人体動作に連動した 3D 水流ビジュアルエフェクトの生成

佐々木 智弘[1,a)]　床井 浩平[2]

概要：本研究では，ユーザーの身体動作に連動し，水滴を飛散させる 3D ビジュアルエフェクトの生成を行った．本研究の目的は，水を人体の自由な位置から発生させる，水滴の物理的な挙動を管理するといった現実にはない表現手法の開発を行うことである．ユーザーの動きを MotionCapture で取得し，取得したデータに合わせて水滴に任意の直径や重力加速度の影響などの振る舞いを制御するパラメータを付与し，飛散させるというものである．水は metaball で実装し，描画には MarchingCubes を利用し，見た目は投影マッピングにより表現する．対話的な速度で生成可能なため，ゲームや VFX を用いたステージ演出への利用が期待できる．

1. はじめに

コンピュータグラフィックスにおいて，自然現象のシミュレーションやアルゴリズムの改良に関する研究は盛んに行われている．これらの研究の多くは，より正確な計算結果を求めたり，より高速な処理を可能にし，災害の対策など技術的分野で貢献している．また，3DCG はゲームやアニメなど，エンターテイメントの分野にも非常によく使用され，近年では特に，アーティストのステージ演出や VR・AR などのコンテンツでも利用されている．

エンターテイメント分野では物理的挙動の正確性より，ユーザー（＝視聴者）の作品から受ける印象や体験が重要であるため，これらのステージ演出や，一般的なゲームやアニメなどの映像作品には現実的な表現に誇張表現で演出を付加された，非現実な表現が用いられることが多い．

これらの表現の中で，VFX などの近年になってから一般的になり始めた演出では，まだ確立されている表現手法があまり見られない．製作者の個人による独自の手法が多いため，一般化されていないことが多い．このことから，VFX 用いたステージ演出の利用などに期待できるビジュアルエフェクトを対話的な速度で生成できる手法を開発することは有用であると考えられる． 本研究では，物理シミュレーション分野で多くの研究対象になっている流体のうち，水を対象に 3D ビジュアルエフェクトを生成する手法の開発を行った．本研究では開発環境として OS:Windows10Pro 64bit，CPU : Intel (R) Core (TM) i5-7500 CPU@3.40Ghz 3.40GHz, RAM:16GB, GPU : NVIDEA GeForce GTX1070 の構成の PC 上で Unity[*1]を使用した．

[1] 和歌山大学大学院システム工学研究科
[2] 和歌山大学システム工学部
[a)] sasaki.tomohiro@g.wakayama-u.jp

2. 水の生成

水の表現では，物理シミュレーションに使われる現実の挙動を正確に再現するものから，自由自在に コントロール可能であるようなゲームやアニメなどで使われるものまで様々である．本研究では水を構成する要素として，形状，色（透明度），粘度に分解した．本節では，形状と色（透明度）について述べる．

2.1 形状の生成

メタボール (Metaball) という，水のような多次元有機的モデルの滑らかな曲面を再現する手法を用いて水の形状を生成する．この手法は，正確な形状は生成できないが，軽量であるため，多くの粒子を用いた流体のシミュレーションや，人体などの曲線が主なモデリングなどに使われている．

3 次元空間上のある点に置かれたモデルに，中心点から距離が大きくなるにつれて値が小さくなる密度分布を与え，密度分布を持った複数のモデルを互いに近づけ，それぞれの持っている密度場を足し合わせた密度場を得る．この新しく得られた密度場がある閾値と等しくなる位置に合わせてモデルの形の表面を描画することにより，滑らかな曲線で繋がったモデルを得る（図 1）．

2.2 メッシュの生成

2.1 節で述べた手法により水の形状を生成した後，表面の形状のメッシュを生成する．水の動きを動的に再現する

[*1] Unity Technologies によって開発されたゲームエンジン

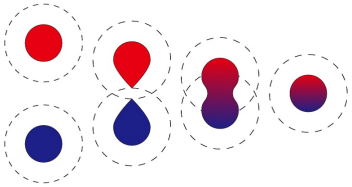

図 1　メタボールの曲面生成

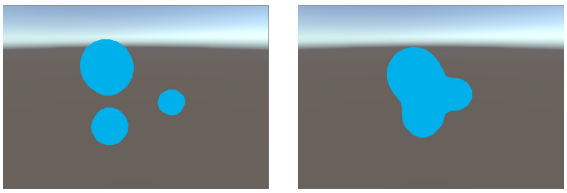

図 2　メタボールによる水の生成

ため，メッシュの生成は高速に行う必要がある．そのため，3 次元空間を分割し，格子内のスカラーデータで構成されるボクセルデータの等値面を近似し，ポリゴンデータに変換し，近似を用いてメッシュを生成するマーチングキューブ法を用いる．

3 次元空間を分割し，格子内のスカラーデータで構成されるボクセ ルデータの等値面をポリゴンデータに変換しレンダリングする手法である．各格子点が持ってい るスカラ値と等値面のスカラ値を比較し，スカラ値の大小を求め大きいほうを 1，小さいほうを 0 とする．1 辺の両端の格子点にそれぞれ 0 と 1 が与えられているとき，その辺に等値面を通過させ，1 辺の両端が共に 0 もしくは 1 のとき，等値面は通過させない．この等値面を通過させた点を つなぎ 3 角形メッシュの生成を行う．3 角形メッシュの生成パターンは 8 つの各頂点が 0 もしくは 1 なので $28 = 256$ 通りであるが，回転を行った際に同形であるものを除き，また表裏を問わない場合は 15 通り（図 3）になる．この 15 通りの 3 角形メッシュでモデルの近似したメッシュを生成する．

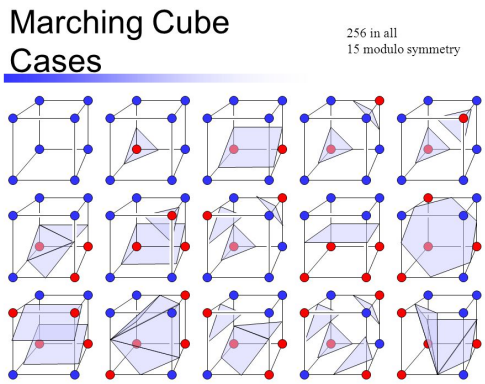

図 3　文献 [8]:近似の 15 パターン

2.3　見た目の決定

形状以外の要素では，色（透明度）が水と認識するうえで重要な要素である．水の持つ特性の光の屈折を用いることにより，一層「水らしさ」を持つモデルの生成が可能である．本節では「水らしさ」を，正確な物理的挙動を考慮しない，見た目だけを表すものと定義する．

2.3.1　投影マッピング

水による光の屈折を表現するため，投影マッピング（プロジェクションマッピング）を Unity Shader を用いて実装した．視点からのベクトルが 3.1，3.2 節で生成した水を通り，背面にあるオブジェクトに到達した場所の画素を取得し，取得した画素をテクスチャとする．テクスチャを水の表面に貼り付けることで，背面オブジェクトの色を持った見た目の水を生成する．

2.3.2　屈折マッピング

2.3.1 節で取得したテクスチャを水に適用する際に，テクスチャを水の形状に合わせて変形させることにより水による光の屈折を表現する．視線ベクトルと正規化した水の頂点（面）の法線ベクトルの内積を求める．視線ベクトルと面が直交していれば内積は 1 を返し，平行であれば 0 を返す．1 から内積の絶対値を引いた値により，透明度をする．視線と面のなす角度が大きいほど不透明になり，陰影ができ水の輪郭が強調される．また，テクスチャに水の面の法線ベクトルで補正し，歪ませることで疑似的に光の屈折を再現した．

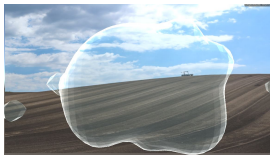

図 4　屈折マッピング無し

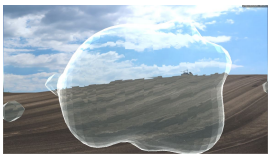

図 5　屈折マッピング有り

3.　身体動作の取得

身体動作の取得に Kinect を使用する．Kinect から取得した人体の Depth（深度）データに基づき，人体のボーンデータを作成する．ボーンデータでは Unity 内の座標系の座標を取得できるので，Unity 内の座標データにより動きを判断する．Kinect では身体動作の取得漏れが発生するので，取得漏れに対しては直前の座標を代用する．

4.　水の動作

取得したモーションに合わせて水を飛散させるため，水の生成時に外力を与える．身体部位の移動した前後の座標の差を微分し，単位時間当たりの加速度を求め，運動方程式を用いて外力を決定した．

$$F = m\frac{\Delta p}{\Delta t}$$

5. 実行例

2 節で述べた手法を用いて水を生成した．メタボールで複数の水滴から曲面を生成したのち，マーチングキューブ法を用いて表面のメッシュを作成したモデル描画した（図 6）．曲面の滑らかさの表現に対し，近似する格子の大きさが大きく円の頂点にあたる角が目立つが，動的にレンダリングするので FPS[*2]が高い場合の動作では，実用範囲内の見た目であった．

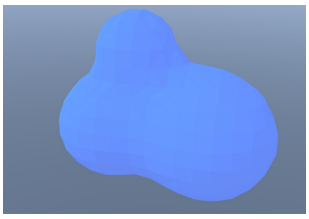

図 6 メタボールと MC 法を用いて水を生成

次に，生成した水に 2.3 節で述べた投影マッピングと屈折マッピングを適用し，見た目の設定を行った．2.3.2 節で述べた手法を用いて実装した屈折マッピングのうち，輪郭部分の陰影付けを行っていない図 7 と陰影付けを行った図 8 を比較したところ，図 8 がより水のように見えるという評価になった．

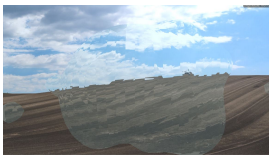

図 7 輪郭の強調無し

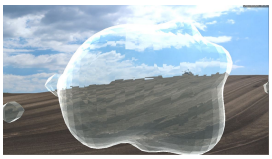

図 8 輪郭の強調有り

最後に 3 節で述べた手法で身体動作を取得し，身体部位から 4 節に則り水を生成した（図 9）．この例では身体のうち，手の位置から水を生成している．

図 9 手から水を生成

[*2] frames per second：単位時間あたりに処理するフレーム数

6. まとめ

本研究では，ユーザーの身体動作に基づいて，水滴が飛散するビジュアルエフェクトを生成する手法の開発を行った．その結果，ユーザーの身体動作に連動して水滴を生成・飛散させる，3D ビジュアルエフェクトを生成することができた．しかし，マーチングキューブ法の格子空間の粒度を細かくすると処理が大幅に重くなり，対話的速度で実行できない問題がある．

また，表現手法としての拡張性の面において，炎や砂塵などの他の流体への応用も行っていく必要がある．

参考文献

[1] Bourke，Paul．”Polygonising a scalar field．”(1994).

[2] Lorensen，William E.，and Harvey E. Cline. ”Marching cubes: A high resolution 3D surface construction algorithm．” ACM siggraph computer graphics. Vol. 21. No. 4. ACM，1987.

[3] 林健一，笠晃一．”パラメータ変化による陰関数曲面の 3 次元 CG アニメーション．” 電気関係学会九州支部連合大会講演論文集 平成 21 年度電気関係学会九州支部連合大会 (第 62 回連合大会) 講演論文集． 電気・情報関係学会九州支部連合大会委員会， 2009

[4] 金森由博，” レイトレーシングによるコンピュータグラフィクス入門 -メタボール-” 2014

[5] 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．” 情報処理学会研究報告グラフィクスと CAD (CG) 2007．70 (2007-CG-127) (2007)：13-18.

[6] 中村 薫，杉浦 司，高田 智広，上田 智章：KINECT for Windows SDK プログラミング Kinect for Windows v2 センサー対応版，`https://www.buildinsider.net/small/bookkinectv2`. (2018 年 2 月 12 日閲覧)

[7] KristoferSchlachter(kristofe):UnityGPUMarchingCubes，`https://github.com/kristofe/UnityGPUMarchingCubes`. (2017 年 12 月 16 日閲覧)

[8] John C. Hart : Volume and Solid Modeling，`http://slideplayer.com/slide/9271343/`. (2018 年 2 月 12 日閲覧)

[9] 新谷幹夫， et al. ”映像表現およびコンピュータグラフィックスの研究動向．” 映像情報メディア学会誌 70.1 (2016)：149-152.

[10] MetaBalls：`http://wiki.unity3d.com/index.php/MetaBalls` (2018 年 2 月 2 日閲覧)

[11] 西祐貴， and 床井浩平．”インタラクティブに落雷アニメーションを生成するビジュアルシミュレーション手法の開発．” 研究報告エンタテインメントコンピューティング (EC) 2014．67 (2014)：1-6.