ISSN 2186-5647

P-22

GPUを用いたレイトレーシングの並列化

日大生産工（学部）　○鈴木　俊孝　　日大生産工　岡　哲資

1 まえがき

近年、コンピュータグラフィックスは映画やテレビ、ビデオゲームなどに用いられるようになり、その技術は産業において重要な役割を果たしている。特に映画では複雑なシーンを違和感なく再現する必要があるため、より写実的な画像を得るための研究が行われている[1)]。

写実的な画像を生成するアルゴリズムの1つとしてレイトレーシングがある。アルゴリズムは単純であるが、実際の物理現象を再現しているため、写真のような画像をコンピュータ上に生成することができる。

レイトレーシングの問題は膨大な計算量とレンダリング時間を要することである。解像度640×480の画像では302,700回光線を生成する必要があり、さらにシーンが複雑になれば計算量と時間は増加する[2)]。このような問題を解決するため、データ構造の改善やレイトレーシングに適したデバイスを利用したGPUレンダリングが行われるようになった[3)]。

GPUの利点はCPUに比べ多くのプロセッシングコアを所有していることである。そのため並列計算に特化しているといえる。GPUは本来3Dグラフィックやゲームなどをディスプレイへ描画するための専用ハードウェアであったが、近年、数値解析・シミュレーションなど他の用途に用いられるようになっている。またGPU専用の開発環境CUDAの登場により、科学研究分野においてGPUの並列計算能力を生かした多くの計算が行われるようなった[4)]。

交差判定を減らすためのデータ構造としてAxis Aligned Bounding Box(AABB)がある。AABBは3Dモデルを立方体で包含し、置き換えることでモデルの各ポリゴンとの交差判定を減らすものである。

2 目的

本研究ではAABBとGPUを用いたレイトレーシングの並列化と実装を行い、その有効性を評価する。

3 レイトレーシング

レイトレーシングは光の反射、屈折を再現できるため現在普及しているスキャンライン法に比べ写真に近い画像を生成することができる[5)]。

図1にレイトレーシングの概要図を示し、アルゴリズムの手順を以下に述べる。

(1) 視点からスクリーン上のピクセルに向けてレイを生成する。

(2) レイと最初に交差した物体が交差点となる。

(3) 交差点が陰であるか判断するために光源に向けてレイを生成する。

(4) 物体の表面が反射、屈折する素材ならば新たにレイを生成し(2)、(3)の手順を繰り返す。

以上の処理を全ピクセル行うことで最終的な画像が得られる。

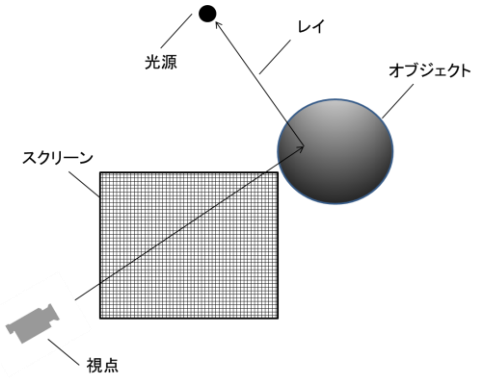

図1 レイトレーシングの概要

4 AABB

4.1 AABBの生成

実際のAABBの生成手順を示す。本研究では3Dモデルデータとして頂点座標、法線ベクトルなどが記述されたOBJ形式のファイルを使用する。AABBは立方体の最小座標と最大座標で構成される。そのためOBJファイル読み込みの際にモデルの最小座標と最大座標を求め、これらの値をAABBの最小座標と最大座標とする。

4.2 AABBを用いた交差判定

Parallel Ray Tracing using a GPU

Toshitaka SUZUKI and Tetsushi OKA

図2にAABBを用いた交差判定の概要図を示す。レイとAABBが交差した場合のみモデルの各ポリゴンと交差判定を行うことで計算量の削減を実現する。

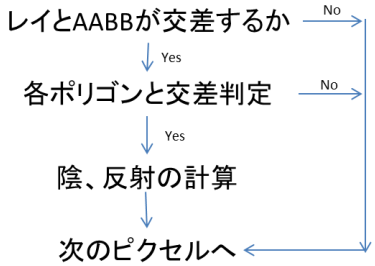

図2 AABBを用いた交差判定

5 GPUを用いた並列処理

5.1 マルチ・スレッドによる処理

プログラムの実行の最小単位はスレッドと呼ばれ、このスレッドが一度に大量に生成されたのちにGPU上の各コアで実行されることで並列処理を実現している。

5.2 GPUでの開発環境CUDA

CUDAはNVIDIA社から提供されるGPU専用のC言語開発環境であり、NVIDIA製のハードウェアのみ使用が可能である。

6 実装

6.1 使用機材およびソフトウェア

本研究ではCPUのみのレイトレーシング、GPUを用いたレイトレーシングを実装する。図3に実装環境を示す。

CPUによるレイトレーシング	GPUによるレイトレーシング
OS:windows8 CPU:Intel(R) Core(TM) i7-4700HQ@2.40GHz 実装メモリ:12.0GB	OS:windows8 CPU:Intel(R) Core(TM) i7-4700HQ@2.40GHz 実装メモリ:12.0GB GPU: GeForce GT 750M GPUクロック: 1.09GHz CUDA: 6.5 CUDA Capability 3.0

図3　実装環境

6.2 GPUを用いたレイトレーシング

図4に実装の概要図を示す。GPUでの処理を行う前にCPU側ではあらかじめ視点座標とスクリーン座標、光源座標の設定を行い、3Dモデルデータを読み込む。3Dモデルデータは OBJ形式のファイルを使用する。この際AABBを計算する。以上のデータをGPU上のカーネルプログラムへ転送する。

実際のGPUのプログラムの起動はCPUから行う。GPUではCPUから受け取ったデータをもとにレイを生成する。レイと物体の交差判定をGPU上の各スレッドで計算させることで並列処理を実現する。最終的に得られたピクセルデータを再びCPU側に転送し、CPU側でディスプレイへの表示を行う。

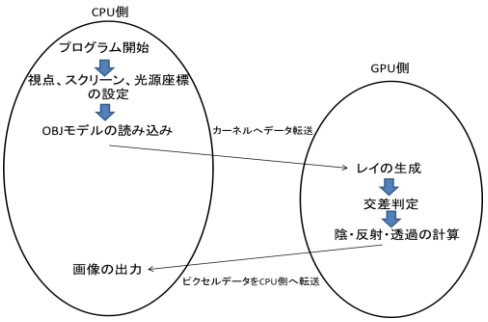

図4 実装の概要図

7 評価方法

本研究で用いた方法の有効性を検討するために評価実験を行う。まずポリゴン数の異なるモデルを用意する。各モデルに対し、AABBを使用した場合と使用しないレンダリングをCPU、GPUで行う。この時レンダリング要した時間を測定する。

CPUのみを用いた処理時間とGPUを用いた並列処理時間から速度向上率を計算する。速度向上率はCPU処理の実行時間を並列処理の実行時間で割った値である。速度向上率の値により並列化による効果の評価をする。

またポリゴン数と速度向上率の関係から、どの程度の複雑な形状のモデルに有効であったかを評価する。

8 まとめと今後の検討

本研究ではAABBとGPUを用いたレイトレーシングの並列化と実装を行う。

実装したレイトレーシングを用いて実験を行い有効性について考察を行う。

「参考文献」

1) Christensen, P.H., Fong j, Laur D.M Batali D “Ray Tracing for the Movie ‘Cars’” IEEE Symposium on Interactive Ray Tracing(2006).
2) 塩川 厚「コンピュータグラフィックスの基礎知識」 オーム社 (2000).
3) F.Karlsson “Ray tracing fully implemented on programmable graphics hardware,”,Chalmers University of Technology(2004).
4) nvidia CUDAとは？ http://www.nvidia.co.jp/object/cuda-jp.html
5) Turner Whitted “An Improved Illumination Model for Shaded Display” Communications of the ACM 23 (1980) p.343-349.