グラフィクスとCAD 92−3
（1998. 10. 15）

水面下の光の模様の高速レンダリング手法

徳山 哲朗† 山本 強†† 高井 昌彰††
†北海道大学大学院 工学研究科
††北海道大学大型計算機センター

本研究が最終的に目指している事は、破砕を伴う水の挙動をモデル化し、その様子を3次元CGによってリアルタイムにレンダリングを行う事である。この目的を達成する事により、仮想現実空間内での水流の挙動や、波の破砕を含むCG作成システム、エンターテイメントソフトでの利用を狙っている。
また、これらのシステムを実行する環境としては、広く普及しているプラットホームを利用したいと考えている。そのため、本研究では並列機などの特殊で高価な機器の利用を前提とするのではなく、一般に普及しているPC上でこれらのシステムを動作させようとしている。
そのような事から、水の挙動やレンダリングに関して厳密な物理シミュレーションを行うのではなく、前述した適用分野に用いた場合に視覚的に満足できるものを作成する事を目指している。
本論文では、利用している水流の2次元の場合のモデリング手法について簡単に説明を行った後、3次元化を行った場合に必要になるレンダリング技法を解説する。具体的には、ハイトフィールド (Height Field) モデル上の水によって発生する屈折現象を、テクスチャマッピングを利用したポリゴンレンダリングによって行う手法を提案する。

A rapid rendering method for generating light illuminations on the ground in the water.

Teturo Tokuyama† Tsuyoshi Yamamoto†† Yoshiaki Takai††
†Graduate school of Engineering, Hokkaido University
††Hokkaido University Computing Center

The purpose of our research is both modeling water motion including breaking phenomenon and rendering its appearance by 3D computer graphics in the real time. We want to apply the result of our research to the following three areas;virtual realities, CG systems for rendering wave breaking, and the entertainment softwares. Furthermore, We try to run these systems in widespread environment, I mean, not in special hardwares such as parallel computers but in general PC.
In order to achive the things mentioned above, we focus on not physical accuracy but visible satisfaction. In this paper, we explain our model of water and propose a rapid rendering method using polygon based rendering techniques with texture mapping for refraction phenomena which lead to both distorted image of ground in water and complex light illuminations.

1　はじめに

本研究では、リアルタイム性を追求し、物理的な正確さを求めるのではなく、視覚的に満足できる水流のモデルの構築とレンダリング手法の提案を行う事を目的としている。本研究は、水流のモデリング部とレンダリング部の２つの部分に分ける事が出来る。

水流のモデリングと関連のある従来の研究として、海岸に打ち寄せる波のシミュレーションを行ったものがある [3][4]。これは、波の発生源を複数個設定し、その波源から任意の地点までの距離と地面の形状等から位相を変化させる事で波のモデリングを行う手法である。この手法では、三角関数を用いてモデリングを行っているため、水の追加・削除と言った操作が不可能である。

２Ｄのナビエ・ストークスの方程式を解き圧力を算出して速度を求め、水を移動を行うモデリング手法がある [5]。このシステムでは、サイズにもよるがリアルタイムにシミュレーションが可能である。しかし、水の深さを考慮していない点や、水の移動の手法が困難な点が挙げられる。

本研究では、文献 [2] のハイトフィールドモデルをベースとし、水の追加・削除、水の速度の考慮などが出来るように変更を行ったモデルを利用している。また、ハイトフィールドモデルで表現不可能な部分につてはパーティクルモデルを用いるようにしている。

レンダリングに関しては、レイ・トレーシングを利用する研究が多いが、リアルタイム性を追求したいと考えているので、本研究ではポリゴンベースの手法を用いるようにしている。

本論文では、利用しているハイトフィールドモデルについて述べた後、水面で発生する屈折現象のレンダリング技法について解説する。

2　利用している水流のモデル

本研究では、ハイトフィールドモデルとパーティクルモデルを併用したモデリング手法を用いている。

ハイトフィールドモデルは地面上を流れる水に利用している。パーティクルモデルは空中を舞う水しぶきに利用する。この２つのモデルの間で水の量を行き来させることで水の挙動をモデル化している。今回は、ハイトフィールドモデル上の水面のレンダリング技法について述べるので、本論文ではパーティクルモデルや、２つのモデル間での水の変換アルゴリズムなどについては言及しない。

2.1　ハイトフィールドモデル

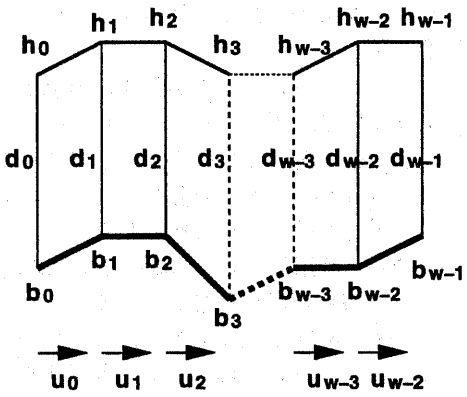

Fig.1 ハイトフィールドモデル

ハイトフィールドモデルは図に示してあるように、水面の高さ h、水の水平方向の速度 u、地面の高さ b をデータとして保持している。ここで、速度 u は格子点上の水の速度ではなく格子点間の水の速度として扱っている。

基本的には参考文献 [2] で述べられているアルゴリズムを利用するが、ハイトフィールドモデル上で水の追加･削除を行ったり、水の速度の変更を行わせたいので、若干の変更を加え、次の式を用いてハイトフィールドモデル上の水流の制御を行うようにしている。

$$\frac{\partial h}{\partial t} + d\frac{\partial u}{\partial x} = Ha$$

$$\frac{\partial u}{\partial t} + g\frac{\partial h}{\partial x} = Ua$$

ここで、gは重力加速度、dは水量を表しており、$d = h - b$ である。Ha と Ua は、ハイトフィー

ルドの外部から与えられる、単位時間当りの水量と水平方向の加速度を表している。

これらを離散化して解いて行くわけだが、単純なオイラー法などでは、タイムステップを十分小さくしなければ安定した解を得られない。リアルタイム性を追求したいので、ある程度タイムステップをを大きくとっても安定した結果を得たい。そのために、文献 [2] に述べられているように、陰的な解法を用いると、最終的に、水面の高さに関しては、

$$\begin{aligned} Ah_i(n) &= 2h_i(n-1) - h_i(n-2) \\ &\quad +\Delta t(Ha_i(n-1) - Ha_i(n-2)) \\ &\quad +\frac{\Delta t^2}{\Delta x}\left(\frac{d_{i-1}+d_i}{2}\right)Ua_{i-1}(n-1) \\ &\quad -\frac{\Delta t^2}{\Delta x}\left(\frac{d_i+d_{i+1})}{2}\right)Ua_i(n-1) \end{aligned}$$

速度に関しては、

$$\begin{aligned} Bu_i(n) &= 2u_i(n-1) - u_i(n-2) \\ &\quad +\Delta t(Ua_i(n-1) - Ua_i(n-2)) \\ &\quad +g\frac{\Delta t^2}{\Delta x}Ha_i(n-1) \\ &\quad -g\frac{\Delta t^2}{\Delta x}Ha_{i+1}(n-1) \end{aligned}$$

を解くことになる。但し、n はステップ数である。また、wをハイトフィールドの格子点の個数とすると、

$$A = \begin{pmatrix} e_0 & f_0 & & & & & \\ f_0 & e_1 & f_1 & & & & \\ & f_1 & e_2 & f_3 & & & \\ & & & \cdots & & & \\ & & & & e_{w-3} & f_{w-3} & \\ & & & & f_{w-3} & e_{w-2} & f_{w-2} \\ & & & & & f_{w-2} & e_{w-1} \end{pmatrix}$$

$$B = \begin{pmatrix} j_0 & f_1 & & & & & \\ f_0 & j_1 & f_2 & & & & \\ & f_1 & j_2 & f_3 & & & \\ & & & \cdots & & & \\ & & & & f_{w-4} & j_{w-3} & f_{w-2} \\ & & & & & f_{w-3} & j_{w-2} \end{pmatrix}$$

$$\begin{aligned} e_0 &= 1+g(\Delta t)^2\left(\frac{d_0+d_1}{2(\Delta x)^2}\right) \\ e_k &= 1+g(\Delta t)^2\left(\frac{d_{k-1}+2d_k+d_{k+1}}{2(\Delta x)^2}\right) \\ e_{w-1} &= 1+g(\Delta t)^2\left(\frac{d_{w-2}+d_{w-1}}{2(\Delta x)^2}\right) \\ j_k &= 1+g(\Delta t)^2\left(\frac{d_k+d_{k+1}}{(\Delta x)^2}\right) \\ f_k &= -g(\Delta t)^2\left(\frac{d_k+d_{k+1}}{2(\Delta x)^2}\right) \\ & \quad (1 \le k \le w-2) \end{aligned}$$

である。

2.2 3次元化について

本研究では、3次元空間内のハイトフィールドは右手座標系のXZ平面に存在しており、Y軸方向が高さ方向になるように設定している。3次元空間内のハイトフィールドの場合には考慮しなければならない次元が1つ増えるため、上記のマトリックスが大きいものになってしまい、計算コストが増大する。そこで、先ほどの手法を単純にX軸方向と、Z軸方向に独立に直接適用する事によって、計算コストを省く手法がある。

ただし、この手法では速度の算出が正しく行われない。しかし、水の挙動そのものには不自然さは現われない。そのため、水の挙動のみであれば、つまり、ハイトフィールド上の速度を計算する必要がなければ、この手法は高速化には有効である。今回はこの手法によって水の移動を行った。

3 レンダリング技法

現実世界において、水面では屈折現象が発生する。そのため、観測者からは、水面下の地面が歪んで見える。また、水面に入射する光も屈折されるため水面下の地面に複雑な光の模様が発生する。これは、光の屈折により、水面下の地面に届く光の分布に粗密が生じるためである。それゆえ、光が多く集まる部分は明るくなり、光が粗になる部分は暗くなるという現象が生じる。

ここでは、ハイトフィールドモデル上の水面における屈折現象によって生じる光の模様や水面下の画像が歪む現象を、テクスチャマッピングを利用したポリゴンを用いるレンダリング技法について解説する。

3.1 水面下の光の模様のレンダリング

水面下の地面に発生する光の様子を、物理的に正確にレンダリングを行うためには、水面下の地面の1点に対して水面全体からの光の入射を考慮する必要がある。これは、水面での光の屈折により、水面へ入射した光が不均一になるために、地面上で光の粗密が生じることや、入射場所が水面上で離れていても地面上の同じ場所に光が集中するすることによって地面の明るさが変化するためである。

本研究では以下のように、光源側から光を入射させる手法を利用する。まず、ハイトフィールドモデルから水面を構成するポリゴンを作成する。次に、そのポリゴンの各頂点に、光源からの光を入射させ、光の屈折方向を求める。屈折した方向にハイトフィールドを探索して行き、屈折した光と地面との交点を算出する。そして、交点からなるポリゴンを生成する。交点では地面の法線を求め、輝度を次式によって決定する。

$$(\overrightarrow{L_{refracted}} \cdot \overrightarrow{N_{ground}}) \left\{ \frac{S}{S'} (\overrightarrow{L} \cdot \overrightarrow{N_{polygon}}) \right\}$$

ここで、$\overrightarrow{L_{refracted}}$は水面で屈折された光線ベクトル、$\overrightarrow{N_{ground}}$は地面と屈折した光線ベクトルとの交点おける地面の法線ベクトル、Sは水面のポリゴンの面積、S'は地面に生成されたポリゴンの面積、$\overrightarrow{L}$は水面のポリゴンに入り込む光線ベクトル、$\overrightarrow{N_{polygon}}$は水面のポリゴンの法線である。

式の前半は、地面に届く光の拡散反射による光の強度を表しており、後半部分は、地面上のポリゴン全体に届く光の強さを表している。

レンダリングを行う際には、ポリゴンの色は白とし、先ほど交点で算出した輝度値をアルファ値としてポリゴン内で補間しながらアルファブレンディングを行い、輝度値をフレームバッファに加算して行く。ブレンディングに用いた式は以下の通りである。αはアルファ値、C_{light}は光源の色を、$C_{framebuffer}$はフレームバッファ上の色を表している。

$$C_{framebuffer} = \alpha \times C_{light} + C_{framebuffer}$$

このようにすることで、近似的に光の粗密を実現することができ、複数の場所からの光の入射による重ね合わせも実現できる。

ここで生成した画像の上に、地面用のテクスチャをレンダリングする。その際に次式によって陰影を付ける。

$$C_{framebuffer} = C_{texture} \times C_{framebuffer}$$

ここで生成された画像を、次に説明する水面のレンダリングの際にテクスチャとして利用する。

3.2 水面のレンダリング

水面のレンダリングの際にも、水面下の光の模様の生成と同様な手法を利用する。ただし、入射するのは光源からの光ではなく、視点からのレイを入射する事になる。

水面に入射したレイを屈折させ地面との交点を求める。先ほどは交点で輝度値を算出したが、ここでは交点の座標をテクスチャ座標に変換する。

水面をレンダリングする際には、ここで求めたテクスチャ座標を利用して、水面下の光の模様の生成の際に作成したテクスチャをポリゴンに張り付ける事によってレンダリングを行う。

また、水面では反射も起こるので、周囲の景色をテクスチャで用意し、環境マッピングを施す事で水面への景色の写り込みを実現することができる。ここでは環境マッピングは、視線の反射ベクトルのX成分とZ成分をテクスチャ座標として用いてレンダリングを行った。

3.3 ハイトフィールド上の水面のポリゴンの生成法

前述した手法を利用するためには、ハイトフィールド上の水面を表現するポリゴンを生成す

る必要がある。本研究では、ボリュームレンダリングの際に用いられるマーチングキューブ法と同様な手法を利用する。マーチングキューブ法では、閾値以上のボクセルと、それ以下のボクセルの間にポリゴンを生成することで等値面を形成する。ここでは、水が存在する格子点と存在しない格子点の間にポリゴンを生成して、水面を形成する。

そのために、ハイトフィールド上を4つの格子点で構成される四角形の領域ごとに見て行く。四角形の領域のエッジの一方の頂点に水が存在している場合、エッジ上で頂点を次のようにして求める。地面の高さは場所によって異なっているので、エッジの傾きによって、下図のような4つの状態にわけて考える。

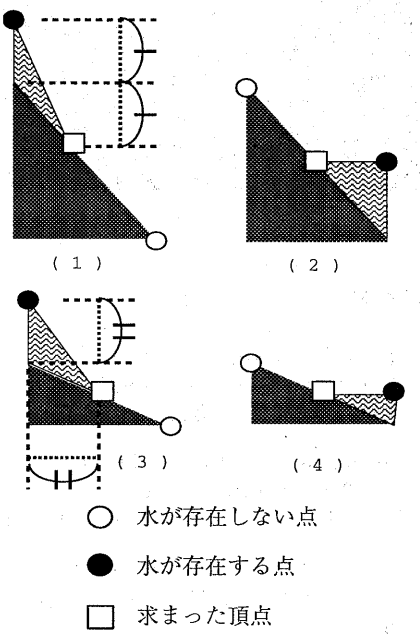

Fig.2 ポリゴンの頂点の求め方

(1) の様に、地面の傾きが1より大きい場合で、高い方に水が存在している場合は、水の高さと等しい分だけ低い位置に頂点を生成する。(3) の様に、地面の傾きが1以下で高い方に水が存在している場合には、水の高さと等しい分だけ、水平方向に移動した場所に頂点を作成する。(2)(4) の様に、地面の低い側に水が存在している場合は、水と同じ高さの部分にポリゴンを生成するようにしている。

この手法で問題になるのは、着目している四角形の内部で対角線上に水が存在している場合、ポリゴンの張り方としては下図のように2通り存在する事である。

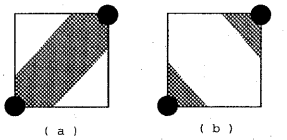

Fig.3 対角線上に水が存在する場合

どちらを用いるかを決めるために、対角線上でそれぞれの頂点を求め、頂点の位置が互いに交わる場合には上図の (a) を、それ以外は (b) のようにポリゴンを張るようにした。

4 結果

実験環境は PentiumII(400Mhz)、メモリ 128MB、OS は Windows98 、3D アクセラレータにはカノープスの PURE3DII(Voodoo2) を利用した。また、座標変換は自前のライブラリーで行いポリゴンなどの描画には Glide を利用した。スクリーンサイズは 800×600 で、実質的なレンダリング領域は 470×380 程度であった。

ハイトフィールドの大きさ	FPS
16×16	21.74
32×32	12.82
64×64	4.10
128×128	1.05
256×256	0.28

この計測時間には、水の移動のための計算時間が含まれていない。しかし、今回利用した水の移動アルゴリズムの場合であれば、水の移動時間を考慮しても大きな差は見られなかった。処理時間が、ハイトフィールドの面積にほぼ比例している事がわかる。これは、ハイトフィールド上でレイと地面の交点を求めるための探索範囲が広がるためである。そのため、水の深さによってもレンダリング時間は変化する。ハイトフィールドのサイズが 128×128 の場合の画像を本論文の最後に掲載しておく。

5 おわりに

本論文で、ハイトフィールド上の水面による屈折現象をテクスチャマッピングを用いた手法で高速にレンダリングする事が可能な手法を提案した。また、サイズにもよるがリアルタイムにレンダリングする事も可能である。

今回は、水の深さによる光の減衰などは考慮しなかったが実装することは容易であろう。また、一次屈折しか考慮していないので、水面に入ったレイが再び水面から出て行く場合に不都合が生じる問題点が存在する。

今後の課題としては、現在水面下の陰影をポリゴン単位で行っているため、ハイトフィールドのサイズが小さい場合に陰影の解像度が荒らくなってしまう問題が存在する。

また、本研究ではパーティクルモデルも利用するので今回提案した手法に加えて、パーティクルのレンダリングのための手法も作成しなければならない。

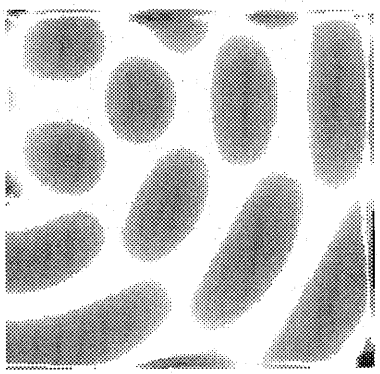

Fig.4 水面下の光の様子

Fig.5 陰影づけられたテクスチャ

Fig.6 屈折の様子

参考文献

[1] 徳山哲朗、山本強、「水流のモデリングとレンダリングに関する研究」情報処理北海道シンポジウム '97 講演論文集 pp.37-39

[2] Michael Kass and Gavin Miller "Rapid, Stable Fluid Dynamics for Computer Graphics" Proc. SIGGRAPH'90(August 1990), pp.49-57

[3] Darwyn R. Peachey "Modeling Waves and Surf" Proc.SIGGRAPH'86 (August 1986), pp.65-74

[4] Alain Fournier "A Simple Model of Ocean Waves" Proc.SIGGRAPH'86 (August 1986), pp.75-84

[5] Jim X.Chen, Niels Da Vitoria Lobo, Charles E.Ughes, and J.Michael Moshell 'Real-Time Fluid Simulation in a Dynamic Virtual Environment" IEEE Computer Graphics and Applications, Vol.17, No.3, May-June 1997 pp.52-61