GPU を用いた光シミュレーションのためのレイトレーシングの研究

蔵野 裕己
Yuki KURANO

1　はじめに

近年，オフィスにおけるワーカーの快適性および知的生産性の向上に注目が集まっており，オフィスの環境を改善することで，知的生産性が向上すると報告されている [1]．我々は，オフィスの光環境に着目し，個々のワーカーの要求に応じた明るさを提供する知的照明システムの研究を行っている [2]．しかし，知的照明システムは照度センサや調光機器など，大量の機器を必要とし，その複雑性と価格的コストにより導入の敷居が高い．そのため，実システム無しで知的照明システムの研究を可能とするような，シミュレータの必要性は高いといえる．またシミュレータの実現に際して，より実世界に近い表示を行うために，3DCG(3 Dimensional Computer Graphics) による画像表示が必要と考えられる．ただし，高品質な 3DCG を生成するためには，膨大な計算が必要である．そこで本研究では，3DCG の生成手法の 1 つであるレイトレーシングを，GPU を用いて高速に生成する手法を提案し，その性能について評価する．

2　GPGPU

本研究で計算に用いる GPU(Graphics Processing Unit) は，従来，画像処理専用のハードウェアとして利用されてきた．画像処理は，高い並列性を持つ計算が多いため，GPU は並列性の高いハードウェア構成を持つ．この特徴に着目し，GPU で画像処理以外の並列性の高い汎用計算を行う，GPGPU(General-Purpose computing on GPU) の研究が広く行われている [3]．特に本研究では，NVIDIA 社が提供する GPGPU の統合開発環境である，CUDA[4] を用いた．

GPU は複数の SM(Streaming Multi Processor) から成り，SM は複数の SP(Streaming Processor) から成る．これをソフトウェア側では，グリッド，ブロック，スレッドの 3 つの単位で管理する．1 つの GPU は 1 つのグリッドに対応し，1 つの SM は 1 つのブロックに対応する．そして，1 つの SP は 1 つのスレッドに対応する．すなわち，複数のブロックをまとめる単位がグリッドであり，複数のスレッドをまとめる単位がブロックである．また，CUDA では複数種類のメモリを利用できる．代表的な例として，グローバルメモリ，コンスタントメモリ，シェアードメモリおよびレジスタの各特徴を Table 1 に示す．これらのメモリを，目的やデータの特性に合わせて使い分けることで，高速化を図ることが可能である．

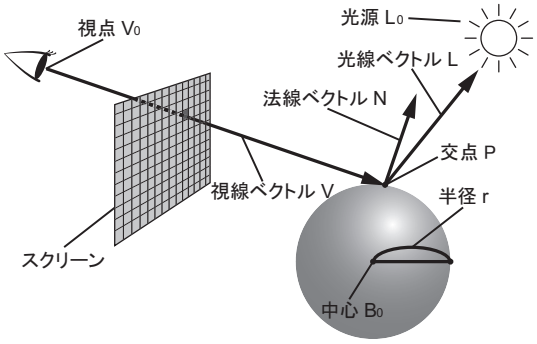

Fig. 1　レイトレーシングの概要図

3　レイトレーシング

本研究では，3DCG の生成手法の 1 つである，レイトレーシングを対象に GPU を用いて高速化する．Fig. 1 にレイトレーシングの概要図を示す．また，レイトレーシングの計算手順は以下の通りである．まず Fig. 1 のように，視点からスクリーン上の画素に向けてレイを発射し，これを視線ベクトルと呼ぶ．計算により視線ベクトルと，空間中に定義される物体との交点を求める．交点が複数存在する場合，視点に最も近い交点を選択する．これらの計算を，交点計算と呼ぶ．また，求めた交点における物体の色，光の色や強さ，反射や屈折の有無などから色を算出し，その色を，視線ベクトルが通過するスクリーン上の画素の色とする．これらの色を求める計算は，シェーディングと呼ばれる．レイトレーシングでは，交点計算が全体の計算の大部分を占める．しかし，交点計算，シェーディングともに，画素ごとに計算が独立しているため，非常に並列性が高い．

4　レイトレーシングの高速化

GPU を用いてレイトレーシングを高速化するために，処理を並列化する．最も基本的な手法としては，画素ごとの計算を並列化する画素並列法が挙げられる．また，

Table 1　GPU 上のメモリ

	容量	速度	共有範囲	キャッシュ
グローバルメモリ	大きい	遅い	同一グリッド内	無し
コンスタントメモリ	大きい	遅い	同一グリッド内	有り
シェアードメモリ	小さい	速い	同一ブロック内	無し
レジスタ	小さい	速い	同一スレッド内	無し

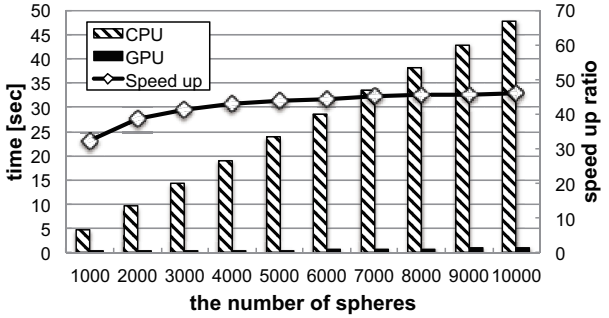

Fig. 2　CPU と GPU の計算時間比較

交点計算を物体ごとに並列化する物体並列法や，空間を 3 次元に区切り，各空間を計算資源に割り当てるボクセル並列法など，様々な並列化手法が提案されている [5]．しかし，物体並列法や空間並列法は，画素並列法に比べてデータの通信量が大きく，計算資源と比例して通信量も増加するため，台数効果が得られにくい．そこで本研究では，画素並列法を採用する．

CUDA で画素並列法を実現するため，計算資源の最小の管理単位であるスレッドと，各画素を 1:1 で対応させて実装する．さらに，1 つのブロックと水平方向の画素 1 行を対応させる．このブロックを垂直方向の画素の数分用意し，これを 1 つのグリッドとする．また，計算に用いるデータは GPU 上のメモリに格納する．レイトレーシングの計算において，物体情報は何度も参照されるが，データ数が大きいため高速なレジスタには格納できない．そのため，指示がなければ，物体情報は大容量なグローバルメモリに格納される．この物体情報を格納するメモリを変更することで，さらなる高速化が見込める．本研究では，物体情報の格納領域に，シェアードメモリおよびコンスタントメモリを利用した実装も行い，計算時間との関係を評価する．

5　評価

本研究で提案した手法の評価のために，画素数を 1000 × 1000 とし，物体数を 1000 から 10000 まで 1000 ずつ増加させ，交点計算とシェーディングにかかる時間を計測した．評価に用いた環境を Table 2 に示す．

CPU と GPU の速度比較の結果を Fig. 2 に示す．評価の結果，GPU による高速化が確認できた．また速度向上比は，球体の増加と比例することがわかり，球の数が 10000 の時には，約 46 倍の速度向上比が得られた．これは，処理に GPU の初期化やメモリの確保，解放などの一定のオーバーヘッドが存在するためと考えられ，負荷が増えるほど GPU の利用が有効であることがわかる．

Table 2　評価環境

CPU	Core i5 2400K (3.1GHz)
Memory	8GB
GPU	GeForce GTX 460
OS	Linux 2.6.38 x86_64
CPU code Compiler	gcc 4.4.5-15ubuntu1
GPU code Compiler	CUDA toolkit 4.0

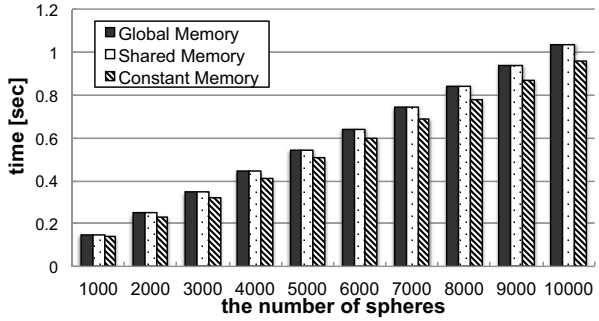

Fig. 3　GPU のメモリと計算時間の関係

GPU で，球の情報を格納するメモリを変更した際の，速度比較の結果を Fig. 3 に示す．Fig. 3 から，グローバルメモリと比べ，シェアードメモリを使用した場合には，大きな速度の変化はなく，コンスタントメモリを使用した場合には高速化することがわかった．また，各スレッドは物体情報に同一の順序でアクセスする．そのため，複数のスレッドによる，同一のデータへのアクセスが同時に起こりやすいといえる．しかし，シェアードメモリ上の同一データへは，複数のスレッドから同時にアクセスすることができず，待ち時間が発生する．そのため，シェアードメモリも高速性を活かすことができず，高速化できなかったと考えられる．一方で，コンスタントメモリはキャッシュを有し，同一データへのアクセスが連続した場合，キャッシュ上のデータに高速にアクセスできるため，高速化されたと考えられる．

6　まとめ

本研究では，3DCG の生成手法の 1 つであるレイトレーシングを，GPU で高速化する手法を提案した．描写する各画素毎に並列性を持つ，レイトレーシングの特徴を生かした画素並列法を GPU で実装することにより，CPU 実行に比べて最大で約 46 倍もの高速化可能であることを確認した．また，物体情報の保持に用いる GPU 上のメモリの種類を変更し，キャッシュを有するメモリの使用時に，更に高速化できることを確認した．

参考文献

1) 大林史明, 冨田和宏, 服部瑶子, 河内美佐, 下田宏, 石井裕剛, 寺野真明, 吉川榮和. オフィスワーカのプロダクティビティ改善のための環境制御法の研究 －照明制御法の開発と実験的評価. ヒューマンインタフェースシンポジウム 2006, 2006.

2) 田中慎吾, 三木光範, 廣安知之, 吉見真聡. 照度分布を基に個別照度環境を実現する知的照明システム. 同志社大学大学院工学研究科修士論文, 2010.

3) 湯川英宜, 平野敏行, 西村康幸, 佐藤文俊. Gpu によるタンパク質高精度静電ポテンシャル計算の高速化. 生産研究, 2009.

4) NVIDIA. Compute Unified Device Architecture Programming Guide, 2007.

5) 平田貴光, 安部毅, 大野義夫. レイトレーシングの並列化. 情報処理学会研究報告. グラフィクスと CAD 研究会報告, 1994.