1ZB-1

GPU を用いたリアルタイムレイトレーシングの検討

上野謙二郎† 孟林‡ 山崎勝弘‡
†立命館大学大学院理工学研究科 ‡立命館大学理工学部

1 はじめに

レイトレーシング法では、光線と物体の交差判定に最も多くの処理時間を必要とする。本研究の目標は、GPU（Graphics Processing Unit）を用いてレイトレーシングの処理時間を 100ms 以下にすることであり、そのために GPU を用いた画面分割の並列化を検討する。画面分割では、画面をブロック分割し、各ブロックをストリーミング・マルチプロセッサ（SM）に割り当ててレイトレーシング処理を行うことで、処理時間を短縮する。本稿では、画面分割を行うために、GPU を用いた並列処理の実装方法について述べる。

2 レイトレーシングの各種手法

2.1 アルゴリズム

レイトレーシングとは、3 次元 CG で物体の座標データ、光源、視点の位置などのデータを計算して、非常にリアルな画像を描画する手法である。図 1 に示すように、レイトレーシングは次の手順で行う。

（1） 視点からスクリーン上の画素を通過する光線を発生させ、光線と物体との交差判定を行う。
（2） 交差判定により、光線と交差する物体が存在するなら、光線と物体との交点を求める。交差する物体が複数ある場合には、すべての物体について交点を求める。交点がない場合は、その画素を背景の輝度とする。
（3） 光線と交差する物体の交点までの距離を求め、最も近い物体を抽出する。
（4） 抽出物体の輝度の計算をする。また、反射、屈折があるならその方向を求め、その方向を光線とみなして（2）、（3）の処理を行い、屈折、反射して見える物体を抽出する。

以上の処理をスクリーン上のすべての画素に対して行い、画像を生成する。

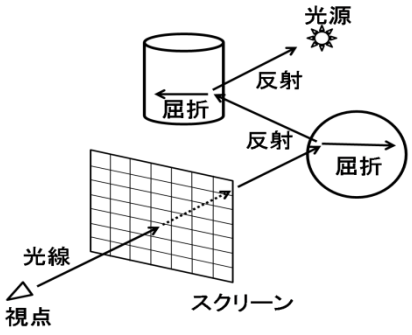

図 1 レイトレーシングの原理

2.2 拡散反射光

拡散反射光とは、物体に陰影をつけることができる手法である。拡散反射光の輝度 Id は以下の式で求めることができる。

拡散反射光を求める点への入射角を θ、光線の光度を Iq、拡散反射係数を Kd とすると、拡散反射光の輝度 Id は、$Id = Kd \times Iq \times \cos\theta$ と表せる。

2.3 鏡面反射光

鏡面反射光とは、物体にハイライトなどテカリを入れることができる手法である。鏡面反射光の輝度 Is は以下の式で求めることができる。

鏡面反射を求める点への入射光を Ip、入射光の正反射光と視線のなす角を α、面反射係数を Ks とすると、鏡面反射光の輝度 Is は、$Is = Ks \times Ip \times \cos^{n}\alpha$ と表せる。

3 GPU を用いた画面分割の並列化の検討

レイトレーシングの処理時間の多くは交差判定に費やされている。レイトレーシングでは、スクリーン上のある画素の輝度値を求める処理が、他の画素の輝度値を求める処理に依存していないので、並列処理の効果を得やすいという特徴がある。

3.1 GPU の構成

本研究では、CUDA 対応 GPU の Tesla C1060 を使用する。図 2 に GPU の構成を示す。Tesla C1060 は 30 個のストリーミング・マルチプロセッサ（SM）が搭載されており、各 SM は 8 個のストリーミング・プロセッサ(SP)から構成されている。つまり、GPU 全体では 240 個の SP をもつことになる。各 SM は 16K バイトの共有メモリと、16K 個の 32 ビットレジスタをもつ。CUDA 対応 GPU では、ある計算処理をスレッドと呼ばれる単位に分割して並列計算を行う。まず、スレッド・スケジューラーが GPU のプログラムをスレッドに分割して、SM の中の 8 個の SP に割り当てる。SP は、クロックサイクル毎に複数のスレッドに対して同一の命令を実行する。これを SIMT（Single Instruction Multiple Thread）と呼ぶ。

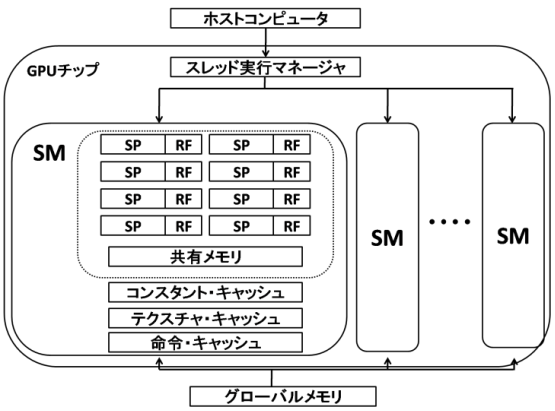

図 2 GPU の構成

Implementing Real -time Raytracing using GPU
Kenjiro Ueno , Lin Meng and Katsuhiro Yamazaki
† Graduate School of Science and Engineering , Ritsumeikan University
‡ College of School of Science and Engineering , Ritsumeikan University

3.2　ストリーミング・マルチプロセッサによる画面分割

本研究では 512*512 の解像度のスクリーンを用いる。ストリーミング・マルチプロセッサによる画面分割では、図 3 に示すようにスクリーンを複数のブロックに分割し、各ブロックをそれぞれ SM に割り当てる。各 SM ではブロック内のスレッドを 32 個単位のワープに分割する。1 つのワープは SP8 個で 4 クロックで実行される。

この方法では、各ブロックに含まれている物体情報のみを SM の共有メモリに格納することにより、メモリアクセスを高速化することができる。

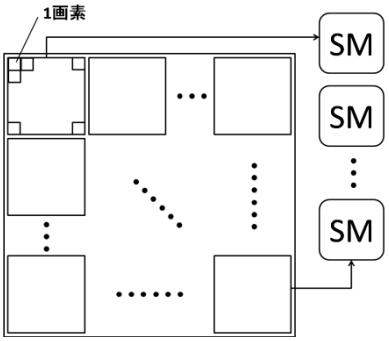

図 3　SM による画面分割

4　GPU 上での並列処理方式

図 4 は現在作成中の処理系の構成を示す。本プログラムは CPU 側と GPU 側の 2 つの処理から成り立っている。

まず、CPU 側でレイトレーシングの実験用のシーンデータである SPD（Standard Procedural Databases）ファイルを読み込む。SPD とはレイトレーシングの実験用のシーンデータであり、この SPD を用いることで、様々な画像を生成することができる。文献[3]では SPD 情報をリスト構造により管理しているが、リスト構造では GPU 側に正しくデータを渡すことができない。そのため、本プログラムでは SPD 情報をリストから配列構造に変換して管理する。CPU 側でブロック数とスレッド数を指定し、SPD 情報を GPU 側に転送する。

GPU 側では受け取ったブロック、スレッド情報を元にスクリーンを SM 数分のブロックに分割する。次に、各ブロックをワープ単位の 32 スレッドに分割する。この時 1 スレッドはスクリーンの画素 1 つの処理を行う。GPU では SP の最低クロック数が 4 なので、一度の動作で 1 つの SP が 4 つのスレッドを実行する。8 台の SP がそれぞれ 4 つのスレッドを動作させるため、32 スレッドで処理を実行する。各スレッドでは 1 画素分のレイトレーシング結果を得るために、光線の方向探索、交差判定、反射、及び輝度計算を行い、スクリーンにレイトレーシング結果を書き込む。

CPU 側でスクリーン情報を受信する。受信した結果を ppm 形式のファイルに変換することで画像を得る。

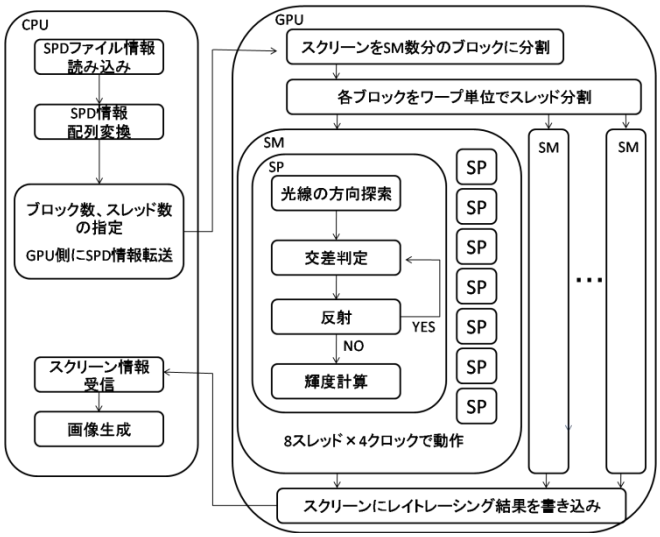

図 4　モジュール構成

5　レイトレーシングによる生成例と考察

C 言語逐次処理で、SPD を用いて生成した画像を図 5 と図 6 に示す。

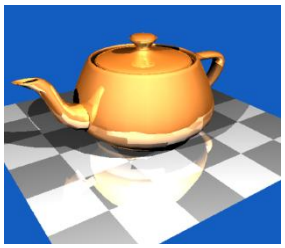

図 5　teapot

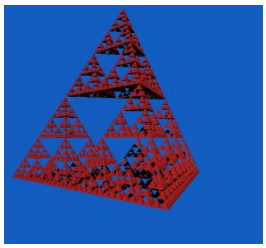

図 6　tetra

図 5 の teapot の実行時間は 399.73(s)、図 6 の tetra の実行時間は 160.18(s)である。ここで、解像度は 512*512、実行環境は Intel(R)Core(TM)2Quad CPU 400@2.66GHz である。リアルタイム性を実現するためには、teapot では 4000 倍の速度向上が必要である。

6　おわりに

本稿では、GPU を用いたリアルタイムレイトレーシングについて検討した。スクリーン全体をブロック分割し、各ブロックを 1 つの SM に割り当てた上で、SM 内で 8 個の SP によるマルチスレッド処理を行う。現在、各画素の処理のデバッグを進めている。

参考文献

[1] James D. Foley, et al : コンピュータグラフィックス理論と実践, オーム社, 2005.
[2] 小山田耕二, 岡田賢治 : CUDA 高速 GPU プログラミング入門, 秀和システム, 2010.
[3] Patrickomatic.com : http://patrickomatic.com/c-ray-tracer