様　式　Ｃ－１９、Ｆ－１９－１、Ｚ－１９　（共通）

科学研究費助成事業　　研究成果報告書

令和　元 年　 ５ 月 ３１ 日現在

機関番号：２２６０４
研究種目：基盤研究(B)（一般）
研究期間：2014～2017
課題番号：２６２８２００９

研究課題名（和文）GPGPUを活用した超高速CGレンダリングサーバーによるASPサービスの構築

研究課題名（英文）Construction of Super High Speed Rendering Server using GPGPU

研究代表者
笠原　信一（Kasahara, Shinichi）

首都大学東京・システムデザイン研究科・客員教授

研究者番号：００４３３１７８

交付決定額（研究期間全体）：（直接経費）　　12,300,000円

研究成果の概要（和文）：GPUの活用による実用的な超高速CGレンダリングシステムを開発すると共に、開発システムをネットワーク経由で利用できる仕組みを構築した。超高速レンダリングのために、計算負荷の軽い箇所はCPUで計算し、計算負荷の重い箇所のみをGPU化し集中的に高速化するという、CPUとGPUのハイブリッドレンダリングのアルゴリズムを考案した。このアルゴリズムをシステムに実装し、様々なタイプのデータに適用することで、レンダリング処理が従来のCPU処理に対して数百倍高速化することを実証した。これによって、これまで1時間を超えるような計算時間を要していた高品質画像を瞬時に生成することを実現させた。

研究成果の学術的意義や社会的意義
建築業界においてCGは設計段階の建物の可視化ツールとして活用されているが、近年のCG表現技術の向上によって画像生成の計算時間も増加し、設計者からのレンダリングの高速化の要望は非常に高い。
本研究開発システムにより、CGレンダリングを超高速に処理し高品質画像を瞬時に生成することを実現した。さらに本システムをネットワークを介してユーザーが手軽に利用できる仕組みも構築し、開発システムの実用性と社会への技術普及の可能性を示した。
レンダリングの超高速処理は、単に時間が短縮されるだけでなく、設計段階におけるCGの利用形態が大きく変化し設計プロセス自体の変化や設計品質向上をもたらすことが期待される。

研究成果の概要（英文）：We have developed a practical super high speed Computer Graphics rendering system whose calculation speed is more than hundreds times faster than ordinal rendering systems by CPU calculation, and going to provide the system through internet for designers to use high end Computer Graphics images quickly and easily at design study.
We propose a hybrid rendering method of CPU calculation and GPU calculation. We extract small process of heavy load calculation from total rendering process, and apply GPU calculation to the heavy load process only. By calculating the rest using CPU huge branched operations by GPU is avoided. We implemented these proposed method for GPU calculation to a global illumination rendering software based on path tracing method. We timed rendering calculation of several test images by the GPU version of the software with a server system consisted of 7 GPUs, and got result of faster hundreds times to thousands times than CPU version of the same software.

研究分野：コンピュータグラフィックス

キーワード：コンピュータグラフィックス　GPGPU　並列処理

様　式　C－１９、F－１９－１、Z－１９、CK－１９（共通）

１．研究開始当初の背景

建築業界において、CG によって設計段階の建物を可視化することにより、実際に建物を建てる前にリアルな臨場感で設計検討ができ、さらに設計期間の短縮やコスト削減にもつながっている。

近年の CG 表現技術の向上によって実物と区別のつかない画像が制作されるようになってきたが、それに比例して画像を生成するための計算時間も増加の一途をたどっている。頻繁なやり取りの要求されるデザイン検討段階での活用においては、すばやい画像の生成が不可欠であり、設計ユーザーからのレンダリングシステムの高速化の要望は非常に高い。

現在のパソコンの計算処理速度は CPU の性能に依存しているが、CPU 性能はこれまでの進歩でほぼ極限に達しており今後は大きな性能向上は期待できない。そのような状況の中で GPU の活用に近年期待が集まっている。GPU はもともと図形処理専用での特殊なハードウエアであったが、近年、GPU を使っての汎用的な計算処理（GPGPU: General- purpose computing on graphics processing units）をおこなうためのプログラミング開発環境が整い始めている。

２．研究の目的

GPGPU 技術を活用し、CPU で動く従来の CG レンダリングソフトウエアに比べて数百倍の速度で処理する実用的な超高速 CG レンダリングシステムを開発する。

GPU を活用したハードウエア構成にすることで、従来のスーパーコンピュータに匹敵する性能を、１つの筐体に収まるコンパクトな形体で実現させる。可搬性があり場所を選ばずに稼働でき、かつ同程度のスーパーコンピュータに比べて 1/10 程度の価格の汎用性に富んだハードウエアを構築する。

ASP(Application Service Provider)サービスの仕組みを構築し、開発した CG レンダリングシステムをユーザーがネットワーク経由で手軽に利用できるようにする。

３．研究の方法

(1) 実用的 CG レンダリングソフトウエアの開発

これまでの研究開発により、本研究代表者はベースとなるレンダリングソフトウエア（CPU で稼働）を保持している。このソフトウエアを GPGPU 化する前段階として、並列処理に相性の良いアルゴリズムやデータ構造に変更する作業を実施する。

(2) GPGPU プログラミングの最適化手法の検討

GPGPU を活用したプログラミング技術は歴史が浅く手法が確立されていない。GPGPU 化はプログラミング方法によって処理速度が大きく左右されるので、GPU ハードウエアの性能を最大限に引き出すための最適なプログラミング技法を調査・実験により解明し確立する。

(3) ソフトウエアの GPGPU 化コンバージョン作業

CPU で稼働する CG レンダリングソフトウエアを、GPU で計算処理するようコンバージョンことによって、CPU での計算処理の数百倍程度の高速化の実現を目指す。

(4) 超高速レンダリングサーバーによる ASP サービスの構築

複数 GPU を搭載したネットワーク経由で画像制作のためのデータを受け取って、GPU で瞬時に画像を制作し、ネットワーク経由で依頼元に送り返す、という一連の作業を行う超高速計算処理サーバーを構築する。高速処理のためには、演算処理の高速化だけでなく、高速なデータ入出力アクセスの機能が不可欠になる。本研究によって開発した超高速 CG レンダリングシステムを、サーバーとしてネットワークに接続し、ネットワーク経由で多くのユーザーが手軽にアクセスして、CG 制作ができる ASP 形態のしくみを構築する。

４．研究成果

(1) 実用的 CG レンダリングソフトウエアの開発

本研究代表者が保持しているレンダリングソフトウエア「Frend」（図 1）は、レイトレーシング法、パストレーシング法をベースにしたグローバルイルミネーションの CPU 版レンダリングソフトウエアで、建築設計用ハイエンドレンダリングの機能を備えている。このソフトウエアを本研究開発のプラットフォームとし、GPGPU 化する前段階として、並列処理に相性の良いアルゴリズムやデータ構造に変更する作業を実施した。その開発項目のリストを表 1 に示す。各開発項目ご

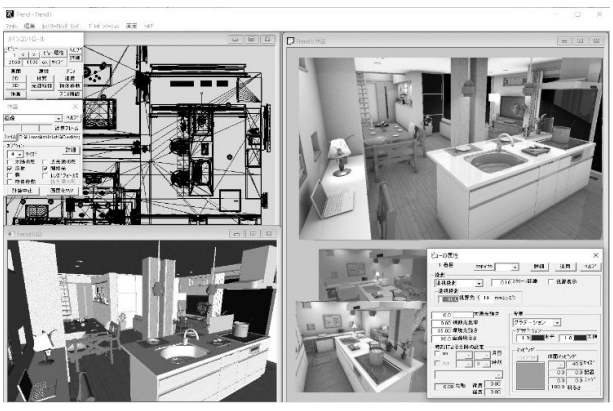

図１　Frend の操作画面

とに詳細ドキュメントが記録されている。

表 1　GPGPU 化の準備ための Frend の改良項目

コード	開発内容
140925	occlusion の交点計算の高速化
150323	間接光のパラメータ設定の目安
151201	F4a の Frend への統合
160608	間接光計算の最大画像サイズ拡張
160625	FrendBS: 入力 Breps 形式の切り替え
160921	struct Indirect のメンバー変数整理
160922	occlusion 計算の関数化
170630	alloc した配列の delete の整理

コード	開発内容
170901	関数に修飾子を付加
170905	struct 内の大サイズ配列を外す
170911	間接光計算で画像のスジのバグ修正
170916	画像のスキャンライン番号の変更
171001	3d ビルボード用画像作成機能の削除
171002	partial rendering 機能の削除
171008	boxel の配列を動的にとる
171009	struct Indirect の中の配列を動的に

(2) GPGPU プログラミングの最適化手法の考案

GPU は数千に及ぶコアによって並列処理計算を行う。個々のコアの計算速度性能は CPU に比べて劣るものの、膨大な数のコアで同時並行処理することで高速処理が可能になる。GPU は同一処理の大量反復が得意で高速性能を発揮するが、一方で分岐処理は不得意であり、分岐処理を含むアルゴリズムでの並列計算では極端に処理速度が落ちる。

CG レンダリングソフトウエアは様々なシェーディング表現機能の集合体であり、その実装は膨大な分岐処理で構成されているため、CPU 版プログラムをそのまま機械的に GPU 用にコンバージョンしても GPU の高速性能は引き出せない。GPU レンダリングのためには、アルゴリズムの膨大な分岐処理に対応できる並列処理アルゴリズムの開発が不可欠になる。

本研究では計算負荷の重い箇所を集中的に高速化するために、計算時間のかかる隠面消去処理をシェーディング処理から分離してあらかじめ GPU で高速計算し、CPU でのシェーディング処理時にその結果を合体させるハイブリッドレンダリング手法を考案した。その処理の流れを図 2 に示す。

マルチ GPU の処理に関しては、画像を細分割領域に分割したうえで、まず上から順に一領域ずつを各 GPU に割り当てて計算を開始し、各 GPU が計算終了したらその都度、次に待機している次の細分割領域を動的に割り当てていく方法を考案した。

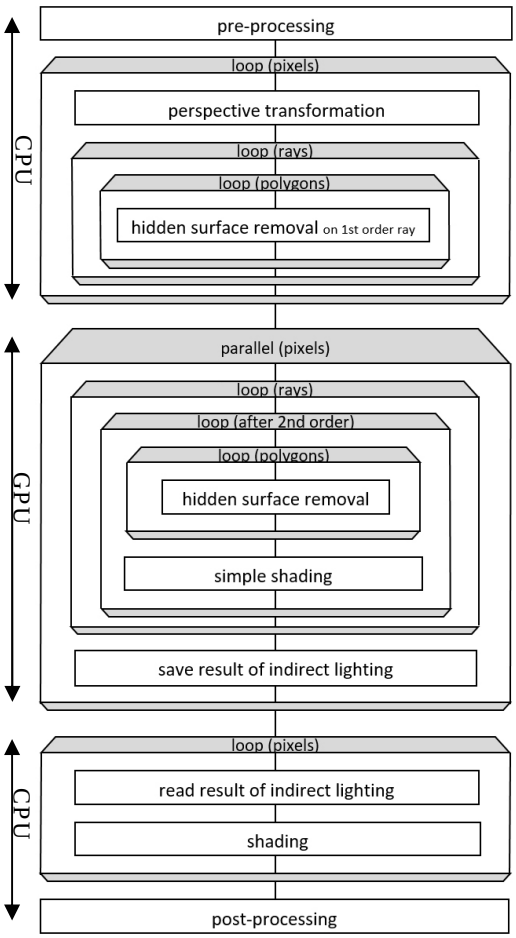

図 2　考案したレンダリング手法

(3) ソフトウエアの GPGPU 化コンバージョン作業

テストプログラムを使って、上記の方法により高速化が実現できることを確認したうえで、Frend に本手法をコンバージョンした。Frend は大規模なプログラム（コーディング約 60,000 行）であるので、作業ミスが発生しないように慎重に作業を進めた。その開発項目のリストを表 2 に示す。各開発項目ごとに詳細ドキュメントが記録されている。

表 2　Frend の GPGPU 化開発項目

コード	開発内容
140630	CUDA テストプログラム　GPUtest200
150712	CUDA 用のヘッダー設定
150713	CUDA のコンソールアプリの新規作成
150810	OpenMP によるマルチスレッドプログラミング
151218	CUDA デバッグのための Nsight の利用

コード	開発内容
180205	cuda でコンパイルするファイルを1つに連結
180206	動的配列を含まない struct を device 側に
180207	multi GPU のためのループ
180208	CUDA のコンパイラーで CPU 版ビルド
180209	乱数発生方法を CPU 版と GPU 版で同一に

180105	scanline を前処理に
180107	scanline の結果を直接光と間接光で共有
180108	col と occ の計算結果を配列での受渡しに
180109	struct Indirect のメンバー変数の整理
180110	間接光計算結果を配列での受け渡しに変更
180111	Rendering のパラメータの struct 化
180112	カーネル関数のピクセルごとに callcall
180114	鏡面反射の交点計算を前処理で
180115	Pre_reflection()のコードの整理
180120	マルチ CPU 化
180121	カーネル関数を call する関数を作成
180122	host と device で使われる関数は device 用に
180124	CUDA 化する関数を別ファイルに移す
180125	CUDA 化する関数プロトタイプを別ファイルに
180126	動的配列を含まない struct を device に
180127	struct 変数を host と device でやり取り
180128	brep_d を device 側に渡す
180129	acel_d scan_d indr_d refl_d rimg_d を device に

180211	CPU 版で mtrl_d[]を動的配列にする
180212	動的配列を含まない device 側 struct の設定
180213	動的配列を含む device 側 struct の設定
180220	関数名に_device_と_host_修飾子をつける
180221	device 側コーディングで strncmp()を使わない
180224	Conc_col_occ()を cpu 側で行う
180307	Indirect_col()と Pre_reflection()をコメント
180308	動的配列 rd1_d[]の device 側の設定
180309	Indirect_occ()を kernel 関数にする
180310	Indirect_col()を kernel 関数にする
180312	Pre_reflection()を kernel 関数にする
180315	後処理での antialiasing
180316	マルチ GPU 化前処理
180317	Indirect_occ のマルチ GPU 化
180409	Indirect_col のマルチ GPU 化
180410	細領域配分によるマルチ GPU
180412	nGPU を自動計算

(4) 超高速レンダリングサーバーによる ASP サービスの構築

開発したソフトウエアをユーザーがネットワーク経由で手軽に利用できる仕組みを実現させるために、複数の GPU を搭載したレンダリングサーバーを構築した。CPU は Intel® Xeon® CPU E5-2697 v2@2.70GHz、GPU は NVIDIA 社製 Quadro K6000（コア数 2880、デバイスメモリ 12GB、転送速度 208GB/s）を 7 台搭載している。GPGPU 開発環境は CUDA (Toolkit 8.0)、OS は Windows10（64bit 版）、コンパイラは Visual Studio Community 2015 Visual C++を使用している。このハードウエアは、１つの筐体の中にすべてが収まったコンパクトな形体で、可搬性があり、設置場所に柔軟性がある。

このサーバーに、開発した GPGPU 版 Frend をインストールしたうえで、ユーザーがネットワーク経由でデータを送り、結果の画像をうけとる Server Renderin の仕組みを構築した。その開発項目のリストを表 3 に示す。各開発項目ごとに詳細ドキュメントが記録されている。

表 3 Server Rendering の開発項目

コード	開発内容
140713	Server Rendering (SR)の作成
140907	SR で Batch Rendering をサポート
141104	SR：リアルタイムに受け取り (Realtime 法)
141105	SR のための FTP サーバー設定
141201	SR: 既存ファイル名への画像ダウンロード
141206	SR: 計算が完了した後ファイルを削除
141207	SR: work のプロジェクトを作成しサーバーに
160331	SR: 計算終了のジョブフォルダの削除

コード	開発内容
160402	SR: レンダリング計算の履歴保持
160403	SR: Frend.exe のファイアウォールへの登録
160509	SR: ストック方式での Panorama In サポート
160826	SR: 複数のプログラムの同時起動
160827	SR: シャットダウン機能の追加
171225	SR: 2 秒制限の対処
180421	SR: マルチ CPU に対応

(5) ASP サービスの実証実験

本開発システムの高速性と実用性の実証実験を行った。実験は CG レンダリング制作を業務としている（株）ワイドソフトデザイン社（兵庫県神戸市）に依頼し、東京に設置されたサーバーにネットワーク経由で神戸からアクセスし、高速性能や応答性能を検証した。実験には実際の業務相当の複雑なモデルが使用された。本実験で作成された画像の例を図 3、図 4 に示す。

図 3 実証実験で作成した画像「Kitchen」

モデル「kitchen」(34 万ポリゴン)のレンダリングに要した時間は 49 秒になり、従来の CPU 版 Frend で計算した 89 分に対して、110 倍の高速化になった。モデル「exterior」（40 万ポリゴン）のレンダリングに要した時間は 35 秒になり、従来の CPU 版 Frend で計算した 429 分に対して、71 倍の高速化になった。

図 4　実証実験で作成した画像「exterior」

本実験で使用した GPU（Quadro K6000）は、システム構築当時（2014 年）は最上位機種であったが、現在では旧世代となっておる。そこで参考のため、本開発システムを現在(2019 年)の最新機種の GPU（Quadro GV100）で構成した場合の高速性能を推定した。その結果、CPU 計算に対してモデル「kitchen」は 2,300 倍、モデル「exterior」は 700 倍になる推定結果が得られた。本研究開始当初に掲げた CPU 計算に対して数百倍のレンダリング処理高速化という目標が達成された。

これらの実験結果を通して、本開発システムが、超高速レンダリング性能を持つと共に、実用に耐えるシステムであることが確認された。

本研究では、レンダリング処理を要素に分け、各要素の計算特性に合わせて、マルチ GPU とマルチ CPU を組み合わせたハイブリッドレンダリングのアルゴリズムを考案した。これによって、レンダリングを超高速化に処理し、これまで数十分から 1 時間を超えるような計算時間を要していた高品質画像を瞬時に生成することを実現した。またこのシステムをネットワークを介してユーザーが手軽に利用できる仕組みも構築し、実用性を検証した。レンダリングの超高速処理は単に時間が短縮されるだけでなく、設計段階における CG の利用形態が大きく変化し、設計プロセス自体の変化や設計品質向上さえもたらす可能性がある。その可能性を実証するために、今後はこのシステムを企業ユーザーにテスト提供し、実務での利用実験を進めていく計画である。

5．主な発表論文等

〔雑誌論文〕（計　2 件）

1. 笠原信一，大久保寛：GPGPU を活用した超高速 CG レンダリングのための効率的並列化アルゴリズム，日本建築学会環境系論文集，第 84 巻，第 759 号，pp543-552，2019.5
   DOI http://doi.org/10.3130/aije.84.543
2. 吉村ももこ，笠原信一：点光源による半影表現のためのレイトレーシング拡張アルゴリズム，日本建築学会環境系論文集，第81巻，第724号，pp563-572，2016.6
   DOI http://doi.org/10.3130/aije.81.563

〔報告書〕（計　2 件）

1. CG レンダリングソフトウエアの並列処理コーディング方法に関する調査報告者、株式会社ワイドソフトデザイン、208 ページ、2016 年 2 月 17 日
2. GPGPU を活用した超高速 CG レンダリングの評価と可能性について、株式会社ワイドソフトデザイン、18 ページ、2019 年 1 月 31 日

6．研究組織

(1)研究分担者

研究分担者氏名：大久保　寛
ローマ字氏名：OOKUBO kan
所属研究機関名：首都大学東京
部局名：システムデザイン学部
職名：准教授
研究者番号（8 桁）：90336446

(2)研究協力者

研究協力者氏名：土肥豊和
ローマ字氏名：DOHI toyokazu