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

055734A 多賀野海人　指導教員：河野真治

1　はじめに

プログラミングで Cell を用いる場合、データの転送やタスクの生成等のこれまで学んだことのない技術が多く存在する。これにより、学生実験の短期間では新しい技術を学ぶことに多くの時間を取られ、ゲームの完成度が向上が困難である。

本研究では、PlayStation3 上でゲームプログラミングを行う際に、Cell の性能を活かしながら、アーキテクチャに依存する記述を排除した、並列プログラミングを目的としている。

2　Cell , SPU を用いたプログラミング

Cell[1] は 1 基の PowerPC Processor Element (PPE) と 8 基の Synergistic Processor Element (SPE) からなる非対称なマルチコアプロセッサであり、高速リングバスで構成されている。

SPE は、LS (Local Store) という 256KB のメモリを持つ。本研究で用いた PS3Linux では、6 個の SPE を制御することができる。

SPE には 256KB しか搭載されていないという問題点がある。これは SPE 内で動作させるコードなども含めてである。メインメモリからデータを持ってくることも可能だが、SPE が直接メインメモリにアクセスすることはできず、メインメモリにアクセスするには DMA を用いる。

DMA (Direct Memory Access) 転送とは、CPU を介さずに周辺装置とメモリとの間でデータ転送ことで、Cell の場合はメインメモリと LS 間でデータの転送を行う。手順としては以下の様になる。

1. SPU プログラムが MFC (Memory Flow Controller) に対して DMA 転送命令を発行する。
2. MFC が DMA Controller を介して DMA 転送を開始。この間、SPE プログラムは停止しない。
3. DMA 転送の終了を待つ場合、SPE プログラム内で転送の完了を待つ。

SPE が直接メインメモリにアクセスできないことと、DMA 転送の待ち時間が SPE でプログラミングを行う際の大きな問題となる。問題の解決策として、DMA の待ち時間の間にも DMA 転送を必要としないような処理を行い、パイプラインを組むまた、SPE を複数用いた並列処理で DMA の待ち時間を隠す方法がある。

Cell 用いたゲームプログラミングを行うために、我々の研究室では Cerium を用意した。Cerium は、オブジェクトのデータやその振る舞い、またはゲームのルールなどゲームを構成する場面 (Scene) を木構造で持つ Scene Graph に代表される Rendering Engine、そしてそれらの実行単位を Task とし、動的に全てのコアが動作する様な割り振りを行うカーネル TaskManager で構成されている。

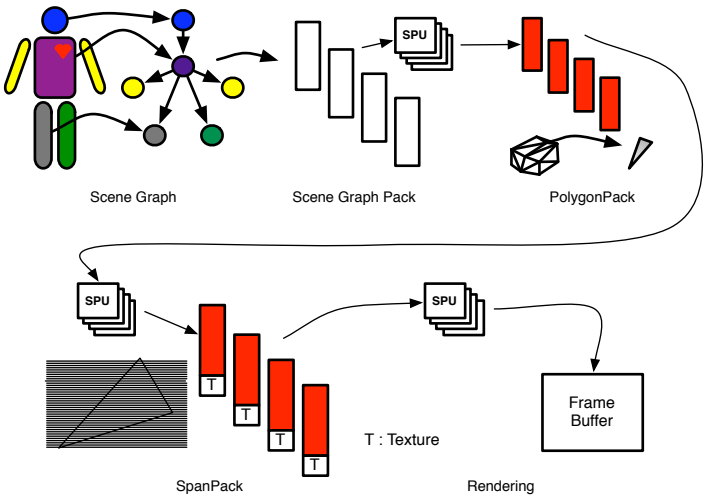

図 1: Cerium の構成要素

Rendering Engine 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 つに分けることが出来る。

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。図 1

3　Rendering 部分の高速化

SPE を用いて計算を行う場合には、その処理（ソースコード）とデータを SPE の LS に転送して行う。しかし、SPE

の LS は 256KB しかないので、一度に Texture データを転送すると容量を超えてしまう可能性がある。

そこで、描画に必要な Texture データを分割して転送するという手法を用いる。分割したデータを Tile とし、分割単位は 8 x 8 pixel とする。図 2 [2] 図 1 の SpanPack に含

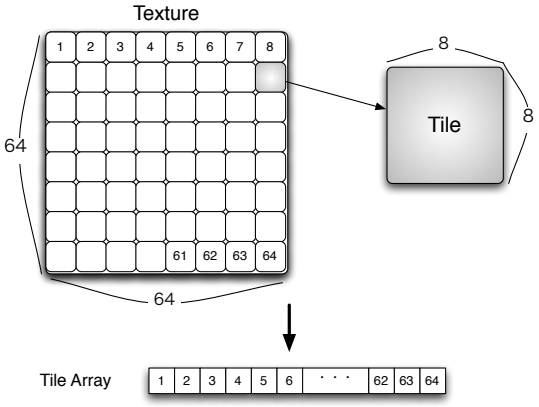

図 2: Texture の分割 (Tile)

まれる Texture のアドレスはこの配列を指している。描画に必要な Tile を計算し、メインメモリから DMA 転送し、pixel の情報を取り出す。

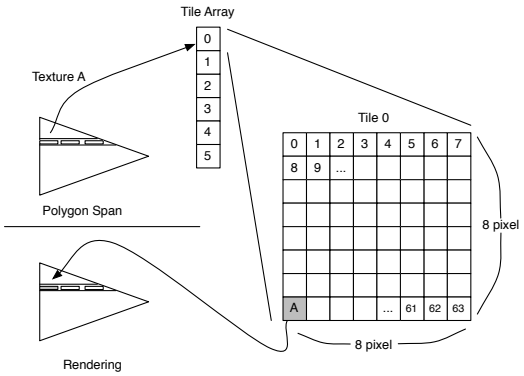

図 3: Span からの Texture の計算

1. Span の 持っている Texture A の座標から A が含まれる Tile を求める。
2. Tile ID と Texture の TOP Adress から、描画に用いられる、Texture の pixel adress を求める
3. Texxture の座標から、pixel が持つ RGB 値を返す。

現在、SPE に転送された Tile は SPE 上に 128 個まで保存される。ハッシュを用いて、既に SPE に Tile が存在するかどうかを調べて、DMA 転送の無駄を省いている。

また、描画されるオブジェクトが遠くにある場合、そのままの大きさの Texture は必要ないので、縮小サイズの Texture (Scale) を用意し、Span の長さによって描画に必要な Texture を取捨選択することにする。Texture は縦横ともに 1/2、1/4、1/8 と、2 分の 1 ずつ縮小させる。

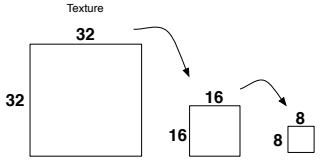

図 4: 縮小画像 (Scale)

以下に最大限に Scale を用いたときの実行速度の比較を示す。

表 1: Scale を用いることによる実行速度の比較

	Scale なし (FPS)	Scale あり (FPS)
Mac OSX	7.0	8.5
PS3Linux (SPU 1)	4.3	5.6
PS3Linux (SPU 6)	10.8	13.5

- FPS とは Frame Per Second の略で、1 秒間に何回画面を書き換えたかを表している。
- 比較に使用したオブジェクトの総 Polygon 数は 19860、Texture は合計 10 枚 (512x384(2) 8x8(3)、616x123(4)、1024x768(1))

表 1 から、Texture の Scale 選択により Playstation 3 上では 30% ほど実行速度が向上していることが分かる。これは、Scale (縮小 Texture) を用いることによって描画に必要な pixel データのヒット率が上昇していることが要因である。

4 今後の課題

現在、Polygon から Span の生成、描画の処理を SPE を用いて行っているが、さらに SceneGraph から Polygon 生成の処理、SPE 上のコードの入れ替えを実装する。

また、評価をする際に、FPS だけではなく、特定の命令数などもっと細かいレベルのデータを用いて検証を行う。

参考文献

[1] Sony Corporation.

Cell BroadbandEngine ™ アーキテクチャ, 2006

[2] Wataru MIYAGUNI.

Cell 用の Fine-Grain Task Manager の実装, 2009