

ソフトウェア RenderingEngine の高機能化と高速化

065725F 金城裕 指導教員：河野真治

1 概要

近年、CPU の消費電力増加や、クロック周波数対効果の停滞により、CPU コア数は増加傾向にある。コア数の増加にともない処理速度の向上が見込める。しかし、CPU のコア数が増加すると、コア数に見合った並列プログラミングを行う必要がある。

並列プログラミングを行う際、Amdahl 則より、並列化を意識してプログラミングしなければならない。しかし、並列化を問題毎に毎回考えるのは手間がかかる。そこで並列プログラミングを裏でサポートしてくれるソフトウェアを開発する。

研究では、Cell の並列プログラミングサポートフレームワークを開発する。当研究室で開発した Cerium は現在十分な並列化、実行速度、機能をサポートできていない。Cerium、特に RenderingEngine 部分を改良し、信頼できる並列化サポートフレームワークを目指す。

2 Cell

Cell は「ヘテロジニアス・マルチコアプロセッサ構成」を採用し、1 基の制御系プロセッサコア (PPE:PowerPc Processor Element) と 8 基の演算系プロセッサコア (SPE:Synergistic Processor Element) で構成される。各プロセッサコアは、EIB (Element Interconnect Bus) と呼ばれる高速なバスで接続されている。また、EIB はメインメモリや外部入出力デバイスとも接続されていて、各プロセッサコアは EIB を経由してデータアクセスをおこなう。

この 2 種類の CPU をプログラマ自身が用途に合わせて適切に使い分けるように考慮する必要がある。

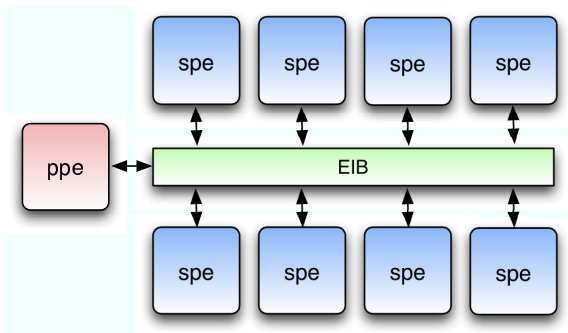


図 1: Cell プロセッサの構成

3 Cerium

当研究室では Cerium と呼ばれるゲーム開発フレームワークがあり、以下の 3 つの要素から構成されている。

- SceneGraph
- Rendering Engine
- Task Manager

Cerium は独自に RenderingEngine を持つ。ゲーム中のオブジェクトの振る舞いやルールは SceneGraph で管理し、それらの動きやレンダリングの処理を動的に SPE に割り振るカーネルとして、TaskMnager が用いられる。TaskManager は、Task と呼ばれる、分割された各プログラムを管理する。Task の単位はサブルーチンまたは関数とし、Task 同士の依存関係を考慮しながら実行していく。現在 Cerium は linux,macosx 上で動作し、コンパイル方法によって Cell の spe を使うかどうかを選択できる。

4 TaskManager を使った例題

TaskManger を使った例題として WordCount を実装した。input された file を mmap でメモリにマッピングし、その data を分割。各 SPE に、WordCount の Task と分割された data を渡す。SPE は渡された data を wordcount し、指定された output 領域に書き込む。全ての wordcount が実行し終わるとその結果を ppe 側で集計し、結果を出力する。その流れを以下に示す。

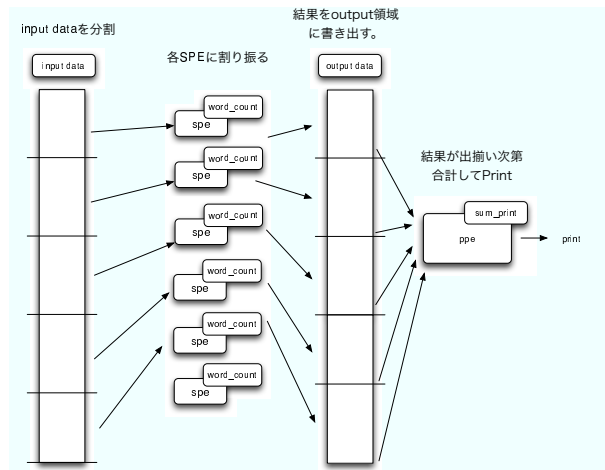


図 2: WordCount の Task 生成、実行の流れ

5 RenderingEngine

RenderingEngine では、SceneGraph から、実際に表示するポリゴンの抽出、ポリゴンから Span の生成、Span に RGB をマッピングし描画する部分と 3 つに分ける事ができる。ここでいう Span とは、ポリゴンに対するある特定の Y 座標に関するデータを抜き出したものである。

5.1 高機能化

RenderingEngine の高機能化に関して、以下のような案を提案する。

5.1.1 光源

RenderingEngine で、未実装だった光源の計算を実装した。各オブジェクトには自身の座標や親子関係などの情報を持っており、その中に法線がある。法線と光のベクトルとの内積を rgb にかけて算することにより光の計算を行っている。以下に光源の計算をした画像を示す。

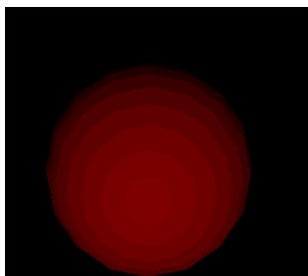


図 3: 光源計算をした描画画像

現在、光源は ppe だけを使った fifo 形式のみに実装されている。

5.2 高速化

RenderingEngine の高速化に関して、以下のような案を提案する。

5.2.1 描画ルーチン

Cerium の RenderingEngine は描画する対象がない部分も毎フレーム描画計算をしている。それでは計算する必要のない部分も計算してしまい、無駄な時間が生じる。そこで、描画する対象がない部分の計算は行わないようにすれば高速化が望める。また画面に背景を描画する場合は、オブジェクトとは別に背景専用のルーチンを設け高速化を計る。さらにオブジェクトが重なり描画する必要のない部分は描画計算を行わない。

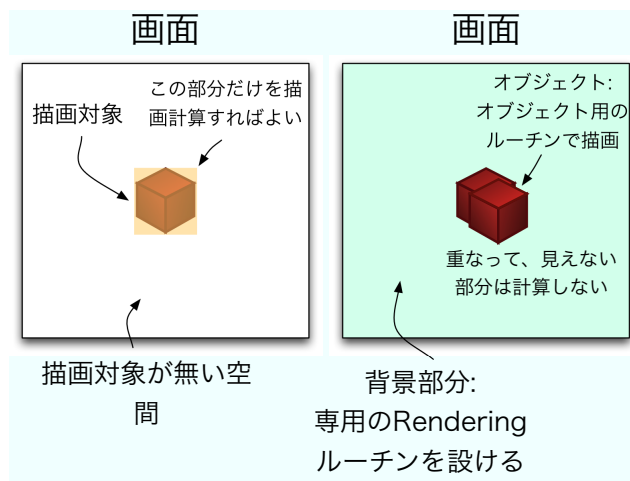


図 4: 描画計算の範囲

5.2.2 SceneGraph から Span を作るまで

Cerium では SceneGraph から Rendering に至るまでに以下の図で示すような流れがある。

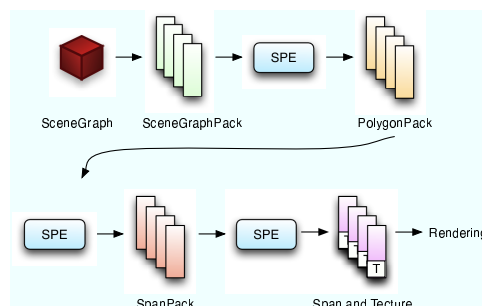


図 5: SceneGraph から Rendering するまでの流れ

以上の工程でそれぞれ SPE に Task を渡し、処理を行っているが、Task は一つにまとめられて一基の SPE で実行されている。この部分の Task を複数に分割、複数の SPE に割当て並列処理を行うことによって高速化を計る。

参考文献

- [1] 宮國渡 ”Implementation of Fine-grain Task Manager for Cell” 平成 20 年度 学位論文 (修士)
- [2] 多賀野海人 ”並列プログラミングを用いたゲームフレームワークの設計と実装” 2008 年 卒業研究中間報告資料
- [3] fixstars: <http://cell.fixstars.com/ps3linux/index.php/> メインページ