NAIST-IS-MT0351006

修士論文

水の実時間アニメーション

天田 崇

2005年2月3日

奈良先端科学技術大学院大学
情報科学研究科 情報処理学専攻

本論文は奈良先端科学技術大学院大学情報科学研究科に
修士 (工学) 授与の要件として提出した修士論文である。

天田 崇

審査委員：　千原 國宏 教授
横矢 直和 教授
眞鍋 佳嗣 助教授

水の実時間アニメーション*

天田 崇

内容梗概

実時間での写実的な流体の表現はコンピュータゲームやバーチャルリアリティにおいてプレーヤをその世界に引き込むための重要な手段の一つである。流体は剛体に代表される他の物体とお互い影響を及ぼしあいながら運動を行うため、写実的な流体の表現に際しては剛体との相互作用を適切に取り扱わなければならない。視覚的にもっともらしい流体の振る舞いを実現するにはには、計算流体力学に基づいたシミュレーションが有効であるが、計算コストが高いためにこれまで流体の実時間アニメーションは実現されていなかった。本論文では、剛体との相互作用を含む Smoothed Particle Hydrodynamics に基づいた粒子ベースの流体シミュレーションの提案とその高速な実装、および反射・屈折・フレネル効果を伴った写実的な水面の描画方法について述べる。提案手法は実時間で剛体との相互作用を行う水のアニメーション生成を可能としている。

キーワード

コンピュータグラフィクス, 物理ベースアニメーション, 計算流体力学

* 奈良先端科学技術大学院大学 情報科学研究科 情報処理学専攻 修士論文, NAIST-IS-MT0351006, 2005 年 2 月 3 日.

Real-Time Animation of Water*

Takashi Amada

Abstract

Real-time rendering of realistic motion of fluids is one of the methods that immerse a player into an interactive application such as computer games. Interaction of fluids with rigid bodies is important because fluids and rigid bodies move influencing each other. Fluid simulation based on Computational Fluid Dynamics (CFD) is useful for rendering visually plausible behaviour of fluids. However, due to the high computational cost of CFD real-time rendering of fluids needs fast simulation. This paper shows the particle-based fluid simulation based on Smoothed Particle Hydrodynamics including interaction between fluids and rigid bodies, its fast implementation and how to render realistic water surface with optical phenomena such as reflection, refraction and Fresnel effect. The proposed method enables real-time animation of water with rigid body interaction.

Keywords:

computer graphics, physics based animation, computational fluid dynamics

* Master's Thesis, Department of Information Processing, Graduate School of Information Science, Nara Institute of Science and Technology, NAIST-IS-MT0351006, February 3, 2005.

目 次

図 目 次

表 目 次

第1章　序論

コンピュータグラフィクスの研究の大きな目的は、様々な現象を写実的に描画することによりユーザをクリエータの作成した世界に没頭させることである。写実的な画像やアニメーションの生成には物理学に基づいた手法が有効であり、多くの研究がなされている。その中で近年ハードウェアの進化とゲームでの表現の要求により、剛体等の実時間物理シミュレーションが研究され実用化にいたっている。そして日常的に接することの多い水や空気などの流体のアニメーション生成は、コンピュータグラフィクスの研究における重要な課題の一つである。

写実的な水の振る舞いの表現には計算流体力学に基づいたシミュレーションが有効であるが、計算コストが高くまたコンピュータグラフィクスの研究と目的も違うためにそのまま適用することは賢明ではない。本研究ではコンピュータグラフィクスにおいて実時間で水らしい振る舞いを表現するには、以下の条件を満たしながら実時間で処理すべきであると考える。

- 流体が凝縮しようとすると構成要素間で大きな反発力が働くこと
- 流体の構成要素が周囲の要素に対して自分と同じ運動を強いること
- 反射や屈折のような水の光学特性が再現されること
- 外力が働いたとき対応して変形すること

従来なされてきた流体アニメーションの研究では計算格子を用いた研究 [1][2] が多いが、これらの手法を用いて見た目として十分な結果を得るためには莫大な量の格子と計算を必要とするため、実時間の描画には向いていないと考えられる。一方、近年粒子を用いた手法 [3] が研究され高速な処理の可能性が示されている。また、ゲーム等のアプリケーションの要求によりグラフィクス処理専用のハード

ウェア (GPU) が急速に高速化し、また今まで固定であったレンダリングパイプラインにおいてユーザが作成したプログラムを実行可能になった。そのため、複雑な光学現象を実時間で表現することが可能となった。

本論文では前述の条件を満たした水の実時間描画を実現するために粒子法である Smoothed Particle Hydrodynamics を用いて剛体との相互作用を含む水の振る舞いをモデル化し、これを解くための高速な実装と水の光学現象を再現する描画方法を提案する。

本論文では、2 章でコンピュータグラフィクスにおける物理ベースアニメーションの現状を述べ、3 章、4 章では、それぞれ提案手法における流体シミュレーション法と描画方法、そしてその実装について論述する。5 章では提案手法を実装、実験を行い考察を行う。最後に 6 章で本研究について総括を行う。

第2章　関連研究

この章では既存のコンピュータアニメーションに関する研究について説明する。

コンピュータグラフィクスの研究においてアニメーションの生成は大きな分野の一つであり、これまで数多くの研究がなされ、様々な方法が提案されてきた。コンピュータアニメーションの研究の大きな目的はユーザにとってもっともらしいアニメーションを生成することである。そのため、物理学に基づいたシミュレーションが有効ではあるが、コンピュータアニメーションの研究においては、物理学に基づき厳密に各現象や物体の振る舞いをシミュレートすることよりも、もっともらしく見えることに重点がおかれ、研究がなされている。

2.1. 剛体アニメーション

剛体とは変形しない物体で、その運動が並進運動と回転運動で表すことができる物体である。したがって剛体の運動は運動を引き起こす剛体に働く力を計算することによってシミュレート可能であるが、その力の計算、すなわち物体が衝突した際の処理や、接触したときに発生する力、接触力の計算が剛体アニメーションにおける問題である。

Moore らは衝突検出と衝突処理の方法について提案し、物体がある条件を逸脱したとき条件から逸脱した量に応じた力を与えることにより接触力を計算した [4]。さらに長谷川らは物体が侵入した量を体積を考慮することにより、より安定なシミュレーション方法を実現した [5]。これらのシミュレーション法は 1 ステップに対して高速な処理が可能であるという特徴がある。

一方 Baraff らは抗力や静止摩擦力、関節のような物体の位置関係を拘束するような力を、方程式を用い解析的に解くことにより実現し剛体の運動を実現した。

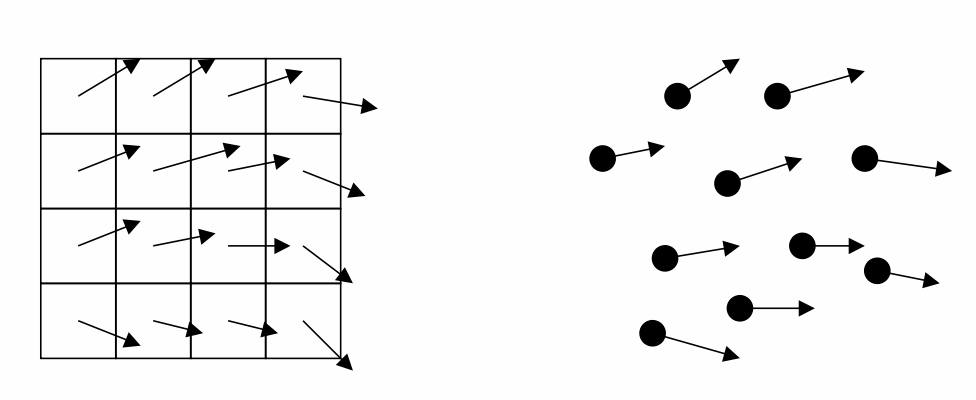

図 2.1 オイラー式記述 (左) とラグランジュ式記述 (右)

1 ステップあたりの計算量は多いものの大きなタイムステップを設定することが可能であり、計算時間が短く済むことが多い [6][7]。

Miritch らは物体の運動が変化する衝突の時刻が重要であると考え、物体が衝突する時刻を求め、衝突時間が一瞬であると仮定し、その時刻で衝突を処理することにより二つの物体に働く力を表現している [8]。さらに Mirtich は多数の物体をシミュレートしたときに、衝突が起こった際、全ての物体をその衝突時刻に同期しなければならなかったが、衝突の影響がない物体の処理を省略することにより効率的にシミュレーションを行う方法を提案している [9]。そして Eran らは物体が複数積み重なったときや摩擦などの物体同士が接触しているときの処理を、処理の順番を考慮することにより改善し、大量の剛体の積み重なりを可能にしている [10]。

2.2. 流体アニメーション

コンピュータアニメーションの研究では剛体に限らずゴムのような弾性体や、ガス状の気体や水などの流体のアニメーションについても様々な手法が提案されている。Fournier ら [11] や Hinsinger ら [12] は海面の変位に注目し、波動方程式を解くことにより海の波のアニメーション生成方法を提案した。Kass らは高さ

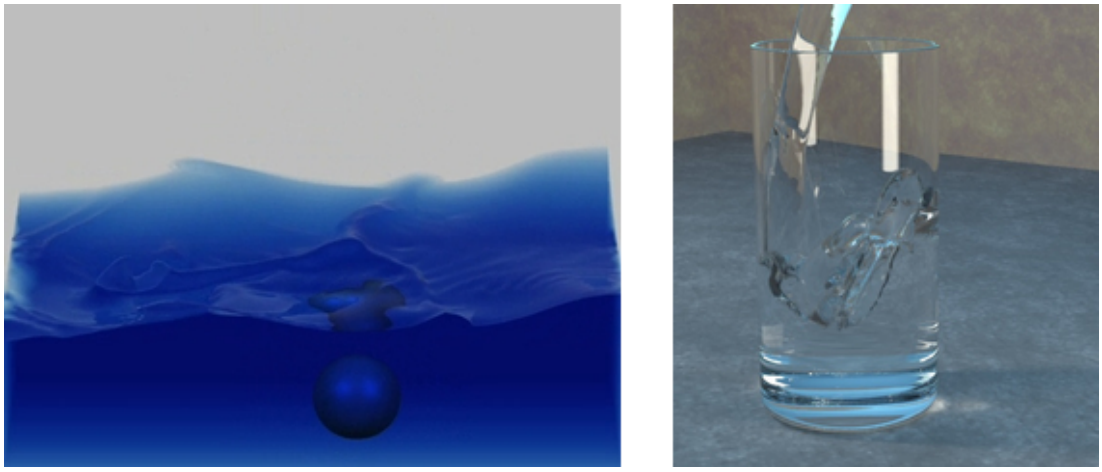

図 2.2 Practical Animation of Water (SIGGRAPH 2002)[1](左)、Animation and Rendering Complex Water Surfaces (SIGGRAPH 2003)[2](右)

フィールドを用いて水面のアニメーションを示した [13]。O'Brien らは高さフィールドと粒子を用い、物体が水面に衝突したときのしぶきと波を表現する手法を提案した [14]。

流体の運動をシミュレートするとき、空間上のある場所に注目し、そこを流れる流体がどのような振る舞いをするかを記述するオイラー式記述 (Eulerian description) と、流体の構成要素に注目し時間とともにどのように振舞うかを記述するラグランジュ式記述 (Lagrangian description) の二つの記述方法が用いられる (図 2.1)。

格子を用いたオイラー式記述の流体シミュレーションは、コンピュータアニメーションでは一般的な方法で、様々な手法が提案されている。

Foster らは Marker-And-Cell(MAC) 法 [15] を用いて水のアニメーションを生成している [16]。また、MAC 法を用いた気体のアニメーションが Foster らによって提案されている [17]。MAC 法は計算流体力学の手法の一つで、流体の存在し得る空間を格子で分割し、それぞれの格子に対して速度場を計算する。格子に配置されたマーカーを速度場に従い移動させることにより、流れの解析を行う。Stam は陰解法と流体要素の流れに Semi-Lagrangian 法を用いてタイムステップによらず、

図 2.3 Rigid Fluid: Animating the Interplay Between Rigid Bodies and Fluid (SIGGRAPH 2004)[20]

安定でインタラクティブな流体シミュレーションを行う方法を提案している [18]。この手法は 2 次元の処理に関してインタラクティブに動作する。Song らは Stam の方法を水に適用した手法を提案している [19]。そして Foster らは MAC 法を用いてより滑らかな水面を表現するために Level Set 法を導入 [1] し、さらに Enright らは Particle Level Set 法を導入、空気の粒子も考慮することによりより良い水面の生成を提案した [2]。(図 2.2) Carson らは剛体と流体の相互作用をシミュレートする方法を提案している。[20] この手法では剛体を流体として扱って処理を行い、その後剛体内の速度場を剛体運動の制約条件を満たすように修正することにより、剛体と流体との相互作用を実現している (図 2.3)。高橋らは Cubic Interpolated Propagation(CIP) 法 [21] を移流方程式の計算に用い、Volume of Fluid(VOF) 法 [22] を用いて水面の追跡を行い、粒子を用いて泡やしぶきを表す方法を提案している [23]。

これらの手法は良好なアニメーションが得られているものの格子を用いているため境界条件の設定が難しく動的に変化するようなシーンでの描画には向いていない。また格子を用いた方法は粒子法と比べて、見た目が十分な結果を得るには

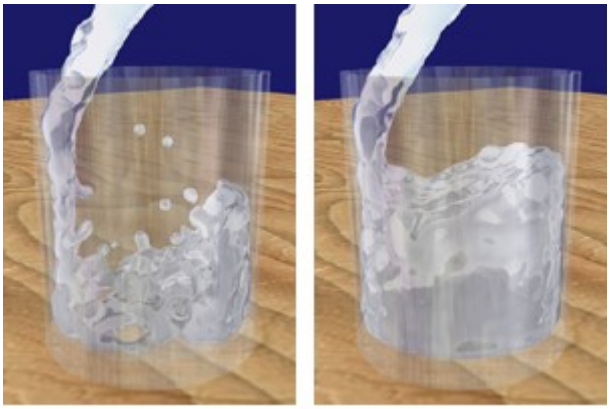

図 2.4　Particle-Based Fluid Simulation for Interactive Application (SIGGRAPH/Eurographics Symposium on Computer Animation 2003)[3]

莫大な数の格子を必要となるため計算量が増え、実時間の描画には適していない。

一方で格子を用いたオイラー式記述ではなく、粒子を用いたラグランジュ式記述の流体シミュレーションを用いた研究もなされている。コンピュータグラフィクスにおいては、流体シミュレーションに粒子を用いる以前から、粒子を用いた表現手法が提案されている。Reeves らは粒子を用いてぼやけたものを表現する方法を提案している [24]。この研究から発展して様々なものが粒子を用いて表現されている。Miller らは粒子を使った粘性流体のアニメーションを提案している [25]。この手法では粒子同士がある程度離れた状態では弱く引き合い、近づくと強く反発するような力が働く粒子間相互作用を用いることによって擬似的に粘性流体を表現している。最近では Graphics Processing Unit(GPU) を用いて大量の粒子を高速に扱う方法が Kipfer によって提案されている [26]。Premoze らは越塚らによる Moving Particle Semi-implicit method (MPS) 法 [27] に基づいた非圧縮流体のアニメーションを提案している [28]。MPS 法は粒子法による非圧縮流体のシミュレーションで、圧力のポアソン方程式を解くことにより非圧縮性を表現している。

Smoothed Particle Hydrodynamics(SPH) は粒子を用いたラグランジュ記述による流体シミュレーションで、元々天体物理学の問題を問うために Monaghan ら

によって導入された [29] が、汎用的な手法であるため様々な流体シミュレーションに用いられている。Stam らは炎などのガス状の流体を SPH を用いて表現している [30]。Desbrun らは SPH を用いた弾性体アニメーションを提案している [31]。Stora らは溶岩の粘性と温度を考慮し溶岩の流れを SPH を使い解いている [32]。また、SPH を用いた高速な非圧縮流体のシミュレーション [3] や外科手術訓練用シミュレータのための血液のシミュレーション [33] が Müller らによって提案されている。

本論文では剛体を粒子で表現し、SPH の枠組みに取り込むことで剛体と流体との相互作用を実現する手法と、実時間のアニメーションを行うための流体シミュレーションの高速な実装を提案する。

第3章　シミュレーション

提案手法で扱う水を含む流体は原子や分子から構成されているが、数 cm^3 程度の空間に莫大な数の原子や分子が存在しているので、巨視的な視点から流体の物理量、つまり質量や運動量等をみたときに、物理量が連続的に分布している連続体として見なすことができる。

提案手法では物理学に基づき剛体との相互作用を伴った流体の振る舞いをシミュレートする。物理学に基づいて剛体との相互作用を伴った水の振る舞いを計算機シミュレートするには、連続体である流体の離散化、つまり体の振る舞いを記述する Navier-Stokes 方程式を離散化しなければならない。また、その流体と相互作用を行う剛体のモデルも必要である。本章では SPH を用いた Navier-Stokes 方程式の離散化と流体と剛体との相互作用、シミュレーションの実装方法について述べる。

3.1. Smoothed Particle Hydrodynamics

Smoothed Particle Hydrodynamics(SPH) は計算流体力学の手法の一つでラグランジュ式記述で粒子を用いて流体の運動を記述する (図 3.1)[29]。

SPH では点 $\mathbf{r}$ における関数 $A(\mathbf{r})$ の補間値は以下の式で与えられる。

$$A(\mathbf{r}) \quad = \quad \int A(\mathbf{r}')W(\mathbf{r}-\mathbf{r}', h)d\mathbf{r}' \tag{3.1}$$

積分は全空間に対してなされる。関数 W は粒子の間の空間で物理量を補間するためのカーネルと呼ばれる関数で、カーネルの有効半径は h である。カーネルには一般的に距離 $|\mathbf{r}-\mathbf{r}'|$ が大きくなるにつれて減少する関数が用いられる。

図 3.1 流体 (左) とその流体要素を表す粒子の分布 (右)

カーネルには以下の二つの特徴がある。

$$\int W(\mathbf{r}-\mathbf{r}',h)dr' = 1 \tag{3.2}$$

$$\lim_{h\to 0} W(\mathbf{r}-\mathbf{r}',h) = \delta(\mathbf{r}-\mathbf{r}') \tag{3.3}$$

実際に数値計算を行う場合は次の離散化された式を用いる。

$$A(\mathbf{r}) = \sum_j m_j \frac{A_j}{\rho_j} W(\mathbf{r}-\mathbf{r}_j,h) \tag{3.4}$$

ここで ρ、m、はそれぞれ流体粒子の密度、質量である。したがって空間上の点 $\mathbf{r}$ における密度は以下の式で表される。

$$\begin{aligned}\rho(\mathbf{r}) &= \sum_j m_j \frac{\rho_j}{\rho_j} W(\mathbf{r}-\mathbf{r}_j,h) \\ &= \sum_j m_j W(\mathbf{r}-\mathbf{r}_j,h)\end{aligned} \tag{3.5}$$

密度はシミュレーションのステップごとに粒子の位置でのみ更新され、粒子間相互作用の計算に用いられる。

カーネルの有効半径が h であるため半径 h 内にある近傍粒子のみを用いて補間を行うことにより補間の際の計算量を減らすことができる。そのため高速に計算

を行うには、それぞれの粒子に対してカーネルの有効範囲内に存在する近傍粒子を探索する必要がある。近傍粒子の探索については 3.4.1 で述べる。

SPH は、物理量の勾配やラプラシアンをカーネルの勾配やラプラシアンと物理量の積との積分によって表せるという特徴がある。一次元で式 (3.1) を表現すると

$$A(r) = \int_{-\infty}^{+\infty} A(r')W(r-r',h)dr' \tag{3.6}$$

となる。$A(r)$ の勾配は各点での勾配を補間すればよいので、

$$\frac{\partial A}{\partial x}(r) = \int_{-\infty}^{+\infty} \frac{\partial A}{\partial x}(r')W(r-r',h)dr' \tag{3.7}$$

になり、部分積分は

$$\begin{aligned}\int_{-\infty}^{+\infty} \frac{\partial A}{\partial x}(r')W(r-r',h)dr' &= [A(r')W(r-r',h)]_{-\infty}^{+\infty} \\ &\quad - \int_{-\infty}^{+\infty} A(r')\cdot(-1)\cdot\frac{\partial W}{\partial x}(r-r',h)dr' \end{aligned} \tag{3.8}$$

になるので、右辺の第一項は消え

$$\frac{\partial A}{\partial x}(r) = \int_{-\infty}^{+\infty} A(r')\frac{\partial W}{\partial x}(r-r',h)dr' \tag{3.9}$$

になり、物理量 A の勾配はカーネルの勾配で表すことができる。同じようにラプラシアンも導出できる。したがって勾配は

$$\nabla A(\mathbf{r}) = \sum_j m_j \frac{A_j}{\rho_j}\nabla W(\mathbf{r}-\mathbf{r}_j,h) \tag{3.10}$$

と表され、ラプラシアンは

$$\nabla^2 A(\mathbf{r}) = \sum_j m_j \frac{A_j}{\rho_j}\nabla^2 W(\mathbf{r}-\mathbf{r}_j,h) \tag{3.11}$$

と表される。

3.2. SPH による流体シミュレーション

SPH を用いた支配方程式の離散化には Müller らの方法 [3] を用いる。非圧縮流体の振る舞いは質量保存則を表す連続の式

$$\nabla\cdot\mathbf{v} = 0 \tag{3.12}$$

と非圧縮流体の振る舞いを記述する Navier-Stokes 方程式

$$\rho\frac{D\mathbf{v}}{Dt} = -\nabla p + \mu\nabla^2\mathbf{v} + \mathbf{f} \tag{3.13}$$

で表される。ρ、$\mathbf{v}$、p、μ、$\mathbf{f}$ はそれぞれ密度、速度ベクトル、圧力、粘性係数、外力である。

連続の式 (3.12) は、シミュレーションで用いる粒子の質量が不変であるため満たされる。また Navier-Stokes 方程式 (3.13) のラグランジュ微分 $\frac{D\mathbf{v}}{Dt}$ は流体要素の速度変化を表すため、単純に速度の時間微分となる。

右辺の圧力項 $-\nabla p$、粘性項 $\mu\nabla^2\mathbf{v}$、外力項 $\mathbf{f}$ それぞれを SPH によって離散化する。式 (3.13) の左辺から粒子の加速度は

$$\mathbf{a}_i = \frac{d\mathbf{v}_i}{dt} = \frac{\mathbf{f}_i}{\rho_i} \tag{3.14}$$

と表される。

3.2.1 圧力

圧力項 $-\nabla p$ は SPH を用いると次のように表される。

$$\mathbf{f}_i^{\text{pressure}} = -\nabla p = -\sum_j m_j \frac{p_j}{\rho_j}\nabla W(\mathbf{r}_i - \mathbf{r}_j, h) \tag{3.15}$$

しかし大抵の場合圧力 p は場所によって異なるため、この式 (3.15) では力が対称ではなく、作用反作用の法則を満たさない。そこで以下のようにして力を対称化にする。

$$\mathbf{f}_i^{\text{pressure}} = -\sum_j m_j \frac{p_j + p_i}{2\rho_j}\nabla W(\mathbf{r}_i - \mathbf{r}_j, h) \tag{3.16}$$

他にも力を対称化する方法が存在する [29] が、安定性や計算コストの観点から式 (3.16) を採用する [3]。

圧力項を計算する前に圧力 p が必要であるが、状態方程式 ($p = k\rho$) を踏まえて、Desbrun ら [31] が提案したように圧力は

$$p = k^{\text{fluid}}(\rho - \rho_0^{\text{fluid}}) \tag{3.17}$$

と計算される。k^{fluid}、ρ_0^{fluid} はそれぞれ流体粒子壱での圧力の大きさを調整するパラメータと流体の密度である。式 (3.17) のように圧力が計算されることにより、流体の密度が ρ_0^{fluid} になるように流体粒子に力が働くことになる。

3.2.2 粘性

粘性項は

$$\mathbf{f}_i^{\text{viscosity}} = \mu\nabla^2\mathbf{v} = \mu\sum_j m_j\frac{\mathbf{v}_j}{\rho_j}\nabla^2 W(|\mathbf{r}_i-\mathbf{r}_j|,h) \tag{3.18}$$

となる。粒子ごとに速度が違うためこの式 (3.18) からは対称な力を得られない。粘性は流体要素間の速度の差によって生じると考えられるので、以下のように対称化する。

$$\mathbf{f}_i^{\text{viscosity}} = \mu\sum_j m_j\frac{\mathbf{v}_j-\mathbf{v}_i}{\rho_j}\nabla^2 W(|\mathbf{r}_i-\mathbf{r}_j|,h) \tag{3.19}$$

3.2.3 表面張力

Navier-Stokes 方程式 (式 3.13) には表面張力に関する明示的な記述はない。Müller ら [3] は表面張力を Morris の提案手法 [34] を用いて表現した。流体が存在している場所を 1、存在しない場所を 0 とすると

$$c_s(\mathbf{r}) = \sum_j m_j\frac{1}{\rho_j}W(\mathbf{r}-\mathbf{r}_j,h) \tag{3.20}$$

とカラーフィールドが定義でき、ある場所の水面の法線方向は

$$\mathbf{n} = \nabla c_s \tag{3.21}$$

となる。また、曲率は

$$\kappa = \frac{-\nabla^2 c_s}{|\mathbf{n}|} \tag{3.22}$$

となる。表面張力は曲率に比例し、法線逆方向に働く力である。したがって、

$$\mathbf{f}^{\text{surface}} = \sigma\kappa\mathbf{n} = -\sigma\nabla^2 c_s\frac{\mathbf{n}}{|\mathbf{n}|} \tag{3.23}$$

となる。σ は表面張力の強さを調整する係数である。

3.2.4 式の対称性

密度の式 (3.5) における、ある粒子から別の粒子への寄与は

$$W(\mathbf{r}_i - \mathbf{r}_j, h) \quad = \quad W(\mathbf{r}_j - \mathbf{r}_i, h) \tag{3.24}$$

で、また、圧力の式 (3.16) は加速度を考慮すると

$$\frac{p_j + p_i}{2\rho_j\rho_i}\nabla W(\mathbf{r}_i - \mathbf{r}_j, h) \quad = \quad -\frac{p_i + p_j}{2\rho_i\rho_j}\nabla W(\mathbf{r}_j - \mathbf{r}_i, h) \tag{3.25}$$

で、また粘性の式 (3.19) は

$$\mu\frac{\mathbf{v_j} - \mathbf{v_i}}{\rho_j\rho_i}\nabla^2 W(|\mathbf{r_i} - \mathbf{r_j}|, h) \quad = \quad -\mu\frac{\mathbf{v_i} - \mathbf{v_j}}{\rho_i\rho_j}\nabla^2 W(|\mathbf{r_j} - \mathbf{r_i}|, h) \tag{3.26}$$

であるので、ある粒子から別の粒子への寄与は、逆の寄与と等しく対称なので、和の計算を行う前にどちらか一方への寄与を計算し、計算された結果を二つの粒子の和の計算で利用することにより、対称性を考慮しない場合の約半分の計算時間で計算可能である。

3.2.5 外力

重力や流体以外の物体による力は流体粒子を単なる質点として扱いその運動を計算し、コップなどの運動しない物体による抗力は、ペナルティ法 [4] を用いて計算する。ペナルティ法は二つの物体間の接触力の計算方法で、二つの物体の距離を d、バネ係数を k^s、ダンパ定数を k^d、物体の法線方向を $\mathbf{n}$、物体の相対速度を $\mathbf{v}$ とすると抗力 $\mathbf{F}^{\mathrm{col}}$ は

$$\mathbf{F}^{\mathrm{col}} \quad = \quad k^s d\mathbf{n} + k^d(\mathbf{v}\cdot\mathbf{n})\mathbf{n} \tag{3.27}$$

と計算される。

3.2.6 カーネル

本研究では [3] で提案されたカーネルを用いる。カーネルは補間に用いられるため、安定性と計算量の少なさが求められる。そのため、以下に挙げたカーネル

は全てカーネルの有効範囲内と外との境界でカーネルの値とその傾きが 0 である。以下にそのカーネルと実際に使用する勾配やラプラシアンを記述する。カーネルの係数の導出は付録 A に記載した。また $r = |\mathbf{r}|$ である。

$$W_{\text{poly6}}(\mathbf{r}, h) = \frac{315}{64\pi h^9} \begin{cases} (h^2 - r^2)^3 & 0 \le r \le h \\ 0 & otherwise \end{cases} \tag{3.28}$$

Poly6 カーネルは密度の計算 (式 3.5) や表面張力の計算に用いられる。このカーネルは距離の自乗を用いるため、計算時間のかかる平方根の計算が必要ない。

$$W_{\text{spiky}}(\mathbf{r}, h) = \frac{15}{\pi h^6} \begin{cases} (h - r)^3 & 0 \le r \le h \\ 0 & otherwise \end{cases} \tag{3.29}$$

$$\nabla W_{\text{spiky}}(\mathbf{r}, h) = \frac{45}{\pi h^6} \begin{cases} (h - r)^2 / r \cdot \mathbf{r} & 0 \le r \le h \\ 0 & otherwise \end{cases} \tag{3.30}$$

Spiky カーネルは圧力の計算 (式 (3.16)) に用いられる。このカーネルの勾配は粒子間の距離が小さくなると値が急激に大きくなり、粒子が極端に近づかないように働く。

$$W_{\text{viscosity}}(\mathbf{r}, h) = \frac{15}{2\pi h^3} \begin{cases} -\frac{r^3}{2h^3} + \frac{r^2}{h^2} + \frac{h}{2r} - 1 & 0 \le r \le h \\ 0 & otherwise \end{cases} \tag{3.31}$$

$$\nabla^2 W_{\text{viscosity}}(\mathbf{r}, h) = \frac{45}{\pi h^6} \begin{cases} (h - r) & 0 \le r \le h \\ 0 & otherwise \end{cases} \tag{3.32}$$

Viscosity カーネルは粘性による力の計算 (式 (3.19)) に用いられる。

カーネルの有効範囲 h は定数なのでそれぞれのカーネルの係数は先に計算しておく。

3.3. 剛体の扱い

剛体は変形しないため構成要素の相対位置が変化せず、その運動が平行移動と重心周りの回転に制限された一種の流体だと考えることができる。そこで剛体を流体と同じく粒子の集合として表し、剛体粒子に与えられた力を並進運動と重心周りの回転運動に変換することにより、剛体運動を表現する [27]。剛体の質量 M は剛体を構成する剛体粒子の質量を m_j とすると

$$M = \sum_j m_j \tag{3.33}$$

となる。剛体における圧力は

$$p = \begin{cases} k^{\text{rigid}}(\rho - \rho_0^{\text{rigid}}) & \rho \geq \rho_0^{\text{rigid}} \\ 0 & otherwise \end{cases} \tag{3.34}$$

で与えられるとする。ここで、ρ_0^{rigid} は剛体の密度、k^{rigid} は剛体粒子と流体粒子間の圧力による反発力を調整するパラメータである。また、流体粒子から剛体粒子への圧力は式 (3.17) ではなく

$$p = \begin{cases} k^{\text{fluid}}(\rho - \rho_0^{\text{fluid}}) & \rho \geq \rho_0^{\text{fluid}} \\ 0 & otherwise \end{cases} \tag{3.35}$$

と計算する。式 (3.17) に似ているが、圧力が負の値になると剛体内に流体粒子が侵入してしまうため、式 (3.35) を用いて負の値にならないように制限する。式 (3.34) により、密度が流体より小さい剛体が流体内に存在すると、剛体粒子が大きな反発力を流体粒子から受けることになる。したがって密度が ρ_0^{rigid} より小さい剛体は浮くことになり、逆に密度が流体より大きい剛体は沈むことになる。

3.3.1 剛体の運動

3.2 節に従い、剛体粒子を流体粒子と同じように扱い、剛体粒子位置での密度、加速度を計算する。そして得られた粒子の加速度を用いて、剛体粒子の速度を更新する。

剛体を構成する剛体粒子 i の速度ベクトルを $\mathbf{v}_i$ とすると、重心の並進速度ベクトル $\mathbf{v}_g$ は、剛体を構成する粒子の数を N とすると

$$\mathbf{v}_g = \frac{1}{N}\sum_j \mathbf{v}_j \tag{3.36}$$

である。角速度 $\boldsymbol{\omega}$ は

$$\boldsymbol{\omega} = \frac{1}{I}\sum_j \mathbf{q}_j \times \mathbf{v}_j \tag{3.37}$$

となる。ここで $\mathbf{q}_j$ は重心 $\mathbf{r}_g$ から剛体粒子への変位ベクトルで、次のように表される。

$$\mathbf{r}_g = \frac{1}{N}\sum_j \mathbf{r}_j \tag{3.38}$$

$$\mathbf{q}_i = \mathbf{r}_i - \mathbf{r}_g \tag{3.39}$$

また、I は慣性モーメントを質量で割った値で、

$$I = \sum_j |\mathbf{q}_j|^2 \tag{3.40}$$

である。

以上の角速度 $\boldsymbol{\omega}$ と並進速度 $\mathbf{v}_g$ を用いて、剛体の運動としての制約が課せられた粒子の速度 $\mathbf{v}_i$ が以下の式で計算される。

$$\mathbf{v}_i = \mathbf{v}_g + \boldsymbol{\omega} \times \mathbf{q}_i \tag{3.41}$$

3.4. 実装

SPH を用いた流体シミュレーションの実装方法について述べる。

提案手法の処理の概要は以下の擬似コードで表される。

```
// 位置や速度などの粒子の初期状態の設定
// 剛体粒子の近傍粒子の探索
initialize
```

```
while simulating {
  // 全ての流体粒子に対してその近傍粒子を探索
  search_neighbours_of_fluid

  // 全ての粒子に対してその密度を計算
  for each particle i {
    for each neighbour particle j in neighbour list of i {
      compute_density(i, j)
    }
  }

  // 全ての粒子に対して圧力、粘性、表面張力による力を計算
  for each particle i {
    for each neighbour particle j in neighbour list of i {
      compute_force(i, j)
    }
  }

  // 全ての流体粒子の位置と速度を更新
  for each fluid particle i {
    update_position_and_velocity(i)
  }

  // 全ての剛体に対して
  for each rigid body r {

    // 剛体を構成する粒子に働く力から剛体の加速度と角加速度を計算
    compute_rigidity(r)
    for each rigid particle i in r {

      // 剛体の運動を剛体粒子に適用し、剛体粒子の位置と速度を更新
      update_position_and_velocity(i)
    }
  }
}
```

3.4.1 近傍粒子の探索

SPH では粒子の物理量とカーネルとの積の和により物理量の補間を行う。カーネルの有効半径を決めることにより、物理量の補間はカーネルの有効半径内にある粒子のみで行えばよいことが保証される。したがって高速に計算を行うために

はカーネルの有効範囲内にある粒子、つまり近傍粒子を探索する必要がある。提案手法では次のようにして近傍粒子の探索を行う [35]。

空間を一辺の長さが (カーネルの有効範囲以上) 全て等しい大きさの格子に空間を分割し、粒子を対応する格子に割り当てる。ある粒子の近傍粒子は、その粒子が割り当てられた格子と近傍粒子を含む 27 つの格子内の粒子同士との距離を、カーネルの有効半径と比較することにより得られる。

シミュレーションに用いられる式 (3.5)(3.16)(3.19)(3.23) は対称なので粒子のペアが求められればよい。そこで全ての粒子に対して、1) 近傍粒子の探索、2) 粒子の格子への割り当て、という順で処理を行うことで最小限の比較回数で近傍粒子を探索できる。ただし、剛体粒子間の位置関係は変わらないので先に剛体粒子間の近傍リストを作成しておき、近傍粒子の探索の際に先に全ての剛体粒子を格子に格納しておく。

以下に近傍粒子の探索アルゴリズムの擬似コードを記述する。

```
// 先に全て剛体粒子を格子に格納する
for each rigid particle i {
  allocate_to_grid(i)
}

for each fluid particle i {
  for each neighbour grid g {
    for each particle j in g {
      if h > distance(i, j)
        add_pair_to_list(i, j)
    }
  }
  allocate_to_grid(i)
}
```

近傍粒子の探索において計算された粒子間の距離は、以後のカーネルの計算で再利用することにより、シミュレーション全体の計算時間を短縮できる。また、現在の時刻と次の時刻での粒子の分布がほとんど変化しないことを利用し、近傍粒子の探索をスキップすることにより計算量を減らすことができる。

3.4.2 SPH による補間の計算

3.2.4 で述べたようにある粒子から別の粒子への寄与は対称なので、寄与を計算後それぞれの粒子の物理量に寄与を加算する。また、カーネルの係数はカーネルの有効範囲 h が定数なので先に計算しておいた値を用いることにより、計算時間を短縮できる。

3.4.3 流体粒子の位置、速度の更新

先の処理で得られた加速度を用いて次の時刻での剛体、流体のそれぞれの粒子に位置と速度を計算し、更新する。次の時刻の位置、速度の計算には Leap-Frog(付録 B) を用いる。Leap-Frog は $\mathbf{r}_i(t)$、$\mathbf{v}_i(t)$、$\mathbf{a}_i(t)$、Δt をそれぞれ時刻 t での粒子 i の位置、速度、加速度ベクトル、タイムステップとすると、

$$\mathbf{r}_i(t+\Delta t) = \mathbf{r}_i(t) + \Delta t \mathbf{v}_i(t+\frac{1}{2}\Delta t) \tag{3.42}$$

$$\mathbf{v}_i(t+\frac{1}{2}\Delta t) = \mathbf{v}_i(t-\frac{1}{2}\Delta t) + \Delta t \mathbf{a}_i(t) \tag{3.43}$$

となる。

また、粘性の計算 (3.2.2) では時刻 t での速度が用いられる。時刻 t での速度は

$$\mathbf{v}_i(t) = \frac{1}{2}\left\{\mathbf{v}_i(t-\frac{1}{2}\Delta t) + \mathbf{v}_i(t+\frac{1}{2}\Delta t)\right\} \tag{3.44}$$

で与えられる。

3.4.4 剛体粒子の位置、速度の更新

式 (3.41) をそのまま適用すると、回転運動を示す $\boldsymbol{\omega} \times \mathbf{q}_i$ は回転軸に対して垂直な方向なので、角速度 $\boldsymbol{\omega}$ が大きい場合、重心から一定半径上にあるべき粒子の位置に誤差が生じ、各剛体粒子間の位置関係が破綻してしまう。そこで、剛体は時刻 t での位置姿勢を表す回転行列 $\mathbf{R}(t)$ と重心位置 $\mathbf{r}_g(t)$ を変数として保持し、まずそれらの変数を以下のように更新する。

$$\mathbf{r}_g(t+\Delta t) = \mathbf{r}_g(t) + \Delta t \mathbf{v}_g(t+\frac{1}{2}\Delta t) \tag{3.45}$$

$$\mathbf{R}(t+\Delta t) = \mathbf{Q}(t+\frac{1}{2}\Delta t)\mathbf{R}(t) \tag{3.46}$$

ここで $\mathbf{Q}(t)$ は時刻 t での重心周りの回転を表す行列である。これらを用いて、次の時刻 $t + \Delta t$ での各剛体粒子 i の位置 $\mathbf{r}_i$ は次のように計算される。

$$\mathbf{r}_i(t + \Delta t) \quad = \quad \mathbf{R}(t + \Delta t)(\mathbf{r}_i(\mathrm{o}) - \mathbf{r}_g(\mathrm{o})) + \mathbf{r}_g(t + \Delta t) \tag{3.47}$$

第4章　描画

本章ではシミュレーションで得られた粒子の分布から水を描画する方法について述べる。概要は以下の通りである。1) 粒子の分布から陰曲面を生成し、2) 陰曲面を多角形の集合に変換し、3) 描画した三角形に対しシェーダを用いて光学現象を再現する。詳しくは以後に述べる。

4.1. 水面の構築

粒子分布から陰曲面を生成し、Marching Cubes[36] を用いた陰曲面のポリゴン化により水面を生成する。陰曲面はある関数 $\phi(\mathbf{r})$ が与えられたとき

$$\phi(\mathbf{r}) \quad = \quad 0 \tag{4.1}$$

を満たす点 $\mathbf{r}$ の全体からなる曲面で、陰曲面の法線ベクトル $\mathbf{n}$ は

$$\mathbf{n} \quad = \quad \nabla\phi(\mathbf{r}) \tag{4.2}$$

で与えられる。例えば原点を中心とした半径 R の球面は次の関数 $\phi(\mathbf{r})$ で表される。

$$\phi(\mathbf{r}) \quad = \quad |\mathbf{r}| - R \tag{4.3}$$

提案手法では $\phi(\mathbf{r})$ として流体の密度を表す式 3.5 から以下の関数を用いた。

$$\phi(\mathbf{r}) \quad = \quad \sum_j m_j W_{\mathrm{poly6}}(\mathbf{r} - \mathbf{r}_j, h) - \rho^{\mathrm{iso}} \tag{4.4}$$

ρ^{iso} は陰曲面生成のためのしきい値である。

4.1.1 Marching Cubes

Marching Cubes [36] [37] は陰曲面やボリュームデータを、グラフィクスハードウェアが扱える多角形の集合で近似するアルゴリズムである。

Marching Cubes は空間を複数のセル (立方体) に分割し、セルのそれぞれの辺と陰曲面との交点を求める。セルと陰曲面が交わっていれば、交点が存在するセルの面の辺を陰曲面の内側に属する頂点の方向に探索し、最初に見つかった交点とを結んだ線分が陰曲面を近似する多角形を構成する辺とし、順に交点を探索、それらの辺を用いて多角形を生成する。

実践的には、セルの各頂点が陰曲面の中であるかどうかが決まると各交点が他のどの交点と結ばれて多角形の辺を構成するかがわかるので、その情報をテーブルとして保存し、テーブルを参照して多角形を生成する。立方体の頂点の数は 8 つなのでテーブルには全部で $2^8 = 256$ のエントリが存在する。

他の陰曲面を多角形の集合で変換するアルゴリズムとして、セルを 5 つないし 6 つの四面体に分割しその四面体ごとに多角形を生成する方法がある [37]。

4.2. 光学現象

異なる二つの媒質の境界面に光が達すると図 4.1 のように光の反射と屈折が起きる。反射角は入射角と等しく、屈折角はスネルの法則により決定される。

スネルの法則は入射角を θ_i、屈折角を θ_t、二つの媒質の屈折率を η_1、η_2 とすると次の式で表される。

$$\eta_1 \sin\theta_i \quad = \quad \eta_2 \sin\theta_t \tag{4.5}$$

反射光の方向を表すベクトル $\mathbf{R}$ は法線ベクトルを $\mathbf{N}$、入射光ベクトルを $\mathbf{I}$ とすると以下の式で表される。

$$\mathbf{R} \quad = \quad \mathbf{I} - 2(\mathbf{N}\cdot\mathbf{I})\mathbf{N} \tag{4.6}$$

また屈折光の方向を表すベクトル $\mathbf{T}$ は次の式で表される。

$$\mathbf{T} \quad = \quad -\mathbf{N}\cos\theta_t - \frac{\sin\theta_t}{\sin\theta_i}\{(\mathbf{N}\cdot\mathbf{I})\mathbf{N} - \mathbf{I})\} \tag{4.7}$$

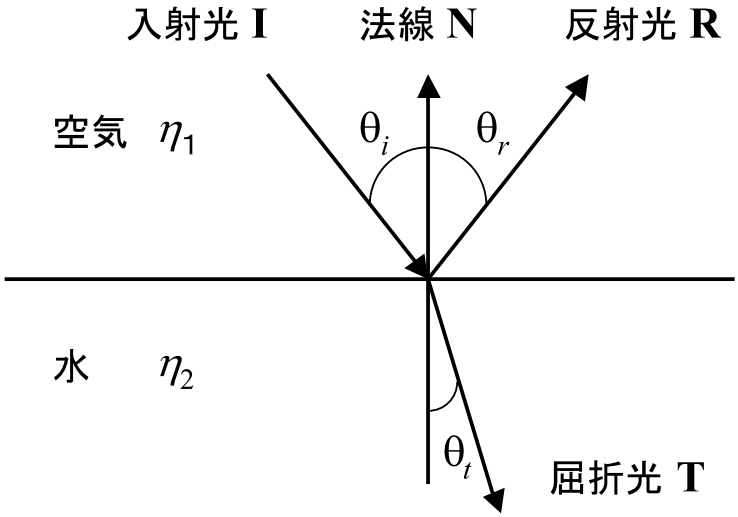

図 4.1 水面における光の反射と屈折

また、実生活において水を水面に対して垂直な方向から見たときは水の中が良く見え、ほぼ水平な方向から見たときは鏡のように周囲のものを反射して見えるように、見る方向、つまり光の入射角に依存して反射光と屈折光の強さが変化する、フレネル効果が起こる。フレネル効果は、入射角が θ_i のときの反射光の強度を $R(\theta_i)$、屈折光の強度を $T(\theta_i)$ とすると以下の式で表される。

$$R(\theta_i) = \frac{1}{2}\left\{\frac{\sin^2(\theta_i-\theta_t)}{\sin^2(\theta_i+\theta_t)} + \frac{\tan^2(\theta_i-\theta_t)}{\tan^2(\theta_i+\theta_t)}\right\} \tag{4.8}$$

$$T(\theta_i) = 1 - R(\theta_i) \tag{4.9}$$

ただし $R(0)$ は以下の式で与えられる。

$$R(0) = \frac{(\eta_1-\eta_2)^2}{(\eta_1+\eta_2)^2} \tag{4.10}$$

しかし、この式は複雑で計算時間がかかり実時間の描画には適さないので、以下のように近似する [38][39]。

$$R(\theta_i) = R(0) + (1-R(0))(1-\cos(\theta_i))^5 \tag{4.11}$$

4.2.1 実装

前節で記述した水面で起こる光学現象を厳密に再現するには光線追跡法 [40] が最適であると考えられるが、実時間での処理は困難である。提案手法では、ラスタライズ処理を行う GPU を有効に活用するために描画される面でのみ反射、屈折が起こるとする。

反射

反射光の参照先としてキューブマップテクスチャによる環境マップを用いて実装する。環境マップはシーン中のオブジェクトとそれを取り囲む環境が十分に遠いと仮定し、環境を表すテクスチャを用いて反射等を表す手法である [41]。キューブマップテクスチャは、3 次元の方向ベクトルでテクセルを参照できるテクスチャで、オブジェクトを取り囲む環境マップが立方体からなるとし、立方体の 6 つの面それぞれに環境マップを表す正方形のテクスチャを持つ [42](図 4.2)。キューブマップテクスチャを構成する各面はそれぞれ座標軸に垂直で、原点から見た座標軸の方向 (+X、-X、+Y、-Y、+Z、-Z) に対応している。

方向ベクトルが与えられたとき、参照するテクセルは以下のように決定される。

1. 方向ベクトルの成分で、絶対値が最も大きな座標成分からキューブマップテクスチャのどの面を使うかを決定する (例えばベクトル $(3, 2, -9)$ が与えられたとき-Z 面が用いられる)

2. 最も大きな座標成分を m、残りの座標成分を u、v とすると、使われる面のテクスチャ座標 (s, t) は

$$s = \frac{1}{2}\left(\frac{u}{|m|} + 1\right) \tag{4.12}$$

$$t = \frac{1}{2}\left(\frac{v}{|m|} + 1\right) \tag{4.13}$$

で与えられる。

図 4.2 キューブマップテクスチャ(Peter Murphy 提供)

屈折

先に述べたように、屈折を厳密に表現するには、光線追跡法が最適であると考えられるが、実時間の描画は難しいので、近傍物体は屈折による光線方向の変化を無視し、遠方の物体は反射の処理と同じように環境マップを用いて屈折を表現する。処理の手順を以下に擬似コードで示す。

```
// 最初に近傍オブジェクトが描画される近傍テクスチャを用意しておく
initialize neighbour texture

...

procedure render_water_surface {

  // アルファ値 0 で近傍テクスチャをクリア
  clear neighbour texture with alpha 0

  // フレームを描画するカメラと同じ位置姿勢で
  // 遠方の物体をキューブマップテクスチャに描画
  render far objects to cube map texture
```

```
  // フレームを描画するカメラと同じ位置姿勢で
  // 近傍の物体をアルファ値 1 で近傍テクスチャに描画
  render near objects with alpha 1 to neighbour texture

  // 水面を構成する全てのフラグメント f に対して
  for each fragment f in water surface {

    // フラグメントに対する他の処理
    ...

    color refract_color

    // f の座標と同じ座標にあるテクセルを近傍テクスチャから取得
    color c = fetch texel at f.xy from neighbour texture

    // テクセルのアルファが 0 以上、
    // つまりオブジェクトが描き込まれているかどうか
    if (c.alpha > 0) {

      // オブジェクトが描き込まれているので
      // 屈折光の色は近傍の物体の色になる
      refract_color = c;
    }
    else {

      // オブジェクトが書き込まれていないので屈折ベクトルの計算
      vector T = compute refract vector

      // 屈折ベクトルを用いてキューブマップから遠方の物体の色を取得し、
      // それを屈折光の色とする
      refract_color = fetch textel at T from cube map texture
    }

    // フラグメントに対する他の処理
    ...
  }
}
```

第5章　実験

本章では実装したシミュレーションの処理時間と生成された画像について述べる。

5.1. 実験環境

提案手法について表 5.1 の環境で実装、実験を行った。

表 5.1 実験環境

CPU	Intel Pentium4 2.8GHz
GPU	NVIDIA GeForce 6800 Ultra
コンパイラ	Microsoft Visual Studio .NET 2003
言語	C 言語
グラフィクスライブラリ	OpenGL
シェーダ言語	Cg [43]

シーン 1) ユーザが操作可能な円筒形の容器に入った流体粒子 2000 個で表された水、シーン 2) 木片 (剛体粒子 192 個、密度 0.5g/cm^3) と相互作用する水 (流体粒子 3808 個)、シーン 3) 石柱 (剛体粒子 192 個、密度 3.0g/cm^3) と相互作用する水 (流体粒子 3808 個) の三つのシーンを用意した。シェーダを用いて表現した光学現象の効果を確かめるために、粒子のみで表示したもの、構築された水面に対して拡散反射のみで表示したもの、シェーダを用いて光学現象を表現して表示したものの 3 つの画像を生成した。また、粒子の数による生成された画像の差を調

べるために粒子の数が 1000 個、2000 個、4000 個で、粒子一つあたりの質量がそれぞれ 4g、2g、1g のときの画像を生成した。

どのシーンにおいても 1 フレームごとに近傍リストの構築を 2 回、流体粒子の位置、速度の更新を 4 回、描画を 1 回を行い、タイムステップは 5ms とした。また、容器は描画していない。

5.2. 結果と考察

シーン 1、シーン 2、シーン 3 で生成された画像を図 5.1、図 5.2、図 5.3、図 5.4、図 5.5 に示す。シェーダ使用による差を表した画像を図 5.6 に示す。粒子の数による差を示した画像を図 5.7 に示す。またフレームレートは表 5.2 のようになった。

表 5.2 フレームレート

シーン	**フレームレート (fps)**
シーン 1	35
シーン 2	17
シーン 3	17
粒子の数 1000 個	70
粒子の数 2000 個	34
粒子の数 4000 個	13

シーン 1 ではユーザの対話的な操作で円筒形の容器を左に動かし、容器の動きに従い流体が変化していることがわかる。シーン 2 では水より密度が低い木片が浮き、シーン 3 では水より密度が高い石柱が沈んでいることがわかる。また、どのシーンにおいても流体要素がが多く集まった場所の圧力が高まり粒子の存在しない方向に向かい運動し、場合によっては飛沫が上がっていることがわかる (図 5.3)。また、例えば図 5.1 や図 5.3 を見ると、コップを左に動かしたときに、左上に向かって移動している粒子が粘性により他の粒子を引きずり、一部の粒子がまとまってが左上に移動しているのがわかる。

5.2.1 シェーダによる効果

実装されたシェーダにより、屈折により背景が歪んで見え、反射により周囲のオブジェクトが写りこんでいることがわかる。フレネル効果により、屈折と反射が合わさりより高い写実性をアニメーション結果に与えている (図 5.6)。

5.2.2 粒子の数による違い

図 5.7 を見ると粒子の数が増えるにつれて滑らかな水面が生成されていることがわかる。1000 個の場合は水面に顕著な凹凸が見られ、大きな違和感があった。しかし、計算時間は粒子の数が少ないほど短く、4000 個の粒子を用いたものは非常に遅いため水の振る舞いとしては違和感があった。

5.2.3 計算時間

表 5.2 より、どのシーンにおいても十分対話的に動作可能なフレームレートで描画できていることがわかる。フレームごとに粒子の位置、速度の更新を 4 回行っているので、1 フレームでシミュレーション内の時間は 20ms 進む。したがって、どのシーンにおいても完全に実時間で処理が行われているとはいえないが、大きな違和感は感じられなかった。

5.2.4 本手法の制限

擬似的に非圧縮性を実現しているため、大量に粒子が積み重なるとゴムのように圧縮と膨張を繰り返してしまうことがある。これは式 (3.17) の k^{fluid} を大きくすることによりある程度改善できるが、安定にシミュレーションを行うには、タイムステップを小さくしなければならなく、計算時間がかかることになる。また、粒子一つから生成される陰曲面が等方的であるため、流体と他の物体との境界に見られるような鋭い形状の表現が難しい。

5.2.5 将来への展望

提案手法は並列な実装が可能であり、筆者らはGPUを用いた並列実装を提案している [44]。結果は本論文の提案手法の実験結果と比べると芳しくないが、GPUでの汎用計算は最近になり大きく注目され始めたため、近い将来汎用計算で用いられるGPU機能が改善され、高速な処理が実現されると考えられる。また、今後並列化による高速化がさかんになり、多くのマルチコアのプロセッサが現れることになると予想される。したがって、並列化可能なアルゴリズムは今後有効でプロセッサの進化に伴い高速に処理されると考えられる。

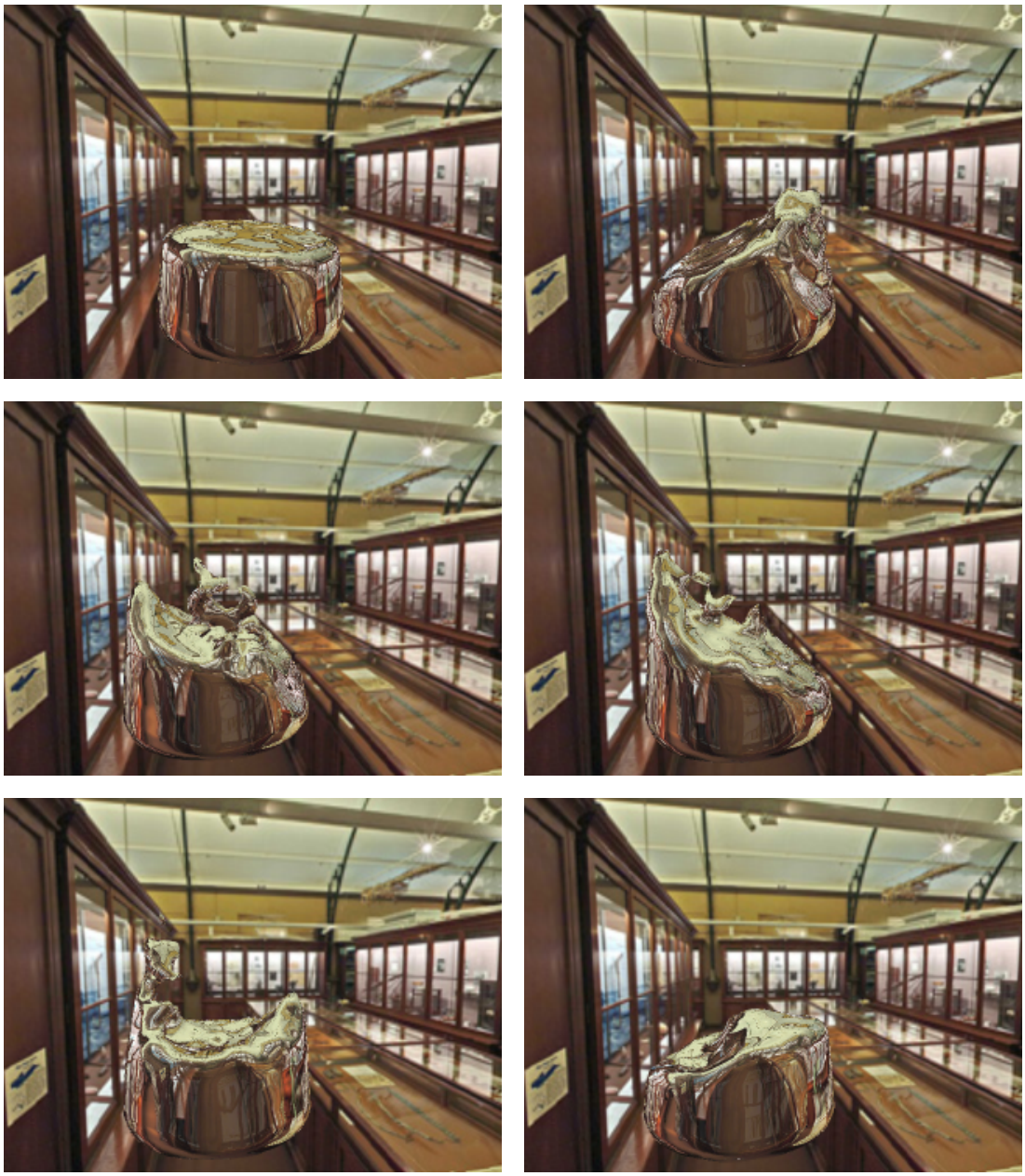

図 5.1　シーン 1 容器を左に動かすことにより水が変形している様子

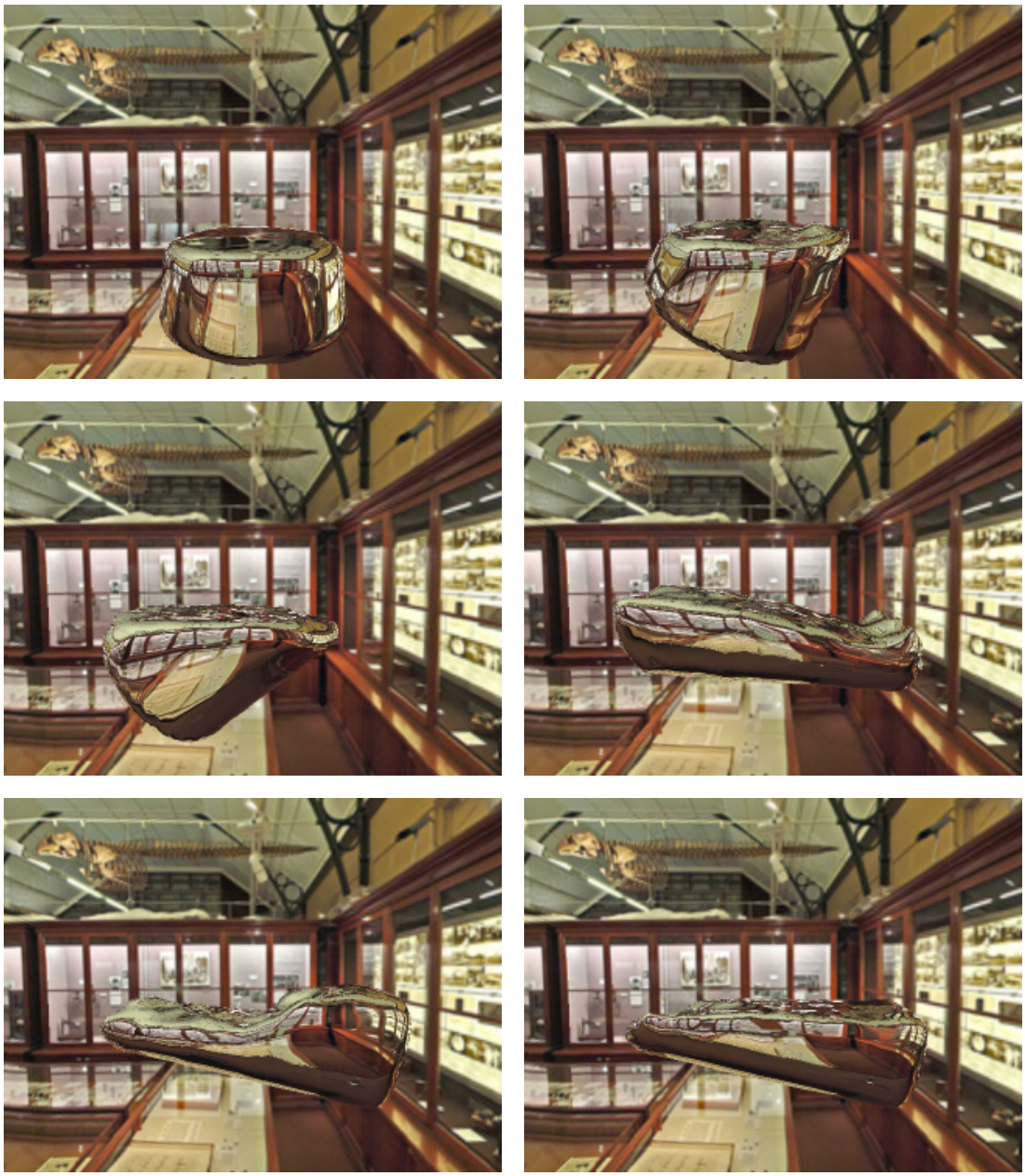

図 5.2　シーン 1 容器を傾けることにより、水が流れている様子

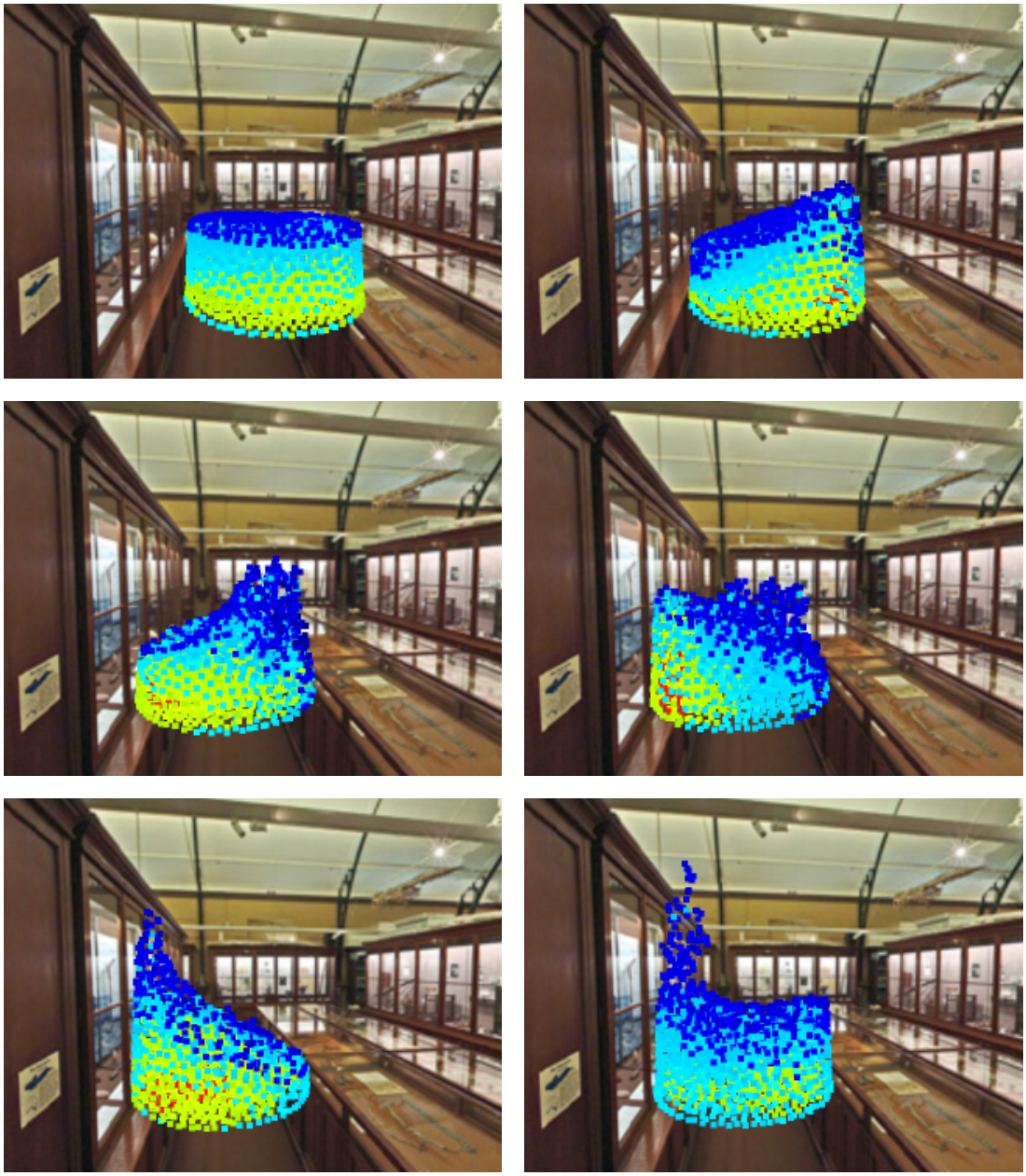

図 5.3 シーン 1 容器を右から左に動かしたときの圧力分布 (青 → 黄 → 赤の順に圧力が高い)

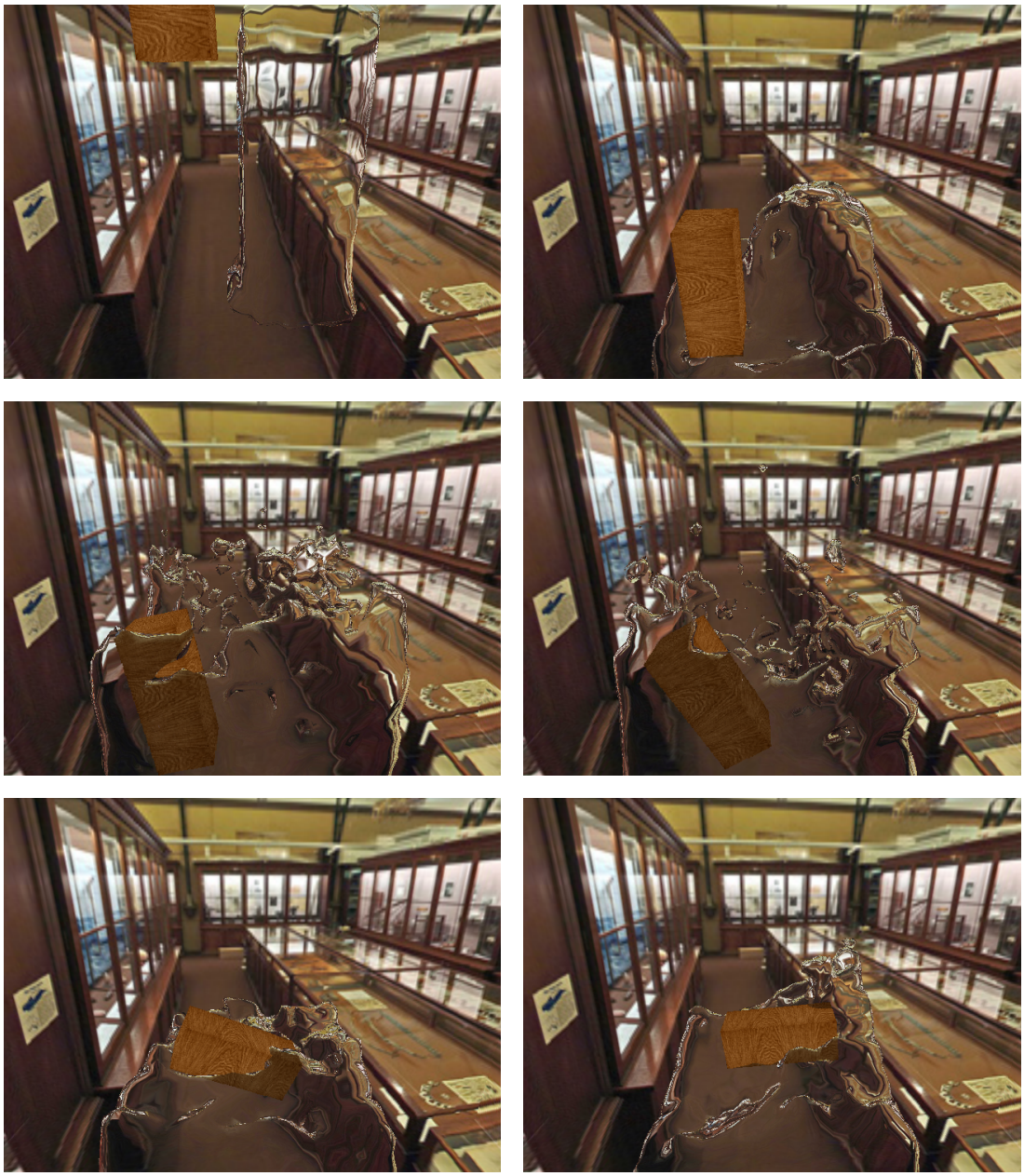

図 5.4　シーン 2 木片と相互作用する水

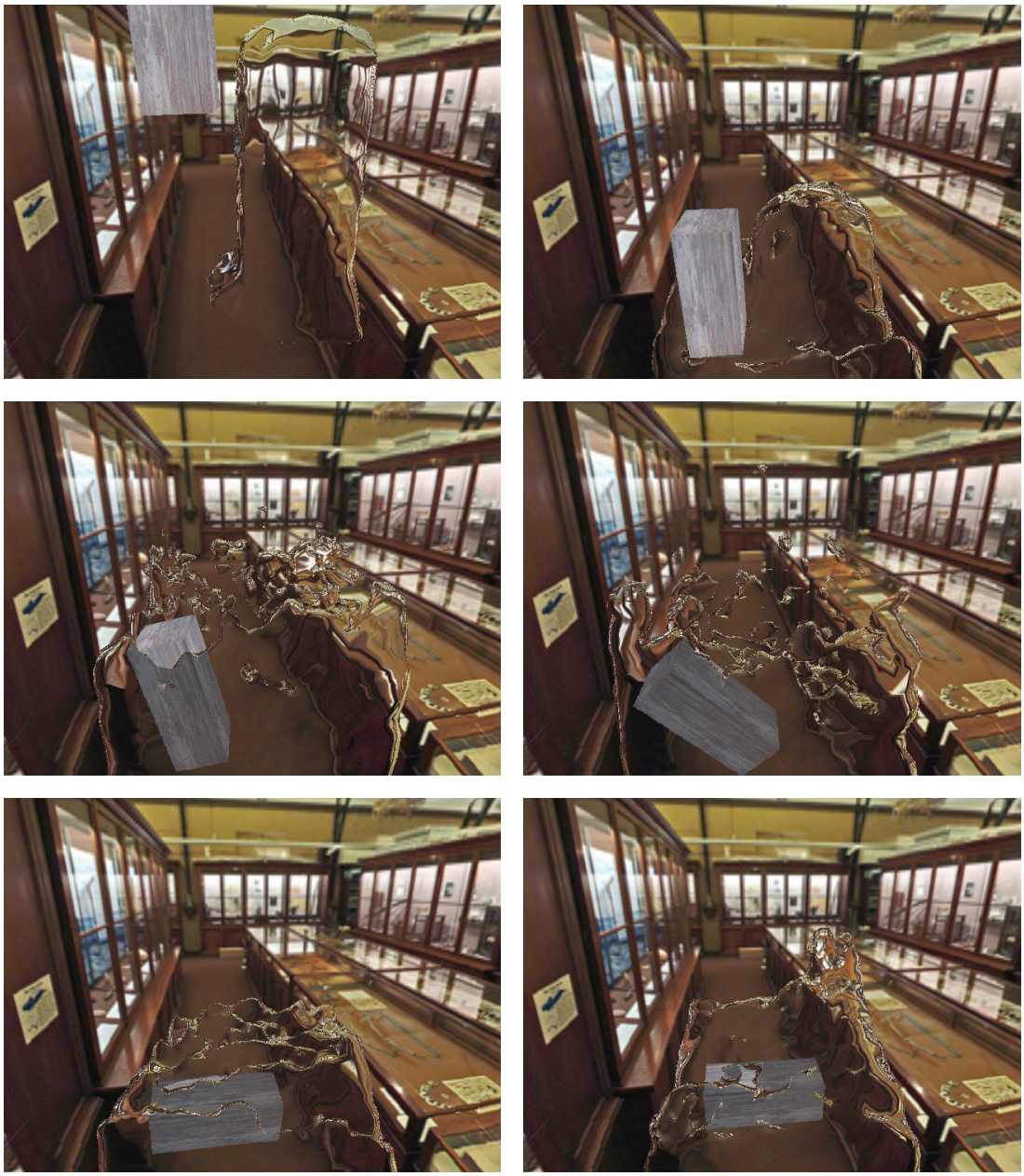

図 5.5 シーン 3 石柱と相互作用する水

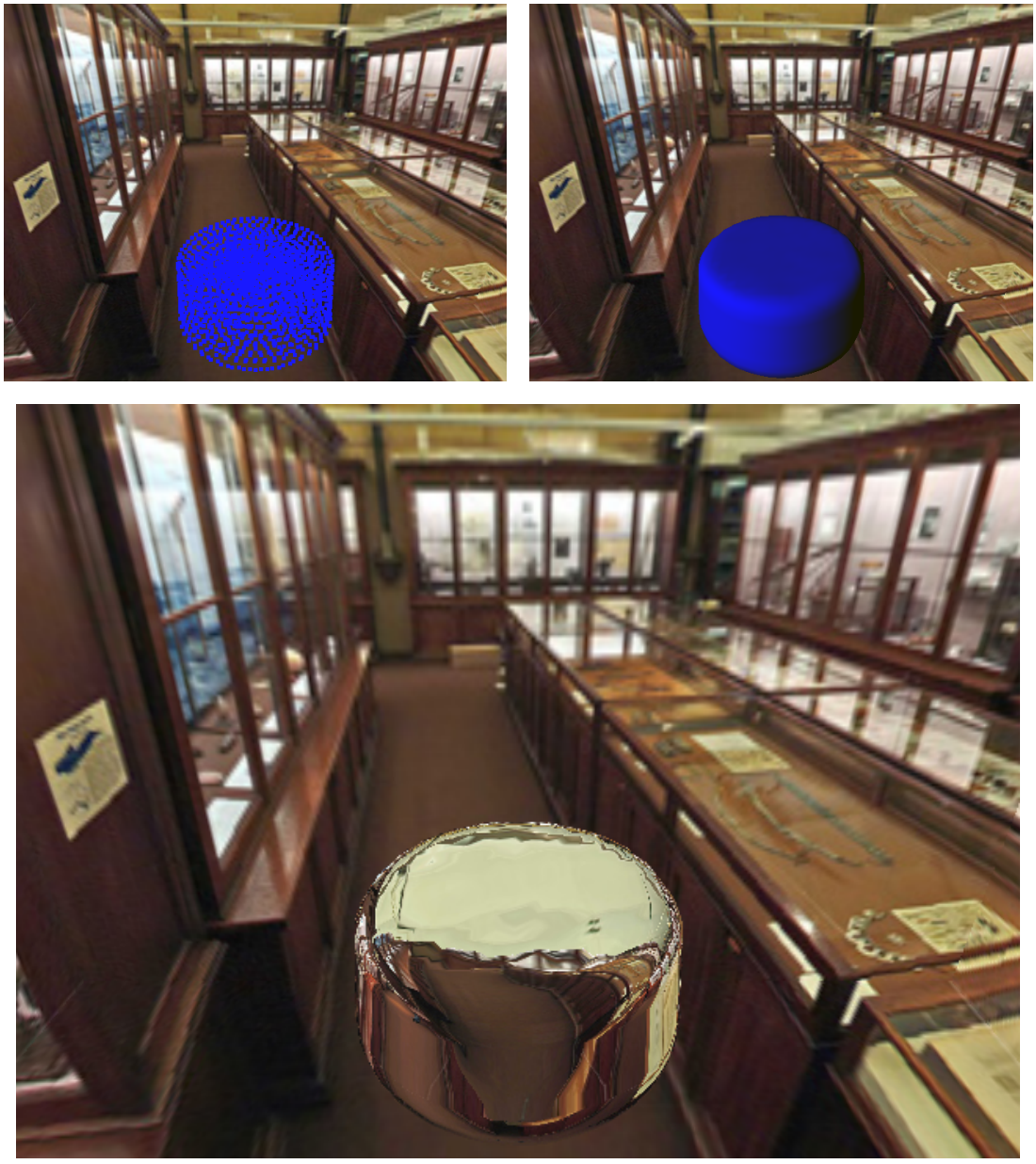

図 5.6　粒子による表示 (左上)、水面に対して拡散反射のみを適用した表示 (右上)、光学現象を表現した表示 (下)

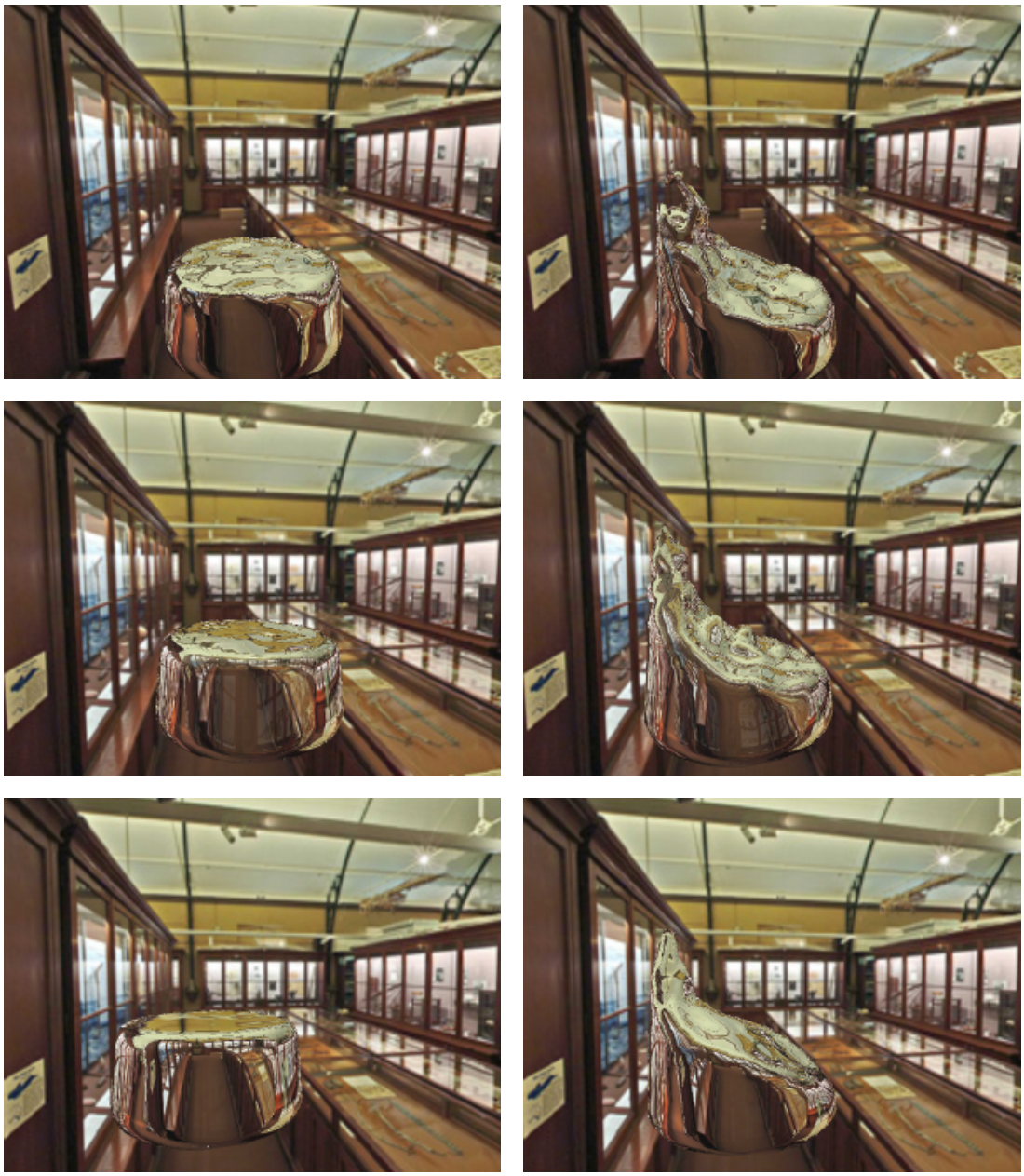

図 5.7 粒子の数 1000 個 (上)、粒子の数 2000 個 (中)、粒子の数 4000 個 (下)

第6章　結論

本論文では、剛体との相互作用を含むSmoothed Particle Hydrodynamicsに基づいた粒子ベースの流体シミュレーションとその高速な実装、反射や屈折、フレネル効果を伴った写実的な水面の描画方法について述べた。本研究では剛体を流体の一種とみなして粒子で構成し、粒子全体が剛体としてふるまうように剛体を構成する粒子の運動を制約することにより、剛体と水との相互作用を実現し、高速な実装を行うことによりインタラクティブなスピードでのシミュレーションを可能にした。また、水面における光学現象をGPUを用いた実装で表現することで、写実的なアニメーションが得られた。

提案手法は、今までゲームやVRアプリケーションでは見られなかった剛体との相互作用を行う汎用性の高い流体のシミュレーションを実現している。実際にゲームやVRアプリケーションに組み込むこむ際に、本手法以外の処理が発生することを考えると、処理速度が十分とはいえないが、提案手法は並列処理が可能でハードウェアの進化の恩恵を受けやすいため、将来ゲームやVRアプリケーションでの利用が期待される。

謝辞

本研究を行うにあたり、終始暖かいご指導を頂いた情報科学研究科像情報処理学講座 千原國宏教授に深く感謝申し上げます。副指導教官として御助言を頂いた視覚情報メディア講座の横矢直和教授に深く感謝申し上げます。御指導を賜わり、また、数々の有益なご助言を与えて下さいました像情報処理学講座眞鍋佳嗣助教授に、深く感謝申し上げます。

ミーティングにおいて、様々なアドバイスをしてくださった像情報処理学講座安室喜弘助手にに厚く御礼申し上げます。本研究や他の活動にあたり種々の相談にのって戴き、数多くのご指導、ご指摘を戴いた像情報処理学講座井村誠孝助手に深く御礼申し上げます.

研究指針に対しご助言をしてくださったCOE 研究員増田泰氏、COE 研究員長縄美香氏、神戸大学佐々木博史助手、熊本大学病院末永貴俊助手に厚く御礼申し上げます。つねに暖かく研究活動をご支援頂いた村上満佳子氏 (元像情報処理学講座教務職員) に感謝します。

研究やそれ以外の様々な面で日頃からご協力を頂いた像情報処理学講座博士後期および前期課程の皆様に感謝します。日ごろからお世話になった像情報処理学講座 川本桂子秘書に感謝します。

研究に必要なキューブマップテクスチャを提供してくださったPaul Bourke 氏とそのテクスチャの作者であるPeter Murphy 氏に感謝します。

最後に様々な面で私をささえてくれた父と母に感謝します。

参考文献

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

[2] Douglas Enright, Stephen Marschner, and Ronald Fedkiw. Animation and rendering of complex water surfaces. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pp. 736–744. ACM Press, 2002.

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

[4] Matthew Moore and Jane Wilhelms. Collision detection and response for computer animationr3. In *Proceedings of the 15th annual conference on Computer graphics and interactive techniques*, pp. 289–298. ACM Press, 1988.

[5] Shoichi Hasegawa and Makoto Sato. Real-time rigid body simulation for haptic interactions based on contact volume of polygonal objects. *Comput. Graph. Forum*, Vol. 23, No. 3, pp. 529–538, 2004.

[6] D. Baraff. Analytical methods for dynamic simulation of non-penetrating rigid bodies. In *SIGGRAPH '89: Proceedings of the 16th annual conference on Computer graphics and interactive techniques*, pp. 223–232. ACM Press, 1989.

[7] David Baraff. Fast contact force computation for nonpenetrating rigid bodies. In *SIGGRAPH '94: Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pp. 23–34. ACM Press, 1994.

[8] Brian Mirtich and John Canny. Impulse-based simulation of rigid bodies. In *SI3D '95: Proceedings of the 1995 symposium on Interactive 3D graphics*, pp. 181–ff. ACM Press, 1995.

[9] Brian Mirtich. Timewarp rigid body simulation. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pp. 193–200. ACM Press/Addison-Wesley Publishing Co., 2000.

[10] Eran Guendelman, Robert Bridson, and Ronald Fedkiw. Nonconvex rigid bodies with stacking. *ACM Trans. Graph.*, Vol. 22, No. 3, pp. 871–878, 2003.

[11] Alain Fournier and William T. Reeves. A simple model of ocean waves. In *SIGGRAPH '86: Proceedings of the 13th annual conference on Computer graphics and interactive techniques*, pp. 75–84. ACM Press, 1986.

[12] Damien Hinsinger, Fabrice Neyret, and Marie-Paule Cani. Interactive animation of ocean waves. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pp. 161–166. ACM Press, 2002.

[13] Michael Kass and Gavin Miller. Rapid, stable fluid dynamics for computer graphics. In *SIGGRAPH '90: Proceedings of the 17th annual conference on Computer graphics and interactive techniques*, pp. 49–57. ACM Press, 1990.

[14] J. F. O'Brien and J. K. Hodgins. Dynamic simulation of splashing fluids. In *CA '95: Proceedings of the Computer Animation*, p. 198. IEEE Computer Society, 1995.

[15] F. H. Harlow and J. E. Welch. Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface. In *Physics of Fluids, vol. 8*, pp. 2182–2189, 1965.

[16] Nick Foster and Dimitri Metaxas. Realistic animation of liquids. *Graph. Models Image Process.*, Vol. 58, No. 5, pp. 471–483, 1996.

[17] Nick Foster and Dimitris Metaxas. Modeling the motion of a hot, turbulent gas. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pp. 181–188. ACM Press/Addison-Wesley Publishing Co., 1997.

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

[19] Oh-Young Song, Hyuncheol Shin, and Hyeong-Seok Ko. Stable but non-dissipative water. *ACM Trans. Graph.*, Vol. 24, No. 1, pp. 81–97, 2005.

[20] Mark Carlson, Peter J. Mucha, and Greg Turk. Rigid fluid: animating the interplay between rigid bodies and fluid. *ACM Trans. Graph.*, Vol. 23, No. 3, pp. 377–384, 2004.

[21] T. Yabe, T. Ishikawa, P. Y. Wang, T. Aoki, Y. Kadota, and F. Ikeda. A universal solver for hyperbolic equations by cubic-polynomial interpolation II. Two- and three-dimensional solvers. *Computer Physics Communications*, Vol. 66, pp. 233–242, September 1991.

[22] C. W. Hirt and B. D Nichols. Volume of fluid (vof) method for the dynamics of free boundaries. *J. Comput. Phys.*, Vol. 39, pp. 201–225, 1981.

[23] Tsunemi Takahashi, Hiroko Fujii, Atsushi Kunimatsu, Kazuhiro Hiwada, Takahiro Saito, Ken Tanaka, and Heihachi Ueki. Realistic animation of fluid

with splash and foam. *Comput. Graph. Forum*, Vol. 22, No. 3, pp. 391–400, 2003.

[24] W. T. Reeves. Particle systems a technique for modeling a class of fuzzy objects. *ACM Trans. Graph.*, Vol. 2, No. 2, pp. 91–108, 1983.

[25] Gavin S. P. Miller and Andrew Pearce. Globular dynamics: A connected particle system for animating viscous fluids. *Computers & Graphics*, Vol. 13, No. 3, pp. 305–309, 1989.

[26] Peter Kipfer, Mark Segal, and Rüdiger Westermann. Uberflow: A GPU-based particle engine. In *Proceedings Eurographics Conference on Graphics Hardware*, p. to be published, 2004.

[27] 越塚誠一. 数値計算流体力学. 培風館, 1997.

[28] Simon Premzoe, Tolga Tasdizen, James Bigler, Aaron Lefohn, and Ross T. Whitaker. Particle-based simulation of fluids. *Comput. Graph. Forum*, Vol. 22, No. 3, pp. 401–410, 2003.

[29] J.J. Monaghan. Smoothed Particle Hydrodynamics. *Annual Review of Astronomy and Astrophysics*, Vol. 30, pp. 543–574, 1992.

[30] Jos Stam and Eugene Fiume. Depicting fire and other gaseous phenomena using diffusion processes. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pp. 129–136. ACM Press, 1995.

[31] Mathieu Desbrun and Marie-Paule Gascuel. Smoothed particles: a new paradigm for animating highly deformable bodies. In *Proceedings of the Eurographics workshop on Computer animation and simulation '96*, pp. 61–76. Springer-Verlag New York, Inc., 1996.

[32] Dan Stora, Pierre-Olivier Agliati, Marie-Paule Cani, Fabrice Neyret, and Jean-Dominique Gascuel. Animating lava flows. In *Proceedings of the 1999*

conference on Graphics interface '99, pp. 203–210. Morgan Kaufmann Publishers Inc., 1999.

[33] Matthias Müller, Simon Schirm, and Matthias Teschner. Interactive blood simulation for virtual surgery based on smoothed particle hydrodynamics. *Technol. Health Care*, Vol. 12, No. 1, pp. 25–31, 2004.

[34] J.P.Morris. Simulating surface tension with smoothed particle hydrodynamics. *International Journal for Numerical Methods in Fluids*, Vol. 33, pp. 333–353, 2000.

[35] David Love Tonnesen and Demetri Terzopoulos. *Dynamically coupled particle systems for geometric modeling, reconstruction, and animation.* PhD thesis, 1998.

[36] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pp. 163–169. ACM Press, 1987.

[37] Jules Bloomenthal. An implicit surface polygonizer. pp. 324–349, 1994.

[38] Matthias Wlock. Fresnel reflection, 2002. http://developer.nvidia.com/.

[39] Christophe Schlick. A Customizable Reflectance Model for Everyday Rendering. In *Fourth Eurographics Workshop on Rendering*, No. Series EG 93 RW, pp. 73–84, Paris, France, 1993.

[40] Turner Whitted. An improved illumination model for shaded display. *Commun. ACM*, Vol. 23, No. 6, pp. 343–349, 1980.

[41] James F. Blinn and Martin E. Newell. Texture and reflection in computer generated images. *Commun. ACM*, Vol. 19, No. 10, pp. 542–547, 1976.

[42] Ned Greene. Environment mapping and other applications of world projections. *IEEE Comput. Graph. Appl.*, Vol. 6, No. 11, pp. 21–29, 1986.

[43] William R. Mark, R. Steven Glanville, Kurt Akeley, and Mark J. Kilgard. Cg: a system for programming graphics hardware in a c-like language. *ACM Trans. Graph.*, Vol. 22, No. 3, pp. 896–907, 2003.

[44] Takashi Amada, Masataka Imura, Yasumuro Yasumuro, Yoshitsugu Manabe, and Kunihiro Chihara. Particle-based fluid simulation on gpu. In *ACM Workshop on General-Purpose Computing on Graphics Processors*, pp. C–36, 2004.

付録

A. カーネルの係数

提案されたカーネルの特徴を持ちながら式 (3.2) を満たすようにカーネルの係数を決定する。Poly6 カーネルの係数を k_{poly6} とすると、Poly6 カーネルは次の式で表される。

$$W_{\text{poly6}}(\mathbf{r}, h) = k_{\text{poly6}} \begin{cases} (h^2 - r^2)^3 & 0 \leq r \leq h \\ 0 & otherwise \end{cases} \tag{A.1}$$

ここで、Poly6 カーネルは $\mathbf{r}$ の長さ $|\mathbf{r}| = r$ のみに依存し、$0 \leq r \leq h$ なので式 (3.2) の左辺は

$$\int_0^h W_{\text{poly6}}(\mathbf{r}, h) \cdot 4\pi r^2 dr = \int_0^h k_{\text{poly6}}(h^2 - r^2)^3 \cdot 4\pi r^2 dr \tag{A.2}$$

になり、この積分値が 1 になるように係数 k_{poly6} を決定すればよい。

式 (A.2) は

$$\begin{aligned} & \int_0^h k_{\text{poly6}}(h^2 - r^2)^3 \cdot 4\pi r^2 dr \\ &= 4\pi k_{\text{poly6}} \int_0^h (h^6 r^2 - 3h^4 r^4 + 3h^2 r^6 - r^8) dr \\ &= 4\pi k_{\text{poly6}} \left[\frac{1}{3} h^6 r^3 - \frac{3}{5} h^4 r^5 + \frac{3}{7} h^2 r^7 - \frac{1}{9} r^9 \right]_0^h \\ &= 4\pi k_{\text{poly6}} \cdot \frac{16}{315} h^9 \\ &= \frac{64\pi h^9}{315} k_{\text{poly6}} \end{aligned} \tag{A.3}$$

となるので、この積分値が 1 になるには

$$k_{\text{poly6}} = \frac{315}{64\pi h^9} \tag{A.4}$$

であればよい。同様に Spiky カーネル、Viscosity カーネルの係数が決定される。

B. Leap-Frog

変数tの関数$f(t)$についてΔtを微小量としたとき$f\left((t+\frac{1}{2}\Delta t)+\frac{1}{2}\Delta t\right)$、$f\left((t+\frac{1}{2}\Delta t)-\frac{1}{2}\Delta t\right)$を第二項までテイラー展開すると

$$f\left((t+\frac{1}{2}\Delta t)+\frac{1}{2}\Delta t\right) = f(t+\frac{1}{2}\Delta t)+\frac{1}{2}\Delta t\frac{df}{dt}(t+\frac{1}{2}\Delta t) \quad \text{(B.5)}$$

$$f\left((t+\frac{1}{2}\Delta t)-\frac{1}{2}\Delta t\right) = f(t+\frac{1}{2}\Delta t)-\frac{1}{2}\Delta t\frac{df}{dt}(t+\frac{1}{2}\Delta t) \quad \text{(B.6)}$$

となり、式 (B.5) から式 (B.6) を引くと

$$f(t+\Delta t)-f(t) = \Delta t\frac{df}{dt}(t+\frac{1}{2}\Delta t) \quad \text{(B.7)}$$

を得る。また、同様にテイラー展開を用いて

$$f(t+\frac{1}{2}\Delta t)-f(t-\frac{1}{2}\Delta t) = \Delta t\frac{df}{dt}(t+\Delta t) \quad \text{(B.8)}$$

を得る。ここで、

$$\frac{d\mathbf{r}_i}{dt}(t) = \mathbf{v}_i(t) \quad \text{(B.9)}$$

$$\frac{d\mathbf{v}_i}{dt}(t) = \mathbf{a}_i(t) \quad \text{(B.10)}$$

だから式 (B.7)(B.8) をそれぞれ式 (B.9)(B.10) で置き換え変形すると、

$$\mathbf{r}_i(t+\Delta t) = \mathbf{r}_i(t)+\Delta t\mathbf{v}_i(t+\frac{1}{2}\Delta t) \quad \text{(B.11)}$$

$$\mathbf{v}_i(t+\frac{1}{2}\Delta t) = \mathbf{v}_i(t-\frac{1}{2}\Delta t)+\Delta t\mathbf{a}_i(t) \quad \text{(B.12)}$$

が得られる。