物理ベースのレンダリングを柔軟性を持って行えるアーキテクチャの開発

1. 背景

実用的な物理ベースレンダラの必要性

近年、レンダリングアルゴリズムは急速な進歩を遂げている。Kajiya らによって物理ベース CG の基盤となるレンダリング方程式が提唱されてから二十年、物理学的に正確な理論に基づいたレンダリングアルゴリズム（以下物理ベースアルゴリズムと記す）は実用の域に達している。

しかし、実際には現在プロダクションレベルで使われているのは古典的な手法が殆どである。レイトレーシング法やフォトンマッピング法の拡張を施したものも見受けられるが、物理学的に正確な演算をしているとは言い難いものが多い。しかし、既存の物理ベースのレンダラは、写実的な画像は正確に計算できても、アーティスティックな表現は難しいといった欠点がある。

レンダリングアルゴリズム開発環境

レンダリングアルゴリズムを設計する際には、具体的な評価を行うために実際にレンダラを開発する必要がある。しかし、レンダラは非常に規模の大きなプログラムのため、新規開発することは非常に多くの時間を要する。そうとはいえ簡易レンダラでは、モデルシーンはレンダリングすることができても、実践的なシーンをレンダリングすることは困難である。これにより、誤った性能評価が行われることが多い。

レンダリング向け分散計算システム

レンダリングは、莫大な計算を必要とする。そのため、ある程度の規模になると、専門にレンダーファームと呼ばれるクラスタが組まれ、分散計算が行われる。

しかし、既存の分散計算システムの実装には、特定のレンダリングアルゴリズムに依存するといった問題点も多い。

また多くのシステムは、非常に信頼性の高く、かつ性能が同一の計算ノードを要求する。しかし最近の PC の低価格化の傾向をみていると、低価格・低信頼性のノードを大量に活用したほうが有効となる場合があると考えられる。

2. 目的

背景で述べたような問題を解決するレンダリングアーキテクチャを設計し、プロトタイプ実装を行う。また、レンダリングに特化した分散計算システムを開発する。

3. 開発の内容

統合型レンダリングアーキテクチャ

レンダリングアルゴリズムは、物理ベースアルゴリズムと古典的アルゴリズムの2つに大別できる。物理ベースアルゴリズムは、物理学（光学）に基づき、正確なシミュレーションを行うのが特徴で、写実的な画像を得意とする。古典的アルゴリズムは Appearance-based とも呼ばれ、結果ありきで計算式が設計されている。物理学的な根拠には欠けるが、幅広い表現が可能なのが特徴である。

いままでは、異なるタイプの画像を生成するためには、この2種類のレンダラを使い分け

る必要があった。

そこで、本プロジェクトでは、物理ベースレンダラに古典的レンダラを統合し、あらゆるタイプの画像が生成できるようなシステムを設計・実装した。物理レンダラをベースにする理由はいくつかあるが、最大の理由は計算の安定性である。古典レンダラの計算式はエネルギー保存則を満たさないものも多く、計算が発散してしまうことが多々ある。物理レンダラを基盤にし、放射照度計算をすべて物理レンダラ側に任せてしまうことで、この問題を避けることができる。

しかし、物理レンダラだけでは、物理法則に逆らうような特殊効果や誇張表現をうまく実現できない。そこで、エフェクト部分の処理を古典的レンダラベースのエフェクタに分離した。これにより、計算の安定性、生成画像の写実性、表現の多様性をすべて実現することができる。

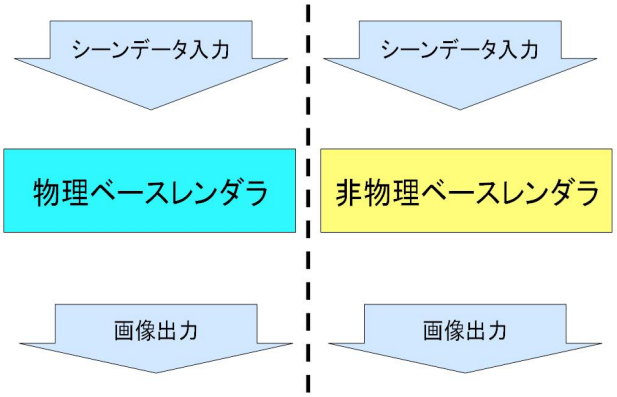

図１：既存レンダラブロック図

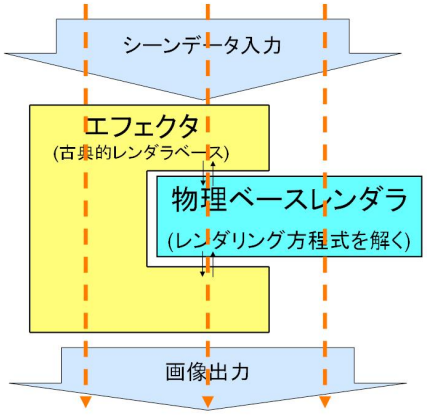

図　２：提案レンダラブロック図

“PBS”(Physically-based Shader)システム

古典的レンダラには、複雑な材質を表現するのに「シェーダ」と呼ばれるスクリプトが使われる。これにより、自由に材質の見た目が表現できるようになっている。

しかし、これを前述の統合型アーキテクチャにそのまま適用しようとすると、様々な問題が生じる。

まず、要求される情報の量が異なる。古典的レンダラでは視点方向からの応答のみを記述すればレンダリングが可能であったが、物理ベースレンダラはさらに厳密な関数定義を必要とする。（図5）

また、統合型アーキテクチャでは照明計算部とエフェクト部が完全に分離しているため、シェーダプログラムも材質の反射特性定義とエフェクト定義に適宜分離する必要がある。

PBS システムは、これらの問題を解消した統合型アーキテクチャ向けのシェーダシステムである。ツリー状の UI でシェーダをビジュアルプログラミングすることによって、統合型アーキテクチャの複雑なパイプラインを意識することなく、シェーダプログラムを記述することが可能になっている。

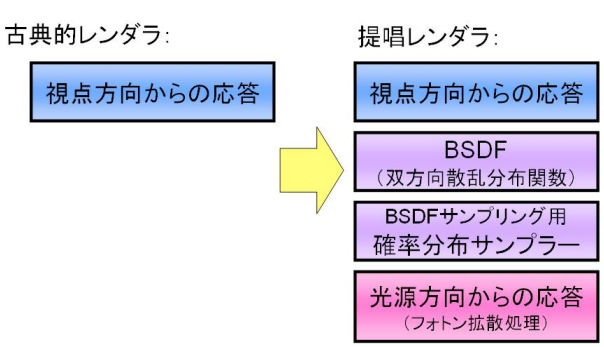

図　5：シェーダ情報量の比較

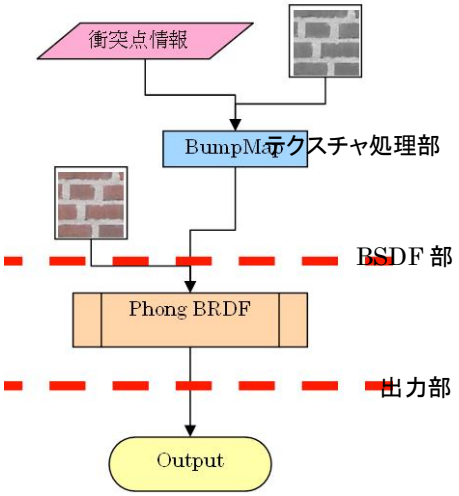

図　6：シェーダ分解

レンダラ向け分散計算ライブラリ: libpolatsk

レンダラ向けに最適化した分散計算ライブラリ“libpolatsk”を開発した。

libpolatsk は一般的な分散計算システムとは異なり、タスクシステムを基盤にしている。ユーザーからは、ネットワーク上に一つのタスクプールがあるように見え、そこにタスクを投げ込むことで最適なノードに自動的に転送され、実行される。

タスクの処理ノード選択には、タスクの粒度に応じて二種類のアルゴリズムを使い分けている。粒度が細かいタスクに対しては、単純にノード性能で重み付けをした乱数によって決定するアルゴリズムを採用し、粒度が粗いタスクに対しては、実行時間予測によるヒューリスティクスを使ったアルゴリズムを採用している。粒度が細かいタスクは、処理時間が短いため、予測が仮に失敗しても問題は少ない。それよりも予測自体の高速性が求められる。これに対し、粒度の粗いタスクは、処理時間が長いため、多少時間をかけても正確な予測を行う必要がある。二種類のアルゴリズムを用意することで、両方のケースに対応している。

タスクシステムの実装にあたり、Coroutine（ユーザレベルスレッド）の実装が必要になった。短期間で可搬性のある実装を書くのは困難と判断し、Io 言語の VM から該当部分をライブラリとして抽出し、C++言語のラッパーを書いた。

また、libpolatsk は非対称な動的ネットワーク環境に対応している。これは、低信頼性のノードでレンダーファームが構成された場合を考慮している。PC の高性能化、低価格化の傾向の中、こういった環境に対応することは重要であると考えられる。

“nytr”レンダラ

上記の統合アーキテクチャ、PBS システム、分散計算システムを実装したレンダラ“nytr”を開発した。

開発には、C++言語を用いた。Microsoft Windows XP, Linux, Mac OS X の主要プラットフォームで動作する。可搬性が高いコードになっているため、他プラットフォームへの移植も容易である。

現在、nytr レンダラは GNU Public License の下、フリーで公開している。

4. 従来の技術との相違

統一型アーキテクチャ・PBS システム

このアーキテクチャを採用したレンダラでは、従来型の単機能レンダラとは違い、全ての種類の3D 画像を一つのレンダラプログラムで生成可能である。これにより、データ資産の活用や、レンダーファームの構築が容易になる。

また、このアーキテクチャでは、物理ベースアルゴリズムと古典的アルゴリズムの併用が可能なため、より幅広い用途で物理ベースアルゴリズムの写実性・計算安定性が享受できる。特に、実写と組み合わせた特殊効果では、古典的レンダラと比較してきわめてリアルな画像を生成することができる。

レンダラ向け分散計算システム

MPI や GridRPC といった他の分散計算ライブラリと違い、今回開発した分散計算ライブラリ libpolatsk はレンダラ向けに最適化している。具体的には引数サイズの小さなタスク RPC コールが非常に高速に行えるようになっている。

分散 RPC ライブラリである OmniRPC[1]とパフォーマンス比較を行った結果（表1）、リモートノード上の sin 関数の平均呼び出し時間は libpolatsk の方が約 1.38 倍早いことがわかる。

表1：libpolatsk、OmniRPC で 100 回 sin()関数の RPC コールを行った場合の処理時間
（OmniRPC では同一 rsh コネクションを使用できた場合）

	試行1	試行2	試行3	平均
Libpolatsk	381msec	392msec	373msec	**382msec**
OmniRPC	532msec	524msec	528msec	**528msec**

5. 期待される効果

nytr レンダラの登場によって、高品質な物理ベースの演算と芸術的表現を組み合わせた新しい表現が可能となった。これを活用した新しいアート表現の登場が期待される。また、市販ソフト並の機能を持つレンダラをフリーで提供することにより、アマチュア 3DCG 作家の活躍が期待できる。

レンダラ用分散計算ライブラリ libpolatsk は、nytr レンダラから独立しており、他アプリケーションから使用することもできる。nytr レンダラ以外の、又はレンダラと特性が近い他アプリケーションへの応用が期待される。

6. 普及の見通し

現時点では、ソースコードのみをウェブサイト上で配布している。近日中にインストーラ等を配備し、導入を容易にする予定。

7. 開発者名（所属）

上野　康平（千葉大学　理学部先進科学プログラム）

（参考）開発者URL
http://nyaxtstep.com

[1] OmniRPC ウェブサイト：http://www.omni.hpcc.jp/OmniRPC/index.html.ja