芸術科学会論文誌 Vol. 6 No. 3 pp. 167 - 178

GPU の先駆的利用の研究動向と将来像

Research trend and future vision in the advanced application of GPU

新庄貞昭 Sadaaki SHINJO (1), 高橋誠史 Masafumi TAKAHASHI (2) ,
木村秀敬 Hidetaka KIMURA (3), 白井暁彦 Akihiko SHIRAI (4) ,
宮田一乗 Kazunori MIYATA (5)
(1,2,3) 北陸先端科学技術大学院大学 知識科学研究科 JAIST, School of Knowledge Science
(4) ENSAM Presence & Innovation Laboratory
(5) 北陸先端科学技術大学院大学 知識科学教育研究センター JAIST, Center for Knowledge Science
(1) s0550608@jaist.ac.jp, (2) masa-t@jaist.ac.jp, (3) hide.kimura@gmail.com,
(5)miyata@jaist.ac.jp

概要 本解説では, グラフィックプロセッサ(GPU)を取り巻く環境やその利用法, 開発環境の現状, および GPU のコンピュータビジョンへの適用事例を紹介し, GPU コンピューティングの可能性を探る. GPU はプログラマブルシェーダの導入後, シェーダのバージョンアップにあわせてスペックの向上が行われてきた. また CPU と比較して制約は多いが, プログラミングを工夫することにより速度向上が可能である. GPU は幅広く利用されているが, 特筆すべき適用事例としてコンピュータビジョン, 物理シミュレーションがあり, 汎用計算においては分散コンピューティングによるナノテクへの応用も報告されている. 本解説では, GPU に加えて最近の CPU アーキテクチャの革新であるマルチコアプロセッサ CPU についても触れるとともに, 最新 OS のひとつである Windows Vista でのグラフィックユーザインタフェースへの利用にも触れ, 今後の応用の可能性を探る.

Abstract In this paper, we introduce background, application and current environment for software development of the graphic processor (GPU). In addition, we explore possibility of the GPU computing. After the programmable shader was introduced into the GPU, the specification of GPU has been improved with the version up of shader. Although GPU has more restriction than CPU, it is able to speed up by fully programming. GPU has wide application. Especially remarkable applications are seen in the Computer Vision and the physics simulation. And it is reported that GPU is applied in the nano-technology by the distributed computing. In addition to GPU, we touch some topics of multi-core processor technologies that are innovative architectures on the latest CPU, and some usages of graphic user interfaces in the "Windows Vista" which is the one of the latest OS. Finally, we explore possibility of the future applications.

グラフィックプロセッサ, GPGPU, シェーダモデル, 並列計算, マルチコア
Graphic Processor, GPGPU, Shader Model, Parallel Processing, Multi-core CPU

1. はじめに

グラフィックプロセッサを汎用計算の目的で利用するGPGPU (General Purpose computation on GPUs)という研究は，もう 20 年以上も前から行われている[01, 02, 03]. しかし，それらの研究の多くは，自ら開発した専用プロセッサを利用したものであり，現在のような市販のハードウェアを利用したものは，ここ数年の歴史しかない[04]. GPGPU 研究の応用分野における幅の広さと有用性には目を見張るものがあり，今後の計算工学分野のメインストリームになりうる可能性が高いと考える.

なぜ，GPGPU の研究はここまで熱狂的とも言えるほどに盛り上がってきたのだろうか. その理由のひとつとして，CPU は互換性維持という足かせがあるのに対し，GPU はアーキテクチャの劇的な改変が許容されるため，進化が一足飛びに進んでいることがあげられる. もう一方で，GPU による映像処理能力の進歩はビデオゲームの表現力向上に直結しているため，常に新しい刺激を求めてやまない産業分野における，高い技術革新モチベーションの維持と，それに対価を惜しまないユーザが多数いることも大きな理由として考えられる.

GPU は，その初期よりピクセルパイプラインなどの大量のデータを並列処理することが可能なように設計されており，処理のスループットを格段に向上させることが可能であるが，すべての処理に対して高速化が望めるわけではない. 例えば，for ループの一括処理などは高速処理が可能であるが，データ間の相互依存があるような処理の高速化は望めない. また近年の動きとして注目すべき，HLSL などの高級言語開発環境の整備とGPU 特有のアーキテクチャ利用により，テクスチャデータやベクトル演算のように，特性上 CPU 上よりも扱いが，シンプルで高速な応用例も存在する.

以上のような背景から，本解説では，GPU を取り巻く環境やその利用法，開発環境の現状，およびGPU のコンピュータビジョンへの適用事例を紹介し，GPU コンピューティングの可能性を探る.

2. シェーダモデルの動向

GPU のプログラマブルシェーダは，2000 年にマイクロソフト社が DirectX 8.0 の仕様に盛り込んだことから始まり，対応するハードウェアは 2001 年に登場した. 最初のシェーダモデルとなるバージョン 1.1 から，現在までに 3 回の大きなバージョンアップが行われた. その変遷を，表 1 にまとめる.

表 1 DirectX におけるシェーダモデルの変遷

バージョン(登場年)	1.1(2001)	2.0(2002)	3.0(2004)	4.0(2006)
命令数(VS)	128	256	512 以上	64,000
命令数(PS, テクスチャ+算術命令数)	4+8	32+64	512 以上	64,000
定数レジスタ(VS)	96 以上	256 以上	256 以上	16×4,096
定数レジスタ(PS)	8	32	224	16×4,096
一時レジスタ(VS)	12	12	32	4,096
一時レジスタ(PS)	2	12	32	4,096
入力レジスタ(VS)	16	16	16	16
入力レジスタ(PS)	4+2（テクスチャ+色）	8+2（テクスチャ+色）	10	32
サンプラ数	8	16	16	16
テクスチャ数(VS)			4	128
テクスチャ数(PS)	8	16	16	128
フロー制御(VS)	なし	静的	静的/動的	動的
フロー制御(PS)	なし	なし	静的/動的	動的

OpenGL 環境においては，当初はメーカ独自の拡張方式による GPU 利用が「許可されていた」が，その後 ARB 拡張と呼ばれる API 経由に代わり，2004 年の OpenGL 2.0 で API からの GPU 利用が標準仕様となった．本章では，この流れにおける近年のシェーダモデルの変遷と，それに伴う GPU の変遷について述べる．

2.1 シェーダモデルの変遷

近年の GPU ハードウェアの進化にともない，それを利用するソフトウェアモデルを「シェーダモデル（Shader Model 以下 SM）」と呼んでいる．本来は，シェーダ(shader)とは，陰影計算におけるシェーディングを指すため，必ずしも語義として正しくはないが，GPGPU 的な利用では一般的に普及している呼称である．レンダリングパイプラインにおける頂点操作とピクセル操作をそれぞれ「頂点シェーダ（Vertex Shader 以下 VS）」，「ピクセルシェーダ（Pixel Shader 以下 PS）」と呼んでいる．GPU ハードウェアの革新にともなうバージョンアップによって，スペック上の数値としては，常に命令数やレジスタ数増加が行われてきた．加えてメジャーなアップデートにおいては，数値以外の「自由度」が向上してきたといえる．

SM2.0 への変更では，VS に静的なフロー制御が加わった．PS では，命令数の増加が行われた． SM2.0 には 2.0a と 2.0b というマイナーバージョンが存在し，命令数の増加や静的なフロー制御のサポートなど SM3.0 の仕様を先取りしたものがあるが，その分 GPU ごとの機能的な方言を認めることになった．

SM3.0 では，VS におけるテクスチャ参照が加わった．従来，VS においてテクスチャメモリや PS と連携した処理を行うことは難しかったが，これによりディスプレイスメントマッピングのような手法が可能になった．しかしながら，すべての GPU で実装が仕様では義務づけられていなかったため，同世代の GPU でもサポートされている機能に応じた個別の実装が必要になっている．

SM4.0 における大きな機能の追加は，それまでの VS と PS に加えて，ジオメトリシェーダ（Geometry Shader 以下 GS）というシェーダプログラミングのステージが加わることである．GS は，レンダリングパイプラインにおける，VS 処理後に施す三角形化（フラグメントと呼ばれる GPU 上の最小処理単位）などの処理をプログラマブルにしたものである．これにより GPU だけでシャドウボリュームの稜線処理や，頂点分割が可能になる．

2.2 シェーダモデルの変遷に対するGPUの変遷

SM は，ハードウェアに依存した実装であるため，SM のバージョンアップは GPU のモデルチェンジを意味する．すなわち，SM のバージョンアップ時には，GPU の命令ステップ，レジスタ数の増加，動的条件分岐のサポートなどが行われてきた．なお SM では処理速度は定義されず，非常に低速ではあるが CPU でのエミュレートも可能となっている．

SM2.0 世代の GPU では，PS に対する要求が増えたことから命令数の増加が行われた．VS では，フロー制御がサポートされた．PS におけるフロー制御は 2.0 では見送られたが，バージョン 2.0a と 2.0b では静的なものがサポートされている．さらに従来までの整数型のテクスチャ（RGBA 各 8Bit）から浮動小数点テクスチャ(32bitFloat)の利用が可能になった．

SM3.0 世代の GPU において，NVIDIA 製 GPU では VS においてテクスチャ参照が可能になったが，ATI 製の GPU ではこの機能は実装されていない．PS ではフロー制御のサポートが仕様に盛り込まれた．

SM4.0 世代の GPU では，従来までの VS や PS の演算ユニットを共通化し，汎用的なコアを実装したハードウェアが登場した．これにより頂点処理やピクセル処理の負荷に応じてシェーダの演算コアを効率的に利用できるようになった．

当初，命令数やレジスタ数に関して，SM ごとの GPU の仕様は，厳密に規定されていたが，実際の実装は緩やかに規定されていたため，同世代の GPU でも細かな面で実装が異なっていた．SM2.0 では，PS の演算精度が 32bit と 24bit のものが混在し，場合によっては同じシェーダプログラミングでも結果が異なることがあった．この問題は，SM3.0 において PS の演算精度が 32bit に規定されたため解決した．一般的な GPU プログラミングにおいて，プログラマはどのバージョンの VS,PS を利用するかをまず記述するため，過去の GPU プログラミング手法が将来において不具合を起こすという状況は考えづらい．また SM2.0～SM3.0 世代において存在した各

ベンダー間の非互換性問題はSMバージョンがあがるにつれ解消されている. SM4.0において残るベンダー間非互換の問題は, 浮動小数点テクスチャ利用時におけるフィルタ処理結果の非同一性問題である.

3. CPU vs. GPU 構造の変化

本章では, 従来存在した「CPU vs. GPU」といった比較・対立構造のここ数年における変化を述べる.

3.1 "GPU神話"の崩壊

GPUを汎用計算機として用いるGPGPU研究は, 以下のような難しい要素を含んでいる場合がある.

(1) GPUを利用することに科学的な新規性を見出せない
(2) 既存のCPU実装方式に比べ, 速度を含め大きな利点が見出せない
(3) GPUで実装するために処理に制限を加えている

これらの要素に加え, 2004年以降に新たに加わった難しい要素が「CPUのマルチコア化」である. 最新のCPUはHyper Threading (HT)技術やマルチコア技術により, 1つのプロセッシングユニットによって複数のCPUが存在するかのような処理が可能である. 具体的には処理の並列化やマルチスレッドプログラミングによって劇的な高速化が望めるだけでなく, OSそのもののメモリ管理パフォーマンス向上などの利益がある.

対して, GPUプログラミングはSMが上がるにつれ自由度は増してはいるが, この2年間で劇的な高速化が実現されたわけではなく, また基本的にメモリ制約やデータ構造などに制約が多い. つまりGPGPUの研究分野において「GPUで処理しているから速い」という研究アプローチは早くも崩壊している面もある.

3.2 ツール・プログラミング環境

前節で列挙したとおり, CPUでの実装に比べGPU上での実装には多くの制約がある. 当初はアセンブラを用いた機種依存性の高いプログラミングや, GPUプログラミングコンテストで優勝した「Frogger」 [05]で実装されたスタイルのように, 全ての処理をGPU化する方法が主流であったが, 近年では主流とはいえない. GPU環境をより簡単に利用できるようにする「ShaderMonkey」(ATI)や「FX Composer」(NVIDIA)といったツールがGPUベンダーから提供され, API面ではCgやHLSLといった言語環境や開発環境が多く登場してきている. これらは基本的に可読性の高いC言語的なテキストで記述され, 実行時に環境に合わせて動的コンパイルされ, GPUにアップロードされ実行されるため従来のアセンブラ手法と比較して速度的なデメリットはほとんどないといえる. またシェーダを変更するだけであれば, 実行プログラムのリコンパイルは必要なく, 開発や学習においては非常に強力な環境である. またプログラミング手法は従来のようなGPUのみに依存した方法ではなく, CPUで複雑な処理や互換性維持のための処理を記述し, 利用しているハードウェア環境, GPUに利点のある処理のみGPU上で実装するなど, 記述方法を工夫する. つまり「CPU vs. GPU」ではなく「CPU and GPU」といった方向が主流になりつつあるといえる.

3.3 パラダイムシフト

前述したように, 「CPU vs. GPU」は以前のように「速ければ良い」という構図から「汎用性を持ったマルチコアCPU, 柔軟性と速度をもったGPU」という構図に変わりつつある. CPUマルチコアプログラミングはOpenMP [06]のようなプロジェクトにより, より簡単にマルチコアに最適されたライブラリなどが多く開発されていくことが期待されている. CPUに比べ, GPUはコンピュータにおけるすべての演算処理が行えるわけではない, という基本構造は変わらないが, 実際のプログラミングにおける難易度やグラフィックスプログラミング特有の幾何処理などにおける優位性, 言語習得速度を考えると, GPU上における, HLSLやCgといった(C++ではない)C言語ライクな言語環境とテキストファイルで実験できるシェーダコーディングにも多くのアドバンテージがあるといえよう.

また, マルチコア化の流れが安定した後, 再び, PLAYSTATION®3(ソニー・コンピュータエンタテインメント社)のような, 「マルチコアCPU-GPU」といった処理系統になることも容易に想像がつき, 現状のGPU利用技術や特有のプログラミングスタイルは基盤技術として今後も生き続けていくといえるだろう. 現在はこのパラダ

イムにおける転換期にあるといえ，今後も注目が必要といえる．

4. マルチコア CPU の現状

コンピュータゲームの分野では，リアルな世界を表現するために，高精細の画像処理ができ計算速度の速いプロセッサが求められている．その一方で，18ヶ月毎に性能が倍になるというムーアの法則に従って発展してきたCPUだが，ここ数年はその限界がみえてきた．プロセッサの発熱のためクロック周波数が予定より上がらないことや，トランジスタ数が増えても計算能力が比例的に向上しないことから，性能の向上がムーアの法則に沿わなくなってきた．この技術的限界を超え世間の要求に応えるため，複数のプロセッサをつかうマルチコアCPUが脚光を浴びている．

マルチコアCPUは，同一のコアを複数使用するアーキテクチャと，異なる性能のコアを複数使用するアーキテクチャにわけられる．同一のコアを複数使う代表的なマルチコアCPUにはデュアルコアのAthlon 64 X2（AMD社），クアッドコアのCore 2 Extreme（インテル社），さらには8コアのUltraSPARC T1（サンマイクロシステムズ社）がある．また，インテル社が同一パッケージに数十個のCPUを集積させるメニイコアというアーキテクチャを発表している[07]．異なる性能のコアを複数使うアーキテクチャはヘテロジニアスマルチコアと呼ばれ，代表的なものにソニー・コンピュータエンタテインメント社と東芝社およびIBM社が共同で開発したCell Broadband Engine™（以下Cell）がある．このCellのアーキテクチャは今後のCPUのトレンドになるとみられている[08]．

Cellは一個の64bit汎用RISCプロセッサのPPE (PowerPC Processor Element)と8個の信号処理専用プロセッサSPE (Synergistic Processor Element)から構成される[09]．SPEは単一命令／複数データ(SIMD)演算ユニットのSPU (Synergistic Processor Unit)とローカルストレージエリア，およびMFC(Memory Flow Controller)からなっている[10]．PPEはシステム内のSPEに対するタスクの管理と割り当てを行う目的を持ち，SPUは高密度の計算ユニットを必要とし，かつ与えられた命令セットを有効に利用するアプリケーションを可能にする目的をもつ．Cellの浮動小数点演算パフォーマンスは動作周波数4.0GHzの場合で，256GFLOPSとなる．さらにネットワークによるグリッドコンピューティングのような分散処理にも対応しており，将来の高速ネットワーク時代でのコンピューティングにも備えている．Cellの応用として，PLAYSTATION®3やIBM社のスーパーコンピュータRoadrunnerに使用され，さらに大量の画像データをリアルタイムに扱う家電などにも使われる予定である．

CPUのパフォーマンスを示す具体的なデモソフトとして群衆を表現するものがある．Cell での動作事例として，SIGGRAPH Sandbox 2006で発表されたReynoldsらによるPSCrowd [11][12]などがあげられる．これはPLAYSTATION®3向けの群衆シミュレーションするソフトウェアライブラリである．コンピュータゲームに代表されるバーチャルワールドにリアルな雰囲気を持たせるため，その世界に生息する生物や群集，混雑する交通などを表現し，バーチャルワールドを複雑な生命の活動で満ちあふれるようにみせる目的で開発された．特徴としてCellの複数のSPUを使ってシミュレーションを行い，負荷分散を自動的に行う．PSCrowdのデモソフトではバーチャルワールド内において3Dで創られた1万匹の簡単な形状の魚が60fpsで自律的に動き回るという性能を出している．図1にそのスクリーンショットを示す．2Dの魚群であれば1万5千匹が動き回る．これに対してPCではGPGPUの技術でCPUとGPUを使った群衆デモソフトのFastCrowd Systemにおいて35fpsで1万個のキャラクタを動かすという性能にとどまっている．今後PSCrowdはネットワークで繋がれたCellを使ったシミュレーションへの発展や，メモリ効率を上げることが期待されている．

マルチコアCPUのソフトウェアの開発効率をあげるため，Mercury Computer System社のソフト開発キットであるMultiCore Plus SDK [13]や，RapidMind社のソフトウェア製品[14]などがある．また，RTT AG社では，RapidMind社の開発環境下でリアルタイムレイトレーシングをCell上に実装し，高いパフォーマンスを実証済みである[15][16]．マルチコアCPUにおいてはソフトウェアの効率化がハードウェア性能に直結するため，今

後ますますプログラミングの重要性が増すことが予想される.

図 1 PSCrowd のスクリーンショット

5. GPU 利用の現状

本章では, GPU の現状として, メインストリームである CG 技術への応用および, 付加的用途であるコンピュータビジョン, 物理シミュレーションへの応用例について述べる. また, 分散コンピューティングへの応用としてナノテクノロジー分野への利用の試みも紹介する.

5.1 GPUの広がりとCG技術の動向

GPU は PC のグラフィックスを処理する部品として登場したが, 現在では家庭用ゲーム機や業務用アーケード基盤まで幅広く利用されている. 家庭用ゲーム機では, PLAYSTATION®3 やマイクロソフト社の Xbox および Xbox 360 に搭載されており, 業務用プラットフォームでは, タイトー社の Type X [17]やセガ社のリンドバーグ (LINDBERGH) [18]に搭載されている. かつては, 家庭用ゲーム機や業務用アーケード基盤は, PC とは異なるアーキテクチャで設計されていたが, コストや開発効率の面から PC に近いハードウェアへ転換してきている.

特にセガ社が 2005 年に発表した LINDBERGH は非常に PC に近いアーキテクチャで設計されており, CPU は 3GHz で動作する Hyper Threading (HT)対応 Intel Pentium4, GPU は NVIDIA 社の GeForce6800Ultra 相当の GPU を搭載した業務用プラットフォームである. LINDBERGH を使ったビデオゲームでは, セガの「バーチャファイター5」が特徴的で, このゲームでは, 積極的に GPU を使ったテクニックが用いられている[19]. 業務用プラットフォームでは, PC アーキテクチャと比較して実行環境が安定かつ統一されているため GPU ごとの差異による配慮にとらわれずに開発できるメリットがある. そのため本作品の VS ではテクスチャ参照(VTF: Vertex Texture Fetch)などの GPU 依存の高いテクニックが利用されている.

LINGBERGH のように統一化されたプラットフォームにおける産業応用の場合, GPU の利用・開発手法が統一できるというメリットがあり, 特に PC アーキテクチャを対象に提案された先行研究や論文, 技術デモがそのまま実装できる利点もある.

PC では, SM4.0 のハードウェアが 2006 年の後半に登場した. NVIDIA 社は, GPU コア G80 のパフォーマンスを示すデモとして実在のモデルを再現することに挑戦した[20]. このデモでは, 1 ピクセルあたり 15 のレンダリングパスを使い総命令数 1,400 のシェーダプログラムでレンダリングを行っている. こうしたテクニックなども数年後には消費者向けのシステムで利用されると予想する. さらに NVIDIA 社は, PC やサーバーの外部に設置する, マルチ GPU を実装したグラフィックコンピューティングシステムである NVIDIA Quadro Plex [21]により, ビジュアライゼーションの高度化だけでなく GPGPU への適用の可能性を広げた.

5.2 コンピュータビジョン への応用

GPU は本来, 映像生成用に設計されているが, その構成技術は高速で高度なピクセル処理, ベクトル演算器, 大容量高速メモリであり, 他の画像処理用途, たとえばコンピュータビジョン(CV), マシンビジョンといった利用の可能性もある. この種の研究では Fung らによる OpenVIDIA [22]が有名であるが, 2005 年以降大きな成果が出ていない.

白井らは ATI 社製, NVIDIA 社製両 GPU 上において DirectX, HLSL を用いた高速なコンピュータビジョンプロジェクト GPUVision [23]を発表している. IEEE1394 経由の DV, WebCam, ビデオファイルといった各種のメディアから, テクスチャレンダリングのステップにピクセルシェーダを用いて実装していくスタイルをとっている. PS はその構造上, 並列処理は並列パイプラインにより高速に処理できるが, 画像上のピクセルをカウントする処理などは不向きである. つまり処理の 100%を

GPU 化することは高速化にはつながらず，基本的に GPU は CPU の前処理となる各種フィルタ処理を担当することになる．カラートラッキングや，マーカなしモーショントラッキング，背景差分処理などに効果があり，速度もカメラのフレームレートを大きく上回る 600fps などの処理が可能である．また，簡易なメディアアート作品などへの利用を想定し，パラメータ変更はプログラムの再コンパイルが不要で，画像ファイルとテキストファイル(HLSL)によって操作できる．さらに，VR アプリケーションオーサリングツールである Virtools へのツール環境への通信プログラムの提供や，「AceSpeeder2」といった実際のゲームプロジェクトの人体モーションインタフェース化[24]も試みている． 図 2，図 3 にそのスクリーンショットをそれぞれ示す．

GPU 上の CV はグラフィックスが最終出力ではない場合もあり，CPU→GPU→CPU といった処理になることが多い．つまり，メインメモリから GPU 側のビデオメモリに画像を転送し，さらにその結果を CPU 側に引き戻す必要があるため，同じ処理でもデュアルコア CPU 処理に対し，大きなアドバンテージをつけられるわけではない．しかしピクセルパイプラインによる処理の並列化，SM3.0 以降や PCI Express に含まれる GPU 側からのデータリードバックを向上させる技術により，今後，この状況は大きく変わる可能性がある．特にリアルタイムビデオや Augmented Reality，特に携帯電話や PDA などにおける処理において発展が期待されるところである．

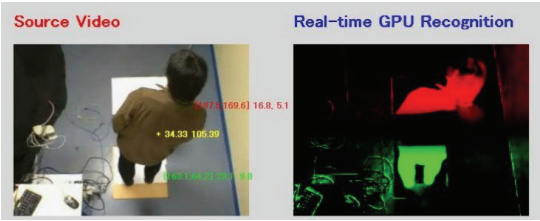


図 2　GPUVision のスクリーンショット

図 3 「AceSpeeder2 Motion Input Version」GPUVision のヒューマンインタフェース応用

5.3 物理シミュレーションの現状

GPU の登場によって，ビデオゲームにおける質感表現の向上が進んだ．しかし，ユーザが感じるビデオゲームにおけるリアルさの指標はグラフィックスのみをもって測るものではない．具体的には，サウンドや人工知能(AI)，物理シミュレーションなどの品質もリアルなビデオゲームを実現するための重要な技術的要素である．この中で，GPU に関連がある分野として物理シミュレーションがある．

物理シミュレーションは，可視化することへの要求が大きいためグラフィックスと親和性の高い話題である．ビデオゲームにおいては，剛体や流体などの力学の物理シミュレーションがよく用いられる．剛体は，3D 空間内の物体の衝突処理や移動の制御などに用いられ，流体は，水や煙などの表現に使われることが多い．ビデオゲームに物理シミュレーションを取り入れるためのライブラリと実行環境を物理エンジンと言い，商用のものからオープンソースのものまで様々なものがある．その中でも GPU 技術と関連したユニークな実装を行っている物理エンジンが 2 つある．1 つは，AGEIA 社の PhysX [25]であり，もう 1 つは Havok 社の Havok FX [26]である．これら 2 つの物理エンジンは，それぞれビデオゲームで良く使う物理演算の高速化やインタラクションが必要ない演出効果のための物理演算（効果物理と呼ばれる）などの，並列化の効く手法の処理を CPU 以外のプロセッサに処理をさせる．

AGEIA 社の PhysX は，Physics Processing Unit (PPU)と呼ばれる物理演算を行う専用のプロセッサを設計し，CPU の演算の負荷を減らす試みが行われている．これは，グラフィックスを CPU から専用のチップを設計して計算を割り当てる GPU の発想に近い．PPU は物理演算専用のコプロセッサであり，グラフィックスと非同期で処理される物理エンジンである．後述の HavokFX とのもっとも大きな違いは，GPU のような一般化したプロセッサではないという点である．現状，PPU はノート PC で利用することができないことと，PPU が PCI バスで接続されているため物理演算で使うデータをレンダリングで利用するためのデータの取り出しでバス間のデータ転送が生じるといったデメリットがある．

Havok 社の物理エンジンの Havok の 1 機能である Havok FX は，物理シミュレーションの演算を GPU に行わせることで高速化を行う．Havok FX では，NVIDIA 社の SLI，AMD 社の CrossFire のようなマルチ GPU 環境において，グラフィックス処理を行う GPU と物理演算を行う GPU をわけて，効率的な演算が出来るような仕組みもある．通常，3D のジオメトリデータは，GPU 上のメモリの上にデータが載っているが，これを対象に物理演算を CPU で行う場合には，PCI-Express などのバス間でデータをやりとりする必要がある．この処理は，ビデオゲームにおいて大きな処理のボトルネックになる．Havok FX では，物理演算を GPU で行うことでこのバス間の転送を減らすことが可能になる．

5.4 ナノテク応用

GPU を汎用計算に利用する試みの事例として，スタンフォード大学で進められている蛋白質の折り畳みの分散コンピューティングプロジェクト「Folding@home 」[27]がある．Folding@home は計画に賛同する PC に解析ソフトウェアをダウンロードし，ネットワークを介して複数のコンピュータで計算させる分散コンピューティングを行うプロジェクトである．その目的は蛋白質の折り畳みのシミュレートを行い，計算結果を集め，構造を解析することである．その結果は，アルツハイマー病，癌，ハンチントン病，骨形成不全症，パーキンソン病などの治療の助けになると期待されている．

Folding@home の分散コンピューティングを行うソフトウェアは，マイクロソフト社の Windows をはじめアップル社の MacOSX や Linux などの各種 OS に対応している．

さらにこの Folding@home に GPU の計算能力を利用するソフトウェアのテストが始まっている[28]．これは，特定計算において GPU の能力が CPU の数倍のパフォーマンスを持つためである．その計算能力は，ATI 社の X1900 クラス GPU を使った場合，100GFLOPS に達する．加えて 2007 年 3 月からはマルチコアプロセッサの Cell を搭載したコンピュータエンタテインメントシステム PLAYSTATION®3 がプロジェクトに加わった[29][30][31]．図 4 に，そのスクリーンショットを示す．

プロジェクトに参加している 20 万台のコンピュータに，仮にこれらの高い計算能力をもつプロセッサが普及すれば，総合で 10PFLOPS の計算力に達することになり，現時点で最高性能を持つ IBM 社のスーパーコンピュータ BlueGene/L の 367TFLOPS より高い能力となる[32]．GPU や Cell はこのプロジェクトにおいて，さらに計算力を押し上げる要素として期待されている．

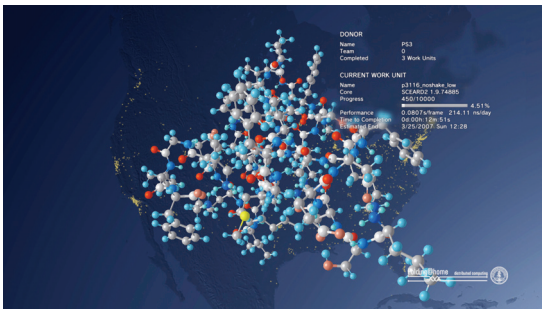


図 4 「Folding@home」PLAYSTATION®3 版

6.今後の方向性

本章では，これまで記した様々な構造・歴史を省みるとともに，近い将来の GPU 環境の将来について，特に新しい OS の登場による変化を予測できる範囲でまとめる．

6.1 旧来の対立構造

現在 PC クライアント OS では Microsoft 社の「WindowsXP」が主に使われている．Windows シリーズは，OS という市場構築の歴史とともにデファクトスタンダードの地位を獲得してきたが，一企業によって市場が

独占されている状態から，Windows に反発する動きも出てきた[33]．オープンソースの Linux [34]がその代表である． Linux は OS のソースコードが公開されているため，開発者にとっては自由度の高い OS と言える．そのため多くのライブラリやソフトウェア，さらにはカスタマイズされた OS 自身が草の根的に開発されている．また，動作の安定性や，導入コストの低さも魅力である．これらの理由から Linux は主に企業向けサーバーや学術向けに使われるようになってきたが，数の上では依然 Windows の圧倒的な市場シェアが続いている．

この OS における対立はグラフィック開発用 API (Application Programming Interface)まで巻き込んだものになった．Windows のマルチメディア機能を強化するための API 群である DirectX [35]と，ハードウェアや OS に依存しない API の OpenGL [36]の対立である．こちらも OS での対立と同様に，一企業提供で Windows にしか対応しない DirectX か，オープンな仕様の OpenGL かという比較がされた．

PC ゲームやマルチメディア処理を中心に発展してきた DirectX は Windows 上で性能を発揮し，その表現力を向上する役割を担ってきた．一方 OS に依存しない OpenGL は，機種を超えてソフトウェアの移植性に優れるという特色を持つ反面，オープンゆえに仕様の確定に時間がかかると言った弊害もあった．

OpenGL は ARB (Architecture Review Board) [37]と呼ばれるコンソーシアムが管理し，仕様の承認，改訂を行っていたが，2006 年に約 100 社の会員企業から出資された費用で運営される Khronos™ Group に管理を移行し，API 仕様を確立する体制になった [38]．この移行により仕様決定のスピードアップと各種プラットフォーム間で統一した仕様の確定が期待される．

さらにこれらの OS, API が使われる組み込み機器やゲーム機，PC といった開発ターゲットにおいても対立がみられた．PC と組み込み機器のどちらがテクノロジードライバとなり技術や産業を牽引してゆくか，ということが経済に大きな影響を与えるためである．加えて，機器にどのような OS を搭載するか，どのような API で開発するかということがターゲット機器の特徴を決定することも関係している．

PC では機器構成が Windows に合わせて設計されているため，Windows が使われる事が多い．一方の組み込み機器では多様な CPU，ユーザインタフェースなどが使われるという特徴を持つため，柔軟な開発・メンテナンスが可能で，生産性が高く，ネットワークへの対応，画像処理の高度化対応への必要性から，Linux が使われるようになった[39]．

このような対立はメーカ，ユーザ，開発者まで巻き込んだ熾烈なものだったが，最近では内容が変貌している．

6.2 対立構造の終焉:Vista世代

前節で述べた対立は，現在では一般ユーザの PC には Windows，サーバーなどの専門用途の PC や組込機器には Linux というように，ほぼ棲み分けが確立しそれぞれの分野で発展し続けているため，対立そのものが意味をなさなくなり終焉を迎えた観がある．

API においても，ハードウェアに近い位置づけでアマチュアゲームプログラマに人気が高い Direct3D (DirectX の 3D グラフィックス用コンポーネント)は，シェーダプログラミングが可能となり高度なプロ向け API の位置を確立した．一方の OpenGL は，一部の CAD 用グラフィックワークステーション向け用途としての位置を確立した．加えて，OpenGL には GPU メーカ固有のコードに互換性こそ無いが，ARB を通じて試験実装可能なので，OpenVIDIA のような実験的な取り組みも多く存在している．OS とハードウェアの中間部分であるドライバソフトウェアも従来は，GPU ベンダーからは Windows 環境向けのドライバしか提供されてこなかったが，現在は MacOSX や Linux といった各種プラットフォーム向けドライバも公式に提供されることが珍しくなくなってきた．

このように棲み分けが確立した後の世界において動向が注目されるのが PC の OS とアプリケーションの開発環境である．最新の Windows である Windows Vista ではゲームや特定のアプリケーションのみに使用が限定されていたリッチな 3D グラフィックスが OS のグラフィックユーザインタフェースに使われるようになった[40]．ユーザインタフェースの生成，操作性の統一のために Windows 用プレゼンテーションサブシステムである

WPF (Windows Presentation Foundation) [41]が使用される. たとえば文字の描画といった GUI の基幹部分における描画処理にも GPU によるフォントレンダリングやアンチエイリアシングといった技術が標準利用される. このプラットフォームにおいては, GPU の高いパフォーマンスが一般的用途で発揮されることが期待される. またエンドユーザ・デベロッパは XAML (Extensible Application Markup Language)というテキストで記述できる言語を用いて WPF を活用することができる. XAMLを用いたコンパイル不要の 3Dコンテンツ開発が, メディアアートなどへの応用も含め, 大きく GPU の利用者の裾野を広げていくだろう.

ハイエンドな開発には Windows Vista がサポートする Direct3D 10 [42]が活用されていくと予想される. Direct3D 10 では, SM4.0 をサポートした GPU を用いたプログラミングが可能となる. 前世代の GPU と比べ大きく変化した点はグラフィックスパイプラインが進化し, VS, PS に加えて新たに GS が追加された点である. この GS は単一の頂点入力から複数頂点を出力することを可能とし, シャドウボリュームやディスプレイスメントマッピングに応用される.

さらに Direct3D 10 では, ハードウェアサポート機能の相違を表現していた CAPS (Capability Set)が廃止される. これは「Direct3D 10 を用いたプログラムならどのハードウェアでも動作する」ことを意味するので, 開発者は従来までのように各 GPU が持つ機能の差を考慮する必要がなくなる. 技術革新により新たな機能が追加される場合には Direct3D 10.1 というようにマイナーバージョンアップによる対応がなされるとアナウンスされている.

7. まとめ

以上, GPU の取り巻く現状として, SM の変遷, GPU の利用法, 開発環境の現状, および GPU の具体の適用事例を紹介した. また, CPU の最新動向であるマルチコア化の現状についても触れ, 近未来の GPU をとりまくコンピューティング環境の方向性を示した.

総じていえば, 前回の GPU に関する解説論文(2004 年末)[04]のまとめで述べた「予想された状況」が現実になり, 具体的なパラダイムシフトの渦中にあるという状況が現在(2006 年末)ではないだろうか. GPGPU 研究の黎明期はすでに終わりを迎え, GPU プログラミング環境は, Vista プラットフォームや Linux により, さらなる柔軟性を得ていくだろう. 一方では 2 年前の緒言とした「GPU コンピューティングを支える人材の育成」も相変わらず課題として残されている(この 2 年間で GPU プログラミングを主題とした一般技術書籍はほとんど刊行されていない).

あえて強調するまでもなく, この分野の新陳代謝のスピードは非常に速く, 本論文が掲載される頃には一部内容が陳腐化している可能性がある. しかし, 区切りとなる機会ごとに歴史をまとめることで頭の中を整理し, 将来の研究への道筋を見出す一助となればと思い, ここに述べた次第である.

謝辞

本論文の一部は JSPS 海外特別研究員(平成 17 年度)の支援による. ここに謝意を表する.

参考文献

[01] Ikonas http://www.virhistory.com/ikonas/ikonas.html

[02] Potmesil, M. and Hoffert, E. M. "The pixel machine: a parallel image computer," In Proc. of SIGGRAPH '89. pp.69-78. 1989

[03] Fuchs, H., John Poulton, John Eyles, Trey Greer, Jack Goldfeather, David Ellsworth, Steve Molnar, Greg Turk, Brice Tebbs, Laura Israel "Pixel-planes 5: a heterogeneous multiprocessor graphics system using processor-enhanced memories," In Proc. of SIGGRAPH '89. pp.79-88, 1989

[04] 宮田, 高橋, 黒田, "GPU コンピューティングの動向と将来像",芸術科学会論文誌, Vol.3, No.1, pp.13-19, 2005

[05] http://www.beyond3d.com/content/articles/83

[06] OpenMPのHP http://www.openmp.org/drupal/

[07] Hiremane, R. ムーアの法則からインテルのイノベーションへ: 予測を現実に Technology@Intel Magazine, 2005

http://download.intel.com/jp/developer/jpdoc/moores-law-0405_j.pdf

[08] 後藤弘茂,「決定的となったヘテロジニアスマルチコアへの潮流」, PC Watch, 2006.08.18 http://pc.watch.impress.co.jp/docs/2006/0818/kaigai295.htm

[09] "Cell Broadband Engine™ processor –based systems White Paper" IBM Corporation Systems and Technology Group 2006.9

[10] "Cell Broadband Engine™ アーキテクチャ Version 1.0 "Sony Computer Entertainment Inc. 2005.8.8

[11] "2006 ACM SIGGRAPH Video Game Symposium "Sandboxのサイト http://sandbox.siggraph.org/archives/program06.html

[12] Reynolds,Craig "Big Fast Crowds on PS3" Sony Computer Entertainment,US R&D 2006 http://research.scea.com/pscrowd/

[13] http://www.mc.com/products/productdetail.aspx?id=2826

[14] http://www.rapidmind.net/product.php

[15] http://www.rtt.ag/rtt/index.html

[16] http://www.rapidmind.net/pdfs/RapidMindRaytracer.pdf

[17] http://www.typex.taito.jp/

[18] http://sega.jp/corp/release/2005/0901/

[19] http://www.nzone.com/object/nzone_adrianne_home.html

[20] http://www.watch.impress.co.jp/game/docs/20061025/3dvf5.htm

[21] http://www.nvidia.co.jp/page/quadroplex.html

[22] Fung, J. and Mann, S. "OpenVIDIA: parallel GPU computer vision". In *Proceedings of the 13th Annual ACM International Conference on Multimedia*. 2005.

[23] Akihiko Shirai, Masafumi Takahashi, Kiichi Kobayashi, Hideki Mitsumine, Simon Richir, "A new real-time video synthesis method for virtual studio environments using GPU and projected screens", SIGGRAPH 2005 Poster.

[24] Akihiko Shirai, Takahiro Nakatani, Erik Geslin, Hidetaka Kimura, Masafumi Takahashi, Kazunori Miyata, Simon Richir, "Physical evaluation of new computer entertainment interfaces under natural play conditions", Sandbox: an ACM SIGGRAPH Video Game Symposium, Boston , 2006.

[25] http://www.ageia.com

[26] http://www.havok.com

[27] "Folding@home", Stanford University, http://folding.stanford.edu/

[28] "Folding@home on ATI GPU's: a major step forward", Stanford University, http://folding.stanford.edu/FAQ-ATI.html

[29] "Folding@home on the PS3", Stanford University, http://folding.stanford.edu/FAQ-PS3.html

[30] "Folding@home on the PS3 Advanced features for the PS3", Stanford University, http://folding.stanford.edu/pics/PS3-shot-00008.jpg

[31] Sony Computer Entertainment America Research and Developmentのサイト http://research.scea.com/2006-09-folding@home/index.html

[32] "Folding@home Petaflop Initiative (FPI)", Stanford University, http://folding.stanford.edu/FAQ-FPI.html

[33]「日本OSS推進フォーラムとは」日本OSS推進フォーラムのサイト http://www.ipa.go.jp/software/open/forum/

[34]「Linuxとは」www.linux.or.jp 管理グループ(Webmasters)のサイト
http://www.linux.or.jp/general/linux.html
[35]「Windows DirectX テクノロジ概要」Microsoft Corporationのサイト 2006
http://www.microsoft.com/japan/windows/directx/productinfo/overview/default.mspx
[36]「OpenGL About OpenGL OpenGL Overview」日本SGIのサイト
http://www.sgi.co.jp/visualization/opengl/
[37] "About the OpenGL Architecture Review Board Working Group" OpenGL.org 2006
http://www.opengl.org/about/arb/
[38]「ニュースリリース:OpenGL ARB, クロノス・グループにOpenGL の仕様管理を移行」Khronos.orgのサイト 2006
http://www.khronos.org/news/articles/20060731_Khronos_OpenGL_ARB_jp.pdf
[39] 近藤純司「組み込みLinux業界動向」アットマークIT 2004.11.26
http://www.atmarkit.co.jp/fembedded/trend0411/trend01.html
[40]「Windows Vista™ Developer Center Windows Vista ディスプレイドライバモデル」Microsoft Corporation のサイト 2006
http://www.microsoft.com/japan/msdn/windowsvista/general/WinVistaDisplayDriverModel.aspx
[41] Windows Presentation Foundation Microsoft Corporation のサイト
http://www.microsoft.com/products/expression/ja/wpf/default.mspx
[42] Blythe, D. "The Direct3D 10 System". ACM Trans. Graph. 25, 3 (Jul. 2006), 724-734.

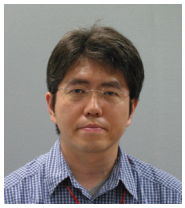
著者略歴

新庄 貞昭
1983 年金沢大学工学部電気工学科卒. 現在, 北陸先端科学技術大学院大学知識科学研究科博士前期課程 MOT コース在学. 株式会社ソニー・コンピュータエンタテインメント CS 部勤務. 芸術科学会, 研究・技術計画学会 各会員.

高橋 誠史
2003 年多摩大学経営情報学部経営情報学科卒. 2005 年北陸先端科学技術大学院大学知識科学研究科知識社会システム学専攻修了. 現在, 同研究科博士後期課程在学. 情報処理学会会員.

木村秀敬
2005 年茨城工業専門学校機械・電子制御工学専攻了. 2007 年北陸先端科学技術大学院大学知識科学研究科知識社会システム学専攻修了.現在, 株式会社 jig.jp 勤務.

白井 暁彦
1996 年東京工芸大学工学部写真工学科卒. 1998 年同工学研究科画像工学専攻修了. 2004 年東京工業大学総合理工学研究科博士後期課程了. 博士(工学). 現在, フランス国立工芸大学(ENSAM)客員研究員. 芸術科学会, 日本 VR 学会, ACM SIGGRAPH 各会員.



宮田一乘
1984 年東北大学工学部応用物理学科卒. 1986 年東京工業大学大学院総合理工学研究科物理情報工学専攻修了. 同年日本アイビーエム(株)東京基礎研究所入社. 1998 年東京工芸大学芸術学部助教授. 2002 年より, 北陸先端科学技術大学院大学知識科学教育研究センター教授. 博士(工学). 情報処理学会, 芸術科学会, ACM,IEEE 等会員.