2006年度　卒　業　論　文

ボリュームレンダリングによる
エネルギー波の表現手法

指導教員：渡辺 大地講師

メディア学部 ゲームサイエンス
学籍番号　M0103196
坂井　悠基

2006 年度　　卒　業　論　文　概　要

論文題目

ボリュームレンダリングによる
エネルギー波の表現手法

メディア学部 **学籍番号：M0103196**	**氏名**	坂井　悠基	**指導教員**	渡辺 大地講師

キーワード	ボリュームレンダリング、レイキャスティング法、テクスチャベース法 CG エフェクト

近年、アニメーションやビデオゲームの分野において 3 次元コンピュータグラフィクスを利用した数多くのコンテンツが開発されており、これらの中にはエネルギーの塊のような物体が強く発光する表現がよく使われる。このような現象は現実には存在しないが、似たような特徴を持つ身近なものにレーザー光がある。本稿では、アニメーションやビデオゲームなどのコンテンツで使われるレーザー光のような表現を総称してエネルギー波と呼ぶこととする。エネルギー波の定義は「空間中に存在するエネルギーの密度が高い部分が強く発光する現象」とし、エネルギーとは空間中に存在するエネルギー波を構成する要素で、3 次元空間上に分布し強い熱量を持っているものとする。3 次元のビデオゲームの分野では、このようなエネルギー波は 2 次元のテクスチャとして表現する方法が一般的である。視点の変更に応じてテクスチャのマッピングされたポリゴンを回転したり、テクスチャを切り替えたりすることでエネルギー波を表現している。しかし、この方法は 1 枚の画像としてエネルギー波を表現するため、視点が変わるとエネルギー波の光の強さを正確に表現することができなくなってしまう。

そこで本稿では、エネルギー波を表現する新しいアプローチとしてボリュームレンダリングを利用したエネルギー波の表現方法を提案する。テクスチャで表現されたエネルギー波と、同じ視点から見たボリュームレンダリングによるエネルギー波の 2 つの画像を比較することで、エネルギー波の光の強さを正確に表現できているかを検証する。

目 次

図 目 次

第 1 章

はじめに

1.1　研究の背景と目的

アニメーションやビデオゲームなどのコンテンツにおいて、エネルギーの塊のような物体が強く発光する表現がよく使われる。代表的な例として少年漫画「ドラゴンボール [1]」に出てくる気功波や、「機動戦士ガンダム [2]」に出てくるビーム系の武器、ビデオゲームに出てくる魔法のエフェクトなどがある。このような現象は現実には存在しないが、似たような特徴を持つ身近なものにレーザー光がある。レーザー光とは光の束の一種であり、自然には存在しない人工的な光である。その大きな特徴は単色性や指向性に優れ、強い熱を持つことである。このようなレーザー光の特徴がアニメーションやゲームなどのコンテンツによって形を変えたものが「ドラゴンボール」に出てくる気功波であり、「機動戦士ガンダム」に出てくるビーム系の武器と考えることができる。本稿ではこのような表現を総称してエネルギー波と呼ぶことにする。エネルギー波の定義は「空間中に存在するエネルギーの密度が高い部分が強く発光する現象」とし、エネルギーとは空間中に存在するエネルギー波を構成する要素で、3 次元空間上に分布し強い熱量を持っているものとする。

3 次元のビデオゲームの分野では、このようなエネルギー波は 2 次元のテクスチャとして表現する方法が一般的である。視点の変更に応じてテクスチャのマッピングされたポリゴンを回転したり、テクスチャを切り替えたりすることでエネ

ルギー波を表現している。しかし、この方法は1枚の画像としてエネルギー波を表現するため、視点が変わるとエネルギー波の光の強さを正確に表現することができなくなってしまう。

そこで本研究では、エネルギー波を表現する新しいアプローチとして、ボリュームレンダリングを利用したエネルギー波の表現方法を提案する。ボリュームレンダリングとは3次元空間中に分布するデータを可視化する手法を指し、一般的には煙や霧のように微粒子の集合として定義された物体や、内部構造を持った物体を正確にレンダリングするために用いられる。視点の変更に応じてボリュームレンダリングを行うことで、常に正確なエネルギー波の光の強さを表現することができる。なお、本稿ではエネルギー波から発生する光が他の物体に及ぼす影響は考慮していない。

1.2 本論文の構成

本論分は全6章で構成する。第2章で既存のエネルギー波の表現方法について述べ、第3章でボリュームレンダリングの手法について述べる。第4章でボリュームレンダリングによるエネルギー波の表現方法を述べ、第5章でその実行結果とエネルギー波の光の強さを正確に表現できているかの検証を行う。最後に第6章でまとめと今後の展望について述べる。

第 2 章

既存のエネルギー波の表現方法

本稿で対象とするエネルギー波の概要と既存の表現方法について述べる。

2.1　エネルギー波

本稿ではエネルギー波を「空間中に存在するエネルギーの密度が高い部分が強く発光する現象」と定義する。エネルギーとは空間中に存在する、エネルギー波を構成する要素で、3 次元空間上に分布し強い熱量を持っているものとする。

エネルギー波が世間に広まったきっかけは、松本零士原作の「宇宙戦艦ヤマト[3]」や、鳥山明原作の「ドラゴンボール」などのアニメーション作品だと考えることができる。特に「ドラゴンボール」は、漫画やアニメーション、ビデオゲームなど多くの分野で世界的な人気を誇っており、作中に出てくる「かめはめ波」や「元気玉」などの気功波は、エネルギー波の中で最も多くの人に馴染みがあるものと考えることができる。また、「ドラゴンボール」以外にも、エネルギー波はたくさんのアニメーションやゲームで使われている。

2.2　ビデオゲームにおけるエネルギー波の表現

3次元のビデオゲームでは、エネルギー波はテクスチャとして表現されるのが一般的である。視点の変更に応じてテクスチャのマッピングされたポリゴンを回転したり、テクスチャを切り替えたりすることでエネルギー波を表現する。

ビデオゲームでエネルギー波を表現する方法を説明する。まずは何種類かの視点から見た画像が描かれたテクスチャを準備する。図2.1は3種類の視点から見たテクスチャの例である。左の画像はエネルギー波を左側から見た画像であり、同様にエネルギー波を真ん中から見た画像と右側から見た画像を1つのテクスチャとして準備しておく。

図2.1: テクスチャの準備

次に、視点とテクスチャのマッピングされたポリゴンが垂直になるようにこれらを配置する。視点の位置が変わった場合、テクスチャのマッピングされたポリゴンを視線と垂直になるように回転する。図2.2は視点の位置に応じてポリゴンを回転している例である。視線とポリゴンのなす角度を常に垂直にすることで、エネルギー波が1枚の画像で表現されていることをわかりづらくする。

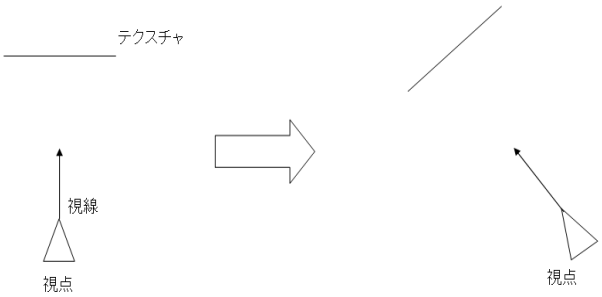


図 2.2: 視点の変更に応じたポリゴンの回転

さらに、視点の位置によってはポリゴンにマッピングするテクスチャを他の視点から見たテクスチャに切り替える。図 2.3 は 3 つの視点からエネルギー波を見た時のテクスチャ切り替えの例である。

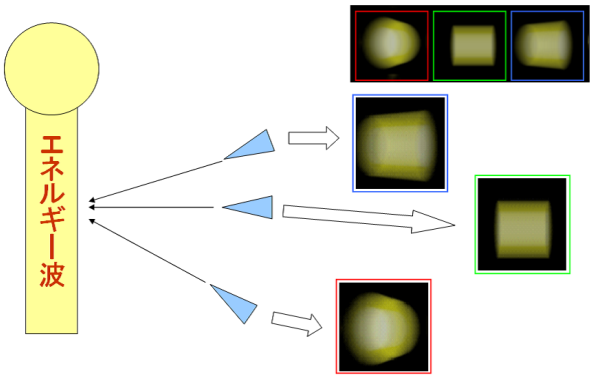


図 2.3: 視点の位置に応じたテクスチャの切り替え

このように 3 次元のビデオゲームでは、エネルギー波を 1 枚の絵として表示し、視点が変わった場合はポリゴンの回転とあらかじめ用意した絵を切り替えることでエネルギー波を表現している。しかしこの方法では、正確にエネルギー波の光の強さを表現しているのはあらかじめ絵を準備した 3 箇所の位置に視点がある時のみであり、それ以外の位置ではエネルギー波の光の強さを正確に表現することができない。

そこで本研究では、どのような視点から見ても正確にエネルギー波の光の強さ

を表現することを目的とする。そのための新しいアプローチとして、本稿ではレイキャスティング法とテクスチャベース法と呼ばれる 2 つのボリュームレンダリングの手法を利用する。

第 3 章

ボリュームレンダリングの手法

本稿で対象とするボリュームレンダリングの概要について述べる。

3.1 ボリュームレンダリング

ボリュームレンダリングとは 3 次元空間中に分布するデータを可視化する手法を指し、一般的には煙や霧のように微粒子の集合として定義された物体や、内部構造を持った物体を正確にレンダリングするために用いられる [4]。ボリュームレンダリングにはいくつかの手法があり、その手法によって描画速度や得られる画像の画質に大きな違いがある。また、ボリュームレンダリングを行うことに特化したツールやライブラリ、ハードウェアも多く存在する [5][6][7]。

本来、ボリュームレンダリングはボリュームレンダリング方程式を解くことで行う。ボリュームレンダリング方程式とは、光がある領域内を微少距離進む間に視点方向に出て行く光の強さがどう変化するかが記述されているものである。しかし、このようなボリュームレンダリングは高精度な画像を得られる反面、計算量が膨大である [8]。そこで現在主流となっている手法は、領域全体をボクセルに分割してボリュームレンダリングを行うというものである。ボクセルとは 3 次元座標上の点からサンプリングされたもので、ボクセルの集合体がボリュームである。ボリュームは密度、温度、電荷などの観測値や計算値を持たせることにより、空

間全体をデータとして保持する。このようなボリュームデータを使うことで、レンダリング方程式を解く方法よりも大幅に計算量を減らすことができる。ボクセルを用いたボリュームレンダリングの代表的な手法として「レイキャスティング法」が有名である。

3.1.1 レイキャスティング法

ボリュームレンダリングの一般的な方法で、視点からピクセルに視線を飛ばすことからレイキャスティング法と呼ばれる。視線上にあるボクセルの輝度を足し合わせるという処理を、視線上のボクセルがなくなるまで繰り返してピクセル値を求める手法である。図 3.1 は視線上のボクセルを足し合わせた結果をスクリーンのピクセルに表示している例である。

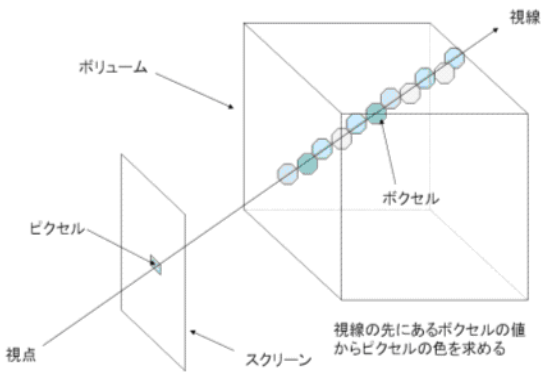


図 3.1: レイキャスティング法

レイキャスティング法はレンダリング方程式を解く方法と比べると計算時間が大幅に短縮されるため、リアルタイムボリュームレンダリングには一般的にこの方法が採用されている。

3.1.2　テクスチャベース法

近年、GPU の性能の向上により、その機能をうまく使った「テクスチャベース法」[9][10][11][12] と呼ばれる手法が提案されている。テクスチャベース法は GPU を使用したボリュームレンダリングにおける代表的なアルゴリズムであり、レイキャスティング法の考えを発展し処理の一部を GPU で行うことで高速な処理を可能にする方法である。テクスチャベース法ではボリュームはテクスチャとして GPU のビデオメモリに読み込まれる。このテクスチャをスライスと呼ばれるポリゴンの平面にマッピングし、複数のスライスによってボリュームを再構成する。図 3.2 はこれらの工程の図解である。

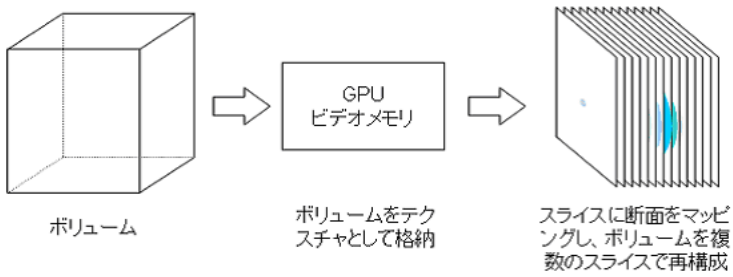


図 3.2: GPU への読み込みとボリュームの再構成

次に、ボリュームをスライスの重ね合わせによって再構成し、視点から遠いスライスから順に透明度を考慮して色を加算し最終的な画像を得る。図 3.3 は各スライスの透明度を考慮して色を加算し、スクリーンに最終的な色を表示している例である。

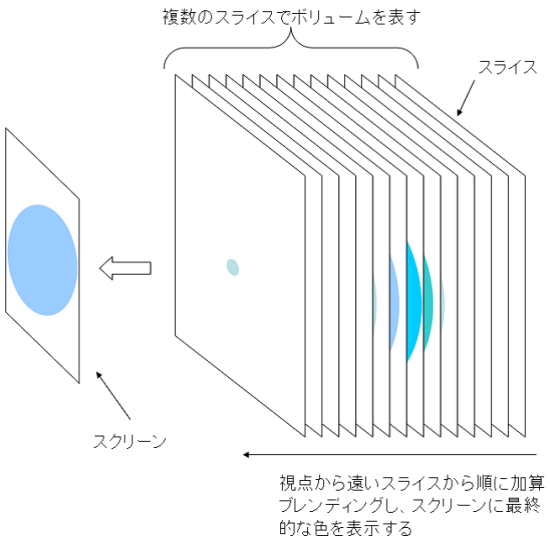


図 3.3: ボリュームからスクリーンへの投影

この手法を用いることで GPU の機能を利用した高速な処理が可能である。

第 4 章

ボリュームレンダリングによるエネルギー波の表現

本稿ではエネルギー波を表現する新しいアプローチとして、ボリュームレンダリングを利用したエネルギー波の表現方法を提案する。ボリュームレンダリングの手法としてはボクセルを用いたレイキャスティング法と GPU の機能を生かせるテクスチャベース法の 2 つの手法を用いる。このことにより、視点が変わってもエネルギー波の光の強さを正確に表現する。

4.1 レイキャスティング法によるエネルギー波の表現

本研究では、レイキャスティング法によるエネルギー波の表現を以下のような手順で行った。

1. エネルギー波を作るための関数を定義
2. 積分区間の決定
3. 空間内のエネルギーの強さの算出
4. シンプソン法による数値積分

5. エネルギー波の表示

レイキャスティング法では、視点の位置からボリュームに向かって視線を飛ばし、その視線上にあるボクセルの輝度を足し合わせるという処理を繰り返す。ボリュームはエネルギー波を表す関数とし、ある区間内を一定の間隔で分割し、分割点におけるボクセル値を数値積分を行うことで足し合わせ最終的なピクセルの色を得る。以下、個々の処理について説明する。

4.1.1 エネルギー波関数の定義

まず、エネルギー波を作るためにエネルギー波の関数を定義する。今回は位置ベクトル $\mathbf{P}(x, y, z)$ に対して、球を表す関数 $F(\mathbf{P})$ と円柱を表す関数 $G(\mathbf{P})$ を組み合わせた関数 $E(\mathbf{P})$ を定義する。

$$E(\mathbf{P}) = F(\mathbf{P}) + G(\mathbf{P}) \tag{4.1}$$

$F(\mathbf{P})$ は球の中心から $\mathbf{P}$ への距離が増えるほどエネルギーの値が小さくなる関数であり、球の中心から $\mathbf{P}$ への距離とパラメータ a で構成された関数である。図 4.1 は $F(\mathbf{P})$ を表した図である。

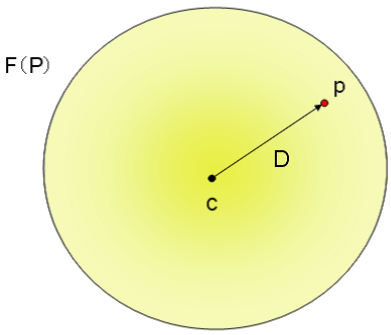


図 4.1: $F(\mathbf{P})$

図 4.1 にある点 p とはエネルギーの計算を行う対象点を表す。また、点 c は球の中心を表し、点 c の位置ベクトルを $\mathbf{C}$ とする。$\mathbf{D}$ は球の中心から対象点までのベ

クトルを表している。$F(\mathbf{P})$ は図 4.1 の値を用いて次式のように定義する。

$$|\mathbf{D}| = |\mathbf{P} - \mathbf{C}| \tag{4.2}$$

$$F(\mathbf{P}) = a/|\mathbf{D}|^2 \tag{4.3}$$

式 (4.3) 中の a には任意の値を設定し、エネルギー波の形を変えるために用いる。a の値を大きくすればするほど対象点のエネルギーは大きくなる。

$G(\mathbf{P})$ は円柱の中心から $\mathbf{P}$ への距離が増えるほどエネルギーの値が小さくなる関数であり、円柱の中心から $\mathbf{P}$ への距離とパラメータ b で構成された関数である。図 4.2 は $G(\mathbf{P})$ を表した図である。

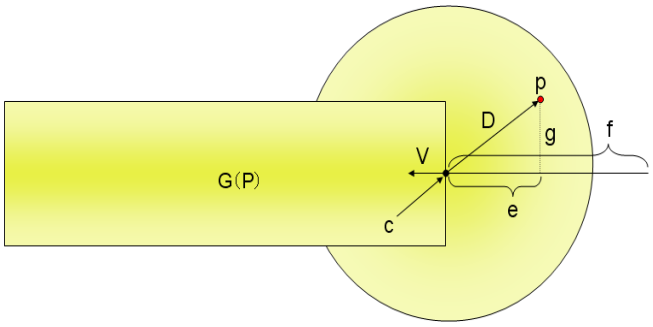


図 4.2: $G(\mathbf{P})$

図 4.2 にある点 p, ベクトル $\mathbf{D}$ は $F(\mathbf{P})$ で使用したものと同じ値である。点 c は球の中心であり、円柱の先端の位置でもある。$\mathbf{V}$ は円柱の向きを表す正規化されたベクトルであり、図 4.2 では $(-1, 0, 0)$ の方向を指している。e はベクトル $\mathbf{D}$ をベクトル $\mathbf{V}$ 上に射影した結果求められる値であり、f にはプログラム実行者が任意の値を設定する。まず、円柱の向き $\mathbf{V}$ とエネルギーの計算を行う対象点までのベクトル $\mathbf{D}$ との内積 s を求める。

$$s = \mathbf{V} \cdot \mathbf{D} \tag{4.4}$$

式 (4.4) で求めた内積の値を使用することで、点と直線の距離の式 [13] から図 4.2 中の g の長さを求めることができる。g は円柱の中心線から対象点までの距離を表

している。

$$g = |\mathbf{P} - \mathbf{C}| - |s|^2 \tag{4.5}$$

g の値が大きくなればなるほど円柱のエネルギーの値は小さくなる。結果的に得られるエネルギー波関数は、図 4.2 の値を用いて式 (4.6) のように表すことができる。

$$E(\mathbf{P}) = a/|\mathbf{D}|^2 + b/g \tag{4.6}$$

今回の実装では先に述べた光線型のエネルギー波を表現するために、パラメータ b に設定する値を内積 s の値によって変更している。図 4.3 は内積 s によって処理を分岐し、b に設定する値を変更しているプログラムの実装例である。

```
if(s > 0.0) {   //内積が0より大きい時
    b = b;
} else if(e < f) {   //内積が0より小さい時
    b = (1.0 - e / f) * b;
} else {   //それ以外の時
    b = 0.0;
}
```

図 4.3: プログラム実装例

内積は 2 つのベクトルの位置関係を表す数値である。この例では、2 つのベクトルが同じ向きを向いているか、同じ向きを向いていないかで処理を分岐し、b に設定する値を変更している。a、b、f の値をプログラム実行者が適切に設定し、2 つのベクトル $\mathbf{D}$、$\mathbf{V}$ の内積の値によって図 4.3 のような処理を行うことで、かめはめ波のようなエネルギー波を表現することができる。

図 4.4 は最終的に生成されるエネルギー波の例である。

図 4.4: 得られるエネルギー波

4.1.2　積分区間の決定

レイキャスティング法では空間中のエネルギー波をスクリーンに表示するために、視点からピクセルごとに視線を飛ばし、その視線上にあるエネルギーの強さの値を積分で足し合わせることでそのピクセルの色を決定する。そのために、まずは視線上のどこからどこまでの区間を積分するかを決定する必要がある。図 4.5 は積分区間とスクリーンの位置関係を表した図である。

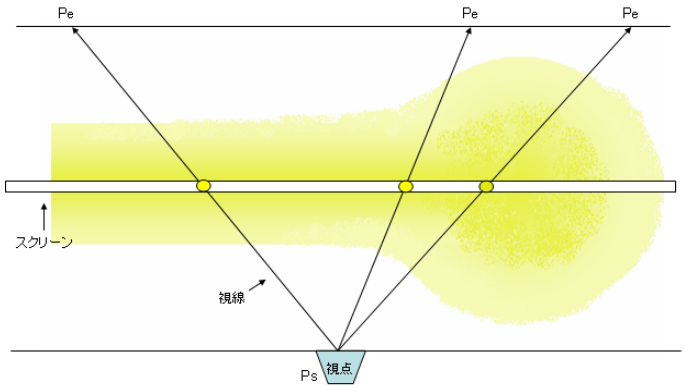


図 4.5: 積分区間とスクリーンの位置関係

本研究ではレイキャスティング法によってボリュームレンダリングを行うため、積分区間の始点を図 4.5 のように視点の位置 P_s からはじめることとする。今回の実装ではスクリーンは原点の位置に配置し、始点の位置からスクリーンの各ピク

セルに向かって視線を飛ばしその視線上にあるボクセルを足し合わせる。この処理を始点からスクリーンまでの距離の 2 倍の位置にある終点 P_e になるまで繰り返すことで、最終的に表示される各ピクセルの色を決定することができる。

4.1.3 空間内のエネルギーの強さの算出

空間内のエネルギー波関数のある位置 $\mathbf{P}(p_x, p_y, p_z)$ におけるエネルギーの強さは、

$$E(p_x, p_y, p_z) = F(p_x, p_y, p_z) + G(p_x, p_y, p_z) \tag{4.7}$$

で求めることができる。この時算出されるエネルギーの強さが、レイキャスティング法におけるボクセル値に相当する。視線上のボクセルごとに式 (4.7) の計算を行いその位置におけるボクセル値を計算し、その値を順次足し合わせていく。この処理を終点まで行うことでその視線上のエネルギーの強さの総和を求め、最終的なピクセルの色を決めることができる。

4.1.4 シンプソン法による数値積分

シンプソン法とは数値積分法の一つで、エネルギー波関数のようななめらかな曲線を持つ関数を積分するときに高速な近似値を得ることができる手法である。本稿ではシンプソン法を利用して積分を行う。

シンプソン法では、関数 $f(t)$ と t 軸の区間 $[t_0, t_n]$ で囲まれた部分の面積 S を

$$\int_{t_0}^{t_n} f(t)dx = S \tag{4.8}$$

とすると、求めたい面積は次の式 (4.9) で表すことができる。

$$S = \frac{h}{3}[f(t_0) + 2\sum_{i=1}^{\frac{n}{2}-1} f(t_{2i}) + 4\sum_{i=1}^{\frac{n}{2}} f(t_{2i-1}) + f(t_n)] \tag{4.9}$$

ここで、n とは区間 $[t_0, t_n]$ を等分割した際の部分区間の個数であり、h は各部分区間の長さを表す。式 (4.9) を合成シンプソン公式と呼び、これをプログラムで実装することで最終的なピクセルの色をシンプソン法の積分によって求めることが

できる。実際のプログラミングを行うときは奇数項の合計と偶数項の合計を計算することで数値積分を行えばよい。

4.1.5　エネルギー波の表示

視点から終点の位置までをある一定の分割数で分割し、分割した点におけるエネルギーの強さをエネルギー波関数から求める。その値をシンプソン法によって積分していき、これを全てのピクセルごとに行うことでエネルギー波を表示することができる。図 4.6 はこれらの処理を図示した例である。

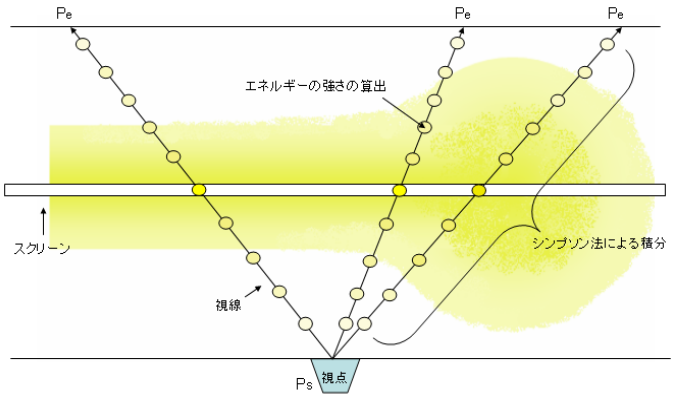


図 4.6: レイキャスティング法

以上がレイキャスティング法におけるエネルギー波の表現方法である。

4.2　テクスチャベース法によるエネルギー波の表現

本研究では、テクスチャベース法によるエネルギー波の表現を、以下のような手順で行った。

1. エネルギー波関数の定義
2. 3 次元配列にエネルギー波の密度分布を格納
3. 3 次元配列をテクスチャとして GPU に読み込ませる
4. 視点に垂直な矩形ポリゴンを複数枚配置
5. テクスチャの回転
6. 矩形ポリゴンにテクスチャの断面をマッピング
7. 視点から遠い順に加算ブレンディング

テクスチャベース法では、エネルギー波の密度分布をテクスチャとして GPU に読み込ませ、テクスチャの断面がマッピングされた複数枚のポリゴンを視点から遠い順に α ブレンディングすることで最終的なピクセルの色を得る。以下、個々の処理について説明する。

4.2.1　エネルギー波関数の定義

レイキャスティング法と同様に、まずはエネルギー波を作るためにエネルギー波の関数を定義する。今回テクスチャベース法で描画するエネルギー波は、ドラゴンボールに登場する「元気玉」や「かめはめ波」のような光球、光線型のエネルギー波である。光線型エネルギー波のエネルギー波関数は、前節で紹介したレイキャスティング法でのエネルギー波関数を用いる。レイキャスティング法でのエネルギー波関数の手法をテクスチャベース法でもそのまま適用し、これによって得られたエネルギーの値を対応する 3 次元配列に格納する。光球型エネルギー

波のエネルギー波関数は、一般的に用いられている球の方程式 (4.10) を使用して定義する。

$$(x-a)^2+(y-b)^2+(z-c)^2=r^2 \tag{4.10}$$

式 (4.10) は球の中心が (a,b,c)、半径 r の球を表す。半径の値に応じた色情報を 3 次元配列に格納することでエネルギー波の密度分布を生成することができる。図 4.7 は半径 r の値に応じて格納する色情報を分けている例である。

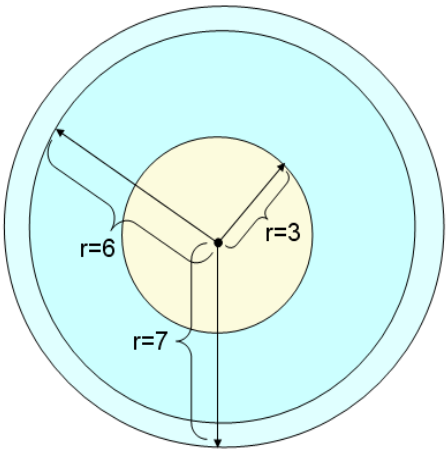


図 4.7: 光球型エネルギー波の色情報の設定方法

図 4.7 のように、半径の値が 3 以内の場合は、光球型のエネルギー波の中心付近の色は白色に近い色情報を 3 次元配列に格納する。半径が 3 より大きく 6 以下の場合は、エネルギー波独自の色が現れるような色情報を 3 次元配列に格納する。半径が 6 より大きく 7 以下の場合は、少し薄めのエネルギー波の色情報を 3 次元配列に格納する。このように、光球型エネルギー波は光線型エネルギー波と違い、半径の値に応じてエネルギー波の色情報をプログラム製作者が自由に設定し、ドラゴンボールに登場する「元気玉」のような光球型エネルギー波に近づくように色情報を 3 次元配列に格納する。

4.2.2 ボリュームデータの生成

まずはエネルギー波をエネルギーの密度分布として 3 次元配列に格納する。空間を六面体の格子状に分割し、その各セルに密度値と配列要素を対応させる。配列の各要素は 4 つの値 RGBα をそれぞれ格納する。この RGBα の値がボクセル値に相当し、3 次元配列がボリュームデータに相当する。ある場所でのエネルギーの密度を変えたい場合は、不透明度を表す α の値を変更することでその場所でのエネルギーの密度を変更するのと同じ効果を得ることができる。エネルギーの密度を高くするには不透明度を上げればよく、密度を下げる場合は不透明度を下げればよい。

光球型エネルギー波は球の方程式を用い、光線型エネルギー波はレイキャスティング法で使用したものと同じ関数を用いて密度分布を生成する。二つのエネルギー波は中心付近が白色で、外側に行くほどエネルギー波独自の色が現れ、その光の強さは弱まっていく。これを表現するため、光球型エネルギー波では、中心部分の α の値は大きめにし、外側に行くにつれて α の値を下げ、エネルギーの密度を低くする。また、配列の各要素に対し、球の内部であれば白色、球の外部であれば黒色を設定する。こうすることで、3 次元配列に格納した値に対応した密度分布を生成することができる。

4.2.3 ボリュームの読み込み

テクスチャは色や透明度の情報を持つ多次元配列である。テクスチャ配列内の個々のデータをテクセルと呼ぶ。生成したボリュームは GPU にテクスチャとして読み込ませる。GPU が 3 次元テクスチャ[9] をサポートしている場合、ボリュームを 3 次元テクスチャとして扱う方法が可能である。今日の GPU のほとんどは 3 次元テクスチャをサポートしているため、本研究でも 3 次元テクスチャを使用してテクスチャベース法を実装する。

ボリュームは CPU のメインメモリから GPU のビデオメモリにグラフィクス

API を介して転送され、3 次元テクスチャとして保持される。この際、テクスチャ座標は通常そのサイズによらず、0〜1 の範囲に正規化される。3 次元テクスチャの場合は x, y, z 座標のそれぞれが 0〜1 の範囲に収まることに注意が必要である。図 4.8 は 3 次元テクスチャが 0〜1 の範囲に正規化されて保持されることを表している。

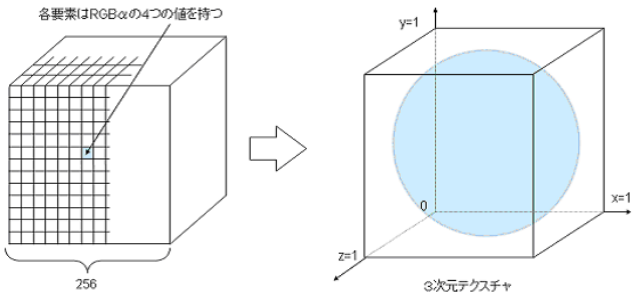


図 4.8: ボリュームの読み込み

4.2.4 スライスの配置

ボリュームをテクスチャとして GPU に読み込んだ後は、ポリゴンの平面であるスライスをワールド座標系に配置する。まず、エネルギー波を画面中央に表示するためスクリーンの中心とワールド座標系の原点を一致させ、視点を Z 軸上に、視線方向はワールド座標系の原点に設定する。次に、全てのスライスの中心が Z 軸上にあるようにスライスを等間隔で配置する。このスライスに 3 次元テクスチャの断面をマッピングすることでエネルギーの密度分布を再構築する。図 4.9 はスライスを Z 軸上に等間隔で配置した例である。

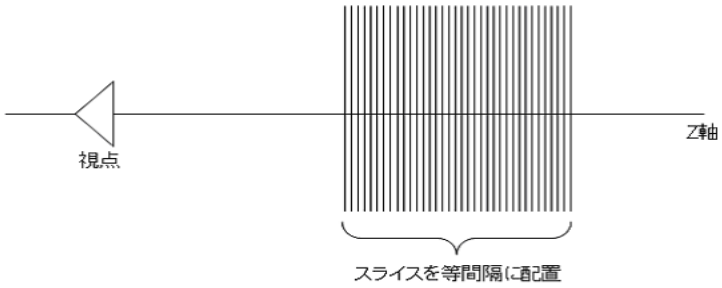


図 4.9: スライスの配置

4.2.5 テクスチャの回転

今回の実験では、視点の変更はテクスチャの回転によって行うものとする。これは、視線とスライスを常に垂直にするためである。視線とスライスの成す角度が小さくなると、それにつれて投影面積が小さくなりサンプリングされるボクセルの数が減少する。結果、正確なボリュームレンダリングを行うことができなくなる。そこで今回の実験では、テクスチャ座標に回転行列を掛けることでテクスチャを回転し、視点が変わっているのと同じ効果を得るものとする。

4.2.6 テクスチャマッピング

テクスチャマッピングとはポリゴンに画像を貼り付ける操作のことである。テクスチャベース法ではスライスに 3 次元テクスチャの断面をマッピングすることでボリュームレンダリングを行う。スライスにマッピングをする際は、スライスの Z 座標の値に合わせて 3 次元テクスチャの Z 座標値を設定する。こうすることでスライスの位置に合わせてマッピングされる 3 次元テクスチャの断面が変化することになる。図 4.10 はスライスの位置に合わせて 3 次元テクスチャの断面が変化していることを表している。

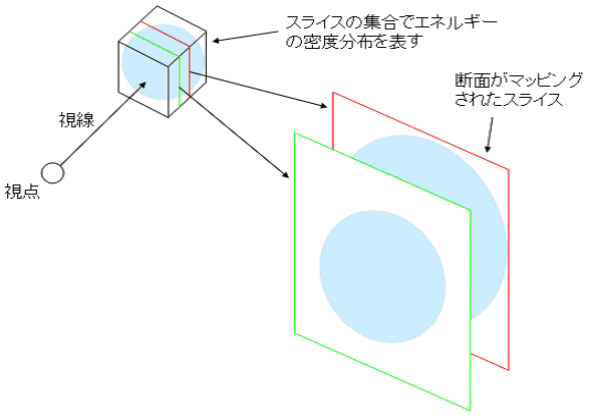


図 4.10: テクスチャマッピング

4.2.7 加算ブレンディング

3次元テクスチャの断面が貼り付けられたスライスを視点から遠い順に順次加算ブレンディングする。加算ブレンディングとはある画像の上に別の画像を重ね合わせるときに、透明度を表わす α 値にもとづいて画像の色を足し合わせる処理のことである。ある画像の上に別の画像を加算ブレンディングを用いて重ね合わせることで、ピクセルの持つ透明度を考慮した画像を得ることができる。レイキャスティング法ではエネルギーの値を数値積分の一つであるシンプソン法によって計算したが、テクスチャベース法では加算ブレンディングを用いることでシンプソン法に近い計算を行う。以上がテクスチャベース法におけるエネルギー波の表現方法である。

第 5 章

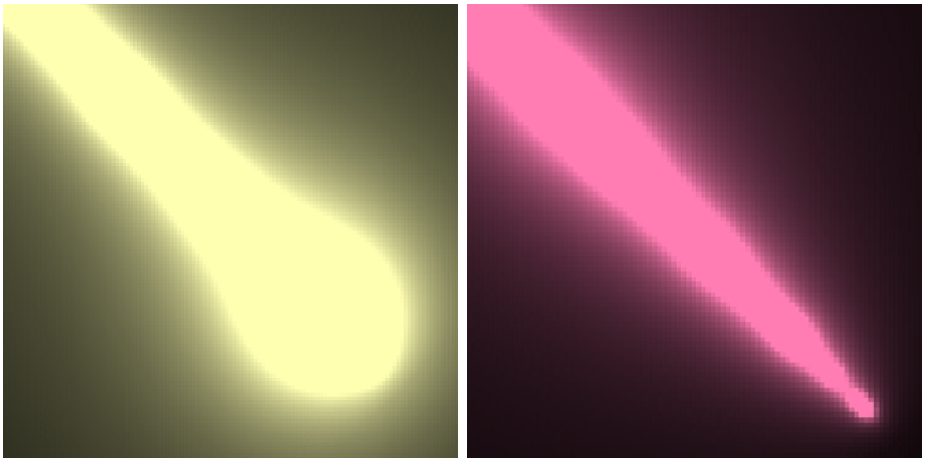
検証と考察

5.1 実行結果

今回はレイキャスティング法とテクスチャベース法によるボリュームレンダリングを行いエネルギー波を表現する。

5.1.1 レイキャスティング法

レイキャスティング法におけるボリュームレンダリングでは、OpenGL ベースの 3DCG クラスライブラリである FK System[14] を使用し、４章で説明した内容で実装を行った。図 5.1 はプログラムを実行して得られた画像である。



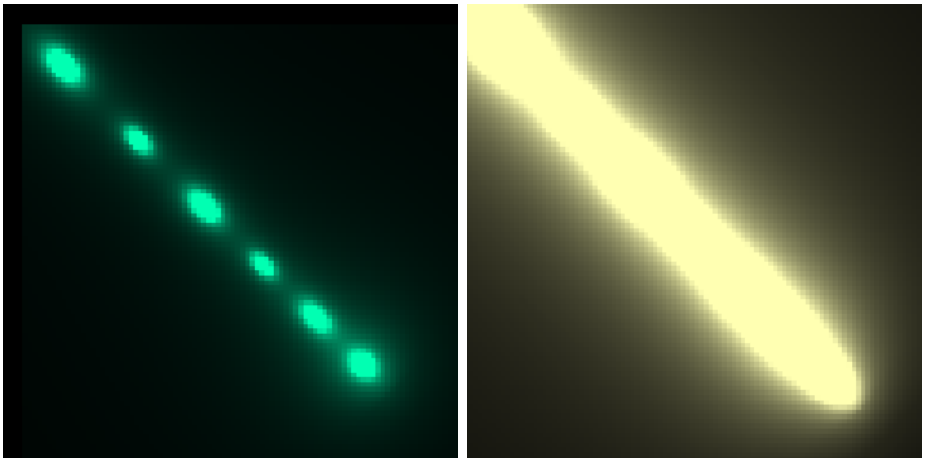


図 5.1: レイキャスティング法によるボリュームレンダリング

それぞれのエネルギー波は中心から離れるごとに光の強さが弱くなっている。各パラメーターを適切に変えることで様々なエネルギー波を作成できることが確認できる。レイキャスティング法ではエネルギー波を関数として定義しているため、上図のように関数のパラメータを変えることで容易にエネルギー波の形を変えられることが大きな利点である。しかし、この手法はエネルギーの計算や視点から飛ばした視線上のボクセルを積分で計算するなど、非常に負荷の高い計算を行う必要があるという欠点もあることに注意が必要である。

5.1.2 テクスチャベース法

テクスチャベース法におけるボリュームレンダリングでは、グラフィックス API に OpenGL[15]、GPU 用のグラフィックス言語に nVidia 社の Cg 言語 [16] を使用し、4 章で説明した内容で実装を行った。図 5.2 はプログラムを実行して得られた画像である。

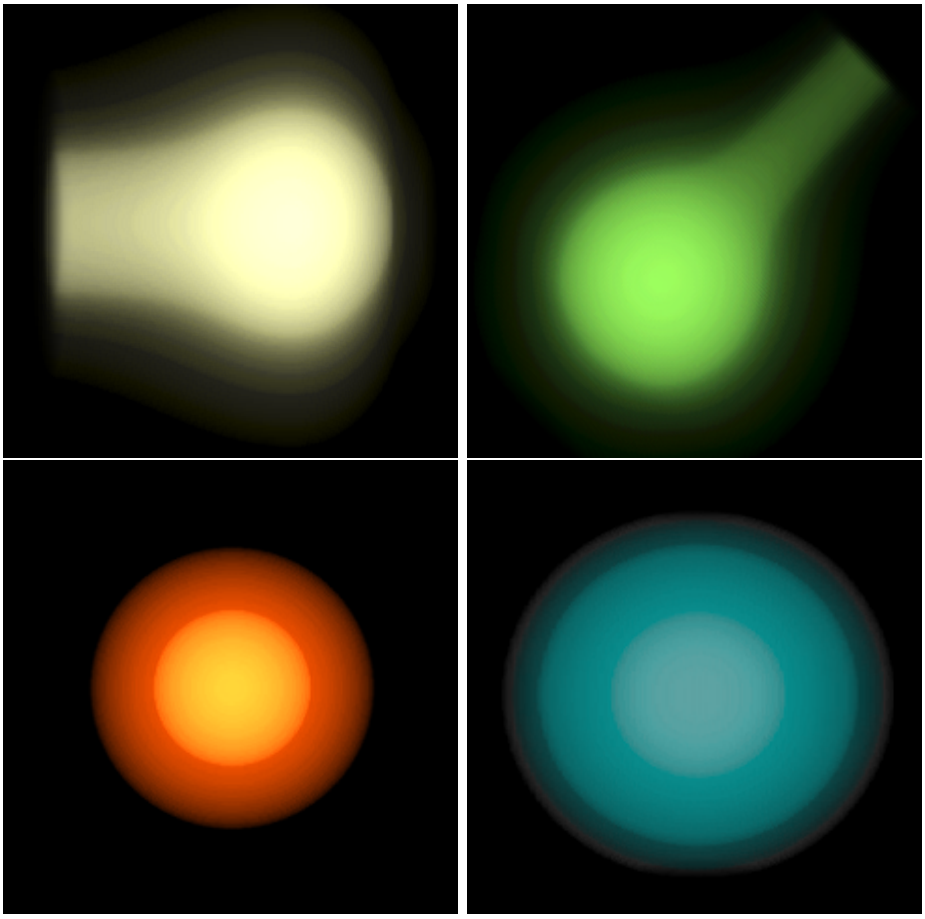


図 5.2: テクスチャベース法によるボリュームレンダリング

上の 2 つの光線型エネルギー波の画像は、レイキャスティング法で用いたエネルギー波関数によって得られた値を 3 次元配列に格納し、それを加算ブレンディングを行うことで得られた画像である。下の 2 つの光球型エネルギー波の画像は、球の方程式を利用して得られた画像であり、エネルギー波の中心付近を白色に、中心から離れるごとにそのエネルギー波独自の色が表れるように調整した。テクスチャベース法の大きな利点は、GPU を用いた高速な処理である。ボリュームデータを 3 次元テクスチャとして保持することで、GPU のテクスチャメモリを用いた高速な処理が可能となった。しかし、この手法は 3 次元テクスチャの解像度が低い場合は高速な処理が可能だが、3 次元テクスチャの解像度が高くなるほどそれだけ多くのメモリ容量が必要になってしまう。そのため、テクスチャベース法はボ

リュームデータの大きさによって処理速度に大きな差が生じてしまうという欠点もある。

5.2　検証と考察

本稿の目的はボリュームレンダリングによってエネルギー波を表現し、３次元空間中のエネルギー波の光の強さをどのような視点から見ても正確に表現することである。そのため、一枚のエネルギー波が描かれたテクスチャとボリュームレンダリングによって得られた画像を見比べ、プログラムの実行結果と併せて検証を行うこととする。

今回検証に用いるエネルギー波はテクスチャベース法で作成されたかめはめとする。図 5.3 が検証に用いるエネルギー波である。

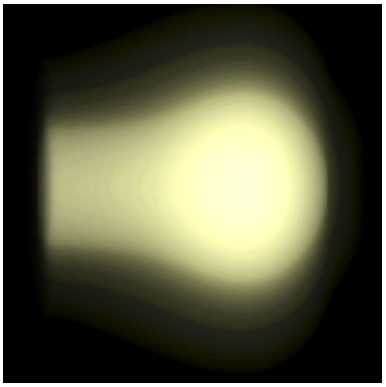


図 5.3: 検証に用いるエネルギー波

エネルギー波のテクスチャと、同じ位置から見たボリュームレンダリングの画像を見比べる。図 5.4 はいくつかの角度から見たテクスチャとボリュームレンダリングの画像である。

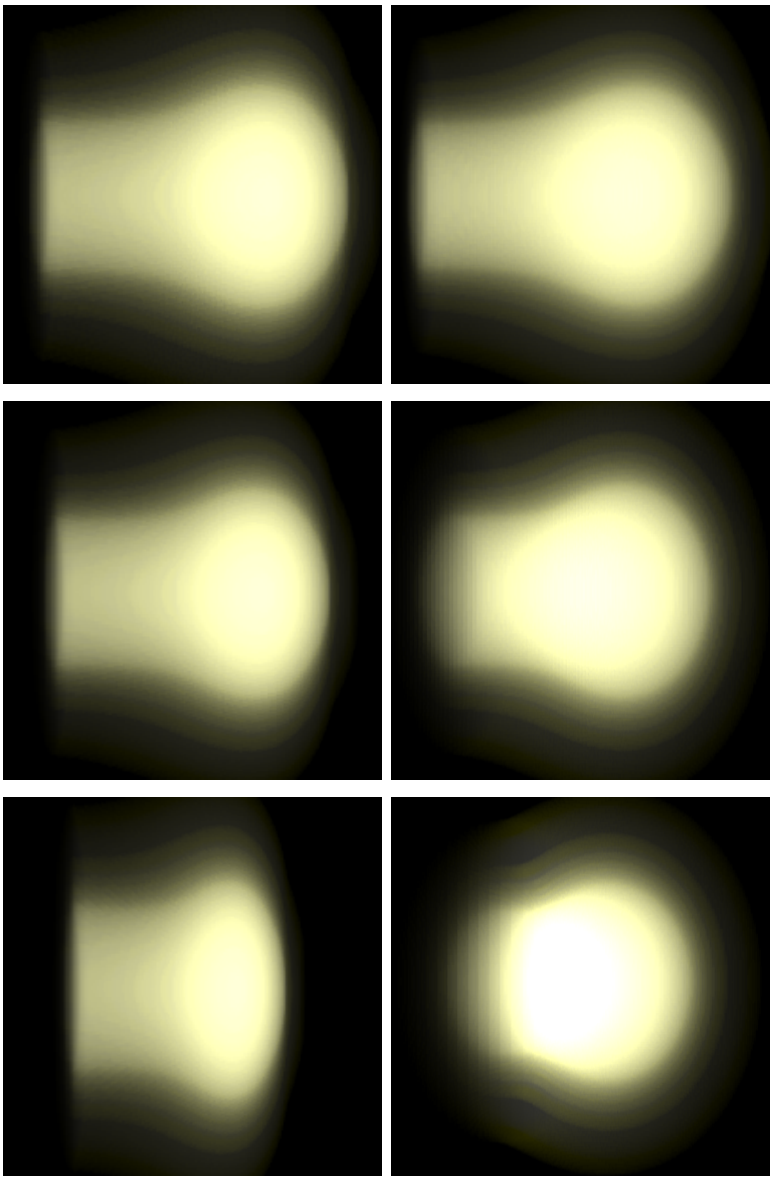


図 5.4: テクスチャとボリュームレンダリング

図は視点の位置を x 軸方向にそれぞれ $-20°,-40°,-60°$ 回転した位置から見た画像であり、左側がテクスチャを回転した画像で、右側がテクスチャと同じ位置から見たボリュームレンダリングの画像である。テクスチャを回転させただけの場合、どの視点から見てもエネルギー波の光の強さは一定である。しかし、同じ位置から見たボリュームレンダリングの画像では、角度によってエネルギー波の見え方は大きく変わっていることが確認できる。これはボリュームレンダリングの

特徴である、「視線上にあるボクセルの値を透明度を考慮して足し合わせる」という処理によって実現可能なものである。

２章で紹介したように、エネルギー波をビデオゲームで表現する場合、エネルギー波は一枚の画像として表示し、視点が変わった場合はポリゴンの回転とあらかじめ用意した画像を切り替えることでエネルギー波を表現している。そのため、この手法では決まった視点の位置でしか正確なエネルギー波を表現することができないという問題点があった。

今回の実験ではエネルギー波を決まった角度から見た時の画像を紹介したが、このような問題点もボリュームレンダリングを用いることで解消することができる。ある決まった角度からではなく、どのような視点から見てもエネルギー波の光の強さを正確に表現できるのである。

以上のことから、テクスチャで表現されたエネルギー波と比べ、本手法を用いることでどの視点の位置においてもエネルギー波の光の強さを正確に表現することができたといえる。

第 6 章

おわりに

本論分の締めくくりとして、まとめと今後の展望について述べる。

本稿ではボリュームレンダリングによるエネルギー波の表現方法を提案した。視線上のボクセルを足し合わせるという処理を繰り返してピクセル値を求めるレイキャスティング法について解説し、OpenGL ベースの 3DCG クラスライブラリである FK System を使用し実装を行った。

また、レイキャスティング法の考えを発展し処理の一部を GPU で行うことで高速な処理を可能にするテクスチャベース法についても解説し、OpenGL と GPU 用のプログラミング言語である Cg 言語を使用し実装を行った。

検証と考察では、視点の変更に応じた光の強さの変化が一番わかりやすいテクスチャベース法で得られたかめはめ波の円柱部分を利用した。テクスチャとボリュームレンダリングによるエネルギー波の画像を見比べることで、視点の位置に応じてエネルギー波の光の強さが変わっており、ボリュームレンダリングを用いることで正確にエネルギー波の光の強さを表現できることを確認した。

しかし、現状では 5 章の実行結果でも述べたように、いくつかの問題点が残っている。レイキャスティング法ではエネルギー波を関数として定義しているため、関数のパラメータを変えることで容易にエネルギー波の形を変えられることができた。しかし、エネルギーの計算や視点から飛ばした視線上のボクセルを積分で計算するなど、非常に負荷の高い計算を行う必要がある。また、テクスチャベー

ス法でも、GPU を用いた高速な処理を行えるという利点がある反面、ボリュームデータの大きさによって処理速度に大きな差が生じてしまったり、エネルギー波関数を 3 次元の配列に格納するという処理に多くの調整が必要だという欠点がある。

これらを踏まえた上で今後の展望について述べる。レイキャスティング法では、非常に負荷の高い計算を行う必要があるという問題点があった。しかし、近年の GPU の機能拡張により、GPU でレイキャスティング法を用いることが可能となっている [17]。そのため、この問題に関しては GPU を用いてレイキャスティング法を行うように改良することで処理速度を向上することができる。また、現状で特に改良の余地があるのは、テクスチャベース法においてエネルギーの密度分布をテクスチャとして 3 次元の配列に保持する処理である。テクスチャベース法ではレイキャスティング法よりも高速な処理が行えるが、3 次元の配列にエネルギーの密度分布を保持する部分に多くの調整を必要とする。

この問題を解決する方法として、エネルギーの密度分布を 3 次元配列に格納する際に、より複雑な密度分布を定義できるようなツールを制作するという方法が考えられる。例えば、ペイントソフトのようなツールで 2 次元の画像をスライスの数だけ描き、それを 3 次元配列に格納することでエネルギーの密度分布を定義するという方法である。エネルギーの密度分布を 3 次元配列に格納することに特化したツールを制作・利用することで、テクスチャベース法におけるボリュームレンダリングで更に様々な形状を表現することができるようになる。

現状では、プレイステーション 3[18] や XBOX360[19] などの次世代機と呼ばれるハードでボリュームレンダリングによるエネルギー波を使用したゲームを開発することは難しい。非常に負荷の高い計算が必要なボリュームレンダリングを用いてエネルギー波を表現するよりも、テクスチャとしてエネルギー波を表現する方が明らかにパフォーマンスが良いからである。しかし、CPU や GPU などのハードウェアの性能は現在も驚くほど向上している。近い将来、このようなハードウェアの発展によってビデオゲームでボリュームレンダリングを用いることも可能になるであろう。テクスチャベース法だけでなく、負荷の高い計算が欠点であるレ

イキャスティング法においても、ビデオゲームでボリュームレンダリングを用いたエネルギー波の表現を行うことができるのではないかと考える。本論文の表現手法がビデオゲームやその他のコンテンツにおけるエネルギー波の表現に役立つことを願い、本論文の締めくくりとする。

謝辞

本研究を締めくくるにあたり、研究の指針から開発の手法、論文の執筆と幅広いご指導ご教授を頂きました、本校メディア学部の渡辺大地講師及び、卒業研究だけでなくゲームに関する研究の心得などをご指導して下さった山路和紀氏、三上浩司氏、中村太戯留氏、小澤賢侍氏に心より感謝いたします。在学中に研究の手助けや、メディア学の在り方、研究者としての心得などをご指導して下さった本校大学院メディア学研究科博士課程の竹内亮太さんに感謝したいと思います。さらに、研究を進めるにあたって様々な意見を交換してくれた、本校大学院の方々や、メディア学部の学友諸氏に感謝します。また、本校メディア学部のゲームサイエンス卒研室のメンバーに感謝します。そして、いつも私を支えてくれた家族と、全ての友人たちに感謝します。

最後に、本研究にご協力いただきました全ての皆様と、この論文に目を通してくださった全ての方々に、厚くお礼を申し上げます。

参考文献

[1] 鳥山　明,「ドラゴンボール」, 集英社, 1985.

[2] 「機動戦士ガンダム」, 日本サンライズ, 1979.

[3] 松本零士, 「宇宙戦艦ヤマト」, 秋田書店, 1974.

[4] 「CGWORLD, 2004,2 月号」, p36〜41. WORKS CORPORATION.

[5] ボリュームレンダリングソフトウェア, 「VGStudio/VGStudioMAX」, 日本ビジュアルサイエンス株式会社, <http://www.nvs.co.jp/archives/01_01_vgstudio/>.

[6] 3D 画像処理エンジン, 「VolumePro」, TeraRecon.Inc, <http://www.terarecon.co.jp/imaging/vol1.html>.

[7] 3 次元テクスチャ・シェーダ, 株式会社ジェーエフピー, <http://www.jfp.co.jp/three_d_t/>.

[8] Lichtenbelt, B., Crane, R. and Naqvi, S, 「Introduction To Volume Rendering」, Hewlett-Packard, 1998.

[9] 篠本雄基, “汎用 GPU を用いたボリュームレンダリングの高速化に関する研究”, 京都大学大学院情報学研究科修士論文, 2006.

[10] 山崎俊太郎, 加瀬究, 池内克史, “PC グラフィクスハードウェアを利用した高精度・高速ボリュームレンダリング手法”, IPSJ CVIM, 2001.

[11] 千葉則茂, 土井章男,「3 次元 CG の基礎と応用 [新訂版]」, サイエンス社, 2004.

[12] Tom McReynolds,Silicon Graphics, “Programming with OpenGL: Advanced Techniques”, pp.144-147, 1997.

[13] Eric Lengyel, 「ゲームプログラミングのための 3D グラフィックス数学」, 株式会社ボーンデジタル, 2002.

[14] 渡辺大地, FK Tool Kit System, <http://www.media.teu.ac.jp/~earth/FK/>.

[15] OpenGL.org, OpenGL, <http://www.opengl.org/>.

[16] NVIDIA, <http://jp.nvidia.com/page/home.html>.

[17] Stegmaier, S., Strengert, M., Klein, T. and Ertl, T, “A Simple and Flexible Volume Rendering Framework for Graphics-Hardware-based Raycasting”, Proceedings of Eurographics/IEEE VGTC Workshop on Volume Graphics, pp.187-195, 2005.

[18] SonyComputerEntertainment, PLAYSTATION3, <http://www.playstation.jp/>.

[19] Microsoft, XBox / XBox360, <http://www.xbox.com/>.