehe [LSC08]).

Bei Verwendung einer Kollisionen-Map sinkt die Performance, da mehr aufwendige Distanzbeschriftungen verwendet werden. Abbildung 7.22 zeigt beispielhafte Konfigurationen unter Berücksichtigung verschiedener Hintergrundbilder. Zeitmessungen werden in Tabelle 7.5 gegeben. Bei den gezeigten Beispielen reduziert sich die Performanz um den Faktor 10 — sie ist jedoch immer noch ausreichend, um eine Interaktion zu gewährleisten.

7.4. Zusammenfassung

Dieses Kapitel thematisierte die Darstellung von Flüssigkeiten — mit dem Schwerpunkt transparenter Flüssigkeiten. Neben der allgemeinen Approximation optischer Effekte wurde die Repräsentation polygonaler Objekte behandelt, die eine Flüssigkeitsoberfläche durchschneiden. Reflexions- und Brechungsphänomene in derartigen Situationen effizient zu behandeln, ist ein bisher ungelöstes Problem und führt in der Regel zu starken Artefakten. Die beschriebene, interaktive Methode approximiert mit Hilfe einer Oberflächenapproximationen und eines Mapping-Ansatzes die Reflexion und Brechung polygonaler Objekte, die die Oberfläche durchschneiden — wobei die brechungstypischen Eigenschaften von Winkelverschiebungen an der Grenzschicht und Skalierung der unter der Oberfläche liegenden Objektteile repräsentiert werden.

Schließlich wird im Bereich der nicht-realistischen Darstellung eine schnelle Methode vorgestellt, die die überdeckungsfreie Beschriftung von Partikelwolken ermöglicht. Diese kann während der Entwicklung Partikel-basierter Flüssigkeitsumgebungen oder der Exploration von Simulationsdaten hilfreich sein, wenngleich die Anwendungsmöglichkeiten weit über die Beschriftung von Flüssigkeitspartikeln hinausgehen. Denn die Methode ist sehr schnell und kann als bisher einziger Beschriftungsansatz beliebige Hintergrundobjekte während der Beschriftung berücksichtigen.

8. Zusammenfassung und Ausblick

Zusammenfassung Die realistische Repräsentation von Flüssigkeiten in der Computergraphik erfordert drei auszuführende Schritte: (1) *Simulation*, (2) *Oberflächenextraktion* und (3) *Darstellung*. Diese wurden zusammenfassend als Liquid-Pipeline bezeichnet. Die einzelnen Schritte der Liquid-Pipeline sind rechentechnisch ausgesprochen aufwendig. Der Großteil existierender Arbeiten befasst sich mit der Ausführung einzelner Schritte der Liquid-Pipeline in nicht-echtzeitfähigen, dreidimensionalen Umgebungen. Eine Umsetzung in Echtzeitumgebungen ist mit einer erheblichen Einschränkung der verfügbaren Rechenzeit verbunden und führt in der Regel zu undetaillierten oder langsamen Animationen oder ist auf die Repräsentation geringer Flüssigkeitsmengen beschränkt. Aus diesem Grund wird im Rahmen dieser Arbeit die Verwendung und Kopplung verschiedener, angepasster Simulationstechniken für interaktive Umgebungen vorgeschlagen, um eine effizientere Ausführung zu erzielen und damit den Detailgrad und/oder das repräsentierbare Volumen zu vergrößern.

Die verschiedenen Methoden können als physikalisch-basiert oder empirisch klassifiziert werden. Der Fokus dieser Arbeit liegt auf physikalisch-basierten Verfahren. Ein Stufenmodell erfasst diese in Hinblick auf Echtzeitumgebungen: Ambiente Wellen, Oberflächensimulation, 2D Strömungssimulation und 3D Strömungssimulation. Die adaptive Verwendung mehrerer der genannten Verfahren in geeigneten Situationen ermöglicht die Darstellung von Flüssigkeitsflächen, physikalischen Effekten und zugleich dreidimensionalen Volumina in interaktiven Umgebungen. Es können mehr Details oder größere Flüssigkeitsmengen repräsentiert werden, als es bei Verwendung einer einzigen Methode möglich wäre. Die Möglichkeiten und Grenzen der einzelnen Verfahren wurden umfassend in Kapitel 4 behandelt und sind zusammengefasst in Tabelle 4.4 dargestellt worden.

In dieser Arbeit wurden neue, konkrete Methoden zu allen drei Schritten der Liquid-Pipeline vorgestellt. Im Rahmen des Stufenmodells wurden neue Simulationskopplungen vorgestellt, die zusammen alle Möglichkeiten des Stufenmodells aufzeigen und deutliche Vorteile gegenüber der Verwendung lediglich eines einzelnen Verfahrens besitzen. Im Einzelnen wurden neue Methoden zur Kopplung folgender Stufen präsentiert:

- Ambiente Wellen + Oberflächensimulation + Bewegte Gitter [CS09b]: Dieser neue Ansatz ermöglicht eine adaptive, Gitter-basierte Simulation auf unendlich ausgedehnten Flüssigkeitsoberflächen (zum Beispiel ein Ozean). So können detaillierte Wellen in definierten Bereichen simuliert werden, die bilateral mit Objekten interagieren.

- Ambiente Wellen + Oberflächensimulation + 2D Strömungssimulation [Cor08]: Die Wellenausbreitung auf bewegten Flüssigkeitsflächen (zum Beispiel Flüssen) kann effizient mit dieser Methode realisiert werden. Zugleich stellt die Methode eine Erweiterung der *Wave Particles*-Methode dar, die lediglich Oberflächenwellen ohne Berücksichtigung einer Strömung repräsentieren kann.

- Ambiente Wellen + Oberflächensimulation + 3D Strömungssimulation [Cor07b]: Durch Kombination einer zwei- und einer dreidimensionalen Simulation können dreidimensionale Strömungen und damit bewegte dreidimensionale Flüssigkeiten mit detaillierten Oberflächenwellen repräsentiert werden. Die ausschließliche Verwendung der dreidimensionalen Simulation würde wesentlich weniger detaillierte Oberflächen ergeben oder einen erheblich höheren Rechenaufwand erfordern.

Des Weiteren beschreibt diese Arbeit die Konstruktion brechender Wellen aus einer physikalisch-basierten zweidimensionalen Strömungssimulation unter Berücksichtigung der natürlichen Symmetrie [CS08]. Besonderes Merkmal der Methode ist die Tatsache, dass eine *interaktive* Repräsentation brechender Wellen erreicht wird — damit stellt die Methode den bisher schnellsten, physikalisch-basierten Ansatz zur Repräsentation brechender Wellen dar.

Die Oberflächenextraktion aus dreidimensionalen Partikelwolken kann effizient mit dem in dieser Arbeit beschriebenen Bild-basierten Ansatz erfolgen [CS09a]. Die Methode wurde für interaktive Umgebungen entworfen und ist schneller als existierende Methoden für dynamische Partikelwolken. Die gute Laufzeit resultiert aus der Tatsache, dass keine polygonale Oberfläche erzeugt werden muß und die Methode vollständig auf der GPU ausgeführt werden kann.

Für die Höhenfeld-basierte Darstellung von Flüssigkeiten wurden einfache und schnelle Approximationen zur Darstellung von Schaum, Kaustiken und Absorption beschrieben. Ein weiterer Fokus lag auf der verbesserten und dennoch schnellen Repräsentation von Reflexionen und Brechungen polygonaler Objekte, die die Flüssigkeitsoberfläche schneiden [Cor07a]. Mit der vorgestellten Methode können die unerwünschten Artefakte existierender Verfahren in diesen Situationen reduziert werden. Zudem kann der Ansatz die brechungstypischen optischen Winkelverschiebungen und Skalierungen der Objektteile darstellen, die unter der Flüssigkeitsoberfläche liegen. Die in dieser Arbeit präsentierte Partikel-basierte Beschriftungsmethode erleichtert die Exploration der physikalischen Parameter Partikel-basierter Flüssigkeiten [LSC08, CLS09]. Darüber hinaus ist sie interessant für viele andere interaktive Umgebungen, in denen eine Beschriftung vonnöten ist. Die Methode ist in der Performanz existierenden Verfahren mindestens ebenbürtig — stellt darüber hinaus jedoch die einzige existierende schnelle Methode dar, die beliebige graphische Objekte beschriften und gleichzeitig Hintergrundobjekte berücksichtigen kann.

Ausblick Obwohl die im Rahmen dieser Arbeit entwickelten Methoden die Flüssigkeitsrepräsentation in interaktiven Umgebungen verbessern, bleibt Raum für zukünftige Arbeiten. Zunächst wäre eine vollständige Umsetzung des hier beschriebenen, allgemeinen Simulationsschemas innerhalb einer einzigen Umgebung wünschenswert, so dass Techniken miteinander kombiniert werden könnten und diese Kombination dann direkt in anderen Programmen verwendet werden könnte. Doch eine derartige Umsetzung ist von großem handwerklichen Aufwand geprägt und würde nur wenig wissenschaftlichen Mehrwert gegenüber den hier gezeigten prototypischen Umsetzungen besitzen. In diesem Rahmen könnte auch eine dreidimensionale FD-Methode hinzugefügt und untersucht werden — in dieser Arbeit wird eine dreidimensionale SPH-Methode verwendet, da sie für interaktive Umgebungen mit wenig Flüssigkeitsvolumen besser geeignet ist. Das vorgestellte Schema schließt jedoch beide Verfahren ein, da es unabhängig von bestimmten Simulationsmethoden formuliert ist.

Die einzelnen präsentierten Methoden bieten auch Raum für Verbesserungen. Die Kopplung mit Videosequenzen wurde in dieser Arbeit nicht betrachtet, wenngleich ihre Anwendbarkeit für gewisse Effekte intuitiv nachvollziehbar ist und den Realismus erhöhen kann. Auch die Erzeugung dreidimensionaler Gischt innerhalb der Höhenfeld-basierten Verfahren erscheint sinnvoll. Insbesondere SPH-Partikel, die das Volumen verlassen, könnten zur Generation verwendet werden. Die beschriebene Oberflächenextraktion kann durch Verwendung kantenerhaltender Filter wesentlich verbessert werden, da hierdurch innere Konturen erhalten bleiben und die Oberflächen weniger verschwommen wirken würden. Diese sind jedoch aufwendig und (noch) nicht in interaktiven Umgebungen anwendbar.

Ein bisher nicht untersuchtes Gebiet im Bereich der Simulation von Flüssigkeiten ist die Verwendung von Techniken aus dem Bereich des *Non-Photorealistic-Renderings*, um zum Beispiel die aus Trickfilmen bekannten Repräsentationen von Flüssigkeiten automatisch generieren zu können. Physikalische Informationen könnten verwendet werden, um Techniken aus diesem Bereich zu adaptieren. Gerade Strömungsinformationen könnten dabei als Metainformationen eine wertvolle Basis für verschiedene Darstellungsstile darstellen. So wäre es beispielsweise möglich, strömende Flüssigkeiten künstlerisch zu schraffieren, wobei die Schraffur entsprechend der Strömungsrichtung orientiert ist. Bei Entwicklung geeigneter Algorithmen unter Verwendung der GPU, könnten derartige Verfahren durchaus mit interaktiven Geschwindigkeiten ausgeführt werden.

Die in dieser Arbeit präsentierten Methoden stellen erste Schritte zur Verbesserung der Qualität computergenerierter Flüssigkeiten in interaktiven Umgebungen dar. Ein grundsätzliches Ziel bleibt es jedoch, den Detailgrad weiter zu verbessern. Die Komplexität und der Detailreichtum realer Flüssigkeiten ist derart hoch, dass die Erzeugung virtueller und zugleich realistischer Flüssigkeiten in interaktiven Umgebungen weiterhin ein wichtiges Forschungsthema in der physikalisch-basierten Simulation in der Computergraphik bleiben wird. Neue Impulse werden in Zukunft auch durch verbesserte Hardware gegeben — insbesondere schnellere GPUs und

Multi-Core-Architekturen werden die parallele Flüssigkeitssimulation in interaktiven Umgebungen maßgeblich beeinflussen. Als Beispiel sei die Entwicklung von *Larrabee* [SCS+08] erwähnt (eine *Many-Core* x86 Architektur von Intel zur Ausführung einer *Software-Rendering-Pipeline*), die hohe Erwartungen in Bezug auf parallele Simulationen und interaktives Ray Tracing schürt — die erste Version wird laut Intel 2009/2010 veröffentlicht.

Fazit Realistische Flüssigkeitsrepräsentationen in der Computergraphik sind aufwendig und in der Regel nicht in interaktiven Umgebungen zu erreichen. Um dennoch interaktive Flüssigkeiten in verschiedensten Szenarien realisieren zu können, wird in dieser Arbeit die Verwendung unterschiedlicher Simulationstechniken vorgeschlagen. So kann die Qualität der Ergebnisse verbessert werden. Gleichzeitig wird die Darstellung von Effekten ermöglicht, die mit einer einzigen Simulation prinzipiell nicht möglich sind. Auch die im Rahmen dieser Arbeit vorgestellten Verfahren zur Oberflächenextraktion und Darstellung beschleunigen oder verbessern die Repräsentation von Flüssigkeiten in interaktiven Umgebungen. Dennoch bleibt es ein generelles Ziel, die hohe Qualität von Nicht-Echtzeit-Flüssigkeiten auch in interaktiven Umgebungen zu erreichen.

A. Smoothed Particle Hydrodynamics (SPH)

In dieser Arbeit wird unter anderem eine SPH-Methode verwendet. Diese und die im Rahmen dieser Arbeit verwendeten Kernel werden der Vollständigkeit halber in diesem Anhang kurz beschrieben. Ein ausführliche Einführung der SPH-Methode zur Lösung der Navier-Stokes Gleichungen wird in [Mon92] gegeben. Die Grundidee liegt darin, das zu simulierende Liquidvolumen mit Hilfe von Partikeln zu diskretisieren. Diese beschreiben die physikalischen Eigenschaften des Liquides in ihrer jeweiligen Umgebung. Jedes Partikel besitzt eine Position $\mathbf{x} = (x, y, z)$, eine Geschwindigkeit $\mathbf{v} = (v_x, v_y, v_z)$ und eine Masse m. Mit Hilfe dieser Informationen können die Navier-Stokes Gleichungen gelöst werden. Dazu wird zunächst ein *Smoothing Kernel* $W_h(\mathbf{x})$ eingeführt. Dieser dient dazu, die physikalischen Informationen, die lediglich an den diskreten Partikelpositionen bekannt sind, auf das kontinuierliche Volumen zu verteilen beziehungsweise zu glätten. Der *Smoothing Kernel* $W_h(\mathbf{x})$ ist somit ein Interpolationskernel, der zum einen die Volumenerhaltung erfüllen muss:

$$\int W_h(\mathbf{x}) d\mathbf{x} = 1. \tag{A.1}$$

Zum anderen muss er im Limit $h \to \infty$ in die Dirac'sche Delta Funktion übergehen:

$$\lim_{h\to\infty} W_h(\mathbf{x}) = \delta(\mathbf{x}). \tag{A.2}$$

Die Smoothing Länge h definiert die Größe des *Smoothing Kernels*:

$$W_h(\|\mathbf{x}\| > h) = 0. \tag{A.3}$$

Somit beschreibt h die maximale Distanz, bei der Partikel noch miteinander interagieren. Physikalische skalare Größen $A(\mathbf{x})$ können mit der SPH-Methode dann für n Partikel wie folgt bestimmt werden:

$$\langle A(\mathbf{x})\rangle = \sum_{i=1}^{n} \frac{m_i}{\rho(\mathbf{x}_i)} A(\mathbf{x}_i) W_h(\mathbf{x} - \mathbf{x_i}), \tag{A.4}$$

$$\langle \nabla A(\mathbf{x})\rangle = \sum_{i=1}^{n} \frac{m_i}{\rho(\mathbf{x}_i)} A(\mathbf{x}_i) \nabla W_h(\mathbf{x} - \mathbf{x_i}). \tag{A.5}$$

So kann zum Beispiel der Druck bestimmt werden und zur Integration der Partikel und Geschwindigkeitspositionen verwendet werden. Im Rahmen dieser Arbeit werden Kernel für Druck und Viskosität verwendet, die in der Druckkraft $\mathbf{F}_i^{Druck}$ und der Viskositätskraft $\mathbf{F}_i^{Visk}$ resultieren. Dabei werden in dieser Arbeit diejenigen Kernel verwendet, die in [MCG03] vorgestellt wurden. Im Folgenden werden sie in Anlehnung an [MCG03] gegeben:

$$W_h^{\text{Druck}}(x) = \frac{15}{\pi h^6} \begin{cases} (h-x)^3 & : \quad 0 \leq x \leq h \\ 0 & : \quad \text{sonst}, \end{cases} \tag{A.6}$$

$$W_h^{\text{Visk}}(x) = \frac{15}{2\pi h^3} \begin{cases} -\frac{r^3}{2h^3} + \frac{r^2}{h^2} + \frac{h-2r}{2r} & : \quad 0 \leq x \leq h \\ 0 & : \quad \text{sonst}, \end{cases} \tag{A.7}$$

$$\mathbf{F}_i^{\text{Druck}} = -\sum_j m_j \frac{p_i + p_j}{2\rho_j} \nabla W_h^{\text{Druck}}(\mathbf{x}_i - \mathbf{x}_j), \tag{A.8}$$

$$\mathbf{F}_i^{\text{Visk}} = -\mu \sum_j m_j \frac{\mathbf{v}_j - \mathbf{v}_i}{2\rho_j} \nabla^2 W_h^{\text{Visk}}(\mathbf{x}_i - \mathbf{x}_j). \tag{A.9}$$

Entsprechend Gleichung A.3 fließen in die Berechnung der physikalischen Eigenschaften an einer gegebenen Stelle lediglich die Nachbarn ein, die einen Abstand von weniger als h zur gegebenen Position besitzen. Diese können in der Praxis schnell mit Hilfe einer Gitter-basierten, orthogonalen Datenstruktur mit Gitterabstand h gefunden werden, in der jedes Gitterelement die inneliegenden Partikel kennt. Somit müssen lediglich die Partikel in benachbarten Gitterelementen auf ihren Abstand hin überprüft werden, ob sie die physikalischen Eigenschaften der gegebenen Position mit beeinflussen — so kann der numerische Aufwand wesentlich reduziert werden.

Akronyme und Notationen

Akronyme

CPU	Central Processing Unit
FD	Finite Difference
FDM	Finite Difference Method
FEM	Finite Element Method
FVM	Finite Volume Method
FPS	Frames per Second
GPU	Graphics Processing Unit
MLS	Moving Least-Squares
MPS	Moving-Particle Semi-Implicit
SPH	Smoothed Particle Hydrodynamics

Mathematische Notationen

$a, b, \ldots$	Skalare Größen
$\tilde{a}, \tilde{b}, \ldots$	Durchschnittswerte skalarer Größen
$\mathbf{x} = \begin{pmatrix} x_x \\ x_y \\ \vdots \end{pmatrix}$	Spaltenvektor
$\|\mathbf{x}\|$	Betrag eines Vektors
$\hat{\mathbf{x}}, \hat{\mathbf{y}}, \ldots$	Normierte Vektoren
$\mathbf{x}^y = x_y$	Element eines Spaltenvektors
$\mathbf{x}^T = (x_x, x_y, \ldots)^T$	Zeilenvektor
$\mathcal{M}, \mathcal{R}$	Matrizen
$\mathbf{x} \cdot \mathbf{y}$	Skalarprodukt
$\mathbf{x} \times \mathbf{y}$	Kreuzprodukt
$\frac{\partial}{\partial x}$	Partielles Differential
∇	Gradient, in $\mathbb{R}^3$: $\nabla = (\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z})$
$\triangle$	Laplace-Operator, in $\mathbb{R}^3$: $\triangle = \nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}$

Physikalische Notationen

$\mathbf{x}$	Position
$\mathbf{v}$	Geschwindigkeit
$\mathbf{g}$	Fallbeschleunigung
c	Konstante Geschwindigkeit
t	Zeit
Δt	Zeitschritt
p	Druck
μ	Viskositätskonstante
ρ	Dichte
ω	Winkelgeschwindigkeit
α	Winkelbeschleunigung
ϕ	Potential
Φ	Strahlungsfluss
$\mathbf{F}$	Kraft
$\mathbf{M}$	Drehmoment
J_X	Massenträgheitsmoment bzgl. der Achse X
V	Volumen
$\langle O \rangle$	Erwartungswert der Observablen O
a^t	Variable zum Zeitpunkt t
a_i	Diskreter eindimensionaler Index
$a_{i,j}$	Diskreter zweidimensionaler Index

Literaturverzeichnis

[AIY+04] T. Amada, M. Imura, Y. Yasumuro, Y. Manabe, and K. Chihara. Particle-based fluid simulation on gpu. In *ACM Workshop on General-Purpose Computing on Graphics Processors*, Los Angeles, CA, USA, 2004.

[AL96] M. Aubury and W. Luk. Binomial filters. *The Journal of VLSI Signal Processing*, 12(1):35–50, 1996.

[ALD06] B. Adams, T. Lenaerts, and P. Dutré. Particle splatting: Interactive rendering of particle-based simulation data. Technical Report CW 453, Katholieke Universiteit Leuven, Leuven, Belgium, 2006.

[Ama06] T. Amada. *Real-Time Particle Based Fluid Simulation with Rigid Body Interaction*, pages 189–205. Game Programming Gems 6. Charles River Media, Boston, MA, USA, 2006.

[AMHH08] T. Akenine-Möller, E. Haines, and N. Hoffman. *Real-Time Rendering 3rd Edition*. A. K. Peters, Ltd., Natick, MA, USA, 2008.

[APKG07] B. Adams, M. Pauly, R. Keiser, and L. J. Guibas. Adaptively sampled particle fluids. *ACM Transactions on Graphics*, 26(3):48–55, 2007.

[ASA07] N. H. Anh, A. Sourin, and P. Aswani. Physically based hydraulic erosion simulation on graphics processing unit. In *Proceedings of the 5th international conference on computer graphics and interactive techniques in Australia and Southeast Asia (GRAPHITE '07)*, pages 257–264, New York, NY, USA, 2007. ACM.

[ASHU05] M. Arigovindan, M. Suhling, P. Hunziker, and M. Unser. Variational image reconstruction from arbitrarily spaced samples: a fast multi-resolution spline solution. *Image Processing, IEEE Transactions on*, 14(4):450–460, April 2005.

[Bar97] D. Baraff. An introduction to physically based modeling: Rigid body simulation i - unconstrained rigid body dynamics. In *Course Notes of the 24th annual conference on Computer graphics and interactive techniques (SIGGRAPH '97)*, 1997.

[BD06a] L. Baboud and X. Décoret. Realistic water volumes in real-time. In *Eurographics Workshop on Natural Phenomena*, pages 25–32, Vienna, Austria, 2006. Eurographics Association.

[BD06b] L. Baboud and X. Décoret. Rendering geometry with relief textures. In *Proceedings of Graphics Interface (GI)*, pages 195–201, Toronto, Canada, 2006. Canadian Information Processing Society.

[BDY06] K. Been, E. Daiches, and C. Yap. Dynamic map labeling. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):773–780, 2006.

[Bel03] V. Belyaev. Real-time simulation of water surface. In *Proceedings of GraphiCon 2003*, pages 131–138, Moscow, Russia, 2003.

[Ben07] B. Benes. Real-time erosion using shallow water simulation. In *Proceedings of the 4th Workshop in Virtual Reality Interactions and Physical Simulations (VRIPHYS '07)*, pages 43–50, Dublin, Ireland, 2007.

[BGOS06] A. W. Bargteil, T. G. Goktekin, J. F. O'Brien, and J. A. Strain. A semi-lagrangian contouring method for fluid simulation. *ACM Transactions on Graphics*, 25(1):19–38, 2006.

[BK03] M. Botsch and L. Kobbelt. High-quality point-based rendering on modern gpus. In *Proceedings of the 11th Pacific Conference on Computer Graphics and Applications (PG '03)*, pages 335–343, Washington, DC, USA, 2003. IEEE Computer Society.

[Bli82] J. F. Blinn. A generalization of algebraic surface drawing. *ACM Transactions on Graphics*, 1(3):235–256, 1982.

[BMG+99] D. Blythe, T. McReynolds, B. Grantham, M. J. Kilgard, and S. R. Nelson. Advanced graphics programming techniques using opengl. In *Course Notes of the 26th annual conference on Computer graphics and interactive techniques (SIGGRAPH '99)*, 1999.

[BN76] J. F. Blinn and M. E. Newell. Texture and reflection in computer generated images. *Communications of the ACM*, 19(10):542–547, 1976.

[BR86] J. U. Brackbill and H. M. Ruppel. Flip: A method for adaptively zoned, particle-in-cell calculations of fluid flows in two dimensions. *Journal of Computational Physics*, 65(2):314–343, 1986.

[BSM+06] A. W. Bargteil, F. Sin, J. E. Michaels, T. G. Goktekin, and J. F. O'Brien. A texture synthesis method for liquid animations. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '06)*, pages 345–351, Aire-la-Ville, Switzerland, 2006. Eurographics Association.

[BT07] M. Becker and M. Teschner. Weakly compressible sph for free surface flows. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics*

symposium on Computer animation (SCA '07), pages 209–217, Aire-la-Ville, Switzerland, 2007. Eurographics Association.

[BTT09] M. Becker, H. Tessendorf, and M. Teschner. Direct forcing for lagrangian rigid-fluid coupling. *IEEE Transactions on Visualization and Computer Graphics*, 15(3):493–503, 2009.

[CBP05] S. Clavet, P. Beaudoin, and P. Poulin. Particle-based Viscoelastic Fluid Simulation. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation (SCA '05)*, pages 219–228, New York, NY, USA, 2005. ACM.

[CdVL95] J. X. Chen and d. N. V. Lobo. Toward interactive-rate simulation of fluids with moving obstacles using navier-stokes equations. *Graphical Models and Image Processing*, 57(2):107–116, 1995.

[CGG+03] P. Cignoni, F. Ganovelli, E. Gobbetti, F. Marton, F. Ponchio, and R. Scopigno. Bdam - batched dynamic adaptive meshes for high performance terrain visualization. *Computer Graphics Forum*, 22(3):505–514, 2003.

[CHJ03] C. S. Co, B. Hamann, and K. I. Joy. Iso-splatting: A point-based alternative to isosurface visualization. In *Proceedings of the 11th Pacific Conference on Computer Graphics and Applications (PG '03)*, pages 325–334, Washington, DC, USA, 2003. IEEE Computer Society.

[CIS07] K. Crane, I.Llamas, and S.Tariq. *Real-Time Simulation and Rendering of 3D Fluids*, pages 633–673. GPU Gems 3. Addison-Wesley, Boston, MA, USA, 2007.

[CJR95] S. Chen, D. B. Johnson, and P. E. Raad. Velocity boundary conditions for the simulation of free surface fluid flow. *Journal of Computational Physics*, 116(2):262–276, 1995.

[CJRF97] S. Chen, D. B. Johnson, P. E. Raad, and D. Fadda. The surface marker and micro cell method. *International Journal for Numerical Methods in Fluids*, 25(7):749–778, 1997.

[CLHM97] J. X. Chen, N. d. V. Lobo, C. E. Hughes, and J. M. Moshell. Real-time fluid simulation in a dynamic virtual environment. *IEEE Computer Graphics and Applications*, 17(3):52–61, 1997.

[CLS09] H. Cords, M. Luboschik, and H. Schumann. Floating labels: Improving dynamics of interactive labeling approaches. In *Proceedings of Computer Graphics, Visualization, Computer Vision and Image Processing (CGVCVIP '09)*, pages 235–238, Algarve, Portugal, 2009.

[CMS95] J. Christensen, J. Marks, and S. Shieber. An empirical study of algorithms for point-feature label placement. *ACM Transactions on Graphics*, 14(3):203–232, 1995.

[CMT04] M. Carlson, P. Mucha, and G. Turk. Rigid fluid: Animating the interplay between rigid bodies and fluid. *ACM Transactions on Graphics*, 23(3):377–384, 2004.

[CMVHT02] M. Carlson, P. J. Mucha, R. B. Van Horn, and G. Turk. Melting and flowing. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation (SCA '02)*, pages 167–174, New York, NY, USA, 2002. ACM.

[Cor07a] H. Cords. Refraction of water surface intersecting objects in interactive environments. In *Proceedings of the 4th Workshop in Virtual Reality Interactions and Physical Simulations (VRIPHYS '07)*, pages 59–68, Dublin, Ireland, 2007.

[Cor07b] H. Cords. Mode-splitting for highly detailed, interactive liquid simulation. In *Proceedings of the 5th international conference on computer graphics and interactive techniques in Australia and Southeast Asia (GRAPHITE '07)*, pages 265–272, New York, NY, USA, 2007. ACM.

[Cor08] H. Cords. Moving with the flow: Wave particles in flowing liquids. *Journals of the 16th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG '08)*, 16(1):145–152, 2008.

[CS08] H. Cords and O. Staadt. Instant liquids. In *Posters at the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '08)*, Aire-la-Ville, Switzerland, 2008. Eurographics Association.

[CS09a] H. Cords and O. Staadt. Interactive screen-space surface rendering of dynamic particle clouds. *Journal of Graphics, GPU & Game Tools (JGT)*, 2009. (accepted).

[CS09b] H. Cords and O. Staadt. Real-time open water environments with interacting objects. In *Proceedings of Eurographics Workshop on Natural Phenomena*, pages 35–42, Munich, Germany, 2009. Eurographics Association.

[DB94] P. J. Diefenbach and N. I. Badler. Pipeline Rendering: Interactive Refractions, Reflections, and Shadows. *Displays (Special Issue on Interactive Computer Graphics)*, 15(3):173–180, 1994.

[DCG07] E. Darles, B. Crespin, and D. Ghazanfarpour. Accelerating and enhancing rendering of realistic ocean scenes. In *Proceedings of the 15th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG '07)*, pages 287–294, Plzen, Czech Republic, 2007.

[DG96] M. Desbrun and M.-P. Gascuel. Smoothed particles: A new paradigm for animating highly deformable bodies. In *Proceedings of the EG Workshop on Animation and Simulation*, pages 61–76. Springer-Verlag, 1996.

[Dil99] G. A. Dilts. Moving-least-squares-particle hydrodynamics - i. consistency and stability. *International Journal for Numerical Methods in Engineering*, 44(8):1115–1155, 1999.

[DS04] C. Dachsbacher and M. Stamminger. Rendering procedural terrain by geometry image warping. In *Proceedings of Eurographics Symposium on Rendering*, pages 103–110, Norrköping, Sweden, 2004.

[DWS+97] M. Duchaineau, M. Wolinsky, D. E. Sigeti, M. Miller, C. Aldrich, and M. B. Mineev-Weinstein. Roaming terrain: Real-time optimally adapting meshes. In *Proceedings of IEEE Visualization '97*, pages 81–88, Phoenix, Arizona, 1997.

[EFFM02] D. Enright, R. Fedkiw, J. Ferziger, and I. Mitchell. A hybrid particle level set method for improved interface capturing. *Journal of Computational Physics*, 183(1):83–116, 2002.

[EMDT06] P. Estalella, I. Martin, G. Drettakis, and D. Tost. A GPU-driven Algorithm for Accurate Interactive Reflections on Curved Objects. In *Proceedings of Eurographics Symposium on Rendering*, pages 313–318, Cyprus, 2006.

[EMF02] D. Enright, S. Marschner, and R. Fedkiw. Animation and rendering of complex water surfaces. *ACM Transactions on Graphics*, 21(3):736–744, 2002.

[EMP+04] D. S. Ebert, K. F. Musgrave, D. Peachey, K. Perlin, and S. Worley. *Texturing & Modeling: A Procedural Approach (The Morgan Kaufmann Series in Computer Graphics)*. Morgan Kaufmann, San Francisco, CA, USA, 2004.

[FAMO99] R. P. Fedkiw, T. Aslam, B. Merriman, and S. Osher. A non-oscillatory eulerian approach to interfaces in multimaterial flows (the ghost fluid method). *Journal of Computational Physics*, 152(2):457–492, 1999.

[FF01] N. Foster and R. Fedkiw. Practical animation of liquids. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques (SIGGRAPH '01)*, pages 23–30. ACM, 2001.

[FLH+06] G. Fuchs, M. Luboschik, K. Hartmann, K. Ali, T. Strothotte, and H. Schumann. Adaptive labeling for interactive mobile information systems. In *Proceedings of the 10th International Conference Information Visualization (IV'06)*, pages 453–459, 2006.

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

[FM97] N. Foster and D. Metaxas. Controlling fluid animation. In *Proceedings of Computer Graphics International (CGI '97)*, pages 178–188, Washington, DC, USA, 1997. IEEE Computer Society.

[FOK05] B. E. Feldman, J. F. O'Brien, and B. M. Klingner. Animating gases with hybrid meshes. *ACM Transactions on Graphics*, 24(3):904–909, 2005.

[FP99] J.-D. Fekete and C. Plaisant. Excentric labeling: Dynamic neighborhood labeling for data visualization. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'99)*, pages 512–519, 1999.

[FR86] A. Fournier and W. T. Reeves. A simple model of ocean waves. In *Proceedings of the 13th annual conference on Computer graphics and interactive techniques (SIGGRAPH '86)*, pages 75–84, New York, NY, USA, 1986. ACM.

[Fre06] J. Frechot. Realistic simulation of ocean surface using wave spectra. In *Proceedings of the First International Conference on Computer Graphics Theory and Applications (GRAPP '06)*, pages 76–83, Setúbal, Portugal, 2006.

[FW91] M. Formann and F. Wagner. A packing problem with applications to lettering of maps. In *Proceedings of the 7th Annuual ACM Symposium on Computational Geometry (SCG'91)*, pages 281–288, 1991.

[GBO04] T. G. Goktekin, A. W. Bargteil, and J. F. O'Brien. A method for animating viscoelastic fluids. *ACM Transactions on Graphics*, 23(3):463–467, 2004.

[GBP04] G. Guennebaud, L. Barthe, and M. Paulin. Deferred splatting. *Computer Graphics Forum*, 23(3):653–660, 2004.

[GD98] J. P. Grossman and W. J. Dally. Point sample rendering. In *Proceedings of the Eurographics Workshop on Rendering Techniques*, pages 181–192, Vienna, Austria, 1998. Springer.

[GDN95] M. Griebel, T. Dornseifer, and T. Neunhoeffer. *Numerische Simulation in der Strömungsmechanik*. Vieweg, Braunschweig/Wiesbaden, Germany, 1995.

[GDN98] M. Griebel, T. Dornseifer, and T. Neunhoeffer. *Numerical Simulation in Fluid Dynamics, a Practical Introduction*. SIAM, Philadelphia, USA, 1998.

[Ger98] S. Gernot. Image-based object representation by layered impostors. In *Proceedings of the ACM symposium on virtual reality software and technology (VRST '98)*, pages 99–104, New York, NY, USA, 1998. ACM.

[GH86] N. Greene and P. S. Heckbert. Creating raster omnimax images from multiple perspective views using the elliptical weighted average filter. *IEEE Computer Graphics and Applications*, 6(6):21–27, 1986.

[GH04] S. Greenwood and D. House. Better with Bubbles: Enhancing the Visual Realism of Simulated Fluid. In *Proceedings of the 2004 SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '04)*, pages 287–296, Aire-la-Ville, Switzerland, 2004. Eurographics Association.

[GHD03] O. Génevaux, A. Habibi, and J.-M. Dischler. Simulating fluid-solid interaction. In *Proccedings of the Graphics Interface (GI '03)*, pages 31–38, Halifax, Canada, 2003.

[GJD05] F. Goetz, T. Junklewitz, and G. Domik. Real-time marching cubes on the vertex shader. In *Eurographics 2005 Short Presentations*, Dublin, Irland, 2005.

[GLQ06] I. Gijbels, A. Lambert, and P. Qiu. Edge-preserving image denoising and estimation of discontinuous surfaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(7):1075–1087, 2006.

[GM77] R. A. Gingold and J. J. Monaghan. Smoothed particle hydrodynamics - theory and application to non-spherical stars. *Monthly Notices of the Royal Astronomical Society*, 181:375–389, 1977.

[Gom00] M. Gomez. *Interactive Simulation of Water Surfaces*, pages 187–195. Game Programming Gems. Charles River Media, Boston, MA, USA, 2000.

[Gos90] M. E. Goss. Motion simulation: A real time particle system for display of ship wakes. *IEEE Computer Graphics and Applications*, 10(3):30–35, 1990.

[Gre86] N. Greene. Environment mapping and other applications of world projections. *IEEE Computer Graphics and Applications*, 6(11):21–29, 1986.

[GSLF05] E. Guendelman, A. Selle, F. Losasso, and R. Fedkiw. Coupling water and smoke to thin deformable and rigid shells. In *Proceedings of the 32th annual conference on Computer graphics and interactive techniques (SIGGRAPH '05)*, pages 973–981, New York, NY, USA, 2005. ACM.

[Har04] M. Harris. *Fast fluid dynamics simulation on the GPU*. GPU Gems. Charles River Media, Boston, MA, USA, 2004.

[HC72] C. W. Hirt and J. L. Cook. Calculating three-dimensional flows around structures and over rough terrain. *Journal of Computational Physics*, 10(24):324–340, 1972.

[HDD+92] H. Hoppe, T. Derose, T. Duchamp, J. Mcdonald, and W. Stuetzle. Surface reconstruction from unorganized points. In *Proceedings of the 19th annual conference on Computer graphics and interactive techniques (SIGGRAPH '92)*, pages 71–78, New York, NY, USA, 1992. ACM.

[HH09] R. Hoetzlein and T. Höllerer. Interactive water streams with sphere scan conversion. In *Proceedings of the 2009 symposium on Interactive 3D graphics and games (I3D '09)*, pages 107–114, New York, NY, USA, 2009. ACM.

[HK05] J.-M. Hong and C.-H. Kim. Discontinuous fluids. *ACM Transactions on Graphics*, 24(3):915–920, 2005.

[HKK07a] T. Harada, S. Koshizuka, and Y. Kawaguchi. Smoothed particle hydrodynamics on gpus. In *Proceedings of Computer Graphics International (CGI '07)*, pages 63–70, Petrópolis, Brazil, 2007.

[HKK07b] T. Harada, S. Koshizuka, and Y. Kawaguchi. Real-time fluid simulation coupled with cloth. In *Proceedings of Theory and Practice of Computer Graphics*, pages 13–20, Bangor, UK, 2007.

[HN81] C. W. Hirt and B. D. Nichols. Volume of fluid (vof) method for the dynamics of free boundaries. *Journal of Computational Physics*, 39:201–225, 1981.

[HNC02] D. Hinsinger, F. Neyret, and M.-P. Cani. Interactive animation of ocean waves. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation (SCA '02)*, pages 161 – 166, New York, NY, USA, 2002. ACM.

[HP01] M. Hiltzig and A. Pham. Synthetic actors guild. *Los Angeles Times*, May 8, 2001.

[HQ07] W. Hu and K. Qin. Interactive approximate rendering of reflections, refractions, and caustics. *IEEE Transactions on Visualization and Computer Graphics*, 13(1):46–57, 2007.

[HS01] Z. S. Hakura and J. M. Snyder. Realistic reflections and refractions on graphics hardware with hybrid rendering and layered environment maps. In *Proceedings of the 12th Eurographics Workshop on Rendering Techniques*, pages 289–300, London, UK, 2001.

[HTKK07] T. Harada, M. Tanaka, S. Koshizuka, and Y. Kawaguchi. Real-time particle-based simulation on gpus. In *Posters of the 34th annual conference on Computer graphics and interactive techniques (SIGGRAPH '07)*, page 52, New York, NY, USA, 2007. ACM.

[HW65] F. H. Harlow and J. E. Welch. Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface. *Physics of Fluids*, 8(12):2182–2189, 1965.

[HW04] N. Holmberg and B. C. Wünsche. Efficient modeling and rendering of turbulent water over natural terrain. In *Proceedings of the 2nd international conference on Computer graphics and interactive techniques in Australasia and South East Asia (GRAPHITE '04)*, pages 15–22, New York, NY, USA, 2004. ACM.

[IDN03] K. Iwasaki, Y. Dobashi, and T. Nishita. A fast rendering method for refractive and reflective caustics due to water surfaces. *Computer Graphics Forum*, 22(3):601–609, 2003.

[IDYN06] K. Iwasaki, Y. Dobashi, F. Yoshimoto, and T. Nishita. Real-time rendering of point based water surfaces. In *Proceedings of Computer Graphics International (CGI '06)*, pages 102–114, 2006.

[IGLF06] G. Irving, E. Guendelman, F. Losasso, and R. Fedkiw. Efficient simulation of large bodies of water by coupling two and three dimensional techniques. In *Proceedings of the 33rd annual conference on Computer graphics and interactive techniques (SIGGRAPH '06)*, pages 805–811, New York, NY, USA, 2006. ACM.

[IND01] K. Iwasaki, T. Nishita, and Y. Dobashi. Efficient rendering of optical effects within water using graphics hardware. In *Proceedings of the 9th Pacific Conference on Computer Graphics and Applications (PG '01)*, pages 374–383, Washington, DC, USA, 2001. IEEE Computer Society.

[IODN06] K. Iwasaki, K. Ono, Y. Dobashi, and T. Nishita. Point-based rendering of water surfaces with splashes simulated by particle-based simulation. In *Proceedings of NICOGRAPH*, Seoul, Korea, 2006.

[IS04] J. Iversen and R. Sakaguchi. Growing up with fluid simulation on *The Day After Tomorrow*. In *Sketches of the 31st International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '04)*, page 142, New York, NY, USA, 2004. ACM.

[JBS03] S. Jeschke, H. Birkholz, and H. Schumann. A procedural model for interactive animation of breaking ocean waves. In *Poster Proceedings of the 11th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG '03)*, Plzen, Czech Republic, 2003.

[JC95] H. W. Jensen and N. J. Christensen. Photon maps in bidirectional monte carlo ray tracing of complex objects. *Computers & Graphics*, 19(2):215–224, 1995.

[Jen01] H. W. Jensen. *Realistic image synthesis using photon mapping*. A. K. Peters, Ltd., Natick, MA, USA, 2001.

[JG01] L. Jensen and R. Golias. Deep water animation and rendering. In *Proceedings of the Game Developers Conference Europe*, London, England, 2001.

[JH03] G. James and M. Harris. Simulation and animation using hardware accelerated procedural textures. In *Tutorial at the Game Developers Conference*, San Jose, CA, USA, 2003.

[Joh04] C. Johanson. Real-time water rendering - introducing the projected grid concept. Master's thesis, Lund University, Lund, Sweden, 2004.

[Kal08] D. Kallin. Real time large scale fluids for games. In *Proceedings of Sigrad*, Stockholm, Sweden, 2008.

[KBKv09] P. Krištof, B. Beneš, J. Křivánek, and O. Šťava. Hydraulic erosion using smoothed particle hydrodynamics. *Computer Graphics Forum*, 28(2):219–228, 2009.

[KBW06] J. Krüger, K. Bürger, and R. Westermann. Interactive screen-space accurate photon tracing on GPUs. In *Rendering Techniques: 17th Eurographics Symposium on Rendering*, pages 319–329, Cyprus, 2006.

[KC05] A. Kolb and N. Cuntz. Dynamic particle coupling for gpu-based fluid simulation. In *Proceedings of the 18th Symposium on Simulation Technique*, pages 722–727, Erlangen, Germany, 2005.

[KC07] T. Kim and M. Carlson. A simple boiling module. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation (SCA '07)*, pages 27–34, Aire-la-Ville, Switzerland, 2007. Eurographics Association.

[KCC+06] J. Kim, D. Cha, B. Chang, B. Koo, and I. Ihm. Practical animation of turbulent splashing water. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation (SCA '06)*, pages 335–344, Aire-la-Ville, Switzerland, 2006. Eurographics Association.

[KEBK05] V. Kwatra, I. Essa, A. Bobick, and N. Kwatra. Texture optimization for example-based synthesis. *ACM Transactions on Graphics*, 24(3):795–802, 2005.

[KEWE03] T. Klein, M. Eissele, D. Weiskopf, and T. Ertl. Simulation, modelling and rendering of incompressible fluids in real time. In *Proceedings of Workshop on Vision, Modelling, and Visualization (VMV '03)*, pages 365–373, 2003.

[KFCO06] B. M. Klingner, B. E. Feldman, N. Chentanez, and J. F. O'Brien. Fluid animation with dynamic meshes. 25(3):820–825, 2006.

[KLL+07] B. Kim, Y. Liu, I. Llamas, X. Jiao, and J. Rossignac. Simulation of bubbles in foam with the volume control method. *ACM Transactions on Graphics*, 26(3):98, 2007.

[KM90] M. Kass and G. Miller. Rapid, stable fluid dynamics for computer graphics. In *Proceedings of the 17th annual conference on Computer graphics and interactive techniques (SIGGRAPH '90)*, pages 49–57, New York, NY, USA, 1990. ACM.

[Kry05] Y. Kryachko. *Using Vertex Texture Displacement for Realistic Water Rendering*, pages 283–294. GPU Gems 2. Addison-Wesley, Boston, MA, USA, 2005.

[KSK04] N. Kondoh, S. Sasagawa, and A. Kunimatsu. Creating animations of fluids and cloth with moving characters. In *Sketches of the 31st annual conference on Computer graphics and interactive techniques (SIGGRAPH '04)*, page 136, New York, NY, USA, 2004. ACM.

[KSW04] P. Kipfer, M. Segal, and R. Westermann. Uberflow: a gpu-based particle engine. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS*

conference on Graphics hardware (HWWS '04), pages 115–122, New York, NY, USA, 2004. ACM.

[KT96] K. G. Kakoulis and I. G. Tollis. On the edge label placement problem. In *Proceedings of the Symposium on Graph Drawing (GD'96)*, pages 241–256, 1996.

[KVG02] H. Kück, C. Vogelgsang, and G. Greiner. Simulation and rendering of liquid foams. In *Proceedings of Graphics Interface (GI '02)*, pages 81–88, Calgary, Alberta, 2002.

[KW06] P. Kipfer and R. Westermann. Realistic and interactive simulation of rivers. In *Proceedings of Graphics Interface (GI '06)*, pages 41–48, Quebec, Canada, 2006.

[Lac07] M. L. Lachman. An open programming architecture for modeling ocean waves. In *Proceedings of IMAGE Conference*, Scottsdale, Arizona, 2007.

[LAD08] T. Lenaerts, B. Adams, and P. Dutré. Porous flow in particle-based fluid simulations. *ACM Transactions on Graphics*, 27(3):49–56, 2008.

[LC87] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. *Computer Graphics*, 21(4), 1987.

[LD09] T. Lenaerts and P. Dutré. Mixing fluids and granular materials. *Computer Graphics Forum*, 28(2):213–218, 2009.

[LGF04] F. Losasso, F. Gibou, and R. Fedkiw. Simulating water and smoke with an octree data structure. In *Proceedings of the 31st annual conference on Computer graphics and interactive techniques (SIGGRAPH '04)*, pages 457–462, New York, NY, USA, 2004. ACM.

[LH04] F. Losasso and H. Hoppe. Geometry clipmaps: terrain rendering using nested regular grids. *ACM Transactions on Graphics*, 23(3):769–776, 2004.

[LKR+96] P. Lindstrom, D. Koller, W. Ribarsky, L. F. Hodges, N. Faust, and G. A. Turner. Real-time, continuous level of detail rendering of height fields. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques (SIGGRAPH '96)*, pages 109–118, New York, NY, USA, 1996. ACM.

[Lov03] J. Loviscach. Complex water effects at interactive frame rates. In *Proceedings of the 11th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG '03)*, pages 298–305, Plzen, Czech Republic, 2003.

[LSC08] M. Luboschik, H. Schumann, and H. Cords. Particle-based labeling: Fast point-feature labeling without obscuring other visual features. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1237–1244, 2008.

[LSSF06] F. Losasso, T. Shinar, A. Selle, and R. Fedkiw. Multiple interacting liquids. *ACM Transactions on Graphics*, 25(3):812–819, 2006.

[LTKF08] F. Losasso, J. Talton, N. Kwatra, and R. Fedkiw. Two-way coupled sph and particle level set fluid simulation. *IEEE Transactions on Visualization and Computer Graphics*, 14(4):797–804, 2008.

[LvdP02] A. T. Layton and van de M. Panne. A numerically efficient and stable algorithm for animating water waves. *The Visual Computer*, 18(1):41–53, 2002.

[LW85] M. Levoy and T. Whitted. The use of points as display primitives. In *Technical Report TR 85-022*, Chapel Hill, NC, USA, 1985. University of North Carolina.

[Max81] N. L. Max. Vectorized procedural models for natural terrain: Waves and islands in the sunset. In *Proceedings of the 8th annual conference on Computer graphics and interactive techniques (SIGGRAPH '81)*, pages 317–324, New York, NY, USA, 1981. ACM.

[MCG03] M. Müller, D. Charypar, and M. Gross. Particle-based fluid simulation for interactive applications. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation (SCA '03)*, pages 154–159, Aire-la-Ville, Switzerland, 2003. Eurographics Association.

[MCZ07] K. Museth, M. Clive, and N. B. Zafar. Blobtacular: surfacing particle system in *Pirates of the Caribbean 3*. In *Sketches of the 34th International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '07)*, page 20, New York, NY, USA, 2007. ACM.

[MFC06] M. M. Maes, T. Fujimoto, and N. Chiba. Efficient animation of water flow on irregular terrains. In *Proceedings of the 4th international conference on Computer graphics and interactive techniques in Australasia and Southeast Asia (GRAPHITE '06)*, pages 107–115, New York, NY, USA, 2006. ACM.

[MFC07] M. M. Maes, T. Fujimoto, and N. Chiba. Low-memory and interactive-rate animation of water-column based flows. *The Journal of the Society for Art and Science*, 7(1):1–13, 2007.

[Mit04] J. L. Mitchell. Real-time synthesis and rendering of ocean water. In *ATI Research — Technical Report*. Marlboro, MA, USA, 2004.

[MMS04] V. Mihalef, D. Metaxas, and M. Sussman. Animation and control of breaking waves. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation (SCA '04)*, pages 315–324, Aire-la-Ville, Switzerland, 2004. Eurographics Association.

[Mon92] J. Monaghan. Smoothed Particle Hydrodynamics. *Anual Review of Astronomy and Astrophysics*, 30:543–574, 1992.

[Mon94] J. J. Monaghan. Simulating free surface flows with sph. *Journal of Computational Physics*, 110(2):399–406, 1994.

[Mor00] J. P. Morris. Simulating surface tension with smoothed particle hydrodynamics. *International Journal for Numerical Methods in Fluids*, 33:333–353, 2000.

[Mot07] K. Mote. Fast point-feature label placement for dynamic visualizations. *Information Visualization*, 6(4):249–260, 2007.

[MP89] G. Miller and A. Pearce. Globular dynamics: A connected particle system for animating viscous fluids. *Computers and Graphics*, 13(3):305–309, 1989.

[MS91] J. Marks and S. Shieber. The computational complexity of cartographic label placement. Technical Report TR-05-91, Harvard CS, 1991.

[MSD07] M. Müller, S. Schirm, and S. Duthaler. Screen space meshes. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation (SCA '07)*, pages 9–15, Aire-la-Ville, Switzerland, 2007. Eurographics Association.

[MSKG05] M. Müller, B. Solenthaler, R. Keiser, and M. Gross. Particle-based fluid-fluid interaction. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation (SCA '05)*, pages 237–244, New York, NY, USA, 2005. ACM.

[MSS94] C. Montani, R. Scateni, and R. Scopigno. Discretized marching cubes. In *Proceedings of the conference on Visualization (VIS '94)*, pages 281–287, Los Alamitos, CA, USA, 1994. IEEE Computer Society.

[MST+04] M. Müller, S. Schirm, M. Teschner, B. Heidelberger, and M. Gross. Interaction of fluids with deformable solids. *Computer Animation and Virtual Worlds*, 15(3-4):159–171, 2004.

[MUM+06] V. Mihalef, B. Unlusu, D. Metaxas, M. Sussman, and M. Y. Hussaini. Physics based boiling simulation. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation (SCA '06)*, pages 317–324, Aire-la-Ville, Switzerland, 2006. Eurographics Association.

[MWM87] G. A. Mastin, P. A. Watterberg, and J. F. Mareda. Fourier synthesis of ocean scenes. *IEEE Computer Graphics and Applications*, 7(3):16–23, 1987.

[NC02] K. H. Nielsen and N. J. Christensen. Real-time recursive specular reflections on planar and curved surfaces using graphics hardware. *Journals of the 10th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG '02)*, 10(3):91–98, 2002.

[NH91] G. M. Nielson and B. Hamann. The asymptotic decider: resolving the ambiguity in marching cubes. In *Proceedings of the 2nd conference on Visualization (VIS '91)*, pages 83–91, Los Alamitos, CA, USA, 1991. IEEE Computer Society.

[NN94] T. Nishita and E. Nakamae. Method of displaying optical effects within water using accumulation buffer. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques (SIGGRAPH '94)*, pages 373–379, New York, NY, USA, 1994. ACM.

[OH95] J. F. O'Brien and J. K. Hodgins. Dynamic simulation of splashing fluids. In *Proceedings of the Conference on Computer Animation (CA '95)*, pages 198–205, Washington, DC, USA, 1995. IEEE Computer Society.

[OZH00] J. O'Brien, V. Zordan, and J. Hodgins. Combining active and passive simulations for secondary motion. *IEEE Computer Graphics and Applications*, 20(4):86–96, 2000.

[PA00] S. Premoze and M. Ashikhmin. Rendering natural waters. In *Proceedings of the 8th Pacific Conference on Computer Graphics and Applications (PG '00)*, pages 23–30, Washington, DC, USA, 2000. IEEE Computer Society.

[Paj98] R. Pajarola. Large scale terrain visualization using the restricted quadtree triangulation. In *Proceedings of the conference on Visualization (VIS '98)*, pages 19–26, Los Alamitos, CA, USA, 1998. IEEE Computer Society.

[Pea86] D. R. Peachey. Modeling waves and surf. In *Proceedings of the 13th annual conference on Computer graphics and interactive techniques (SIGGRAPH '86)*, pages 65–74, New York, NY, USA, 1986. ACM.

[Per85] K. Perlin. An image synthesizer. *Computer Graphics*, 19(3):287–296, 1985.

[PFH00] E. Praun, A. Finkelstein, and H. Hoppe. Lapped textures. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques (SIGGRAPH '00)*, pages 465–470, New York, NY, USA, 2000. ACM.

[PGP03] I. Petzold, G. Gröger, and L. Plümer. Fast screen map labeling – data-structures and algorithms. In *Proceedings of the 23rd International Cartographic Conference (ICC'03)*, 2003.

[PM64] W. J. Pierson and L. Moskowitz. A proposed spectral form for fully developed wind seas based on the similarity theory of S. A. Kitaigorodskii. *Journal of Geophysical Research*, 69(24):5181–5190, 1964.

[PTB+03] S. Premoze, T. Tasdizen, J. Bigler, A. Lefohn, and R. Whitaker. Particle-based simulation of fluids. *Computer Graphics Forum*, 22:401–410, 2003.

[PZvBG00] H. Pfister, M. Zwicker, van J. Baar, and M. Gross. Surfels: surface elements as rendering primitives. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques (SIGGRAPH '00)*, pages 335–342, New York, NY, USA, 2000. ACM.

[RB08] I. D. Rosenberg and K. Birdwell. Real-time particle isosurface extraction. In *Proceedings of the 2008 symposium on interactive 3D graphics and games (I3D '08)*, pages 35–43, New York, NY, USA, 2008. ACM.

[Ree83] W. T. Reeves. Particle systems — a technique for modeling a class of fuzzy objects. In *Proceedings of the 10th annual conference on Computer graphics and interactive techniques (SIGGRAPH '83)*, pages 359–375, New York, NY, USA, 1983. ACM.

[REN+04] N. Rasmussen, D. Enright, D. Nguyen, S. Marino, N. Sumner, W. Geiger, S. Hoon, and R. Fedkiw. Directable photorealistic liquids. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation (SCA '04)*, pages 193–202, Aire-la-Ville, Switzerland, 2004. Eurographics Association.

[RH06] D. Roger and N. Holzschuch. Accurate specular reflections in real-time. *Computer Graphics Forum*, 25(3):293–302, 2006.

[RKL+06] R. Rost, J. Kessenich, B. Lichtenbelt, H. Malan, and M. Weiblein. *OpenGL Shading Language, Second Edition*. Addison-Wesley, Boston, MA, USA, 2006.

[RL00] S. Rusinkiewicz and M. Levoy. QSplat: A multiresolution point rendering system for large meshes. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques (SIGGRAPH '00)*, pages 343–352, New York, NY, USA, 2000. ACM.

[RO98] R. Radovitzky and M. Ortiz. Lagrangian finite element analysis of newtonian fluid flows. *International Journal for Numerical Methods in Engineering*, 43:607–619, 1998.

[SA08] L. Szecsi and K. Arman. Procedural ocean effects. In *Shader X 6*, pages 331–350, Boston, MA, USA, 2008. Charles River Media.

[Sch80] B. Schachter. Long crested wave models. *Computer Graphics and Image Processing*, 12(2):187–201, 1980.

[Sch94] C. Schlick. An inexpensive brdf model for physically-based rendering. *Computer Graphics Forum*, 13(3):233–246, 1994.

[SCP+04] M. Shah, J. M. Cohen, S. Patel, P. Lee, and F. Pighin. Extended galilean invariance for adaptive fluid simulation. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation (SCA '04)*, Aire-la-Ville, Switzerland, 2004. Eurographics Association.

[SCS+08] L. Seiler, D. Carmean, E. Sprangle, T. Forsyth, M. Abrash, P. Dubey, S. Junkins, A. Lake, J. Sugerman, R. Cavin, R. Espasa, E. Grochowski, T. Juan, and P. Hanrahan. Larrabee: a many-core x86 architecture for visual computing. *ACM Transactions on Graphics*, 27(3):1–15, 2008.

[Sim90] K. Sims. Particle animation and rendering using data parallel computation. In *Proceedings of the 17th annual conference on Computer graphics and interactive techniques (SIGGRAPH '90)*, pages 405–413, New York, NY, USA, 1990. ACM.

[SK07] M. A. Shah and J. Konttinen. Caustics mapping: An image-space technique for real-time caustics. *IEEE Transactions on Visualization and Computer Graphics*, 13(2):272–280, 2007.

[SM98] C. Stein and N. Max. A particle-based model for water simulation. Technical Report UCRL-JC-129378, Lawrence Livermore National Laboratory, Livermore, CA, USA, 1998.

[Sou05] T. Sousa. *Generic Refraction Simulation*, pages 295–305. GPU Gems 2. Addison-Wesley, Boston, MA, USA, 2005.

[SP09] B. Solenthaler and R. Pajarola. Predictive-corrective incompressible sph. *ACM Transactions on Graphics*, 28(3), (to be published 2009).

[SRF05] A. Selle, N. Rasmussen, and R. Fedkiw. A vortex particle method for smoke, water and explosions. *ACM Transactions on Graphics*, 24(3):910–914, 2005.

[SS06] N. Suárez and A. Susín. A mesh-particle model for fluid animation. In *3th Ibero-American Symposium in Computer Graphics (SIACG-2006)*, pages 13–20, Santiago de Compostela, Spain, 2006.

[SSK05] O.-Y. Song, H. Shin, and H.-S. Ko. Stable but nondissipative water. *ACM Transactions on Graphics*, 24(1):81–97, 2005.

[Sta99] J. Stam. Stable fluids. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques (SIGGRAPH '99)*, pages 121–128, New York, NY, USA, 1999. ACM.

[Sta01] J. Stam. A simple fluid solver based on the fft. *Journal of Graphics Tools*, 6(2):43–52, 2001.

[Sta03] J. Stam. Real-time fluid dynamics for games. *Proceedings of the Game Developers Conference*, 2003.

[Stö07] H. Stöcker, editor. *Taschenbuch der Physik : Formeln, Tabellen, Übersichten*. Harri Deutsch, Frankfurt am Main, Germany, 2007.

[Str97] T. Strohmer. Computationally attractive reconstruction of bandlimited images from irregular samples. *IEEE Transactions on Image Processing*, 6(4):540–548, 1997.

[Suc01] S. Succi. *The Lattice Boltzmann Equation: For Fluid Dynamics and Beyond*. Oxford University Press, Oxford, UK, 2001.

[SW01] J. Schneider and R. Westermann. Towards real-time visual simulation of water surfaces. *Proceedings of the Vision Modeling and Visualization Conference (VMV '01)*, pages 211–218, 2001.

[SY05] L. Shi and Y. Yu. Taming liquids for rapidly changing targets. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation (SCA '05)*, pages 229–236, New York, NY, USA, 2005. ACM.

[Tat06] N. Tatarchuk. Artist-directable real-time rain rendering in city environments. In *Course Notes of the 33rd annual conference on Computer graphics and interactive techniques (SIGGRAPH '06)*, 2006.

[TB87] P. Y. Ts'o and B. A. Barsky. Modeling and rendering waves: wave-tracing using beta-splines and reflective and refractive texture mapping. *ACM Transactions on Graphics*, 6(3):191–214, 1987.

[Tes01] J. Tessendorf. Simulating ocean water. In *Course Notes of the 28th International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '01), Simulating Nature: Realistic and Interactive Techniques (Course 47)*, 2001.

[Tes04] J. Tessendorf. *Interactive Water Surfaces*. Game Programming Gems 4. Charles River Media, Boston, MA, USA, 2004.

[TG02] S. Thon and D. Ghazanfarpour. Real-time animation of running waters based on spectral analysis of navier-stokes equations. In *Proceedings of Computer Graphics International (CGI '02)*, pages 333–346, Bradford, UK, 2002.

[TKR05] N. Thürey, C. Körner, and U. Rüde. Interactive Free Surface Fluids with the Lattice Boltzmann Method. Technical Report 05-4, Universität Erlangen-Nürnberg, Erlangen-Nürnberg, Germany, 2005.

[TMSG07] N. Thürey, M. Müller, S. Schirm, and M. Gross. Real-time breaking waves for shallow water simulation. In *Proceedings of Pacific Graphics*. IEEE Computer Society, 2007.

[TR08] N. Thürey and U. Rüde. Stable free surface flows with the lattice Boltzmann method on adaptively coarsened grids. *Computing and Visualization in Science*, 12(5), 2008.

[TRS06] N. Thürey, U. Rüde, and M. Stamminger. Animation of Open Water Phenomena with coupled Shallow Water and Free Surface Simulation. In *Proceedings of the 2006 Eurographics/ACM SIGGRAPH Symposium on Computer Animation (SCA '06)*, pages 157–166, Aire-la-Ville, Switzerland, 2006. Eurographics Association.

[TS00] C. Trendall and A. J. Stewart. General calculations using graphics hardware with applications to interactive caustics. In *Proceedings of the Eurographics Workshop on Rendering Techniques 2000*, pages 287–298, London, UK, 2000. Springer-Verlag.

[TSS+07] N. Thürey, F. Sadlo, S. Schirm, M. Müller-Fischer, and M. Gross. Real-time simulations of bubbles and foam within a shallow water framework. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation (SCA '07)*, pages 191–198, Aire-la-Ville, Switzerland, 2007. Eurographics Association.

[TUKF02] T. Takahashi, H. Ueki, A. Kunimatsu, and H. Fujii. The simulation of fluid-rigid body interaction. In *Conference abstracts and applications of the 29th annual conference on Computer graphics and interactive techniques (SIGGRAPH '02)*, page 266, New York, NY, USA, 2002. ACM.

[UT92] S. O. Unverdi and G. Tryggvason. A front-tracking method for viscous, incompressible, multi-fluid flows. *Journal of Computational Physics*, 100(1):25–37, 1992.

[VM00] A. Vlachos and J. L. Mitchell. *Refraction Mapping for Liquids in Containers*, pages 594–599. Game Programming Gems. Charles River Media, Boston, MA, USA, 2000.

[Wat90] M. Watt. Light-water interaction using backward beam tracing. In *Proceedings of the 17th annual conference on Computer graphics and interactive techniques (SIGGRAPH '90)*, pages 377–385, New York, NY, USA, 1990. ACM.

[WH04] M. Wiebe and B. Houston. The tar monster: Creating a character with fluid simulation. In *Sketches of the 31st annual conference on Computer graphics and interactive techniques (SIGGRAPH '04)*, page 64, New York, NY, USA, 2004. ACM.

[Whi80] T. Whitted. An improved illumination model for shaded display. *Communications of the ACM*, 23(6):343–349, 1980.

[Wit99] P. Witting. Computational fluid dynamics in a traditional animation environment. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques (SIGGRAPH '99)*, pages 129–136, New York, NY, USA, 1999. ACM.

[Wre03] M. Wrenninge. Fluid simulation for visual effects. Master's thesis, Linköping University, Linköping, Sweden, 2003.

[WS03] M. Wand and W. Straßer. Real-time caustics. *Computer Graphics Forum*, 22(3):611–620, 2003.

[WT90] G. Wyvill and A. Trotman. Ray-tracing soft objects. In *Proceedings of the eighth international conference of the Computer Graphics Society (CG International '90)*, pages 469–476, New York, NY, USA, 1990. Springer-Verlag.

[WW99] H. Weimer and J. Warren. Subdivision schemes for fluid flow. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques (SIGGRAPH '99)*, pages 111–120, New York, NY, USA, 1999. ACM.

[Wym05a] C. Wyman. An approximate image-space approach for interactive refraction. *ACM Transactions on Graphics*, 24(3):1050–1053, 2005.

[Wym05b] C. Wyman. Interactive image-space refraction of nearby geometry. In *Proceedings of the 3rd international conference on Computer graphics and interactive techniques in Australasia and South East Asia (GRAPHITE '05)*, pages 205–211, New York, NY, USA, 2005. ACM.

[WZC+06] Q. Wang, Y. Zheng, C. Chen, F. Tadahiro, and N. Chiba. Efficient rendering of breaking waves using mps method. *Journal of Zhejiang University SCIENCE A*, 7:1018–1025, 2006.

[WZF+03] X. Wei, Y. Zhao, Z. Fan, W. Li, S. Yoakum-Stover, and A. Kaufman. Blowing in the wind. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation (SCA '03)*, pages 75–85, Aire-la-Ville, Switzerland, 2003. Eurographics Association.

[YA91] T. Yabe and T. Aoki. A universal solver for hyperbolic equations by cubic-polynomial interpolation - ii. two- and three-dimensional solvers. *Computer Physics Communications*, 66:233–242, 1991.

[YCL02] M. Yamamoto, G. Camara, and L. A. N. Lorena. Tabu search heuristic for point-feature cartographic label placement. *GeoInformatica*, 6(1):77–90, 2002.

[YCL05] M. Yamamoto, G. Camara, and L. A. N. Lorena. Fast point-feature label placement algorithm for real time screen maps. In *Proceedings of the Brazilian Symposium on GeoInformatics (GEOINFO'05)*, 2005.

[YHK07] C. Yuksel, D. H. House, and J. Keyser. Wave particles. *ACM Transactions on Graphics*, 26(3):99–107, 2007.

[YNBH09] Q. Yu, F. Neyret, E. Bruneton, and N. Holzschuch. Scalable real-time animation of rivers. *Computer Graphics Forum*, 28(2):239–248, 2009.

[YPZL05] X. Yang, X. Pi, L. Zeng, and S. Li. Gpu-based real-time simulation and rendering of unbounded ocean surface. In *Proceedings of the Ninth International Conference on Computer Aided Design and Computer Graphics (CAD-CG '05)*, pages 428–433, Washington, DC, USA, 2005. IEEE Computer Society.

[ZB05] Y. Zhu and R. Bridson. Animating sand as a fluid. *ACM Transactions on Graphics*, 24(3):965–972, 2005.

[ZPvBG01] M. Zwicker, H. Pfister, van J. Baar, and M. Gross. Surface splatting. In *Proceedings of the 28th annual conference on Computer graphics and*

interactive techniques (SIGGRAPH '01), pages 371–378, New York, NY, USA, 2001. ACM.

[ZSP07] Y. Zhang, B. Solenthaler, and R. Pajarola. Gpu accelerated sph particle simulation and rendering. In *Posters of the 34th annual conference on Computer graphics and interactive techniques (SIGGRAPH '07)*, page 9, New York, NY, USA, 2007. ACM.

[ZYP06] W. Zheng, J.-H. Yong, and J.-C. Paul. Simulation of Bubbles. In *Proceedings of the 2006 SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '06)*, pages 325–333, Aire-la-Ville, Switzerland, 2006. Eurographics Association.

Selbstständigkeitserklärung

Ich erkläre, dass ich die eingereichte Dissertation selbstständig und ohne fremde Hilfe verfasst, andere als die von mir angegebenen Quellen und Hilfsmittel nicht benutzt und die den benutzten Werken wörtlich oder inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Hilko Cords
Rostock, 31. Juni 2009

Lebenslauf

Hilko Cords
geboren am 4. April 1978 in Oldenburg (Oldb)

Wissenschaftlicher Werdegang

seit 15.02.2006	Promotionsstudium, Institut für Informatik, Universität Rostock
14.02.2006	Erlangung des akademischen Grades *Diplom-Physiker*, Universität Rostock (Diplomarbeit angefertigt an der Universität Kiel)
06.04.2004	Erlangung des akademischen Grades *Diplom-Informatiker*, Universität Rostock
1999 - 2000	Vordiplom Physik, Universität Stuttgart
1998 - 2000	Vordiplom Informatik, Universität Stuttgart

Liste der wissenschaftlichen Veröffentlichungen und Fachvorträge

Zeitschriften

H. Cords and O. Staadt. **Interactive screen-space surface rendering of dynamic particle clouds**. *Journal of Graphics, GPU & Game Tools (JGT)*, 2009. *(accepted)*.

M. Luboschik, H. Schumann, and H. Cords. **Particle-based labeling: Fast point-feature labeling without obscuring other visual features**. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1237–1244, 2008.

H. Cords. **Moving with the flow: Wave particles in flowing liquids**. *Journals of the 16th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG '08)*, 16(1):145–152, 2008.

Begutachtete Konferenzbeiträge

H. Cords, H. Schumann, and M. Luboschik. **Floating labels: Improving dynamics of interactive labeling approaches**. In *Proceedings of Computer Graphics, Visualization, Computer Vision and Image Processing (CGVCVIP '09)*, pages 235–238, Algarve, Portugal, 2009.

H. Cords and O. Staadt. **Real-Time open water environments with interacting objects**. In *Proceedings of Eurographics Workshop on Natural Phenomena (EGWNP '09)*, pages 35–42, Aire-la-Ville, Switzerland, 2009. Eurographics Association.

H. Cords. **Mode-splitting for highly detailed, interactive liquid simulation**. In *Proceedings of the 5th ACM international conference on computer graphics and interactive techniques in Australia and Southeast Asia (GRAPHITE '07)*, pages 265–272, New York, NY, USA, 2007. ACM.

H. Cords. **Refraction of water surface intersecting objects in interactive environments**. In *Proceedings of the 4th Workshop in Virtual Reality Interactions and Physical Simulations (VRIPHYS '07)*, pages 59–68, Dublin, Ireland, 2007.

Sonstige Veröffentlichungen

H. Cords. **Ein Konzept zur Integration der Visualisierungsumgebung *SimVis* in die Simulationsumgebung *James II***. *Technical Report CS-04-09, University of Rostock*, Rostock, Germany, 2009.

H. Cords and O. Staadt. **Instant liquids**. *Poster at the ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '08)*, Dublin, Ireland, 2008.

H. Cords. **Real-time liquid simulation**. *Proceedings of the Games Track at Eurographics '07*, Prague, Czech Republic, 2007.

Vorträge

Real-Time open water environments with interacting objects, *Eurographics Workshop on Natural Phenomena (EGWNP '09)*, Munich, Germany, 01.04.2009.

Instant liquids. *Poster at the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '08)*, Dublin, Ireland, 07.07.2008.

Moving with the flow: Wave particles in flowing liquids, *16th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG '08)*, Plzen, Czech Republic, 05.02.2008.

Mode-splitting for highly detailed, interactive liquid simulation, *5th ACM international conference on computer graphics and interactive techniques in Australia and Southeast Asia (GRAPHITE '07)*, Perth, Australia, 04.12.2007.

Refraction of water surface intersecting objects in interactive environments, *4th Workshop in Virtual Reality Interactions and Physical Simulations (VRIPHYS '07)*, Dublin, Ireland, 09.11.2007.

Thesen

1. Die physikalisch-basierte Repräsentation von Flüssigkeiten in der Computergraphik umfasst die drei Schritte der (Strömungs-) *Simulation*, *Oberflächenextraktion* und *Darstellung*. Die Erzeugung realistischer Animationen mit Hilfe dieser drei Schritte ist rechentechnisch aufwendig. Um dennoch interaktive Bildraten erzielen zu können, muss jeder einzelne der drei Schritte möglichst effizient ausgeführt werden.

2. Das Strömungsverhalten von Flüssigkeiten wird physikalisch durch die Navier-Stokes Gleichungen beschrieben. Die aufwendige, numerische Simulation dreidimensionaler Strömungen unter Berücksichtigung freier Oberflächen ist jedoch in vielen Szenarien nicht unbedingt notwendig. Oftmals ist eine zweidimensionale Repräsentation unter Verwendung von Höhenfeldern zur Darstellung ein guter Kompromiss zwischen Qualität und Performanz.

3. Die gleichzeitige Verwendung verschiedener Simulationsmethoden mit unterschiedlichen Dimensionalitäten ermöglicht eine effiziente, interaktive Repräsentation verschiedener Flüssigkeitseffekte und kann den Detailgrad, das darstellbare Volumen bzw. die Möglichkeiten gegenüber der Verwendung einer einzigen Methode vergrößern.

4. Die verschiedenen Methoden können als *physikalisch-basiert* oder *empirisch-basiert* klassifiziert werden und im Rahmen eines Stufenmodells systematisch erfasst werden. Der Schwerpunkt dieser Arbeit liegt auf den physikalisch-basierten Ansätzen. Diese können entsprechend ihren Repräsentationsmöglichkeiten und ihrem Aufwand in *Ambiente Wellen, Oberflächensimulation, 2D Strömungssimulation* und *3D Strömungssimulation* unterteilt werden.

5. Die Kopplung dieser unterschiedlichen Simulationstechniken ermöglicht die interaktive Repräsentation detaillierter und großer Flüssigkeitsflächen und zugleich dreidimensionaler Effekte in einer Anwendung. In Kombination mit einer starren Körper-Simulation können zudem bilaterale Interaktionen zwischen Objekten und der Flüssigkeit modelliert werden.

6. Die Oberflächenextraktion dreidimensionaler, dynamischer Partikelwolken kann effizient mit dem in dieser Arbeit vorgestellten Bildraum-basierten Ansatz erfolgen. Die Methode benötigt keine Vorberechnungen und kann

vollständig auf der GPU ausgeführt werden. Da keine polygonalen Oberflächen erzeugt werden, erzielt der Ansatz auch für Millionen von Partikeln interaktive Bildraten.

7. Mit Hilfe eines *Environment Mapping*-Ansatzes auf Objektbasis können typische optische Reflexions- und Brechungseffekte auch für polygonale Objekte realisiert werden, die eine Flüssigkeitsoberfläche schneiden. Die vorgestellte Methode ermöglicht interaktive Bildraten ohne die sonst üblichen Artefakte an der Grenzschicht. Zudem approximiert sie die typischen Winkelverschiebungen und Größenskalierungen der unter der Oberfläche liegenden Objektteile.

8. Brechende Wellen können als Schichtung zweidimensionaler Simulationsdaten repräsentiert werden. Mit dem ebenfalls im Rahmen dieser Arbeit präsentierten Ansatz zur Oberflächendarstellung kann die Welle mit Reflexions- und Brechungseigenschaften bei hohen Bildraten dargestellt werden.

9. Die Visualisierung der Parameter einer Partikel-basierten Flüssigkeitssimulation kann mit der vorgestellten *Partikel-basierten Beschriftungsmethode* erfolgen. Diese erleichtert die konkrete Entwicklung und interaktive Analyse von Simulationsverfahren und bietet Anwendungsmöglichkeiten in vielen weiteren Bereichen der Computergraphik.

10. Die in dieser Arbeit vorgestellten neuen Techniken zur Simulation, Oberflächenextraktion und Darstellung ermöglichen in ihrer Kombination eine physikalisch-basierte und detailgetreue Repräsentation von Flüssigkeiten in Echtzeitumgebungen.