Real-Time Rendering of Flow of Water Drops on a Windshield

Yuu Gotou Non-member (Meijo University, m0830012@ccmailg.meijo-u.ac.jp)
Tosimitu Tanaka Non-member (Meijo University, toshitnk@ccmfs.meijo-u.ac.jp)
Yuzi Sagawa Member (Meijo University, sagawa@ccmfs.meijo-u.ac.jp)

Keywords : Computer Graphics, Real-time rendering, Water drop, Meta Ball, GPU

1. Introduction

This research presents a method to display realistic flow of water drops on windshield in real time. The method selects one from two movement models by the distance from windshield wipers. Shape of each water drop is defined by a height map computed from 2D metaballs. The height maps are accumulated and then non-linearly compressed. Its details are shown below.

2. Switching of Movement Models

The mesh model segments surface of windshield to a grid. To each cell of the grid, a random value is given in order to simulate fluctuation of the surface. A water drop moves from a cell to its surrounding cell. The destination of the water drop is determined by evaluating its motion vector and the random values, in order to create irregular motion of the water drop. However, if windscreen wipers scoop many water drops in one cell, they fuse to one huge water drop.

On the other hand, the particle system is suitable to handle water scooped by windshield wipers. The particles push each other, and they never fuse. This nature is suitable to display overcrowded water drops appearing along windshield wipers. Thus we apply the particle system to the area around windshield wipers and apply the mesh model to the remaining area, as shown in Fig.1. The area for the particle model moves according to motion of the wiper.

When a rain drop hit the windshield, a water drop on the mesh model is placed at the position. When a water drop on the mesh model contacts the wiper, it is converted to water drops on the particle system. In opposition, when a water drop on the particle system drops out the area for particles, it is reconverted. However if other water drops appear around it, its reconversion is suspended till the drops move away.

3. Display Technique

Shape of each water drop is defined as one or more 2-dimensional metaballs, because its computing cost is much cheaper than the 3-dimensional metaballs. Its shape is preserved as a height map. The synthesized shape of all water drops on the entire windshield is defined by accumulating the height map for each water drop at the position of it. However, as shown in Fig.2, unnecessarily sharp peaks appear in the place where the drop of water is overcrowded.

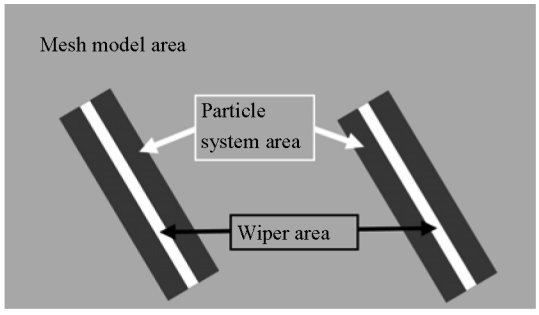


Fig. 1. Coverage of model

Our method makes the peaks smooth as follows. At first the threshold V_0 is set. Then, at each pixel of the height map, stored value V is converted to V_m by Eq. 1. Here k is selected in (0, 1).

$$V_m = \begin{cases} k(V - V_0) + V_0 & V > V_0 \\ V & V \le V_0 \end{cases} \quad \cdots\cdots (1)$$

This operation is several times repeated by increasing the threshold V_0. It is the approximation of no-linear filtering. This implementation is chosen to run the process on GPU.

Figure 3 was created by applying 5 times iteration of Eq.1 to the shapes shown in Fig.2, by increasing V_0 from 0.5 at intervals of 0.1. Since the peaks in Fig.3 are smooth, quality of the shapes is sufficient to simulate water drops in real world.

Then bump map is created from the height map. It is used for the refraction mapping. The rendering process is also executed on GPU. Figure 4(a) is created by our method and (b) was created by the previous method from 3D metaballs. Each images is a part of the original image and it is expanded. Our 2D metaballs are comparable to the 3D metaballs in image quality. And the quality is enough to visual simulation of water drops on windshield.

4. Results

We evaluated our method by using Pentium4 3.0Ghz CUP and Geforce 7600GS GPU. Animated images containing two wipers and approximately 3000 betaballs were constantly created in 30-40fps. Its image size was 1024 × 1024. The speed is sufficient in real-time rendering. In the images, flows of water drops that leave traces were displayed in realistic. Dollops of water pushed by wiper were also displayed. Quality of the images was sufficient for visual simulation. Since transfer between of the two models was very smooth, unnatural motion and shape did not appear.

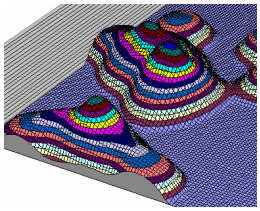


Fig. 2. Original height map.

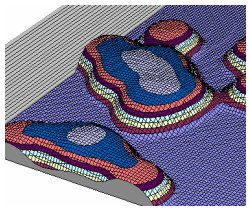


Fig. 3. Converted height map.

(a) By our method (b) By 3D metaballs

Fig. 4. Parts of generated images.

論　文

フロントガラスを流れる雨滴の実時間表示

非会員　後藤　　優*　　非会員　田中　敏光*
正　員　佐川　雄二*

Real-Time Rendering of Flow of Water Drops on a Windshield

Yuu Goto*, Non-member, Tosimitu Tanaka**, Non-member, Yuzi Sagawa**, Member

This research presents a method to display realistic flow of water drops on windshield in real time. The grid system well simulates irregular movements of the water drops on glass surfaces. Realistic shape of the water drops can be displayed by combination of metaballs. Fusion and separation of water drops can be expressed by gathering and distancing metaballs, respectively. However, if windscreen wipers scoop many metaballs in one cell of the mesh, they fuse to one huge water drop. It is very strange. On the other hand, the particle system is suitable to display thin dollop of water along windscreen wipers, because particles push each other. Thus, by the distance from the wipers, our method selects one from those two methods properly. Although shapes of the water drops are defined with 2D metaballs, quality of the shapes is comparable to 3D metaballs, both in nature and smoothness, because our method compresses height of the 2D metaballs non-linearly. Our method can manipulate and render water drops on windshield in real-time by using GPU.

キーワード：コンピュータグラフィックス，リアルタイムレンダリング，水滴，メタボール，GPU
Keywords：Computer Graphics, Real-time rendering, Water drop, Metaball, GPU

1. はじめに

自然現象をリアルに表現することは CG の重要な研究課題の一つである。窓ガラスの上を水が流れていく様子も，表現すべき自然現象の一つである。窓ガラスは，我々が利用する建物や乗り物のほとんどに使われているので，それを表示する必然性は高い。特に，ドライブシミュレータやトレインシミュレータ，カーレーシングゲーム，CG アニメーションなどで降雨時の景観を表示する[9][10]ときには，窓ガラスを流れる水滴のリアルな表現が不可欠となる。

ガラス面上の水は，その状態により様々な姿を見せる。少量の水が付着している場合には，半球状水滴としてガラスに留まる。しかし，やがて，他の水滴と融合したり，重力や風力によって尾を引いて流れ出したりする。電車や車のフロントガラスでは，水滴はワイパーで集められ，ガラス面から除去される。その際，水滴が押し合って形を変えたり，ワイパーの端からこぼれたりする。インタラクティブなシミュレータでは，このような複雑な水滴の振る舞いを実時間でアニメーション表示する必要がある。

* 名城大学理工学部
〒468-8502　愛知県名古屋市天白区塩釜口 1-501
Faculty of Science and Engineering, Meijo university
1-501 Shiogamaguchi, Tenpaku-ku, Nagoya 468-8502

2. 従来の研究

ガラス面上を流れる水滴のシミュレーションでは，水滴に働く重力，風力，摩擦力を考慮した運動方程式を解くことで，水滴の動きを決めている。さらに金田ら[1]は，ガラス面をメッシュ状に分割し，各区画に傷や汚れなどのガラス面の不均一性を示す情報を親和係数として設定した。そして，水滴の移動先を親和係数を使って確率的に決めることで，水滴が蛇行して流れる様子を再現した。これに続く研究[2]~[4]では，対象を曲面に広げ，風の影響を考慮するようにモデルが拡張されている。さらに海野ら[5]は，親和係数を動的に変えることで，水滴が一度通った場所は他の水滴が通りやすくなる様子を表示できるように，モデルを改良した。ただし，これらの手法では，水滴を 1 つの粒子として扱っているため，水滴の融合や分離，衝突時の形状変形などは表現できない。

益村[6]は，それまで半球形状で表示していた水滴を 2 次元メタボールで表現することで，水滴が尾を引いて流れる様子を表現した。静止している水滴同士の融合条件に制限を加え，水滴同士が触れていても一つに融合しない場合も想定したことで，ある程度複雑な形状の水滴を残すことにも成功している。しかし，静止状態の水滴の複雑な形状は表示できていない。

五十住ら[7]は，1 個の水滴を複数のメタボールで表現することで，ガラスに付着した水滴の複雑な形状や，融合時の形状の変化，水滴の尾の生成と消滅などを表現した。さらに，水滴形状を 3 次元メタボール（2 次元のポテンシャル関数）で定義し，リフレクション／リフラクションマッピングで周辺景色の映りこみを表現するように改良した[8]。この手法は，従来手法に比べるとはるかに自然な水滴を表示できるのだが，3 次元メタボールの計算に時間がかかるため，リアルタイム表示はできない。また，メッシュの 1 区画に入った水滴は全て融合するため，水滴の密集した場所には不自然に巨大な水滴ができてしまう。

水滴を実時間表示する研究としては，マッピングを利用する方法[11]やグラフィックハードウェアを利用する方法[12][13]などが提案されている。また，3 次元メタボールを GPU（Graphic Processing Unit）で実時間表示する手法[14]~[16]も提案されている。しかし，これらの手法では，水滴をパーティクルとして扱っており，融合分離は表現できていない。

3. 提案手法の概要

本研究では，五十住らの手法[7]を拡張し，ワイパーの作用を含めてフロントガラスを流れる水滴を表現する。雨滴がガラス面に接触した瞬間に，その場所に水滴が発生し，メッシュモデルに従ってガラス面を運動する。ただし，メッシュモデルでは水滴が衝突すると融合して 1 つの水滴となるため，ワイパーで水滴が集められると不自然に巨大な水滴ができてしまう。これを防ぐために，ワイパーの近傍に入った水滴を，一時的に均一の大きさの粒子に変換する。そしてワイパーや近隣粒子との相互作用を計算することで，水滴が他の水滴に押される様子や，水滴が集まって塊になる様子を表現する。

水滴の形状は処理の軽い 2 次元メタボール（2 次元のポテンシャル関数）で表現する。ただし，メタボールを高さ方向に圧縮することで，3 次元メタボールを使った手法[8]とほぼ同等の品質を確保する。品質の比較は〈5･6〉節に示す。これらの処理を GPU に実装することで，ガラス面を流れる水滴をリアルタイム表示する。

4. 水滴運動の処理

この章では，先行研究で使われていたメッシュモデルと本研究で導入する粒子モデルを解説し，その切り替え方法を示す。

〈4･1〉 メッシュモデル　ガラス面を等間隔のメッシュに区切り，メッシュ単位でガラス面を管理する。メッシュのセルのそれぞれに親和係数と呼ばれるガラス面の揺らぎを表す値を記録しておく。

個々の水滴は位置と質量のパラメータを持っている。水滴は 1 つずつ，メッシュを単位として移動する。水滴に働く重力，風力，摩擦力などの力を考慮した運動方程式を解いて，水滴の移動ベクトルを求め，この方向にあるセルとその両脇のセルの 3 つから，親和係数を考慮して移動先を確率的に選ぶ。Fig.1 にその様子を示す。この図では，灰色に塗ったセルが移動先の候補となる。水滴の移動距離が運動ベクトルから決まる 1 ステップ分の移動距離に満たない場合には，この処理を繰り返して必要なだけ水滴を進める。これにより，水滴の蛇行運動が表現できる。

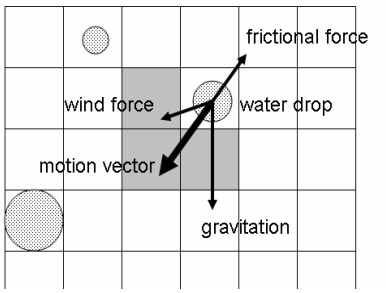


Fig. 1. mesh model.

水滴の質量は可変であるが，1 つのメッシュには 1 つの水滴しか存在を許されない。水滴がすでに存在するメッシュに移動した場合には，融合して質量が合算された 1 つの大きな水滴となる。

五十住らの研究[7]で示されたように，水滴の蛇行運動と融合により，水滴が細かい水滴を集めながら流れる様子がリアルに表示できる。また，水滴が移動した跡に小さな水滴を残しておくことで，尾を引いて流れる様子も表示できる。尾を消すには，尾の水滴を徐々に縮小し，一定時間後に消滅させればよい。

〈4･2〉 粒子モデル　水を均一の大きさの粒子の集合として表現する手法で，滝や川など動きのある水の表示に使われている。本研究ではガラス面上の水滴を対象とするので，2 次元のモデルを使う。この場合，水滴は半径が一定の円として表現される。

粒子モデルの特徴は，水粒子の反発を表現できる点にある。粒子の中心を結ぶ距離が半径の 2 倍以下であれば，衝突したと判定する。衝突した場合には，それぞれの粒子の中心を，中心を結ぶ線上を互いに遠ざかる方向に，重なった長さの半分だけ移動する。これにより，半径の重なりを解消する。加えて，ペナルティ法の考えの基づき，この移動距離に一定の係数をかけた距離だけさらに水滴を遠ざけることで，水滴の反発を表現する。

粒子と物体との衝突処理も必要になる。本研究で対象となる移動物体はワイパーだけである。ワイパーはブレードでガラス面と接しているが，その形状は細長い長方形で近似できる。そこで，長方形の移動方向を向いた長辺と粒子の中心との距離が，半径より小さくなったとき，衝突したと判定する。衝突した場合には，球の中心を，辺の鉛直方向に距離が半径となる位置，つまり，水滴とワイパーが接する位置まで移動する。

〈4･3〉 モデルの適用領域　雨の日のフロントガラスを見ると，ワイパーから離れた場所では，水滴が離散的にガラス面に付着している。このため，水滴単体の運動や融合･分離の表現が必要となる。〈4･1〉節で述べたメッシュモ

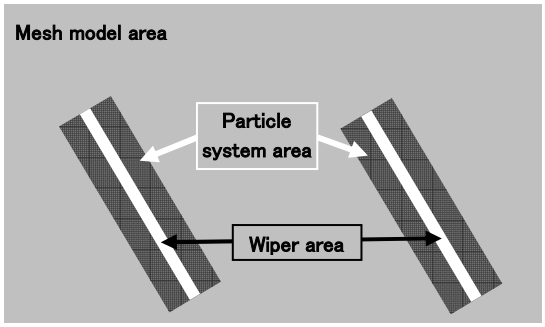


Fig. 2. Coverage of model.

デルを使うと，このような水滴の変化をリアルに表示できる。

しかし，ワイパー周辺では，雨滴が集められて一塊になる。このとき，メッシュモデルを使うと，水滴が融合して不自然に巨大になるが，粒子モデルでは，水滴は周囲の水滴から力を受けて移動し，ガラス面に薄く広がる。粒子モデルの水滴は大きさが一定なので，ぶつかっても大きな水滴ができることは無い。

このように，メッシュモデルは離散配置された水滴の表現に適しており，粒子モデルは密集した水滴の表現に適している。ガラス面上ではワイパーで集めることで水滴が密集するため，ワイパー周囲では粒子モデルを，そのほかはメッシュモデルを適用する。

Fig.2 の白色の線はワイパーを示している。その周囲の色の濃い部分が粒子モデルの領域である。その長辺は，ワイパーの長さと等しく，短辺は，ワイパーの側面における粒子の重なりを考慮して，左右それぞれ粒子 4 個分の幅を持たせている。それ以外の灰色の領域ではメッシュモデルが適用される。

メッシュモデルと粒子モデルは独立に処理されるので，メッシュモデルの水滴と粒子モデルの水滴が同じ位置を占めても，それらの間には融合処理も反発処理も行われない。ただし，表示では，どちらのモデルの水滴もメタボールに変換され，まとめてレンダリングされるので，位置が重なると，そこには大きな水滴が表示される。

〈4･4〉 粒子モデルへの変換 メッシュモデルの水滴がワイパーと衝突したときに，その水滴を粒子モデルに変換する。Fig.3 では，白い矩形領域がワイパーで，グレーの円がメッシュモデルの水滴，黒の円が粒子モデルの水滴を示している。

Fig.3 の左側はワイパーが水滴に接触する直前の様子を，右側は接触直後の様子を表している。水滴がワイパーに触れたとき，メッシュモデルの水滴が粒子モデルの水滴に変換される。ただし，メッシュモデルの水滴は質量が可変なので，その質量を粒子モデルの水滴の質量で割った数だけ粒子を生成する。Fig.3 では 3 個生成されている。生成される粒子が 1 個の場合には，メッシュモデルの水滴と同じ位置に置くが，複数生成される場合には，それぞれの粒子を，基準の位置から，ランダムに少しずらして配置する。この

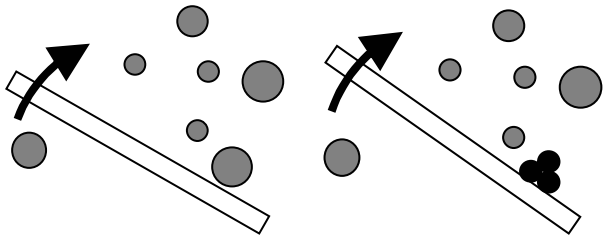


Fig. 3. Conversion from the mesh model to the particle model.

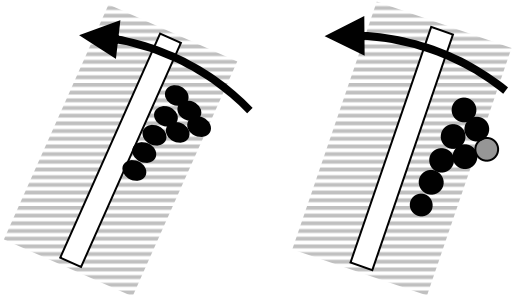


Fig. 4. Conversion from the particle model to the mesh model.

時，生成した粒子やすでにある粒子と重なることがあるが，粒子間の反発処理が働いて移動するため，次のステップでは重なりの無い状態になる。

〈4･5〉 メッシュモデルへの変換 粒子が〈4･3〉節に示した粒子モデルの適用領域を外れたとき，粒子モデルの水滴をメッシュモデルの水滴に変換する。Fig.4 はワイパーが折り返した後の様子を示している。この図の白い矩形領域はワイパーで，その周辺のグレーの斜線の範囲が粒子モデル領域，黒い円が粒子モデルの水滴，グレーの円がメッシュモデルの水滴である。

Fig.4 の左側が変換前の状態で，右側が変換後の状態である。範囲から外れた粒子モデルの水滴が，同じ大きさのメッシュモデルの水滴に変換されている。ワイパーが水滴を集めると，ワイパーの進行方向に水滴の層ができる。このため，メッシュモデルへの変換は，ワイパーからある程度離れたときに行う。

〈4･6〉 変換の保留 粒子モデルからメッシュモデルに変換される際，ワイパーの折り返し地点では，一度に多くの水滴が粒子モデルの領域から外れる。このとき，これらの水滴を全てメッシュモデルに変換すると，同一のセルに多数の水滴が入るため，融合して巨大な水滴ができてしまう。これを防ぐために，条件を設定して変換を保留する。

まず変換先のセルを中心とした正方領域内のセルを探し，閾値以上の大きさの水滴が存在するなら変換を保留する。水滴が小さいうちは，摩擦力によりガラス面にとどまるが，大きくなると重力が静止摩擦力を上回り，流れ始める。そこで，大きさの閾値を水滴が単独で流れ始める条件以上にして，変換が保留されても，その原因がすぐに解消されるようにする。また，この大きさの水滴があっても融合しない距離に探索範囲を選ぶことで，変換直後に大きな水滴ができてしまうことを防いでいる。このため，閾値と

探索範囲はガラス面の摩擦係数や撥水率などの値により変化するが，後の実験で示す通常のフロントガラスの条件では，大きさの閾値は半径 2.2mm となる。実験では 1 セルの間隔を 2mm に選んでいるので，水滴が接しないようにするには中心間を 4 セル離す必要がある。このため，探索範囲を 7×7 セルに選んでいる。

〈4･7〉　水滴の形状表現　メッシュモデルの水滴は，単独で存在しているため，複雑な形の表現が必要となる。そこで，五十住らの手法[7]を使って，1 つの水滴の形状を複数のメタボールで定義する。この手法では，水滴の位置には，主たる形状を表現するため，水滴の大きさに比例したメタボール（メインメタボール）が置かれる。その周囲には，形状を複雑にするための小さなメタボールが数個置かれる。これをサブメタボールと呼ぶ。サブメタボールはメインメタボールからの相対位置で定義される。サブメタボールの位置や数は水滴の状態により変化する。たとえば，ガラス面に衝突した直後には，周囲に 3 個のサブメタボールが置かれ，一定時間かけてメインメタボールの位置まで移動し，メインメタボールに融合する。一方，粒子モデルの水滴は常に集まっているので，個々の形状を考慮する必要は無い。このため，1 個の粒子を 1 個のメインメタボールで表現する。本研究では，メインメタボールもサブメタボールも，2 次元のメタボールとして定義する。

5.　水滴のレンダリング

この章では，前章で求めたメタボールの位置と大きさから，リアルな水滴の画像を実時間で表示する手法を述べる。

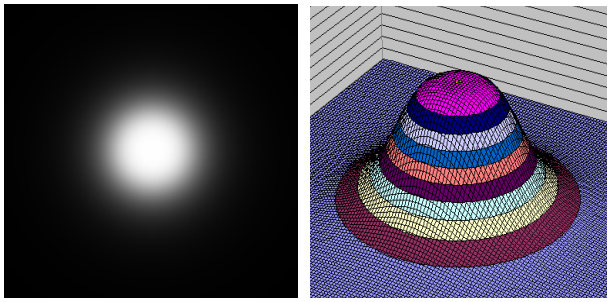
〈5･1〉　2 次元メタボール　3 次元のメタボールによる形状表示手法[14]~[16]も提案されているが，強力な GPU を専用に用意する必要があるので，実用的ではない。そこで本研究では，水滴の形状定義に，処理コストの低い 2 次元のメタボールを使う。2 次元メタボールでは，水滴の凹凸はポテンシャル関数で決まるので，半球形状に近い連続関数を使う必要がある。そこで，本研究では(1)式で 1 つの水滴を定義する。

$$N = 1 \Big/ \left(1 + (h/R)^4\right) \quad \cdots\cdots (1)$$

この式の R は水滴の半径，h は水滴の中心位置からの距離，N が求める濃度となる。

Fig.5(a)は，(1)式の半径を 20，中心座標を(64,64)として求めた濃度を 128*128pixel のハイトマップテクスチャとして出力したもので，Fig.5(b)はそれを 3 次元表示したものである。このハイトマップから，値が閾値を越える部分だけを取り出すと，水滴 1 つの形状となる。Fig.6 は閾値を 0.5 として表示した形状である。

〈5･2〉　ガラス面上に分布する水滴の表現　ガラス面の大きさのハイトマップを用意し，前述した形状表現手法で定まった位置に，単一水滴のハイトマップを水滴の大きさに合わせて拡大縮小した後，加算合成する（Fig.7）。その後，ハイトマップより値が閾値以上の部分を取り出して，



(a) Height map　　(b) 3D view

Fig. 5.　High map made created from the density function.

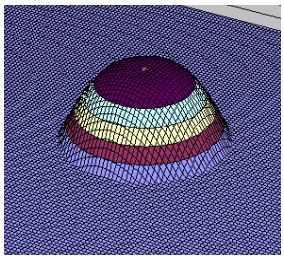


Fig. 6.　Shape of one drop of water.

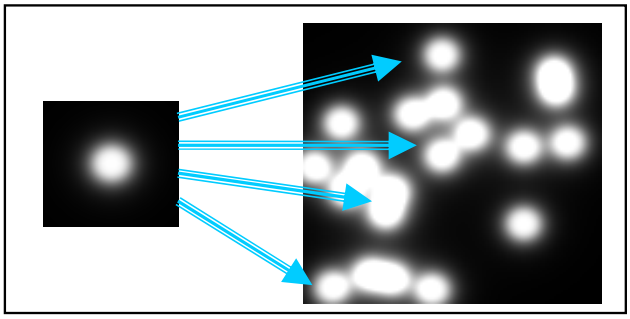


Fig. 7.　Additive synthesis of water drops.

ガラス面上に分布する水滴の形状とする。

〈5･3〉　ハイトマップの補正　前節で作成したハイトマップを立体表示すると，Fig.8 となる。一部で水滴が不自然に高くなっているが，これは，水滴が集まった場所にハイトマップが何回も加算されたためである。しかし，これでは水滴が尖って表示されてしまう。また，法線の傾きも大きすぎるため，実際には起こりえない反射や屈折が表示される。

そこで本研究では，ハイトマップの値が大きい部位分を圧縮することで，高さの変化を滑らかに補正する。具体的には，基準の高さ V_o を決め，1 画素ごとにハイトマップの値を調べる。そして，高さ V を次の(2)式で高さ V_m に置き換える。ただし，補正係数 k は 0～1 の間で決める。

$$V_m = \begin{cases} k(V - V_0) + V_0 & V > V_0 \\ V & V \le V_0 \end{cases} \quad \cdots\cdots (2)$$

1 回の処理では高さを十分に圧縮できないので，同じ処理を高さの基準を変えて何回か行う。Fig.9 は，Fig.8 のハイトマップを V_o=0.5，k=0.3 から V_o を 0.1 ずつ増加しながら 5

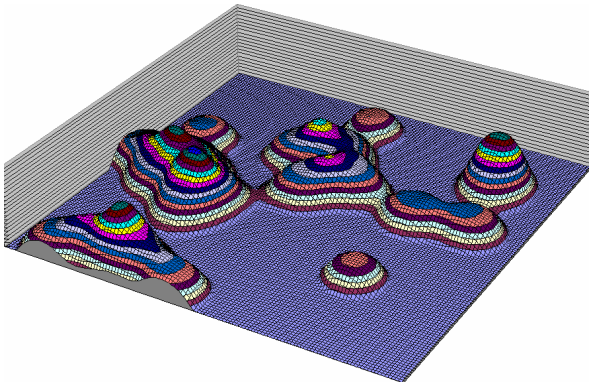


Fig. 8.　Original shape of fused water drops.

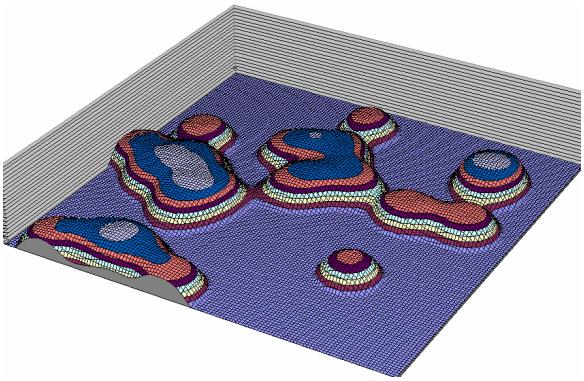


Fig. 9.　Corrected shape of Fig.8.

回補正した結果である。Fig.9 と 8 を比較すると，尖ったところがなくなり，全体に滑らかな形状になっている。このような繰り返し法を用いるのは，後述するように GPU を活用するためである。GPU では，単純な四則演算は高速に行えるが，複雑な計算はできない。そこで，閾値処理と単純な線形操作を繰り返すことで，非線形圧縮を近似している。

〈5･4〉　環境マッピングによる反射と屈折の表現

ガラス面上に付着した水滴には，周囲の風景が歪んで映り込む。これはガラス面に入る光線が水滴によって反射・屈折して起こる現象である。そこで，反射と屈折を環境マッピングで表示する。水滴の凹凸は，視点からガラス面までの距離に比べて十分に小さいので，法線マップで表現する。マップに保存する各点の法線方向は，ハイトマップの値を上下，左右に差分を取って接ベクトルを求め，その外積から計算する。

フロントガラスの水滴に映りこむのは，ほぼ正面の景色に限られるので，背景画像として正面画像 1 枚だけを用いる。もしはみ出した場合には，折り返して参照する。屈折は空気とガラスの境界でも生じるが，影響が小さいので無視する。

本来は Snell の法則で正確な屈折方向を計算すべきであるが，処理コストが大きいので簡易手法を使う。本研究の画像生成条件では，視線はガラス面にほぼ垂直に入射するので，法線ベクトルのうち，ガラス面と平行な X 成分と Y 成分を使う。Fig.10 左側に示すように，視線と交差するガラス

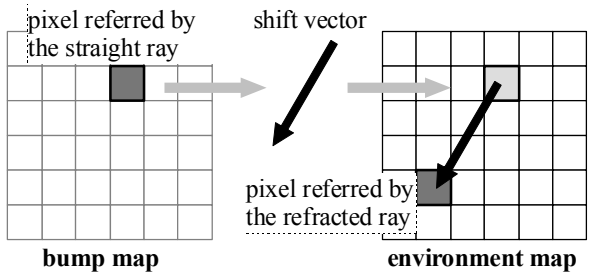


Fig. 10.　Simple refraction mapping with a bump map.

面の座標で法線マップを参照し，その位置の法線の X,Y 成分を求める。次に，その値に画像サイズと画角で決まる係数を掛けて，移動ベクトルを計算する，最後に，その値だけずれた位置の背景画像を参照する。

〈5･5〉　GPU を用いた提案手法の実装　　これまで述べてきた手法を，リアルタイムで処理する為に，GPU を利用する。GPU では複雑な処理は行えないが，並列演算により CPU の数十倍のパフォーマンスを出すことが出来る。最近では Microsoft が提供する DirectX に含まれる HLSL を使うことで，GPU でも C 言語と同じように，ループ処理やフロー制御などを使ったプログラミングが出来るようになった。本研究ではこの HLSL と DirectX の基本的な機能を使い，提案手法をリアルタイム処理する。以下に具体的な手法を述べる。各処理の結果は，処理ごとにバッファに保存し，マルチパスレンダリングを行う。

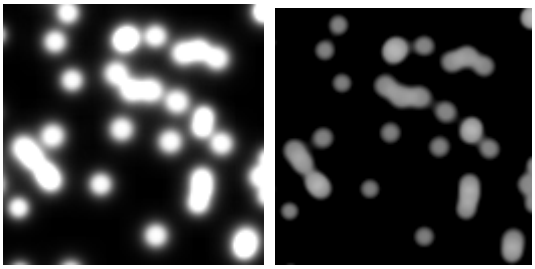
（1）　ハイトマップの加算合成　　あらかじめ Fig.5 左のような一枚のテクスチャとして作成しておいたハイトマップを読み込んで使うことで，濃度マップの計算時間を省略する。ガラス面への加算合成にはポイントスプライトを用いる。メッシュモデルの水滴の大きさは変化するので，基本の大きさの水滴のテクスチャを，水滴の大きさに合わせてバイリニアフィルタで拡大／縮小してから，加算合成する。

（2）　ハイトマップの補正　　HLSL によりフロー制御を行い，基準の高さ以上のピクセルに対して〈5･3〉節に述べた(2)式を適用する。この処理を，基準の高さを上げならが，指定回数だけ繰り返す。

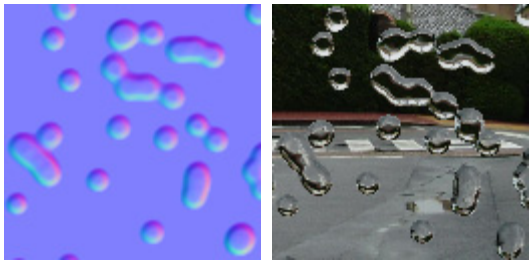
（3）　(2)で求めたハイトマップの各画素で上下・左右の差分を計算して接ベクトルを求める。HLSL の関数を使ってその外積を計算し，法線マップとしてテクスチャバッファに保存する。

（4）　環境マッピング　　ジャギーを抑えるために，前処理として背景画像を 64*64pixel まで縮小し，参照用テクスチャとして保存する。これを〈5･4〉節の方法で参照する。

〈5･6〉　レンダリング結果と従来手法との比較　　一連の処理を GPU で実行した結果を Fig.11 に示す。Fig.11(a)は加算合成したハイトマップで，そこから閾値以上の部分を取り出して高さを補正した画像が(b)，(b)から作成したバンプマップが(c)である。最終的には，Fig.11(d)が表示される。比較のため，〈5･3〉節に示した補正処理を行わない場合の出力結果を Fig.11(e)に示す。補正処理を行わないと水滴がと



(a) Height map (b) Corrected height map of (a)



(c) Bump map (d) Generated image

(e) Created without the correction (f) Created by using 3D metaballs

Fig. 11. Result and comparison of the rendering.

がった形状になる。比較すると，Fig.11(d)のほうが自然な写りこみになっていることが分かる。

Fig.11(f)は五十住の手法[8]で作成した，3 次元メタボールで表現された水滴である。ただし，水滴の配置は異なる。この画像では，個々の水滴を三次元ポテンシャル関数で定義し，合成された表面を反復法により高い精度で計算している。また，反射・屈折方向も正しく計算している。物理的には，こちらのほうが正確な画像だが，提案手法で作成した Fig.11(d)と比べて，水滴の形や映り込みに大きな差は見られない。従って，提案手法は十分な表示品質を備えているといえる。

6. 結果の評価

これまでに提案した運動モデルと表示手法を統合し，映像を作成した。その数コマを Fig.12 に示す。わかりやすいように，ワイパーは矩形ポリゴンで半透明表示している。

Fig.12(a)はワイパーが折り返した直後の画像である。ワイパーに集められた水滴がその右側に集まっている。(b)はそれから少したった後で，集められた水滴の一部が流れている。ワイパーから離れるとメッシュモデルに代わるため，水滴が尾を引いて流れるようになる。しかし，一部は変換を抑制され，粒子モデルのままとどまっている。さらに時間がたつと，多くの水滴は流れ落ちるが，尾の一部は再び水滴となってガラス面上に残る。この画像には，ガラス面

(a) Windshield wiper returned on right edge immediately before

(b) Some seconds are passed after (a)

(c) Additional seconds are passed

(d) The wiper reached to the left edge

Fig. 12. Display of windshield.

に付着する水滴，尾を引いて流れる様子，水滴同士の滑らかな融合，背景の写り込み，ワイパー周辺の水の集まり，などがリアルに再現されている。

Fig.12 に示した映像は，Windows XP，Pentium4 3.0GHz，Geforce7600GS の環境で作成した。出力解像度は 1024*1024，表示されているメタボールの数はおよそ 3000 個である。この場合のフレームレートは 30~40fps の間を推移しており，旧世代の実行環境でも良好な結果が残せた。ためしに運動モデルを統合せず表示したところ，60fps 以上出た。運動モデルはすべて CPU で処理され，サブメタボールの制御などの複雑な処理を含むため，時間がかかるものと推察される。

7. むすび

本研究では，車のフロントガラスを流れる雨滴をリアルに表現するための手法を提案した。この手法は，メッシュモデルと粒子モデルをワイパーからの距離で切り替えることで，蛇行しながら尾を引いて流れ落ちる水滴と，ワイパーで集められて塊になった水滴の両方をリアルに表示することができる。

水滴の表示では，2 次元メタボールを高さ方向に圧縮補正する手法で，3 次元メタボール並みの品質で表示できるようにした。メタボールの補正や，環境マッピングによる表示には GPU を用いるため，実時間表示が可能になった。

今後は，ガラス面を曲面に拡張する。また，運動モデルに車の加減速や走行に伴う風力を加えて，より現実に忠実なモデルにする。さらに，ドライブシュミュレータ等に組み込み，映像を評価することも，今後の課題である。

（平成 21 年 4 月 2 日受付，平成 21 年 8 月 14 日再受付）

文　献

(1) K. Kaneda, T. Kagawa, and H. Yamashita : “Animation of Water Droplets on a Glass Plate”, Proc. Computer Animation'93 Models and Techniques in Computer Animation, Springer-Verlag, Tokyo, pp.77-189 (1993)

(2) 寿山康彦・金田和文・山下英生 :「風の影響を考慮したガラス表面を流れる水滴のアニメーション」，グラフィクスと CAD シンポジウム (1995)

(3) K. Kaneda, Y. Zuyama, H. Yamashita, and T. Nishita : “Animation of Water Droplet Flow on Curved Surfaces”, Proceedings of the Fourth Pacific Conference on Graphics and Applications, pp.50-65 (1996)

(4) K. Kaneda, S. Ikeda, and H. Yamashita : “Animation of Water Droplets Moving Down a Surface”, J. Visual. Comput. Animat. 10, pp.15-26 (1999)

(5) 海野和也 :「ガラス面を流れる水滴のシミュレーション　～可変親和係数を用いた動きの制御～」，電学東海支部若手セミナー（2002 年度第 2 回）予稿集 (2003)

(6) 益村明人 :「透明物体の上を流れる雨粒のシミュレーション」，名古屋大学 工学部 情報工学科 卒業論文 (2000)

(7) T. Isozumi, T. Tanaka, Y. Sagawa, and N. Sugie : “Animation of Deformation of Water Drops on Glass Surfaces”, The journal of the Institute of Image Information and Television Engineers 2006, pp.538-544 (2006) (in Japanese)
五十住拓哉・田中敏光・佐川雄二・杉江　昇 :「ガラス面上を流れる水滴の形状変化の表現」，像情報メディア学会論文誌，Vol.160, No.4, pp.538-544 (2006)

(8) T. Isozumi, T. Tanaka, and Y. Sagawa : “CG Animation of Waterdrops that Flow on Glass Surfaces : Improvement of Representation for Shape”, Proceedings of the IEICE General Conference 2007, p.163 (2007) (in Japanese)
五十住拓哉・田中敏光・佐川雄二 :「ガラス面上を流れる水滴の CG 表現－形状表現の改良－」，信学全大 D-12-47 (2007-3)

(9) 宮沢圭介・藤本忠博・村岡一信・千葉則茂 :「降雨景観のビジュアルシミュレーション」，第 13 回情処学東北支部研 (2001-12)

(10) 間渕友彰・菊池　司・岡崎　章 :「画角追従 LOD-Plane による降雨シーンのビジュアルシミュレーション－CG による降雨シーンの効率的な生成法の開発－」，2005 年度 画像電子学会 第 33 回年次大会 (2005-6)

(11) T. Sato, Y. Dobashi, and T. Yamamoto : “A Method for Real-Time Rendering of Water Droplets Using Texture Mapping”, Proceedings of the Society Conference of IEICE 2001, p.204 (2001-8) (in Japanese)
佐藤倫也・土橋宜典・山本　強 :「テクスチャマッピングを利用した水滴のリアルタイムレンダリング」，信学会 2001 年 情報・システムソサイエティ大会講演論文集，p.204 (2001-8)

(12) T. Sato, Y. Dobashi, and T. Yamamoto : “A Method for Real-Time Rendering of Water Droplet Using Graphics Hardware”, Proceedings of the IEICE General Conference 2001, p.275 (2001-3) (in Japanese)
佐藤倫也・土橋宜典・山本　強 :「グラフィックスハードウェアを利用した水滴のリアルタイム表示手法の開発」，2001 年信学会総合大会講演論文集，p.275 (2001-3)

(13) T. Sato, Y. Dobashi, and T. Yamamoto : “A Method for Real-Time Rendering of Water Droplets Taking into Account Interactive Depth of Field Effects”, Proc. IWEC 2002, pp.110-117 (2002)

(14) Y. Kanamori and T. Nishita : “Fast Rendering of Metaballs on GPUs”, IPSJ SIG Notes 2007, pp.13-18 (2007) (in Japanese)
金森由博・西田友是 :「GPU を用いたメタボールの高速レンダリング」，情処研報 CG-127(3), pp.13-18 (2007)

(15) C. Müller et al. : “Image-Space GPU Metaballs for Time-Dependent Particle Data Sets”, Proc. VMV '07, pp.31-40 (2007)

(16) H. Mishima, T. Tanaka, and Y. Sagawa : “Real-time rendering of metaballs using GPU”, Journal of Information Processing, Vol.49, No.12, pp.4080-4087 (2008-12) (in Japanese)
三嶋　仁・田中敏光・佐川雄二 :「GPU を用いたメタボールの実時間描画」，情処学論, Vol.49, No.12. pp.4080-4087 (2008-12)

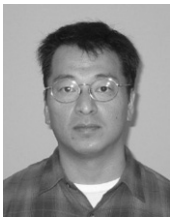
後　藤　　優

（非会員） 1985 年 7 月 19 日生。2008 年 3 月名城大学大学理工学部情報工学科卒業。現在，名城大学理工学研究科情報工学専攻修士課程に在籍。

田　中　敏　光

（非会員） 1959 年生。1982 年，名古屋大学工学部電子工学科卒業。1984 年，同大学大学院情報工学専攻博士（前期）修了。NTT に勤務。1994 年，名古屋大学大型計算機センター助教授。2000 年 4 月より，名城大学理工学部情報科学科（現情報工学科）教授。CG の研究・教育に従事。工学博士。

佐　川　雄　二



（正員） 1963 年生。1987 年，名古屋大学大学院情報工学専攻前期過程修了。1992 年，同大学大学院情報工学専攻後期過程単位取得退学。同年同大助手，同講師を経て，2000 年，名城大学理工学部講師，2008 年，同教授。現在に至る。自然言語処理の研究・教育に従事。工学博士。