2009－CG－134　(14)
2009／2／16

粒子ベース流体シミュレーションの複雑な屈折を含む効率的な可視化

安田　廉[†]　原田隆宏[††]　河口洋一郎[†]

本研究では粒子法による流体の複雑な屈折を含む効率的かつ高速な可視化手法を提案する．本手法ではボクセルベースのメタボールとレイキャスティングを用いて複雑な屈折を含む可視化を行っているが, 粒子の位置情報からボクセル上にメタボールの濃度場を構築する処理は非常に負荷が大きく, ボトルネックになりうる．そこで一般的に流体シミュレーションと完全に別個のプロセスとして処理される可視化プロセスを, 流体シミュレーション (Smoothed Particle Hydrodynamics) の処理内部に一部に置き, さらに流体シミュレーションのデータを一部用いることで可視化プロセスにおける無駄な処理を省き, 処理コストを低減する．この提案手法を用いることで, 最大 3.6 倍の高速化を可能すると共に, 複雑な屈折を含む流体のリアルタイムでの可視化を可能とした．

Effective Visualization of Particle-based Fluid with Multiple Refractions

Ren Yasuda,[†] Takahiro Harada[††] and Yoichiro Kawaguchi[†]

This paper presents a novel algorithm for efficient visualization of particle-based fluid simulation with multiple refractions. In general, particle-based fluid simulation and visualization are processed completely separated. The novelty of our method lives in combination of these two processes to avoid extra processes in visualization, and to reduce computing cost of visualization. We applied the method for a fluid simulation by smoothed particle hydrodynamics and measured the computation time of the method. The present method could visualize results up to 3.6 times faster than the previous method.

1. 序　論

近年の GPU の性能とプログラマブル性の向上に伴い, 流体のリアルタイムシミュレーション (グリッドベース・粒子ベースともに) はもはや困難な問題とは呼べなくなってきている．しかし流体方程式がリアルタイムに解かれる一方で, 流体の可視化技術は未だに未発達である．これはシミュレーションにおいて流体方程式が一度離散化されてしまうため, そこからもう一度滑らかな表面を構築するという難題が存在することに起因する．特に粒子ベースの流体シミュレーションは計算点自体が移動してしまうため, 計算点が固定されたグリッドベースの手法に比べて可視化がさらに困難な問題になっている．そのため粒子ベース流体の可視化手法に関する多くの研究があり, その手法は大まかにボクセルベースやポイントベース, またポリゴンベースに分けられる．また, 多くの場合粒子ベースの流体はメタボールなどの陰曲面関数を用いて可視化されるため, 流体の可視化ではなくメタボールまたは陰曲面関数のレンダリングとしての研究も多い．本研究では粒子ベース流体シミュレーションの可視化を, ボクセルベースの手法を用いて可視化した．ボクセルベースの手法を選んだのは, 流体のよりもっともらしい可視化のために欠かすことのできない複雑な屈折をレイキャスティングによるボクセルのトラバースによって可能にするためである．

次に基本的なボクセルベースの可視化手法であるが, まず用意したボクセル空間上に濃度場を構築し, そのボクセルに対してレイキャスティングを行い等値面を検出するという単純ものである．濃度場の構築は, 通常はどの粒子も等しい濃度分布を保持していると仮定し, 全ての粒子の濃度を足し合わせることで行われる．なお, 各粒子の濃度分布は粒子の中心からの距離の 2 乗に反比例するものとした．しかし粒子が密集している部分の粒子は広い濃度分布を持つ必要はないのは明らかであるため, 全ての粒子が等しい濃度分布を持つのは無駄である．そこでこの無駄を省くために本研究では”粒子存在”を定義し, 濃度場の構築の処理を高速化する．粒子存在は各粒子の周囲 6 方向に粒子がどれだけ存在するかを表し, この粒子存在に応じて各粒子

† 東京大学大学院情報学環・学際情報学府
Interfaculty Initiative in Information Studies, The University of Tokyo
†† Havok

の濃度分布を変形する．この粒子存在の算出は流体シミュレーション中の処理中に組み込むことで, 粒子存在の計算することによる負荷を最小限にとどめることが可能である．

2. 関 連 研 究

粒子間の値の補間には多くの場合メタボールが用いられる．そのため粒子ベースの流体の可視化の研究とは, すなわちメタボールのレンダリング手法の研究とは切り離せない関係にある．メタボールのレンダリングの代表的なものとしてマーチングキューブ法[6] のようにメタボールをポリゴンで表現する手法も存在するが, 近年の SPH シミュレーションのように大量の粒子が存在するような場合には負荷もデータ量も非常に大きくなってしまうため適さない．

Muller らはスクリーンスペースでの反復処理によってメタボールを構成する手法を提案した．しかしこれはスクリーンスペースであるが故に, 複雑な屈折などを扱うのには適さない．また Kanamori らは Depth Peeling と *Bezier Clipping* を用いて可視化している[5]．しかし Kanamori らの手法では反復処理が大きな負荷となると共に, Depth Peeling によって層状にメタボールが検出されるためやはり複雑な屈折には適していない．Iwasaki らはポイントベースの手法によって粒子法の可視化を行ったが[4], 水面のレンダリングのための手法であったため細部の情報は失われていた．

複雑な屈折を計算すると, 視点に一番近い表面だけではなく計算領域全体の情報が不可欠になってくるため, 本手法ではボクセルベースの手法を用いている．

3. ボクセルベースの可視化手法

ここではボクセルを用いた粒子法の可視化手法について簡潔に述べる．本手法では SPH によるシミュレーションを行った後で濃度場をボリューム上に構築するが, ここでいうボリュームとは 3D テクスチャであり, ボリュームの解像度は SPH による流体シミュレーション中で近隣粒子の探索の高速化のために用いるグリッドと同じ解像度とした．これは後で述べるが, 濃度場構築の効率化のためには SPH のグリッドとボリュームの解像度が一致していた方が都合が良いためである．ただし, ボリュームの解像度が SPH のグリッドの解像度の 2^{3n} 倍であれば多少処理を変更すれば可能である．そして全粒子の位置から一定範囲内にあるボクセルに対し各粒子が持つ濃度分布に応じた濃度を足し合わせることで濃度場の構築を行い, その濃度場に対してレイキャスティングを行うことで可視化とする．レイキャスティングとはあるスカラー場をレイに沿って微小な間隔で走査してゆき, 閾値を用いて表面を決定する手法である．実際には表面の位置のずれを抑えるために内分比を用いてより正確に近い位置を表面として決定している．また表面の位置における濃度勾配から法線ベクトルを容易に求めることができるため, その法線に対して屈折を計算することで複雑な屈折を含む可視化が可能になる．

4. 手　　法

4.1 Smoothed Particle Hydrodynamics

本手法では粒子ベースの流体シミュレーション手法として Smoothed Particle Hydrodynamics(SPH)[7] を用いている．非圧縮流体の支配方程式は以下のように表わされる．

$$\frac{D\rho}{Dt} = 0 \tag{1}$$

$$\frac{D\mathbf{U}}{Dt} = -\frac{1}{\rho}\nabla\mathbf{P} + \nu\nabla^2\mathbf{U} + \mathbf{g} \tag{2}$$

ここで ρ,$\mathbf{U}$,$\mathbf{P}$,ν,g はそれぞれ流体の密度, 速度, 圧力, 動粘性係数, 重力であり, SPH ではこの方程式を粒子で離散化してシミュレーションを行う．また,SPH は全粒子の座標・速度・密度に加え, 計算を効率化するためグリッド (格子) を保持する．グリッドはグリッドの各セル中にある粒子の数とその粒子番号を格納しており, 圧力や密度の計算の際に行う近傍粒子探索処理を高速化するために用いる．シミュレーションではまず各粒子の座標における密度を近傍の粒子を用いて算出し, 次に計算した近傍の粒子の密度を用いて粒子に加わる力の計算を行う．以上の処理は処理を完全に並列化することが可能であるため, SPH の GPU 実装に関する研究が存在する[1)3)]．

4.2 濃度場の構築

SPH による流体シミュレーションを行った後, 粒子の位置データを用いて濃度場を構築を構築する．序論で述べたとおり各粒子は中心からの距離の 2 乗に反比例する濃度分布を持つとし, 一定範囲内にあるボクセルに対して分布に従った濃度を足し合わせる．1 つの処理は非常に単純ではあるが, 元々シミュレーションに用いる粒子が非常に多い上に各粒子が多くのボクセルにアクセスする必要があるため, 計算負荷が非常に大きい．そこで濃度場の構築を 2 回の処理に分け, 大まかな構築を行った後に細かな構築を行うことで処理の高速化を行う (図 1)．細かな濃度場の構築の際新たに粒子存在を定義し, さらに処理を効率化する．またレイキャスティングでは一定の閾値を用いて濃度場か

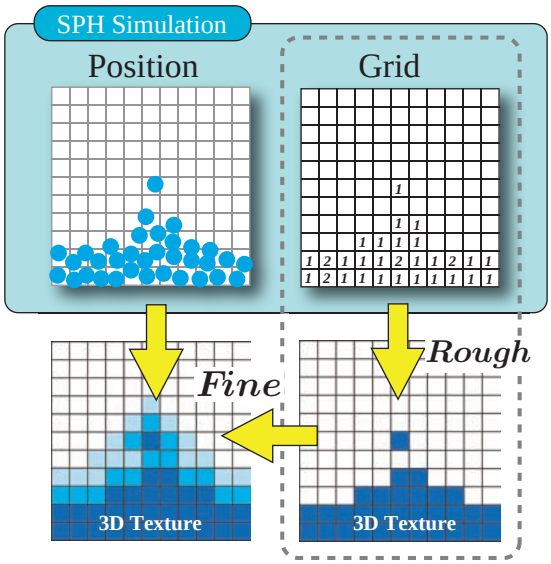

図 1 濃度場構築の流れ：最初に SPH の Grid を用いて大まかな濃度場の構築を行い，次に SPH の粒子座標を用いて細かな濃度場の構築を行う．

ら等値面を検出するため，ボクセルの濃度値がある値以上になった場合にはボクセルに濃度を足し合わせる必要がない．よってボクセルが十分大きな濃度を持っていた場合を”飽和した”とし，大まかな濃度場構築によって飽和したボクセルをスキップすることで細かい構築の際に高速化を行う．大まかな濃度場の構築・細かな濃度場の構築・ボクセルの飽和とスキップのそれぞれについての詳細を以下に述べる．

4.2.1 大まかな濃度場の構築

ここでは SPH におけるグリッドのデータを用いて大まかな濃度の足し合わせ処理を行う．上で述べたとおり,1 つの粒子によって形成される濃度球は最小でも 1 つのボクセルの大きさよりも小さくはならないため，あるボクセルの中に粒子が 1 つでも存在する場合はそのボクセルは必ず閾値以上の濃度を持つことになる．ボクセルの解像度は SPH のグリッドと等しいので，各ボクセルの中に粒子が存在数するか否かは SPH の対応するグリッドセルを参照することで知ることが可能である．そこで各ボクセルは対応するグリッドセルからボクセル中に存在数する粒子数を読み込み，中に粒子が存在する場合は閾値を少し上回る濃度を加え，粒子が存在しない場合は値を 0 にする．この処理はボリュームの初期化を兼ねており，この処理の前にボリュームの値を初期化 (全てのボクセルの濃度を 0 に) する必要はない．

4.2.2 細かな濃度場の構築

前の処理によって大まかな濃度場の構築はされたも

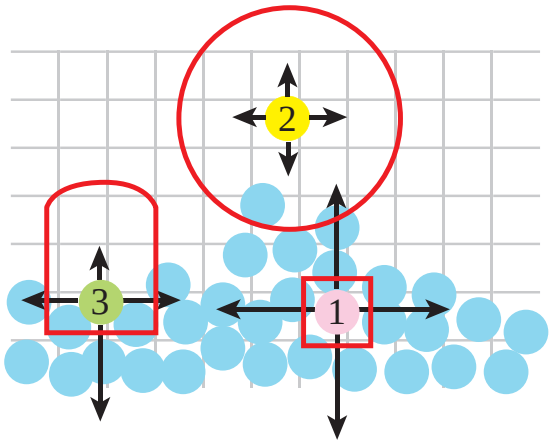

図 2 粒子存在 (簡略化のために図は 2 次元であり，パラメータも 4 方向分しか持たない)．黒の矢印は粒子存在の対応する各方向のパラメータの大きさを表し，赤色の線で囲われた領域が各粒子の持つ濃度分布の形状を表す

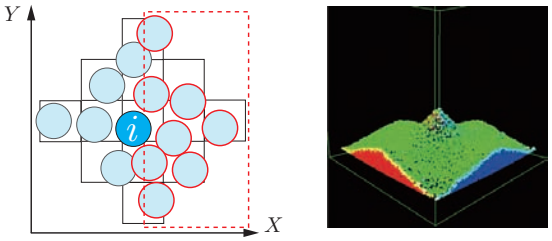

図 3 粒子存在の計算手法．左) 粒子 i の粒子存在の正の X 方向のパラメータを計算するために用いられる粒子．右) SPH による流体の粒子を粒子存在のパラメータを用いて色付けしたもの．x,y,z 方向のパラメータの大きさを r,g,b に対応させている．

ののボリュームは 2 値化されたに過ぎず，表面付近の細かな濃度は全く考慮されていないため詳細な濃度場の構築を行う必要がある．前の処理ではボクセルを処理の 1 単位として大まかな濃度場の構築を行ったが，この詳細な処理では処理単位を粒子として各粒子の回りにあるボクセルに濃度を足し合わせてゆく．ただし，既に飽和したボクセルに関しては濃度を足し合わせる必要はないので，足し合わせ処理の前に濃度を読み込みそのボクセルが飽和しているかを調べてから足し合わせ処理を行う．前の処理によって既に多くのボクセルが飽和しているためかなりの処理をスキップすることができるが，それでもなお足し合わせ処理は負荷が高い．これは対象のボクセルが加算処理をスキップ可能であるかを調べるために濃度を足し合わせる可能性があるすべてのボクセルに対して少なくとも 1 度は

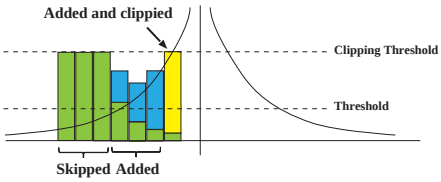

図 4 ある粒子とその周囲のボクセルにおける濃度の加算処理. 緑色の棒は既にほかの粒子によって加えられていた濃度の大きさを表す. Clipping threshold は飽和量を規定する閾値であり, ただの Threhsold は Ray Casting の際に表面を規定する閾値である. 既にそのボクセルが飽和していれば濃度の加算処理はスキップされ, 飽和していなければ濃度分布に応じた濃度が加算される (青色). ただし, 濃度が飽和量を上回ることはない (黄色).

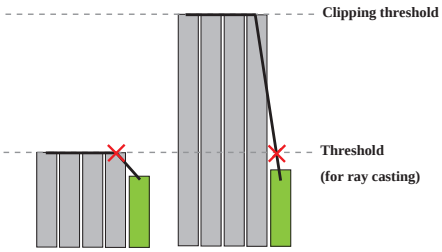

図 5 飽和量がレイキャスティングの閾値と近い場合に起こる問題. 飽和量がレイキャスティングの閾値よりも十分大きければ本来の表面の位置からは非常に近い場所が表面として検出されるが (右図), 飽和量がレイキャスティングの閾値に近い場合は本来の位置から離れた場所が表面として検出されてしまう (左図).

アクセスしなければならないためであり, メモリへのアクセスが浮動小数点よりも遥かに時間を必要とする GPU では大量のメモリアクセスが遅延の原因となりうるからである. そこで本手法では新たに”粒子存在”を定義し, このスキップ処理におけるメモリアクセスを減らす.

粒子存在とは上下左右前後 (+x, -x, +y, -y, +z, -z) の 6 方向に対応する 6 つにパラメータを持ち, ”近傍粒子が各方向にどれだけ存在しているか”を表す. 粒子存在が大きな値をもつ方向は自分以外の粒子が近くに多く存在するということであるから, その方向への濃度分布はさほど広くある必要はない. よって粒子存在を求めた後に図 2 に示すように, 各方向のパラメータの大きさに基づいて各粒子の濃度分布を変形 (縮小) させることで処理の効率化を行う. 図 2 には 3 つの流体の粒子を色付けしてあり, それぞれが異なった状況にある. 粒子の中心から伸びる矢印は粒子存在の各方向のパラメータの大きさを示しており, 赤線の枠は粒子存在に基づいて変形された粒子の濃度分布を表す. なお図 2 は簡単のため 3 次元では無く 2 次元にしており, 従って粒子存在も 6 方向ではなく 4 方向の値を持つとしている. 図 2 中の粒子 1 は水中に存在しており周囲全方向に多くの粒子が存在するため, 粒子存在の全方向のパラメータが大きな値を持つ. このための赤線で表された濃度分布は最小限の大きさになり一方, 粒子 2 は水面上 (水の境界線上) に存在しているため, 周囲の粒子数には方向によって大きく異なる. 従って粒子存在は水中方向は大きく, 真空方向には小さな値を持ち, さらにどちらでもない方向には中間の値を持つことになる. 更に粒子 3 は一つで孤立しており周囲に粒子が存在しない. このため粒子存在の値は全方向において 0 であり, 濃度分布の変形は行われない.

4.2.3 粒子存在の計算手法

ここでは粒子存在の計算方法について述べる. 粒子存在を計算するためには近傍粒子を探索しなければないが, SPH の流体シミュレーションでも近傍粒子を探索するプロセスが存在するため新たに近傍粒子探索のプロセスは追加せず,SPH のシミュレーション中で粒子存在の計算を行う. これによって新たなメモリアクセスは生じないため, この処理のコストは僅かである. 粒子存在の各方向のパラメータを計算するためには, 対応する方向に存在する粒子のみを考慮に入れて算出する. 具体的には図 3 に示すように, 粒子 i の粒子存在の正の x 方向の成分を計算する場合は一定範囲内にある粒子のうちで正の x 方向にある粒子を用い, 算出には以下の式を用いた.

$$[P_i]_{px} = \sum_j \frac{k}{[r_{ij}]_x/([r_{ij}]_y)^2} \tag{3}$$

ここで k は任意の係数であり,r_{ij_x},r_{ij_y} はそれぞれ粒子 i と粒子 j の相対位置の x 成分と y 成分を表す (式 3 も簡単のために 2 次元とした). このようにして求めた粒子存在を用いて粒子の色付けをした物が図 3 の右図であり, 極めて明快に色分けされているのが見て取れる.

4.2.4 濃度の飽和と処理のスキップ

濃度場の構築を大まかな処理と細かな処理を分けることで処理の高速化が可能なのは, ある閾値をボクセルの持ちうる濃度の上限とし, 上限に達したボクセルを”飽和した”として加算処理をスキップすることができるからである. ここでいう閾値とはレイキャスティングによって等値面を構築する場合の閾値とは異なる値であり, レイキャスティングの閾値よりも十分に大きな値を用いる. これは濃度がボクセルによって離散化されているため中間の濃度はテクスチャの線形補間によって得られるため, 上限をレイキャスティングの閾値に近い値にすると本来とは異なる場所が等値面として検出されてしまうのを防ぐためである (図 5). 当然上限を十分大きな値にしてもある程度の位置のずれ

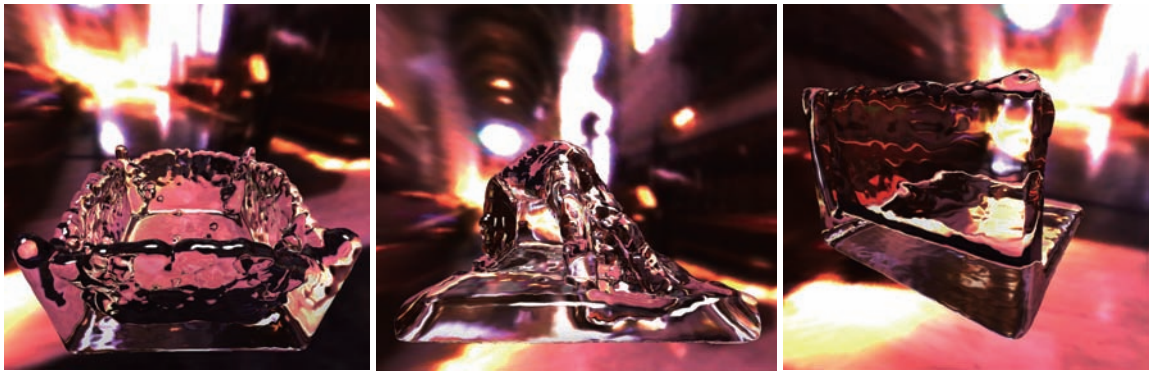

図 6 27000 の粒子と 64 × 64 × 64 のグリッドを用いたリアルタイムレンダリング& リアルタイムシミュレーション. シミュレーション 5 ステップ毎にレンダリングを 1 回行っており, 約 30fps で動作する (シミュレーション, 濃度場の構築, レンダリングなど全ての処理時間含む). 複数回の屈折を計算することによる”水らしい”表現に加えて, 少ない回数の屈折の計算では不可能な”水を通してみた水”という表現も右図に示すように可能になっている.

は避けられないが, 今回のように濃度球が最小でもボクセルの大きさを下回らないようにレイキャスティングの閾値を設定している場合は上限がレイキャスティングの閾値の 3〜5 倍以上であれば見てわかる劣化は生じない.

4.3 レイキャスティング

最後に構築した濃度場に対してレイキャスティングを行い, 水の可視化を行う. なお, 実装は Crane らの実装にならった[2]. レイキャスティングではまず SPH の計算領域を覆うボックスを描画し, 視線ベクトルとの交差点からボリュームの走査を行う. ボクセルベースであることの利点を活かし屈折または全反射を最大で 4 回まで計算することで, 複雑な屈折によるリアリティのある表現や”水を通して見た水”等の表現が可能になる.

5. 結果と考察

本手法は Core 2 Duo 3.0 GHz と Nvidia GeForce GTX 280 そして 2GB のメモリを載せた PC 上で, OpenGL・Nvidia Cg・Nvidia CUDA を用いて実装した. 図 6 は本手法を用いて実装したリアルタイムの流体のシミュレーションと可視化のものである.

表 1 は, 本手法を用いない場合と用いた場合の濃度場構築にかかる計算時間を比較したものである. 本手法を用いない場合というのは, 濃度場構築の処理を分割せずに全ての粒子が同じ濃度分布を持つとした場合を指す. 表 1 からもわかるように,SPH の計算に比べて非常に負荷の高い処理である濃度場の構築にかかる計算時間を大幅に短縮できている. 粒子数が増えるに従って効果が顕著になるが, これは本手法が水中の粒子 (周りに多数の粒子が存在する粒子) の処理の高速化に主眼を置いてた手法であるため, 計算領域が一定であると粒子が多いと水が”溜まる”ために水中の粒子が多くなることに起因する.

	8000	27000	64000	125000
Sim1	14533.0	33058.3	48941.9	72054.3
Sim2	7687.7	12788.7	16180.1	19993.7

表 1 解像度が 64 × 64 × 64 のボリュームを用いた場合のパフォーマンスの比較 : 本手法を用いない場合 (Sim1) と本手法を用いた場合 (Sim2)

本研究では SPH による流体シミュレーションの結果をボクセルベースのレイキャスティングで可視化するために必要な濃度場構築の処理を 2 つに分割し, さらに SPH のデータを用いることで高速化する手法を提案した. 本手法によって複雑な屈折を含む表現がリアルタイムに得ることができるようになるが, 今後の課題として残される問題が存在する. まず, 本手法はレンダリング画像の解像度に大きく依存することである. これはレイキャスティングによるボリュームの走査のためのテクスチャフェッチ回数が解像度に比例することに起因するが, これは Sun らによる GPU ベースの八分木の手法を用いることで改善することができる可能性がある[8]. また次に, 本手法では粒子をボクセルより小さい水滴としてレンダリングすることができない (ボクセルより小さな水滴はレンダリングされない場合があるため). より細かな水滴を表現するため考えうる手法としては, 疎な (密度が小さい) 粒子のために別の濃度場構築のためのより細かいボリュームを用意する, もしくはレンダリング時に SPH のグリッドを参照して疎な粒子は直接陰関数曲面を計算する,

などの手法が考えられる．

謝　　辞

This research was supported by Core Research for Evolution Science and Technology (CREST) of Japan Science and Technology Agency(JST).

参 考 文 献

1) Amada, T., Imura, M., Yasumuro, Y., Manabe, Y. and Chihara, K.: Particle-Based Fluid Simulation on GPU, *ACM Workshop on General-Purpose Computing on Graphics Processors and SIGGRAPH* (2004).
2) Crane, K., Llamas, I. and Tariq, S.: Real-time simulation and rendering of 3d fluids, *GPU Gems*, Vol.3, pp.633–674 (2007).
3) Harada, T., Koshizuka, S. and Kawaguchi, Y.: Smoothed particle hydrodynamics on GPUs, *Computer Graphics International*, pp. 63–70 (2007).
4) Iwasaki, K., Dobashi, Y., Yoshimoto, F. and Nishita, T.: Real-Time Rendering of Point Based Water Surfaces, *LECTURE NOTES IN COMPUTER SCIENCE*, Vol. 4035, p. 102 (2006).
5) Kanamori, Y., Szego, Z. and Nishita, T.: GPU-based Fast Ray Casting for a Large Number of Metaballs, *Computer Graphics Forum*, Vol. 27, No. 2, Blackwell Synergy, pp. 351–360 (2008).
6) Lorensen, W. and Cline, H.: Marching cubes: A high resolution 3D surface construction algorithm, *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, ACM New York, NY, USA, pp.163–169 (1987).
7) Monaghan, J.: Smoothed Particle Hydrodynamics, *Annual Reviews in Astronomy and Astrophysics*, Vol.30, No.1, pp.543–574 (1992).
8) Sun, X., Zhou, K., Stollnitz, E., Shi, J. and Guo, B.: Interactive relighting of dynamic refractive objects (2008).